Probabilistic Dictionaries

Hacker’s LOCKER

Mass Transit Forth

Another Subroutine Technique
The ForthCard

STAND ALONE OPERATION
STD BUS INTERFACE
EPROM/EEPROM PROGRAMMER
RS-232 I/O
PARALLEL I/O
ROCKWELL FORTH CHIP

The ForthCard provides OEMs and end users with the ability to develop Forth and assembly language programs on a single
STD bus compatible card.

Just add a CRT terminal (or a computer with RS-232 port), connect 5 volts and you have a self contained Forth computer.
The STD bus interface makes it easy to expand.

Download Forth source code using the serial port on your PC. Use the onboard EPROM/EEPROM programming capability
to save debugged Forth and assembly language programs. Standard UV erasable EPROMs may also be programmed with
an external Vpp supply.

NEW! Options and Application Notes
Electrically Erasable PROMs (EEPROMs)
FREEZE the dictionary in EEPROM (save in non-volatile memory, to be restored on power up)
Download Software for your IBM PC or CP/M
Non-Volatile CMOS RAM with battery 2K, 8K, optional Clock/calendar
Fast 2MHz clock (4MHz crystal)
Disk Controller Card (5¼"
Self Test Diagnostics
Parallel printer interface

Ask about our ForthBox™
A complete STD bus oriented system including the ForthCard, Disk Controller, Disk Drive(s),
STD Card Cage, Cabinet and power supply.

CALL TODAY FOR COMPLETE INFORMATION!

HiTech Equipment Corporation

9560 Black Mountain Road
San Diego, CA 92126
(619) 566-1892

Evaluation Unit $299
Part #STD65F11-05 includes: ForthCard, Development ROM, 8Kbyte RAM, Manuals

OEM Version as low as $199
Part #STD65F11-00 does not include memory or manuals
FORTH Dimensions
Published by the
Forth Interest Group
July/August 1985
Volume VII, Number 2

Editor
Marlin Ouverson
Production
Cynthia Lawson

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

Subscription to Forth Dimensions is free with membership in the Forth Interest Group at $15.00 per year ($27.00 foreign air). For membership, change of address and to submit material for publication, the address is: Forth Interest Group, P.O. Box 8231, San Jose, California 95155.

Feature 1: Menus in Forth
by Frans Van Duinen
Make life easier for users of application programs — this article includes definitions that make it simple for you to develop menus.

Feature 2: Another Subroutine Technique
by Donald Simard
Want to call code subroutines from either colon or other code definitions, but can't afford the run-time penalty of earlier methods? This one may come closer to the mark for you.

Feature 3: The Hacker's LOCKER
by Cecil McGregor
If a terminal supports line lock, one can use it from Forth to preserve useful data on the screen. Here's a simple way to do it.

Feature 4: Mass Transit Forth
Bus passengers in this English city don't have to wonder when a late bus will arrive or whether an approaching bus is the correct one to board. A Forth application tells them, at the touch of a button.

Feature 5: Forth Spreadsheet, Part II
by Craig A. Lindley
This source code accompanies the article and pseudo-code in the preceding issue, giving Forth users a customizable application program. (For a machine-readable version, turn to the end of the listing.)

Feature 6: Rochester Forth Conference 1985
Forth programmers, project managers and visionaries from several countries convened recently to discuss software engineering, management and productivity. The result was an exciting display of Forth and its adherents at their best.

Feature 7: Probabilistic Dictionaries
by John S. James
Have you wondered how a 60,000-word spelling dictionary can be compressed into RAM? Ultra-fast text searches can be performed using techniques pertinent to a variety of applications.

Departments

5 Letters
9 Editorial: "Journey to the East"
12 Ask the Doctor: "Forth on the Front" by William F. Ragsdale
42 FIG Chapters
MOVE UP TO 16bit FORTH COMPUTING
ENTER THE POWERFUL WORLD OF
68008 PROCESSING TODAY
FOR ONLY $399

HC120 8MHz
- 68008 CPU
CARD

HIGH SPEED
CMOS LOGIC FOR
HIGH NOISE IMMUNITY
& LOW POWER CONSUMPTION
1 MEGABYTE LINEAR ADDRESSING
2 SERIAL PORTS - RS232/422/423
20 PROGRAMMABLE I/O LINES
REAL TIME CLOCK
HARDWARE WATCHDOG
ON BOARD BATTERY BACK-UP

TRIAL OFFER INCLUDES
☆ POWERFUL 16K ROM BASED LAXEN & PERRY'S F83
☆ COMPREHENSIVE 16K DEBUG PACKAGE
☆ BATTERY BACKED 24K RAM & CLOCK
☆ COMPLETE 80 PAGE MANUAL WITH CIRCUITS
☆ FULL MULTI-TASKING F83 FORTH SOURCE LISTING
☆ CAN RUN TOTALLY "STAND ALONE"
ALL THIS FOR JUST $399

DISC BASED FORTH SYSTEMS ALSO AVAILABLE FROM THE HC RANGE.
EXAMPLE:
HC100 - 8 SLOT BACKPLANE $99
HC110 - POWER SUPPLY 110/220/240v (please indicate) $199
HC130 - DMA DISC CONTROLLER WITH 8K BYTES
CACHE FOR SUPERFAST COMPILATIONS $299
F83 - FULL SOURCE ON 5½/3½ DISC (please indicate) $29
AND MORE! - WRITE FOR DETAILS ADD $10 FOR SHIPPING
80 PAGE SYSTEMS MANUAL $25

SEND MONEY ORDER/CHEQUE WITH ORDER, ALL MAIN CURRENCIES ACCEPTED. DELIVERY 2-4 WEEKS.

Ctronix Ltd.
39 HIGH STREET, COWBRIDGE, SOUTH GLAMORGAN, U.K.
TEL. (04463) 2409 & 4661 TELEX 498215 CF77AE
Standard Bearers

Dear Mr. Ouverson:

The Financial Services Division of Computone Systems has developed an applications language specific to insurance using the Forth-83 Standard. We think the remarkable thing about our language is that it runs on 8086/8088, 6502 and Z80-based machines. There is a lot of talk about languages being transportable, but we found that most of it really is nothing but talk. Without the new standard, it is not likely that we could have accomplished the level of transportability that we have.

Computone has been developing software in Forth for the past two and one-half years. Initially, Forth was dictated by the machine we chose to market. We began our support of additional machines by purchasing Forth packages from outside vendors.

One fact became clear early in our development phase: there was no generally accepted standard and little continuity in vocabularies and functions among the Forth versions we were working with. These differences between versions prevented our applications code from being as transportable as we needed. Eventually, we had to decide whether or not to continue our development exclusively in Forth.

When evaluating other languages as possible substitutes for Forth, we found that several of them offered more complete development systems and more liberal licensing agreements. Many of the packages offered better speed and more thorough floating-point tools. On the other hand, there was still no language available that we could alter so that the applications code would be identical on all the machines we were supporting. We found that the better-supported systems were expensive and had their own drawbacks: link/compile time, non-standard extensions, speed or size limitations and highly variable support for machine-specific devices, I/O, etc.

Forth was one language that could be altered at the machine level. If the right version could be found for each processor we were supporting, adjustments could be made to the kernel via the metacompiler to develop the multimachine applications language we needed.

After some investigation, we found that several individuals had produced public-domain versions based on the new Forth-83 Standard. These versions were similar in philosophy and came complete with the source for their compilers, and had most of what we needed for development.

Using the public-domain versions, we were able to extend the machinespecific code and duplicate the current language that our applications team was using. We found that by simply transferring the high-level code from one machine to another and recompiling, our applications would run on virtually all our machines. We eliminated the eventual need for retraining our staff and extended the useful life of the tools and applications we had developed.

Ultimately, our decision to develop exclusively in Forth allowed us to cut our programming costs and development time drastically. We were able to simplify software development by reducing the number of systems we had to learn and use. We increased productivity by reducing the number of application variations that had to be developed and supported. We were able to deliver the software faster and with far fewer problems. The final result was an increase in our bottom line, and after all, the real value of a language is in how it impacts profits.

We are convinced that without the 83 standard, our integration of language and systems could not have been achieved. The standard proved to be a rallying point from which individuals were willing to work to produce systems which were compatible over a wide range of machines and available to the Forth community at low cost. The presence of a standard for high-level code only serves to make the language more transportable and useful to developers who work with an ever-increasing number of machines. The presence of standards within the community need not curtail the adaptability of the language, but rather serves as a bridge to new solutions, tools and ideas. The new standard and its wide acceptance will help Forth gain the industry-wide credibility it deserves.

Sincerely,

Michael D. Pollard
Donald S. Schrader
Computone Systems
Atlanta, Georgia

Hore Today, EXITed Tomorrow

Dear FIG:

Hore (Forth Dimensions VI/6) provides a solution to the so-called "double-loop-exit" problem, as it has been referred to in other quarters. His solution entails the derivation of an alternative DO LOOP structure.

Screen 1 shows yet another alternative. In this solution, the system-dependent word DROPLOOP is created which, when executed, allows a normal EXIT (given as :S in the example, as it is in a fig-FORTH dialect).

DROPLOOP does what the sequence R > R > 2DROP would do if it were included at the equivalent point in the definition of LOOKUP.

For those unfamiliar with the use of EXIT, it forces the end of the word, from anywhere in the word — with the notable exception of within a DO LOOP.

DO LOOPs typically leave two or more values on the return stack. Since EXIT makes use of the value which sits below those two or more values, we have to dispose of them before we can exit. A little brute force (trial and error) experimentation will show you how your system works. The only other element you may need to take care of would be a Forth-83 implementation of DO LOOP, which sets a flag to indicate being in a loop. Your system probably doesn't do this, but the flag should be indicated in your user variables if it does.

The objection raised to the kind of maneuver presented here typically
comes from the Wirth school of programming — the solution used does not have a "common end." And, so the argument goes, without a common end, another person reading a long program will get lost and be unable to follow the flow of the program. Arguing from first principles, a la Dijkstra, it sounds sound.

Evidence, however, has a way of modifying principle. LOOKUP is, in fact, totally unambiguous. A stranger to the code (but not to Forth, of course) would understand immediately the action of the word. The difference which matters here is the size of the word: when the program is trivially small and is made descriptive, it is demonstrably good programming; but when the program is trivially large, it is demonstrably bad. Which, as I think about it, isn’t a bad description of what a Forth program should — and can — be.

I have included screen 2 to show some more uses of those interesting words COMPILE, [COMPILE] and IMMEDIATE. It was my fifth time through Starting Forth before I began to grasp them. Useable examples sometimes teach better than either words or pictures.

The word DO+ saves a little typing and eliminates the clutter of all those OVER + SWAP sequences we use with a range of addresses. LOOP.EXIT performs the functions of DROP.Loop and EXIT, as the F83 word does. I typically keep them separate in my programs, as I have found times when program flow is clearer if I do.

Regards,

Henry J. Fay
Cazenovia, New York

Missing Mathquiz

Dear Marlin,

I’d like to look at the Mathquiz program on pages 13-14 of Forth

Mathquiz

SCR #72
0 ( PROGRAM VARIABLES )
1 ( COMPILE IMMEDIATE"); ( Get and store players choice # )
2 ( COMPIL]E IMMEDIATE"); ( Store result for +, -, * )
3 ( COMPIL]E IMMEDIATE"); ( Maintain players # of correct ans )
4 ( COMPIL]E IMMEDIATE"); ( Store difficulty level )
5
6 ( Delay routines )
7 ( DELAY 15000 0 DO LOOP ; ( Null loop time delay )
8 ( DELAY 15000 0 DO LOOP ; ( Longer time delay )
9
10
11
12
SCR #73
0 ( TITLE PAGE--MATH-QUIZ )
1 ( STAR 42 EMIT )
2 ( STARLINE 40 0 DO STAR LOOP )
3 ( BLANKLINE STAR 38 SPACES STAR )
4 ( INFO-L1 STAR 14 SPACES " MATH QUIZ "
5 ( 14 SPACES STAR )
6 ( INFO-L2 STAR 13 SPACES " VERSION 1.3 "
7 ( 13 SPACES STAR )
8 ( INFO-L3 STAR 10 SPACES " BY "
9 ( 10 SPACES STAR )
10 ( INFO-L4 STAR 12 SPACES " LYLE D. MORTON "
11 ( 12 SPACES STAR )
12 ( MATHTITLE PAGE STARLINE BLANKLINE
13 ( BLANKLINE INFO-L1 BLANKLINE INFO-L2
14 ( BLANKLINE INFO-L3 BLANKLINE
15 ( INFO-L4 BLANKLINE INFO-L3 STARLINE CR CR )

SCR #74
0 ( INSTRUCTIONS FOR MATH-QUIZ )
1 ( INSTRUCT 3 SPACES " HELLO, YOU "
2 ( . " ARE ENTERING THE WORLD " CR
3 ( . " OF MATH-QUIZ. PLEASE ENJOY "
4 ( . " YOURSELF: CR ", " BE ADVISED "
5 ( INVERSE " THE UPPER LEVELS MAY REQUIRE "
6 ( CR " A PENCIL AND PAPER. "
7 ( NORMAL " GOOD LUCK! "
8 ( Continued

FORTH Dimensions 6 Volume VII, No. 2
Dimensions (VI/6) but some of the screens seem to be missing. Words like GETNAME and ANSWER in the text, and many of the words shown in screen 83 are not defined. Could I get a copy of the full set of screens? I’d like to set up the program for my wife's school kids and I'd like to study it to learn something about keyboard input.

Klaus Schleisiek has picked a wonderful place as site of the euroFORML conference. I’ve been in Heilbronn and through the Neckar valley a couple of times. I'll be at the European FORML conference for sure.

Best!

Nathaniel Grossman
Los Angeles, California

Editor's note: Sincere apologies go to our readers and to author Lyle D. Morton who wrote "Mathquiz." In our issue VI/6 we unintentionally printed only four screens (numbered 71, 79, 80 and 83). The accompanying screens 72-78, 81 and 82 are published here to complete the program.

You Screen, I Scream

Dear Marlin,

The Forthodoxy commands that we Forth our programs into screens, the "natural unit of thought." As a confirmed heterodox, I think it is time to examine what screens offer and do not offer, in contrast to their alternative.

Right off, let's admit that screens are not a natural unit of anything except computer memory or CRT display. The mind doesn't inherently work with 1024-byte units — certainly not when any number of those bytes can be blank. To the extent that screens keep us 1K-bound, they make it difficult to deal with our code at flexible levels of detail and generality.

To appreciate how dysfunctional screens are, note their perfect analogy to the numbered lines of BASIC. Like line numbers, screen numbers are inherently meaningless, presenting housekeeping chores and working against self-documentation. The process of copying, moving and inserting screens is tedious (even hazardous),

```
SCR #75
  0 # MATH QUIZ ---- ADDITION MODULE
  1
  2 VARIABLE ADD1 0 ADD1 ! ( addend # 1 )
  3 VARIABLE ADD2 0 ADD2 ! ( addend # 2 )
  4 # ANSWER QUERY 8 "WORD NUMBER DROP ; ( gets players answer
  5 # ADDITION ( MAKE PROB & GET ANS)
  6 LEVEL @ CHOOSE ADD1 ! ( get difficulty level and )
  7 LEVEL @ CHOOSE ADD2 ! ( use CHOOSE to select addends )
  8 CR ADD1 @ ." + "
  9 ADD2 @ ." = " ( 8 & 9 format problem )
 10 ANSWER
 11 ADD1 @ ADD2 @ + DUP RESULT ! ( add the addends & store it )
 12 = IF " CORRECT " 1 SCORE +
 13 ELSE CR " WRONG, THE CORRECT
 14 ." ANSWER IS " RESULT @ . THEN ;

SCR #76
  0 ( SUBTRACTION MODULE )
  1
  2 VARIABLE SUB1 0 SUB1 ! ( minuend )
  3 VARIABLE SUB2 0 SUB2 ! ( subtrahend )
  4 # SUBTRACTION ( MAKE PROB-GET ANS)
  5 LEVEL @ DUP CHOOSE * ( add random number to LEVEL )
  6 SUB1 @ ( to make a minuend ) subtrahend )
  7 LEVEL @ CHOOSE SUB2 ! ( choose subtrahend )
  8 CR SUB1 @ ." - "
  9 SUB2 @ ." = " ( 8 & 9 format problem )
 10 ANSWER
 11 SUB1 @ SUB2 @ - DUP RESULT ! ( get difference and store )
 12 = IF " CORRECT " 1 SCORE +
 13 ELSE CR " WRONG, THE CORRECT
 14 ." ANSWER IS " RESULT @ . THEN ;

SCR #77
  0 ( MULTIPLICTION MODULE )
  1
  2 VARIABLE MULT1 0 MULT1 ! ( multiplicand )
  3 VARIABLE MULT2 0 MULT2 ! ( multiplier )
  4 # MULTIPLICATION ( get problem and players answer )
  5 LEVEL @ CHOOSE MULT1 ! ( randomly choose multiplicand )
  6 LEVEL @ CHOOSE MULT2 ! ( randomly choose multiplier )
  7 CR MULT1 @ ." * "
  8 MULT2 @ ." = " ( 7 & 8 format the problem )
  9 ANSWER
 10 MULT1 @ MULT2 @ * DUP RESULT ! ( get product and store )
 11 = IF " CORRECT " 1 SCORE +
 13 ELSE CR " WRONG, THE CORRECT
 14 ." ANSWER IS " RESULT @ .
 15 THEN ;

SCR #78
  0 ( DIVISION MODULE )
  1
  2 VARIABLE DIV1 0 DIV1 ! ( divisor )
  3 VARIABLE DIV2 0 DIV2 ! ( quotient )
  4 # DIVISION ( GET PROB & ANSW)
  5 LEVEL @ CHOOSE 1+ DUP DIV1 ! ( get > 0 divisor )
  6 LEVEL @ CHOOSE 1+ DUP DIV2 ! ( get > 0 quotient )
  7 * RESULT ! CR ( develop dividend )
  8 RESULT @ ." / "
  9 DIV1 @ ." = " ( 8 & 9 format the problem )
 10 ANSWER DIV2 @ ( compare quotient with player's ans )
 11 = IF " CORRECT " 1 SCORE +
 12 ELSE CR " WRONG, THE CORRECT
 13 ." ANSWER IS " DIV2 @ . THEN ;

Continued
```
hence we create "spaghetti loads" that make source code harder to follow. Indeed, the GOTO command is BASIC's expression at a different level.

In addition to spaghetti loads, screen boundaries encourage "meatball code." Long, horizontally formatted, uncommented definitions are a natural result of a screen boundary that takes effort to cross — and the fact that blank areas get wasted. By the way, index lines — manually added redundancy — are another space waster.

So far, I've said that screens work against flexibility, space economy and self-documentation. Readability suffers from spaghetti and meatballs, plus excess baggage (screen number in decimal and hex, initials, date, index line and line numbers). Screen editors themselves are horribly weak, so even interactivity can suffer. There is hardly a virtue claimed by Forth that screens do not work against.

When resources are limited, screens offer an easily implemented, interactive way to write programs in an incremental fashion. Fortunately, resources are no longer that limited, and file systems with powerful editors have already been implemented. The most natural, least restrictive way to produce code is with good text editors or word processors. I won't dwell on their advantages over Forth editors. To anyone who has tried them, the difference is obvious (in New York, "FIG edit" means "forget it"!)

Besides offering greater power, ease and flexibility, a real editor conserves disk space (I know — big deal) while encouraging commenting and high readability. With a good file system, it offers self-documenting file names for code modules of whatever size the programmer chooses.

Any problems? One is indexing; without it, well-organized code in well-named files is important (a good idea in any case). Another problem is navigation between Forth and the editor; to which fast operating systems, multitasking, desktop utilities or non-screen editors within Forth are all solutions. Lastly, compilation speed may suffer, depending on how politely Forth relates to its host; the trend is toward politeness, which eliminates the problem.
Journey to the East

We recently returned from the annual Rochester Forth Conference. The lively event was sponsored by the Institute for Applied Forth Research in cooperation with the Laboratory for Laser Energetics and the IEEE Computer Society. That says something interesting about the times if anything does. The directors and staff of the Forth Institute assembled a stimulating program that is reviewed elsewhere in this issue in necessarily brief terms — there were about sixty presentors.

In this issue we make good on promises past. Due to a last-minute error, our last cover promised "Menus in Forth" but it was nowhere to be found inside — the article had been rescheduled. This time you'll find it, honest! Also included herein are screens to accompany Morton's "Mathquiz." Thanks to the readers who wrote to request them. As a side note, columnists John Hall and Henry Laxen are absent, at least for the time being.

Add to the list of FIG membership benefits a new one: major medical insurance. In the U.S. only, FIG members, their families and possibly their employees are now eligible for group rates. Current members will be receiving notice in the mail with specifics about the health-care plan. Those of you who work independently or who run a small business may be particularly interested.

And, finally, we would like to welcome five new chapters of the Forth Interest Group. Members are now holding official meetings and other Forth-related activities at MAD Apple FIG Chapter, Madison, Wisconsin; Cache Forth FIG Chapter, Oak Park, Illinois; Permian Basin FIG Chapter, Odessa, Texas; Japan FIG Chapter, Tokyo and Kyoto, Japan; and Rockland County FIG Chapter, Pearl River, New York. Welcome to each group, may your meetings prosper. For a complete listing of all international FIG Chapters, turn to the back pages of this magazine.

—Marlin Ouverson
Editor

I've used a file-and-editor system for three years, and in spite of dire warnings, the only lightning I've been struck by is my own programming speed. I think it is vital that we move toward elimination of the restrictions represented by screens. Unless it can adapt to the progress around it, the Forth may not always be with us.

Sincerely,
Laughing Water
Helena, Montana

Division Floors Him

Dear Marlin,
I have never been one to pride myself on an understanding of divisional mathematics, and the debates between floored-to-zero versus floored-to-negative-infinity have escaped me. Well, you can imagine my delight when I recently had occasion to use division in a situation that illustrates the usefulness of the latter method.

In an appointment scheduler, the day has been divided into five periods. It is necessary to access prior and later periods. A single word periodaway can handle the necessary calculations. adjustperiod# uses /MOD to calculate the number of days crossed and resets period#. This simple math is all that is required for any number of periods. The word dayinterval adjusts the date for a change of n days. It is dependent on the calendar module used and serves no purpose in this discussion.

The periods used are: 0 = dawn, 1 = morning, 2 = afternoon, 3 = evening and 4 = night. If one begins with morning (setting period# to 1), then -7 periodaway resets period# to -6 and -6 5 /MOD returns 4 -2. This adjusts the date back two and resets the period# to 4 (night), as it should be. What could be simpler?

I raise my hat to the standards team on this one.

Zaffar Essak, M.D.
Vancouver, British Columbia
Canada

```
<table>
<thead>
<tr>
<th>Block #1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( periodaway Appointments 850512z)</td>
</tr>
<tr>
<td>\ dayinterval ( n--) Adjusts date by n days.</td>
</tr>
<tr>
<td>0 VALUE period#</td>
</tr>
<tr>
<td>: adjustperiod# ( --) period# 5 /MOD dayinterval \ floored to negative TO period# ; \ infinity ideal !!</td>
</tr>
<tr>
<td>: periodaway ( n--) AT period# ++ period# 0 4 WITHIN 0= IF adjustperiod# THEN ;</td>
</tr>
<tr>
<td>EXIT</td>
</tr>
</tbody>
</table>
```

Volume VII, No. 2
FORTH Dimensions
9
A Case of Brackets

Editor,

Henry Laxen's article in your March/April issue (VI/6) was very interesting for his discussion of using J as a callable compiler. Looking into just how this works in the Perry/Laxen F83 system, described in C.H. Ting's book Inside F83, I find a difficulty. When reading from the terminal, the J in F83 compiles only one line of text. Colon definitions in F83 can have multiple lines because compilation is continued by repeated J calls in QUIT. No return into the definition of colon is made, but none is necessary. Yet, such a return into the defining word is just what is wanted from J as it is used in Henry's CASE, for example.

This restriction seems inherent in trying to use J as a callable compiler instead of some other word. The J is used not just to begin compilation, but is used with J to enclose an interpreted expression embedded in the source text of a compiled definition. Such embedded expressions really require a single interpreter vectored back and forth from compiling to interpreting by brackets, rather than calling and returning. This interpreter, actually the word INTERPRET, already does everything Henry's J does, if called with state set to compile.

Like Henry's J, though, INTERPRET only compiles one line from the terminal. Rather than change J, therefore, I would want to change INTERPRET. The repeated loop over QUERY...INTERPRET in QUIT should be moved into INTERPRET so that a single call will process all the lines. The loop must repeat only when input comes from the terminal instead of disk. The null word executed at the end of a line should return into this loop, while exiting from INTERPRET is done by ; at the end of a definition.

One way to do this is by having INTERPRET call a component word (INTERPRET) to process each line. Null can then be simply EXIT, as in fig-FORTH; it exits (INTERPRET) at the end of a line. Semicolon can be defined simply as an exit one more level, to the caller of INTERPRET, which will be a defining word. The other work now done by semicolon — compiling EXIT, etc. to finish a colon definition — can actually be done in colon. Semicolon then can be used to terminate any compiler construct, not just those using colon. Henry's version of semicolon lacks this generality.

Henry described the compiling J as an outgrowth of Forth-83, which eliminates state-smart words (I still like state-smart words, but they really are irrelevant here). Besides, Forth-83 does not define J as processing text; like Forth-79, it defines both brackets vaguely, as setting state so that text is "subsequently compiled" or "subsequently interpreted." Code using Henry's J therefore may not work on some quite standard systems.

On another subject, credit should be given to D. Val Schorre for his paper on "Adding Modules to Forth" in the proceedings of the 1980 FORML Conference. He presented the same technique described by Carol Pruitt in your March/April issue under the title "Local Definitions."

Sincerely yours,
George Lyons
Jersey City, New Jersey
THE FORTH SOURCE™

MVP-FORTH

Stable - Transportable - Public Domain - Tools

You need two primary features in a software development package a stable operating system and the ability to move programs easily and quickly to a variety of computers. MVP-FORTH gives you both these features and many more. This public domain product includes an editor, FORTH assembler, tools, utilities and the vocabulary for the best selling book "Starting FORTH". The Programmer’s Kit provides a complete FORTH for a variety of computers. Other MVP-FORTH products will simplify the development of your applications.

MVP Books - A Series

- Vol. 1, All about FORTH by Haydon. MVP-FORTH glossary with cross references to fig-FORTH, Starting FORTH, and FORTH-79 Standard. 2nd Ed. $25
- Vol. 3, Floating Point Glossary by Springer $10
- Vol. 4, Expert System with source code by Park $15
- Vol. 5, File Management System with interrupt security by Moreton $25
- Vol. 6, Expert Tutorial for Volume 4 by M. & Derick $15

MVP-FORTH Software - A Transportable FORTH

- MVP-FORTH Programmer’s Kit including disk, documentation, volumes 1 & 2 of MVP-FORTH Series (All About FORTH), 2nd Ed. & Assembly Source Code, MVP-FORTH for IBM-PC, CP/M, CPM®86, $120.00.
- APPLE, STC MP, IBM PCXT/XT, PC/MIC/SOS, Osborne.
- Kaypro, Microdecisions, DEC Rainbow, TI-PC, NEC. $125.00

MVP-FORTH Enhancement Package for IBM-PCXT/XT Programmer’s Kit. Includes full screen editor, MS-DOS file interface, disk display and assembler operators. $110

MVP-FORTH Floating Point Math for IBM PC/XT/AT with 8087 or Apple with Applesoft $85

MVP-FORTH Graphics Extension for IBM PCXT/XT or Apple $65

MVP-FORTH Programming Aids for CP/M, IBM or Apple Programmer’s Kit. Extremely useful for decompiling, callfinding and debugging $200

MVP-FORTH Cross Compiler for CP/M Programmer’s Kit. Generates headerless code for ROM or target CPU $300

MVP-FORTH Meta Compiler for CP/M Programmer’s Kit. Use for applications on CP/M based computer. Includes public domain source $150

MVP-FORTH PADS (Professional Application Development System) for IBM PCXT/XT or PC or Apple II, I+ or Ie. An integrated system for customizing your FORTH programs and applications. The editor includes a logical string search and is a word processor specially designed for fast development. PADS has almost tripled the compiled speed of most FORTH’s and provides fast debugging techniques. Minimum size target systems are easy with or without headers. Virtual overlays can be compiled in object code. PADS is a true professional development system. Specify Computer. $500.00

MVP-FORTH MS-DOS file interface for IBM PC PADS $80

MVP-FORTH Floating Point & Matrix Math see above $85

MVP-FORTH Graphics Extension see above $65

MVP-FORTH Expert-2 System for learning and developing knowledge based programs. Both IF-THEN procedures and analytical subroutines are available. Source code is provided. Specify Apple, CP/M or CP/M. Includes MVP Books, Vol. 4 & 6. $100

MVP-FORTH WRITE, A Word Processor for the IBM PCXT/XT with 256K MVP-FORTH compatible kernel with Files, Edit and Print systems. Includes Disk and Calculator systems and ability to compile additional FORTH words. $150

MVP-FORTH Fast Floating Point includes 9511 math chip on board with disk, documentation and enhanced virtual FORTH for Apple II, I+ and Ie. $450

ORDERING INFORMATION: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), Visa, MasterCard, American Express, COD’s $5 extra. Minimum order $150. No cash, COD or billing or unpaid PO’s. California residents add sales tax. Shipping costs in US included in price. Foreign orders, pay in US funds on US bank, include for handling and shipping.

MVP-FORTH Disks

- APPLE by MM, 83 $100
- Macintosh by MM, 83 $125
- ATARI® by alfFORTH $60
- CP/M by MM, 83 $100
- HP-85 by Lange $90
- HP-75 by Cassady $150
- IBM-PC by MM, 83 $125
- Z80 by MM, 83 $100


- APPLE by MM, F.G & 83 $180
- ATARI® by PNS, F.G & X $90
- CP/M by MM, F & 83 $140
- TRS-80 or IMS by MMS F, X & 79 $130
- C64 by PS MVP, F.G & X $86

Key to vendors:

- HW Harig Wild Software LM Laboratory Microsystems AM Armstrong Micro MMS Mike Micromachines Services MPS Print Noise Studio
- PS Perfect
- W.F. &83
- L Derick
- K Moreton

Forth Manuals, Guides & Documents

- Thinking FORTH by Leo Brodie, author of best selling "Starting FORTH" $16
- ALL ABOUT FORTH by Haydon. MVP Glossary $25
- FORTH Encyclopedia by Derich & Baker $25
- FYS FORTH from the Netherlands
- User Manual $25
- Source Listing $25
- FORTH Tools and Applic. by Fehrlich $19
- The Complete FORTH by Winfield $16
- Learning FORTH by Armstrong $17
- Understanding FORTH by Reymann $3
- FORTH Fundamentals, Vol. I by McCabe $30
- Vol. II Glossary $14
- Mastering FORTH by Anderson & Tracy $18
- Beginning FORTH by Chirlian $17
- FORTH Encycl. Pocket Guide $7
- Rick’s FORTH by Huang, A college level text $25
- FORTH Programming by Scanlon $17

- Installation Manual for fig-FORTH $15
- Source Listings of fig-FORTH, Specify CPU $15

- C64 with EXPERT-2 by PS $99
- Extensions for MS-DOS VMS.BN, 280, 8086 $30
- Software Floating Point $100
- IBM-PC or 8086 $60
- 9511 Support (280 & 8086) $100
- Color Graphics (IBM-PC) $100
- Data Base Management $200
- Friend FORTH by Brodie. Best instructional manual available. (soft cover) $20
- 68000 fig-FORTH with assembler $25
- FORML Proceedings
- 1980 $35
- 1981 Vol 1 $25
- 1983 $25
- 1984 $25
- 1985 Rochester Proceedings $35
- 1983 $25
- 1984 Data
- Bibliography of FORTH $17
- The Journal of FORTH Application & Research
- Vol. 1/1 Vol. 1/2 $20
- Vol. 2/1 Vol. 2/2 $20
- Vol. 2/3 $17
- META FORTH by Cassady $30
- Threaded Interpretive Languages $25
- Systems Guide to fig-FORTH by Ting $25
- Inside F83 Manual by Ting $25
- FORTH Notebook by Ting $25
- Invitation to FORTH $20
- PDP-11 User Manual $20
- 6502 User’s Manual by Rockwell Intl. $10
- FORTH-83 Standard $15
- FORTH-79 Standard $15

- By Air: $5 for each item under $25, $10 for each item between $25 and $49 and $20 for each item over $100. All prices and products subject to change or withdrawal without notice. Single system and/or single user license agreement required on some products.

MOUNTAIN VIEW PRESS, INC.
PO BOX 4556 MOUNTAIN VIEW, CA 94040 (415) 961-4103

NEW BOOKS - "FORTH GUIDE" by Haydon. An exploration of the intricacies of MVP-FORTH $20.00
"FORTH Applications" by Weber $13.00
FORTH Dimensions

William F. Ragsdale
Hayward, California

“Ask the Doctor” is Forth Dimensions’ health maintenance organization devoted to helping you understand and use Forth. Questions of a problem-solving nature, on locating references, or just regarding contemporary techniques are most appropriate. When needed, your good doctor will call in specialists. Published letters will receive a preprint of the column as a direct reply.

This month your faithful practitioner deviates from his usual format. In the last three issues, we examined aids to learning Forth. Interest shifts this month to news and commentary on the latest events impacting the Forth community. The first stop on our morning rounds occurs at the Forth nursery, otherwise known as the Chip Hatching Department.

Novix NC4000 Processor

The hottest topic of interest in the Forth world must be the Novix NC4000P Processor, Charles Moore’s embodiment of Forth in silicon. This sizzler of a processor gives about 10,000,000 Forth instructions per second (10 mips). Up to four Forth instructions (like DUP 1 + ) can execute at each 125 ns clock cycle. A call takes one clock cycle and a return takes none! This rewriting the book on software, obviating interest in linear code and macros.

To bring this topic into focus, we peek behind the scenes to witness a bit of the genesis of the Forth processor. The courtship began in October of 1980. The moment of conception was on January 19, 1981.

First things first. Forth, Inc. had decided to expand its Board of Directors. Bill and Anne Ragsdale entertained Mr. John Peers at dinner in Palo Alto in October 1980, with the ultimate result of Mr. Peers being invited to join the Board. John Peers was the founder of Logical Machines, which had an innovative, extensible language called Adam. He was involved in robotics and in the quest for aware machines.

The spark that set the Forth processor development into motion came just two months later. We owe a debt of gratitude to Christine Colburn (president of Creative Solutions), who gave a $1000 birthday present to husband Don. Don had been examining computer architectures and had developed a passion to see the advantages of Forth directly applied in silicon.

Don thus funded a one-day, project-organizing session with Charles Moore, Bill Ragsdale and a chip design consultant. This was the spark needed for Chuck to confirm that others were seriously in support of his idea for a Forth processor.

John Peers saw the beauty of approaching Forth from several levels. He founded Technology Industries in March 1981, with the encouragement of Chuck Moore and Elizabeth Rather of Forth, Inc. As the story unfolds, Forth Inc. merged into Technology Industries in August 1981. The charter of Technology Industries was to be an umbrella for three divisions of Forth: hardware, software and applications. Unfortunately, the full scope of this plan was not reached and the merger was rescinded in October 1982.

The chip development still continued, funded with the limited resources of Technology Industries. Charles Moore demonstrated a color simulation of the processor in March 1983. A funding partner, Sysorex International, became interested in July, and by March of 1984 the Novix partnership commenced operation. John Golden transferred from Technology Industries as general manager, and continues in the capacity today.

Silicon generation was almost antclimactic. It took four years and a million dollars to get to the detail design stage. It took just seven months and seven hundred thousand dollars to realize the dream of an operational Forth processor in silicon.

EDN magazine had the privilege of introducing the NS4000 in their March 1985 cover article. John Golden, guiding light of Novix, says, “The EDN article put us on the map. Three hundred inquiries were developed. We found that Forth has friends throughout industry just waiting to show their management the opportunities.”

The first production run of development boards is nearly sold out. Mr. Golden says shipments will commence in the latter part of July with full documentation and developmental software (by Gregg Bailey). Production units were demonstrated at the Silicon Valley FIG Chapter meeting and at the Rochester Forth Conference. Purchasers’ names read like a who’s who of technology. There are automobile manufacturers, a computer media firm and a computer company that is into fruit. The processor is even being considered as a controller co-processor for a recent thirty-two-bit family. When queried on their motive, a spokesman replied, “We see this as a way to get a leg up on the rest of the world.”

Forth and Voice Mail

FIG’s own John Cassady is getting no end of attention in the technical and popular press. He was recently “discovered” by Infoworld columnist John Dvorak. Mr. Dvorak, you will recall, is fondly remembered for his quote that he has never seen a credible application done in Forth.

He has met his match, as John Cassady recently introduced the DynaCom Voice Mailbox System. It is programmed in Forth and is a honey of a product. Dvorak devoted an entire column in the San Francisco Chronicle to Cassady’s product and then gave it another hit in his Infoworld column. You may try the system and leave remarks for John Dvorak at his voice mailbox by calling (415) 763-2002. When the system answers, tone in 454.

The system supports twenty-five users, each with his own addressable “box.” Callers leave messages which may be interactively reviewed by the boxholder. But each boxholder may grant up to thirty of his friends their own guest mail boxes for replies! Imagine 750 people with selective use of digitally controlled voice communica-
The box is an IBM PC look-alike processor board with a hard disk, speech digitizer and Touch-Tone signalling. From the outside, it looks like just another industrial box. But on the inside, the VMS has a heart of pure Forth. John Cassidy was previously best known as the implementor of fig-FORTH on the 8080. He published the 8080 assembler in widest use and is the author of the book *Metaforth*.

The DynaCom system is to be demonstrated at the Silicon Valley FIG Chapter meeting on July 28. Too bad our publication date means you will receive this issue of *Forth Dimensions* the following week! FIG even has its own voice mail box. For information on the next Silicon Valley Chapter meeting call (415) 763-2002 and tone in 44414. For a demonstration, key in 564.

**The Rochester Conference**

The Fifth Rochester Forth Conference continues to receive rave reviews from attendees. During early June, over 180 people were hosted by Thea Martin and Larry Forsley. Participants traveled from all over the U.S., Canada, the Netherlands, Germany and the U.K. The program featured ten papers on Forth computers, with three from Novix. Novix' Bob Murphy packed the house with the best rundown yet on the internals of the NC4000.

Major firms are using Forth and appear to be making a significant commitment in their project work. High-visibility attendees at Rochester included Standard Oil, Bell Canada and the GM Delco Division.

The working group on standards, presided over by Mahlon Kelly (no relation to Guy Kelly, Forth Standards Team chairman), got into the topic of transitions from Forth-79 to Forth-83. A significant attitude was expressed about the burden and value of shifting to Forth-83 from an earlier dialect. Some opinion was offered that system and application changes weren't justified by the small perceived benefit.

**Switch or Fight?**

Forth-79's five-year lead has finally produced a comfortable user base. Working group members asked "Why update to Forth-83? Is the effort worth the improvement? What will be the common model of a full implementation?"

fig-FORTH was the common model prior to Forth-79. MVP-Forth carried the FIG model forward, and has become the *de facto* model for Forth-79. F83, the Laxen-Perry model, is only now coming into its own as the leading contender for the Forth-83 Model sweepstakes. Your faithful servant offers a few opinions. (Who said, "Anybody can have the facts, but it takes real character to have opinions"?)

First, it takes four years for a language standard to become accepted and popular. Do you remember the hue and cry about Forth-79? It was supposed to be the worst disease since the plague. Refer to your back issues of *Forth Dimensions* to check it out. We are two-and-a-half years into Forth-83. It will catch full swing in another eighteen months. Just wait!

Second, the Forth Interest Group must abandon fig-FORTH. The system design and listings are four years behind the times, unsupported, impossible for the novice to self-install, and at cross purposes to the learning process. (The only viable replacement for the revenue that would be lost is a disk-based applications library, but this has its own set of complications.)

Third, some complaining occurred at Rochester that the shift to Forth-83 was being pressured upon those complaining. This is a touchy topic. Some standards are required. Building code standards, for example, are enforced by local governments. But Forth usage remains voluntary. The Forth Standards Team went to special lengths to make this clear. Read the copyright notice on Forth-83, to wit:

"The existence of a Forth Standard does not in any respect preclude anyone ... from implementing, marketing, purchasing or using products, processes, or procedures not conforming to the Standard."

The complainers probably are reacting to their perception of the transition around them, and divert attention to the ogre of enforced change when trying to cling to the past. My understand-
FOR TRS-80 MODELS 1, 3, 4, 4P IBM PC/XT, AT&T 6300, ETC.

THE COMPLETE FORTH GETS A MAJOR UPDATE: MMSFORTH V2.4

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

IT'S BETTER THAN EVER.

The total software environment for IBM PC/XT, TRS-80 Model 1, 3, 4 and close friends.
- Personal License (required):
  MMSFORTH V2.4 System Disk ........... $179.95
  (TRS-80 Model 1 requires overlay, IBM, 40-track drive.)
- Personal License (additional modules):
  FORTHCOM communications module .... $49.95
  UTILITIES ................................ 49.95
  GAMES .................................. 29.95
  EXPERT-2 expert system ............. 69.95
  DATABASEL ................................ 59.95
  DATABASEL-PLUS (PC only, IBM only) .. 99.95
  FORTHWRITE word processor ......... 99.95
- Corporate Site License
  Extensions .................. from $1,000
  Bulk Distribution ... from $500/50 units.
- Some recommended FORTH books:
  STARTING FORTH (programming) .... $19.95
  THINKING FORTH (technical) ....... 15.95
  BEGINNING FORTH (in MMSFORTH) ... 16.95
  Shopping/Handling & tax extra. No returns on software.
  Ask your dealer to show you the world of MMSFORTH, or request our free brochure.

FOR MORE INFORMATION

For information on the NC4000P processor, contact John Golden at Novix, 10590 N. Tantau Ave., Cupertino, California 95014; (408) 996-9363.

Data on the DynaCom VMS VoiceStar, contact John Cassady at 339-15th Street, Oakland, California 94612; (415) 763-6636.

MVP-Forth is distributed by Mountain View Press, Post Office Box 4656, Mountain View, California 94040; (415) 961-4103.

F83, the Laxen & Perry Model, is available on diskette at $25 for IBM PC, CP/M 8080 and 68000, from No Visible Support Software, P.O. Box 1344, 2000 Center Street, Berkeley, California 94704.

The Rochester FORTH Conference is held in June, organized by the Institute for Applied FORTH Research, Inc., 70 Elmwood Avenue, Rochester, New York, 14611; (716) 235-0168.

About the Author

Bill Ragsdale has been using Forth since 1977 for personal and business projects. He is married to Anne, and they have children Mary, three, and Michael, one. The family recently completed a Caribbean cruise with 250 magicians. The ship offered computer classes on twelve IBM PCs, which Bill chose to skip. However, he was pleased to find MVP-Forth in the ship's software library.
Menus in Forth

Frans Van Duinen
Toronto, Ontario

Programs that display menus to show the various options available can be made very easy to use, especially for the novice user. This is particularly significant where the user gets minimal training and documentation (if any at all), such as with public-domain software.

The menu program shows all options available at any specific point, i.e., now that you've selected option A, these are the available sub-options... Menus also tend to lead the programmer along in how he organizes his thinking: Think about the most important items first and ignore the details, then go down to a lower level and concentrate on the aspects of only one item, etc.

Menus limit the number of options presented at any one time. With a single item per line, and after allowing for headings and such, you typically put on the screen no more than eight to fifteen items (and eight is better), with an entire line to describe each item—more if needed. That, too, makes for easier use, i.e., better programs.

Even the expert user, who does not need all this detail, can be accommodated. Simply implement a key-ahead facility. If the user knows he wants item A on this menu, then 1 on the next sub-menu and then sub-sub-item C, let him key A1C. Then simply skip directly from the A-level menu down to whatever it is that A1C means or does.

Building Menus

Three things are involved in menus: (1) Display the text of the menu, (2) accept the menu option selected and (3) execute the code corresponding to that selection.

Those are must-haves. For good measure, we'll also throw in (4) a Help facility and (5) the ability to back out of a menu without selecting any of its items.

\ Menu Utility words - version 1
\ Menu Utility words 2 - Version 1

Programs that display menus to show the various options available can be made very easy to use, especially for the novice user. This is particularly significant where the user gets minimal training and documentation (if any at all), such as with public-domain software.

The menu program shows all options available at any specific point, i.e., now that you've selected option A, these are the available sub-options... Menus also tend to lead the programmer along in how he organizes his thinking: Think about the most important items first and ignore the details, then go down to a lower level and concentrate on the aspects of only one item, etc.

Menus limit the number of options presented at any one time. With a single item per line, and after allowing for headings and such, you typically put on the screen no more than eight to fifteen items (and eight is better), with an entire line to describe each item—more if needed. That, too, makes for easier use, i.e., better programs.

Even the expert user, who does not need all this detail, can be accommodated. Simply implement a key-ahead facility. If the user knows he wants item A on this menu, then 1 on the next sub-menu and then sub-sub-item C, let him key A1C. Then simply skip directly from the A-level menu down to whatever it is that A1C means or does.

Building Menus

Three things are involved in menus: (1) Display the text of the menu, (2) accept the menu option selected and (3) execute the code corresponding to that selection.

Those are must-haves. For good measure, we'll also throw in (4) a Help facility and (5) the ability to back out of a menu without selecting any of its items.
I've included three versions of menu words. Each successive version is more elaborate and somewhat more difficult to understand. This progression from simple to reasonably complete is how I developed the various ideas. (This code is available on various RCP/M bulletin boards. The file MXSET.BLK is a program written in F83 to configure the Epson printer. Note that it is not a complete program, but rather a "Study in Menus." There is also EDITOR.BLK, a WordStar-compatible screen editor that I adapted to F83 from the original version published by Laxen in Dr. Dobb's Journal, September 1981.)

Figure one shows the first approach. It is concerned primarily with display.

\ Menu support words - Version 2: A" IANS  
PAD DUP C@ 1+ 2DUP SWAP C1 + C1 ; \ Update response list 

: (.A")  \ Display menu item 
R> COUNT 2DUP + >R \ Step past text 
1- SWAP DUP C@ DUP EMIT \ Display lead char 
IANS \ & save as allowed response 
-" - " 1+ SWAP TYPE OR ; \ Display rest 

: .A"  \ Compile menu item 
COMPILE (.A") ; 

.DIFER (HELP) \ Help word for ? response 

: UKEY  \ Set help reference 
KEY DUP ASCII a ASCII z BETWEEN IF DUP AND ENDF ; \ Get upper case key 

VARIABLE HELP# \ Reference no for Help 

\ Menu support words - Version 2: NEW-MENU  
VARIABLE (M-STK) OE ALLOT \ Stack of Menu RP@ & SP@ 
\ The Menu Stack (M-STK) maintains exit RP@ & SP@ addr for 6 
\ levels of menus. This is used by M-EXIT to abort current menu) 
STK-MENU ( S -> ) 
R> RP@ SP@ (M-STK) DUP @ 2+ + 2! \ RP@ for calling wd 
4 (M-STK) +! >R ; 
USTK-MENU ( S -> RP@ SP@ ) 
-4 (M-STK) +! (M-STK) DUP @ 2+ + 2@ ; 
NEW-SEL (S help# -> ) 
0 PAD ! \ Set new selection, not return addr 
HELP# ! ; \ Set help reference 

: NEW-MENU (S help# -> ) 
NEW-SEL \ Clear screen & valid responses 
STK-MENU ; \ Where to return to on Esc 

\ Menu - M-EXIT - Version 2 

: END-MENU 
USTK-MENU 2DROP ; \ End of menu - clear stack item 

: M-EXIT  \ USTK-MENU \ Over RP@ 1 ABORT" Bug in unstacking menus" 
\< ABORT" Menu unstack error" 
\ SP! RP! ; 
(S exit current menu & return to calling level menu) 
\ Note - Should work but does not.
ing the menu. The word \# compile
the text following it, and the run-time
word (#"). The word (#") displays that
text. It also displays the menu selection
number for that item.

\textbf{M} is the variable for the menu item
number. \textbf{.M} and (\textbf{.M") are used for menu
item zero, if at all. \textbf{M?} accepts a
single digit, zero up to the maximum in
\textbf{M}. The word \textbf{.M} is used to display
an item without a number. It's like
menu item zero, if at all.

\textbf{A Better Version}

Instead of using a counter, I used
(.#II) and
codes in
PAD.

Valid responses are those in
exit and an escape mechanism.

\textbf{A C}

\begin{verbatim}
\&[DEF \# define menu - Version 2]
BEGIN \KEY 0 \ " Get key & set index
PAD COUNT OVER + SWAP \ Scan allowed responses
DO OVER I OB = \ " char index flag
\# LEAVE 1+ LOOP \ -> char index
PAD OB 1- OVER < \ " index past end? - yes
WHILE DROP \ " Drop index, check specials
|DUP 3 = ABOut" Break" | Terminate on "C
DUP ASCII ? = IF (HELP) ELSE
1b = IF M-EXIT ENDIF \ exit calling menu
BEEP ENDIF \ " No BEEP after Help
REPEAT SWAP EMIT CR ; \ " index

\textbf{Figure Three}
\textbf{Enhanced Menu Words}

\textbf{Examples Using Version Two Menu Words}
\end{verbatim}

\textbf{Figure Four}
\textbf{Examples Using Version Two Menu Words}
(positional) `CASE` statement. Every item in the menu had better always be in the same spot. The `ELSE` clause in `MENU-SELX` enters dummy responses in `PAD` to ensure its length does not change. If Exit is the fifteenth word in the `CASE`, then `MENU-SEL` had better always return a 15 when Exit is selected!

The escape mechanism, too, was troublesome. As a general facility, it should clean up the stack as well as exit. Since we are talking about returning to a calling routine/menu, a simple solution is to keep track of the position on the stack before calling the lower-level routine. Should that lower-level routine abort (error exit, etc.), simply restore the stack pointer to that known value.

I generalized this to cover both stacks, and set up a little menu stack. On this is kept the value for the two regular stack pointers. This is to allow nesting of menus and returning one level at a time. I never did get it to work. It should — the concept is sound — but it is fragile code. Messing with the return stack must be done just right, or it will crash the system.

The Ultimate Version

Fortunately, by this time I had solved the problem. Version three is a little more complicated, but much more readable. It actually works.

In version three, the menu items and the corresponding execution word are combined. There is no `CASE` statement. Have a look at the sample in figure six first. `.AX" is now called `.AX" and is followed by the name of a Forth word first and then by the text. `.AX" now not only puts the response code in `PAD` but also the address of the word that is to be executed should that code be selected. The first byte in `PAD` still counts the entries, which are now three bytes long.

We have removed the positional dependence of the `CASE` statement. `MENU-SELX` gets the answer, validates it and executes the word whose address is stored immediately following. We have also done away with the menu stack. If the option is Esc, do nothing; don't execute any word, merely return to the calling word/menu.

The Help concept is shown here in its more rudimentary form, a word called from `MENU-SELX` when ? is pressed. It uses the variable `HELP#` as set in each menu by `NEW-MENU` or `NEW-SEL`. The number in `HELP#` would indicate what help information is relevant for that specific menu.

The one limitation I am aware of (and have fixed since) is that after a Help message is displayed, the screen should be cleaned up and the menu restored. I've done this by passing the address of the word that displays the menu to `MENU-SELX`, which will re-execute it after Help has been invoked.

There it is. I hope it is of use to others as well. Feel free to use this code for any purpose that is legal and moral.

Acknowledgments

The Forth dialect used is Forth-83 as implemented by Henry Laxen and

```
.
Menu support words - Version 3: .AX" !ANSX FVD28Apr84
: !ANSX (S addr char -> ) \ Save allowed response
  PAD COUNT 2DUP 3 + SWAP 1- CI \ Update count byte
  + DUP 1+ -ROT ! ! ;
(P !ANSX stores menu char & rtn addr in pad, updates offset)
  (S -> ) \ Display menu item
  R> DUP @ SWAP 2+ \ -> word, text addr
  COUNT 2DUP + >R \ Step return addr past text
  SWAP DUP 08 \ Get lead char
  DUP EMT -ROT "," - " \ -> addr char count addr
  1+ SWAP 1- TYPE OR \ Display rest -> word addr, char
  !ANSX ; \ & save as allowed responses
(P Disp! menu item & store its select, char & rtn addr in PAD)
  (S -> ) \ Compile menu word & text item
  .AX" \ leaves (.AX"), word, text
  [COMPIL] " ",; \ IMMEDIATE
.
Define menus Version 3 : MENU-SELX " ? " (S -> )
BEGIN UKEY 0 \ Get key & set index
  PAD COUNT OVER + SWAP \ Scan allowed responses
  DO OVER I 0 = \ -> char index flag
  \-LEAVE 3 + 3 +LOOP \ -> char index
  PAD 08 1- OVER < \ Index past end? - yes
  WHILE DROP \ Drop index, check specials
  DIP 3 = ABORT" Break" \ Terminates on "C
  DUP ASCII ? = IF (HELP) ELSE
  1B = IF EXIT ENDFIP \ exit MENU-SELX
  BEEP ENDFIP \ No BEEP after Help
  REPEAT SWAP EMT OR \ -> index
  PAD 2+ @ EXECUTE ; \ Get selected word addr
.
Define menus - HELP : HELP OR OR HELP# @ ". See help no: ", OR 
' HELP IS (HELP)

Figure Five
Execution Version of Menu Words
```
Michael Perry. It is readily adaptable to fig-FORTH, etc., though I don’t know why anyone with an 8080/Z80 system would use anything other than Laxen and Perry’s F83.

For those who are not aware, these two gentlemen implemented a meta-compiled version of Forth per the 83-Standard and put it in the public domain. It includes all kinds of bells and whistles, is one of the best Forth systems around, and the price is hard to beat. Most RCP/M computer bulletin boards around North America carry it as library SIGM154A.LBR and SIGM154B.LBR. Thanks Henry! Thanks, Michael!

---

\( \text{\% Menus - using version 3} \) FVD28Apr84

\[ \text{\% Menus - Line spacing, Page size} \] FVD28Apr84

\[ \text{\% Menu - using version 3} \] FVD28Apr84

\[ \text{\% Menus - Line spacing, Page size} \] FVD28Apr84

\[ \text{\% Menu - using version 3} \] FVD28Apr84

\[ \text{\% Menus - Line spacing, Page size} \] FVD28Apr84

---

\. Menus - using version 3
: MX8GX MX80 GRAFTRAX ; mx10GX MX100 GRAFTRAX ;
: MX8GX MX80 NOGRAFTRAX ; mx10GX MX100 NOGRAFTRAX ;
: MX80/100
2 NEW-SEL TITLE "Set Printer Type" CR CR
  AX" MX8NGX ASet MX80"
  AX" MX8GX BSet MX80 with Graftrax"
  AX" MX10NGX CSet MX100"
  AX" MX10GX DSet MX100 with Graftrax"
MENU-SELX ;
\Note that \( \text{\%AX} \) used will not skip leading blanks, hence only
\one blank before menu text (applies to \( \text{\%AX} \) and \( \text{\%AXw} \))

\. Menus - Line spacing, Page size
: SPACING
3 NEW-SEL TITLE "Set Line Spacing" CR CR
  AX" LS-6 ASet 6 lines per inch"
  AX" LS-8 BSet 8 lines per inch"
  AX" LS-72 CSet line spacing in 1/72 inch increments" (GTR) @ IF
  AX" LS-10 DSet 10.3 lines per inch"
  AX" LS-216 DSet line spacing in 1/216 inch increments" AX" LS-216-1 ESet 1 line spacing in 1/216 inch increments" ENDIF MENU-SELX ;
: PAGE
4 NEW-SEL TITLE "Set Page Size" CR CR
  AX" PG-LN ASet page length in lines"
  AX" PG-IN BSet page length in inches"
  AX" PG-WDTH CSet page width in characters" MENU-SELX ;

Figure Six
Sample of Execution Version of Menu Words

---

Multiuser/Multitasking
for 8080, Z80, 8086

Industrial Strength
FORTH

TaskFORTH™
The First
Professional Quality
Full Feature FORTH
System at a micro price*

LOADS OF TIME SAVING
PROFESSIONAL FEATURES:

\* Unlimited number of tasks
\* Multiple thread dictionary, superfast compilation
\* Novice Programmer Protection Package™
\* Diagnostic tools, quick and simple debugging
\* Starting FORTH, FORTH-79, FORTH-83 compatible
\* Screen and serial editor, easy program generation
\* Hierarchical file system with data base management

* Starter package $250. Full package $395. Single user and commercial licenses available.

If you are an experienced FORTH programmer, this is the one you have been waiting for! If you are a beginning FORTH programmer, this will get you started right, and quickly too!

Available on 8 inch disk under CP/M 2.2 or greater also various 5½” formats and other operating systems

FULLY WARRANTED, DOCUMENTED AND SUPPORTED

DEALER INQUIRIES INVITED

Visa
Master Charge

Shaw Laboratories, Ltd.
24301 Southland Drive, #216
Hayward, California 94545
(415) 276-5953

---

Volume VII, No. 2

19
October 23, 1985 — November 3, 1985

FORML
Forth Modification Laboratory
presents

EuroFORML Conference
Stettenfels Castle
Heilbronn, Federal Republic of Germany
Followed by

SYSTEMS Trade Fair, Munich
Computers and Communications 9th International
Trade Fair and International User’s Congress
and

Selected sightseeing tours and entertainment in Germany
International technical conference October 25-27, 1985 Stettenfels Castle

Software Metrics — Programs and methods to measure program performance, complexity, structure, programmer productivity, development methods, models, tools, program verification aids, and procedures. Individual participation is encouraged and attendees are requested to submit a conference paper. Conference proceedings will be published.

SYSTEMS Trade Fair October 28 — November 1, 1985 Munich Fair Grounds

Computers and Communications — This is a major international event covering computers and communications. The trade fair is scheduled October 28 through November 1, 1985.

Guest and Tour Program — A complete program will be available for guests not attending the technical conference sessions. Sightseeing escorted tours are planned for the group.

Reservations, authors instructions, itinerary, special group rate — Write to EuroFORML, Forth Interest Group, Post Office Box 8231, San Jose, CA 95155 or telephone the FIG Hotline (408) 277-0668. East and West Coast departures are planned. Advance reservations are required.

West Coast Departure $1995.00
East Coast Departure $1795.00
MEMBERSHIP
IN THE FORTH INTEREST GROUP

107 - MEMBERSHIP in the FORTH INTEREST GROUP & Volume 7 of FORTH DIMENSIONS. No sales tax, handling fee or discount on membership. See the back page of this order form.

The Forth Interest Group is a worldwide non-profit member-supported organization with over 5,000 members and 80 chapters. FIG membership includes a subscription to the bi-monthly publication, FORTH Dimensions. FIG also offers its members publication discounts, group health and life insurance, an on-line data base, a job registry, a large selection of Forth literature, and many other services. Cost is $20.00 per year for USA, Canada & Mexico; all other countries may select surface ($27.00) or air ($33.00) delivery.

The annual membership dues are based on the membership year, which runs from May 1 to April 30.

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

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

HOW TO USE THIS FORM

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

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

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

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

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

FORTH DIMENSIONS BACK VOLUMES
The six issues of the volume year (May - April) bound in a single text.

101 - Volume 1 FORTH Dimensions (1979/80) $15/16/18
102 - Volume 2 FORTH Dimensions (1980/81) $15/16/18
103 - Volume 3 FORTH Dimensions (1981/82) $15/16/18
104 - Volume 4 FORTH Dimensions (1982/83) $15/16/18
105 - Volume 5 FORTH Dimensions (1983/84) $15/16/18
106 - Volume 6 FORTH Dimensions (1984/85) $15/16/18

REFERENCE

305 - FORTH 83 STANDARD $15/16/18
The authoritative description of 83-Standard Forth. For reference, not instruction.

300 - FORTH 79 STANDARD $15/16/18
The authoritative description of 79-Standard Forth. Of historical interest.

316 - BIBLIOGRAPHY OF FORTH REFERENCES
2nd edition, Sept. 1984 $15/16/18

An excellent source of references to articles about Forth throughout microcomputer literature. Over 1300 references.

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

513 - 1802/MARCH 81 $15/16/18
514 - 6502/SEPT 80 $15/16/18
515 - 6800/MAY 79 $15/16/18
516 - 6809/JUNE 80 $15/16/18
517 - 8080/SEPT 79 $15/16/18
518 - 8086/88/MARCH 81 $15/16/18
519 - 9900/MARCH 81 $15/16/18
520 - ALPHA MICRO/SEPT 80 $15/16/18
521 - APPLE II/AUG 81 $15/16/18
522 - ECLIPSE/OCT 82 $15/16/18
523 - IBM-PC/MARCH 84 $15/16/18
BOOKS ABOUT FORTH

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

205 - BEGINNING FORTH ....................................... $17/18/21
Paul Chirlian
Introductory text for A detailed look at each FIG-Forth instruction.

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

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

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

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

233 - FORTH TOOLS ............................................ $19/21/23
Gary Feierbach & Paul Thomas
The standard tools required to create and debug Forth-based applications.

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

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

245 - STARTING FORTH (soft cover) ......................... $20/21/22
Leo Brodie (FORTH, Inc.)
A lively and highly readable introduction with exercises.

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

260 - THINKING FORTH (hard cover) ......................... $23/25/28
Leo Brodie
The sequel to "Starting Forth". An intermediate text on style and form.

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

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

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

310 - FORML PROCEEDINGS 1980 ......................... $25/28/35
Technical papers on the Forth language and extensions.

311 - FORML PROCEEDINGS 1981 (2V) .................... $40/43/45
Nucleus layer, interactive layer, extensible layer, metacomilation, system development, file systems, other languages, other operating systems, applications and abstracts without papers.

312 - FORML PROCEEDINGS 1982 ......................... $25/28/35
Forth machine topics, implementation topics, vectored execution, system development, file systems and languages, applications.

313 - FORML PROCEEDINGS 1983 ......................... $25/28/35
Forth in hardware, Forth implementations, future strategy, programming techniques, arithmetic & floating point, file systems, coding conventions, functional programming, applications.

314 - FORML PROCEEDINGS 1984 ......................... $25/28/35
Expert systems in Forth, using Forth, philosophy, implementing Forth systems, new directions for Forth, interfacing Forth to operating systems, Forth systems techniques, adding local variables to Forth.

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

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

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

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

324 - ROCHESTER 1984 (Forth Applications) ........ $25/28/35
Forth in image analysis, operating systems, Forth chips, functional programming, real-time applications, cross-compilation, multi-tasking, new techniques and working group reports.
THE JOURNAL OF FORTH APPLICATION & RESEARCH
A refereed technical journal published by the Institute for Applied Forth Research, Inc.

401 - JOURNAL OF FORTH RESEARCH V.1 #1 $15/16/18
Robotics.

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

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

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

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

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

REPRINTS

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

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

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

422 - DR. DOBB'S 9/82 .................................. $5/6/7

423 - DR. DOBB'S 9/83 .................................. $5/6/7

424 - DR. DOBB'S 9/84 .................................. $5/6/7

HISTORICAL DOCUMENTS

501 - KITT PEAK PRIMER ................................ $25/27/35
One of the first institutional books on Forth. Of historical interest.

502 - FIG-FORTH INSTALLATION MANUAL .... $15/16/18
Glossary model editor - We recommend you purchase this manual when purchasing the source-code listings.

MISCELLANEOUS

601 - T-SHIRT SIZE .................................. $10/11/12
(smaller, medium, large, extra large)

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

616 - HANDY REFERENCE CARD ................. FREE

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

SELECTED PUBLICATIONS
The following publications are recent additions to the Forth Interest Group Order Form as selected by the FIG Publications Committee.

LEARNING FORTH, A Self-Teaching Guide by Margaret A. Armstrong. Learning FORTH takes you step-by-step through the various stages of programming, using a payroll program as an example.

In addition to developing a basic working knowledge of the language, Learning FORTH teaches good programming style and easy debugging techniques. There's also a section on how to teach children to use Forth.

1984 FORML CONFERENCE PROCEEDINGS
The sixth annual FORML Conference was held November 23-25, 1984 at the Asilomar Conference Center in Monterey, California, USA. Excellent articles in this volume include two papers on Expert Systems which won the acclaim of the participants. Many papers contain code you can use immediately, and some are philosophical papers which present challenges for the future of Forth.

FORTH-83 REFERENCE CARD
The Forth-83 Handy Reference Card is now available free of charge with any order. If you are not placing an order and wish to receive a card, please send a self-addressed, stamped envelope and we will be happy to send you one. Many thanks to Robert Berkey and Dave Kilbridge for compiling, designing and producing the card.

PUBLICATIONS SURVEY
If you would like to suggest any other publication for review by the FIG Publications committee for inclusion in the Forth Interest Group Order Form, please complete the information below and return to FIG.

Title: ___________________________
Author: ________________________
Publisher: ______________________
Comments: __________________________________________________________________
__________________________________________________________________________

Your comments on any of the publications we currently carry are most welcome, please complete information below.

Title: __________________________
Comments: __________________________________________________________________
__________________________________________________________________________

Volume VII, No. 2
**FORTH INTEREST GROUP**  
P.O. BOX 8231  SAN JOSE, CALIFORNIA 95155  408/277-0668

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

**OFFICE USE ONLY**

<table>
<thead>
<tr>
<th>By</th>
<th>Date</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shipped By</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>UPS Wt.</td>
<td>Amnt.</td>
<td></td>
</tr>
<tr>
<td>USPS Wt.</td>
<td>Amnt.</td>
<td></td>
</tr>
<tr>
<td>BO Date</td>
<td>Wt.</td>
<td>Amnt.</td>
</tr>
<tr>
<td>By</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ITEM #</th>
<th>TITLE</th>
<th>AUTHOR</th>
<th>QTY</th>
<th>UNIT PRICE</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>107</td>
<td>MEMBERSHIP</td>
<td></td>
<td></td>
<td></td>
<td>SEE BELOW</td>
</tr>
</tbody>
</table>

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

☐ VISA  ☐ MASTERCARD

Card #  Expir. Date

Signature

---

**PAYMENT MUST ACCOMPANY ALL ORDERS**

**MAIL ORDERS**  
Sent to:  FORTH INTEREST GROUP  
P.O. Box 8231  
San Jose, CA 95155

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

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

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

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

**SALES TAX**  
California deliveries add 6%. San Francisco Bay Area add 7%.

---

SUBTOTAL  
10% MEMBER DISCOUNT  
MEMBER #  
CA. RESIDENTS SALES TAX  
HANDLING FEE  
$2.00  
MEMBERSHIP FEE  
(NEW OR RENEWAL)  
$20/27/33  
TOTAL
Another Subroutine Technique

Donald Simard
Severn, Maryland

David Held's article and subsequent letter (Forth Dimensions V/3 and V/5) concerning code definitions callable by colon and code definitions, provided a very useful technique. After understanding his ideas, I came up with a variation which avoids some of the inherent execution-time overhead when his method is used from a colon definition.

Using Mr. Held's technique from a colon definition required storing the address of the subroutine that he wanted to call into the calling routine each time that word is executed. This overhead can be eliminated. Figure one illustrates what a subroutine definition will look like after being compiled using the defining words presented here.

One can see that the code field points to a JSR to the parameter field address. The code that is then to be executed must be a machine language subroutine and must return accordingly. When the word is called from a colon definition, the inner interpreter will be directed through the code field to the JSR PFA. When the machine language subroutine in the parameter field returns a JMP then NEXT is executed, which is the normal exit for CODE definitions. If the subroutine is to be used from a code definition, first it is necessary to use ' to find the PFA, then JSR to this address.

The execution overhead associated with this technique is just an extra JSR and RTS from a normal colon definition; there is no overhead from code definitions. The extra memory required is the four bytes for the JSR and JMP in the definition.

```
: SUB
   [COMPILE] ASSEMBLER
   CREATE SMUDGE
   LATEST PFA CFA
;

: END-SUB
   ASSEMBLER
   HERE OVER !
   [COMPILE] ASSEMBLER
   2+ JSR, NEXT JMP,
   END-CODE
;

EXAMPLE:
   SUB TEST
   41 # LDA,
   PRINT JSR,
   RTS,
   END-SUB

CALLED FROM COLON:
   : PRINT.
   TEST
;

CALLED FROM CODE:
   CODE PRINT.
   ' TEST JSR,
   NEXT JMP,
   END-CODE
```
FORTH INTEREST GROUP PRESENTS

Forth National Convention
September 20 - 21, 1985
Complete conference program, educational seminars, and commercial exhibits.

Hyatt Rickeys in Palo Alto, California USA

---
euroFORML Conference
October 23, 1985 - November 3, 1985
International Technical conference at Stettenfels Castle
SYSTEMS Trade Fair in Munich
Guest and Tour Program in Germany

Complete group travel arrangements from USA to Germany and return.

---
Forth Modification Laboratory
November 29, 1985 - December 1, 1985
A technical conference for advanced Forth practitioners.

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

Complete information available from the Forth Interest Group.

---
The Mirth Dimension

: PROJECT DIRECTOR EXPECT UPDATE VERSION , WARNING IF SILENT
  TOGGLE WRITE CHECK AND EXPECT COLD HANDSHAKE
  ELSE COMPILE AGAIN AND B SILENT ;

: FORTH PROGRAMMER TRAVERSE COLD SILENT OUTER SPACE AND CREATE
  RANDOM DEFINITIONS ;

: FORTH SOURCE CODE EMIT RANDOM WORD AND REPLACE ERROR UNTIL
  SPACE = MAX LIMIT OR TIME OVER ;

— Scott Heiner & Steve Gledhill
Hacker's LOCKER

Cecil McGregor
Santa Clara, California

Many times, when debugging Forth code, it is convenient to use small "throw away" parts that will not be used in the later, bug-free version. This could include a few lines of output that would be handy to keep on the display, or portions of a dump to compare with something else. It is not always desirable to edit this onto a screen and reload from that for changes, nor is it always convenient to print it as hard copy. These small gems then scroll off the top of the screen. Or perhaps you are looking at the stack and would like to keep it around while you scrounge about analyzing a bug. A few lines of typing and it, too, goes off the top of the screen.

If you have a terminal that permits line lock (such as a TeleVideo 950 and many others) you can simulate a window that will lock the contents of the line until you release it. Scrolling will then ignore these locked lines. While the appearance is rather odd at first, the usefulness is great.

To implement this, you must look in your terminal manual for the codes that will lock a line, unlock a line or screen, move up and down a line. As can be seen in the accompanying code, the word LINE-LOCK locks; and the word SCR-UNLOCK unlocks the entire screen. The constants UP-ARROW and DN-ARROW move the cursor up and down a line, respectively. Each of these is terminal dependent and must be tailored to your terminal.

The word LOCKER gets a keystroke, and moves up or down a line if the up and down arrows are input. UP-ARROW and DN-ARROW allow positioning of the cursor onto the line to be locked. Striking an "L" will cause the current line to be locked; a "U" will unlock the entire screen; and any other key will exit LOCKER.

This screen will compile under figFORTH and Laboratory Microsystems' 83-Forth, and it should be easily transportable to almost any standard Forth. If your terminal has line-lock capabilities, you will like this utility.

```
Screen # 110
0 \ LOCKER - lock/unlock lines on page
1    HEX
2    : LINE-LOCK ( -- ) ( TV950 control to lock a line )
3    1B EMIT  21 EMIT  31 EMIT ;
4    : SCR-UNLOCK ( -- ) ( TV950 control to unlock entire screen )
5    1B EMIT  21 EMIT  32 EMIT ;
6 OR CONSTANT UP-ARROW 16 CONSTANT DN-ARROW
7   : LOCKER ( -- ) ( selectively lock lines on crt )
8   BEGIN KEY
9   DUP UP-ARROW = IF EMIT ELSE ( move up )
10  DUP DN-ARROW = IF EMIT ELSE ( move down )
11  DUP 4C ( L ) = IF DROP LINE-LOCK ELSE ( lock )
12  DUP 55 ( U ) = IF DROP SCR-UNLOCK ELSE ( scr unlock )
13  DROP QUIT ( any other key exits, )
14 THEN THEN THEN THEN
15 AGAIN ; DECIMAL
```
Mass Transit Forth

From June onwards, some bus stops in the resort and retirement town of Weston-super-Mare, Somerset, U.K., will never be at a loss for words. Waiting passengers will hear a quiet beeping and, if they then press a button at the source of the sound, a personable female voice will announce the time of day, what buses are using that stop and when they are due.

When a bus is running off schedule, inductive loop sensors buried in the road interrogate circuitry in the bus cab to identify it. The microprocessor controller in the speech synthesizer scans the sensor, detects the oncoming bus and a voice will announce something like, "The bus now approaching is a No. 10." Meanwhile, the controller figures out when the next one is due.

London-based Triangle Digital Services Ltd., a one-man outfit set up by Peter Rush, developed the prototype hardware for the system. Formerly a product manager with General Instrument Microelectronics Ltd., Rush set up shop to provide a systems service to potential users of voice systems.

He has developed a custom speech chip using a digital logic array. This is mounted on a self-contained Eurocard-sized board. To go with it, there is a Forth-language microcomputer board. Rush says he chose Forth because its high-level-language capability speeds development work.

Using the speech-synthesizer board and the microcomputer board (which serves as its own development system), Rush can quickly meet a customer’s requirements, even tailoring the voice to the applications. For Britain’s National Physical Laboratory, for example, Triangle has developed a handheld keypad with sixteen-digit alphanumeric display. It is used to check readings as they are entered into the laboratory computer. With the synthesizer speaking in a brisk manner with truncated enunciation, it can keep up with the user as he keys in a long numeric string.

The terminal can be used both for entering data and instructions and for receiving prompts from the laboratory computer. For example, it could give the operator a cue about what reading to make next or indicate that a reading is out of range. The terminal communicates with the laboratory computer systems through an RS-232-C interface at 9,600 bits per second or through an IEEE-488 port.

Triangle has now acquired rights to the terminal from NPL and is also negotiating with the Department of Transport for the rights to the passenger advisory service, which clearly has potential applications in road, rail and airplane terminals, just for starters.

For those companies with sufficient expertise to assemble their own systems, Triangle also sells its various board products directly. One is a speech-synthesizer board that can be operated directly from relay contacts, asynchronous links over twisted pairs or parallel binary-coded decimal inputs.

Forth Talk

Triangle has also developed a Forth development board based on a Hitachi 6303 — a complementary-MOS variant of a Motorola part — and running Triangle’s own Forth system. The board is a useful tool for real-time applications, since multi-tasking programs can be written in a high-level language.

Moreover, users can extend the language as they go along, building their own expertise into their systems. Rush has used this feature to develop progressively more sophisticated speech-control algorithms. The result has been speech with very acceptable quality.

The Department of Transport system shows off the microcomputer’s multitasking capability. First, it keeps a calendar that tracks the minute, hour, day and month. It responds to changes in summer and winter schedules and switches to the appropriate timetable.

Second, it scans the inductive loop sensors and decodes the detected signal. Next, it controls the speech-synthesizer chip’s vocabulary and adjusts the volume to compensate for ambient noise levels. “The lady shouts when a lorry goes by,” says Rush. There is also a switched-capacitor-filter cut-off.

—Kevin Smith

Reprinted from Electronics, May 31, 1984. Copyright © 1985, McGraw-Hill, Inc. All rights reserved.
Now You Can Add

ARTIFICIAL INTELLIGENCE

To Your Programs Using a Powerful Combination

“SUPERFORTH 64” +AI

By Elliot Schneider & Jack Park

Heres Your Chance to Profit by being on the Forefront, Write 5th Generation Software

Learn How To:
• Create Intelligent Programs
• Build Expert Systems
• Write Stand Alone License Free Programs

Write Intelligent Programs
• Home Use
• Robotics
• Medical Diagnosis
• Education
• Intelligent CAI
• Scientific Analysis
• Data Acquisition

Extended Math Functions
• Fast ML Floating Point & Integer Math
• Double Precision 2E+38 with Auto. Sci Not.
• n^e^ Logx Loge Sin Cos Tan SQR 1/X...
• Matrix and Multidimensional Lattice Math
• Algebraic Expression Evaluator

Easy Graphics & Sound Words
• Hires Plotting
• Windows
• Split Screen
• Printer/Plotter Ctrl
• Sprite & Animation Editor

Easy Control of all I/O...
• RS232 Functions
• Access all C-64 Peripherals

Utilities
• Interactive Interpreter
• Forth Virtual Memory
• Full Cursor Screen Editor
• Full String Handling
• Trace & Decompiler
• Conditional Macro Assembler

Great Documentation
• Easy to Read 350 pg.
  Manual with Tutorials
• Source Screen Provided
• Meets all MVP Forth-79 Industrial Standards
• Personal User Support

A Total Integrated Package for the Commodore 64

Ordering Information: Check, Money Order (payable to MOUNTAIN VIEW PRESS, INC.), VISA, MasterCard, American Express. COD’s $5.00 extra. No billing or unpaid PO’s. California residents add sales tax. Shipping costs in US included in price. Foreign orders pay in US funds on US bank, include for handling and shipping $10.

*Parsec Research Commodore 64 TM of Commodore

SPECIAL INTRODUCTORY OFFER

only $99.00

203rd Value
Limited Time Offer

Call:
(415) 961-4103

MOUNTAIN VIEW PRESS INC
P.O. Box 4656
Mt. View, CA 94040
Dealer for
PARSEC RESEARCH
Drawer 1776, Fremont, CA 94538
Part Two
Forth Spreadsheet

Craig A. Lindley
Manitou Springs, Colorado

1 \ spreadsheet - loader block
2 \ Spreadsheet Compiling)
3 warning off
4 1 45 +thru
5 \ load the spreadsheet into the forth dictionary
6 \ this will save the entire forth system along with the spreadsheet
7 \ sheet as an executable file named "filename". When the file
8 \ is executed, your spreadsheet will execute automatically.

2 \ spreadsheet - case statement
3 \ "Compilation only": immediate
4 \ compile over compile = compile \branch
5 \ compile \branch
6 \ endof \pairs compile \branch here 0 ,
7 \ swap \resolve 4 ; immediate
8 \ andcase \pairs compile \branch drop
9 \ begin sp\csp @ )
10 \ while \resolve repeat
11 \ csp ! ; immediate
12
13 \ spreadsheet - constant & variable declarations

3 \ constant row_max \ # of spreadsheet rows
4 \ constant col_max \ # of spreadsheet columns
5 \ constant col_name_len \ max length of column name
6 \ constant row_name_len \ max length of row name
7 \ constant col_org \ column origin of data on display
8 \ constant row_org \ row origin of data on display
9 \ constant bytes/cell \ # of bytes per cell
10
11 \ define a 2D array for spreadsheet data structure
12 \ each cell contains 6 bytes
13 \ 2 for formula execution address (if any)
14 \ 4 for double number value storage
15 \ define a string array for holding the row names
16 \ define a string array for holding the column names

4 \ variable mode_flag \ auto calculate flag
5 \ variable order_flag \ calculation order flag
6 \ variable format_flag \ number output format flag
7 \ variable cur_col \ top left display column
8 \ variable cur_row \ top left display row
9 \ variable col_disp \ column displacement from
10 \ variable row_disp \ row displacement from
11 \ variable dict_mark \ beginning of formula area
12 \ variable op_stack \ operator stack for algebraic
13 \ equation compilation
14
15 \ create 2dup swap c, c , * * dup here
16 \ swap erase allot
17 \ does> dup c@ 3 roll * 2 roll + over
18 \ 1+ c@ + + 2* ;
19
20 \ create 2d array depth bytes deep
21 \ array \ ( rows cols depth -- ) compile time
22 \ ( row col -- element addr ) run time
23
24 \ create 10 string array depth characters deep
25 \ array \ ( rows depth -- ) compile time
26 \ ( row -- string addr ) run time
27
28 \ define a 2D array for spreadsheet data structure
29 \ 2 for formula execution address (if any)
30 \ 4 for double number value storage
31
32 \ define a string array for holding the row names
33 \ define a string array for holding the column names

6 \ spreadsheet - array definitions

7 \ spreadsheet - constant & variable declarations
8 \ spreadsheet - high level array definitions

9 \ spreadsheet - case statement
7
0 \ spreadsheet - misc word definitions cl 07/04/84 \ spreadsheet - misc word definitions cl 07/04/84
1 \ IBM_key
2 key ?dup 0=
3 if key 128 + then ;
4
5 : d8in
6 pad 1+ 20 2dup blank expect ( -- d)
7 span @ pad cl! pad number ;
8
9 : in d8in drop ;
10 \ input single number
11 ( -- n)
12 if 10 fd.r
13 else 10 d.r
14 then ;

8
0 \ spreadsheet - misc word definitions cl 07/02/84 \ spreadsheet - display word definitions cl 07/04/84
1 \ position cursor on cad line
2 : pos2 02 22 2dup at 0 blot at ;
3 \ position cursor on cad line
4 : y/n " Are You Sure ?: " display msg
5 key upc ascii Y = ;
6 \ return flag
7
8 : mark_cell
9 \ mark cell on display
10 2dup at ascii \ emit swap 11 \ (row# col# -- )
11 + swap at ascii \ emit ;
12 \ mark cell like "( )"
13
14 : unmark_cell
15 \ unmark cell of display
16 2dup at space swap 11 + swap \ (row# col# -- )
17 at space ;
18 \ remove < > marks from display

9
0 \ spreadsheet - misc word definitions cl 07/03/84 \ spreadsheet - display word definitions cl 07/04/84
1 \ return address of cell pointed
2 cur_row @ row_disp @ + \ at by < > display marker
3 cur_col @ col_disp @ + \ ( -- cell addr )
4 cells ;
5
6 : cal_cell_disp_loc
7 \ calculate location on display
8 col_disp @ 13 + col_org + \ of cell display markers
9 row_disp @ \ row_org + ; \ ( -- col row )
10 loop
11
12 : place_cell_marker
13 \ place cell marker around cell
14 80 0
15 \ do i 2 at ascii - emit loop
16
17 : erase_cell_marker
18 \ erase cell marker around cell
19 80 0
20 \ do i 18 at ascii - emit loop
21
22 : cal_cell_disp_loc unmark_cell ;
2 \ display spreadsheet menu of options on right side of display

4 : dis_menu
5 74 3 at . " Menu:"
6 74 4 at . " Col: 74 5 at . " Again:"
7 74 6 at . " Data: 74 7 at . " Equ:"
8 74 8 at . " Form: 74 9 at . " Data:"
9 74 10 at . " Mode: 74 11 at . " New:"
0 74 12 at . " Order: 74 13 at . " Print:"
1 74 14 at . " Quit: 74 15 at . " Row:;
19
0 \ spreadsheet - cell calculation words cl 06/31/84 \ spreadsheet - cell marker positioning words cl 07/02/84

: calculate
  \ calc formula of cell if it has:
  \ move cell marker right 1 cell
  \ move display column right one
  \ marker in right most cell?
  \ if so is it the last cell (2)
  \ move display column right one
  \ is cell at top display pos.?
  \ the spreadsheet?
  \ if not move up a
  \ cell at top display rove it
  \ if already at top ignor
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: calc_c/r
  \ move display column right one
  \ if correct then one cell
  \ move up one cell
  \ if so are we at the top of
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: row_max 0
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: col_max 0
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: order_flag @
  \ by state of order_flag
  \ is cell at top display pos.?
  \ the spreadsheet?
  \ if not move up a
  \ cell at top display rove it
  \ if already at top ignor
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: order_r/c
  \ move display column right one
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: calc_r/c
  \ move display column right one
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: one ( cell addr -- )
  \ marker in right most cell?
  \ if so is it the last cell (2)
  \ move display column right one
  \ is cell at top display pos.?
  \ the spreadsheet?
  \ if not move up a
  \ cell at top display rove it
  \ if already at top ignor
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

20
0 \ spreadsheet - cell calculation words cl 07/04/84 \ spreadsheet - cell marker positioning words cl 07/02/84

: calc_cells
  \ determine which to calc first:
  \ move cell marker up 1 cell
  \ move cell marker down 1 cell
  \ move cell marker right 1 cell
  \ move cell marker left 1 cell

: order
  \ prompt user for calc order
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: pos1 ." Specify calculation order"
  \ prompt user for calc order
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

21
0 \ spreadsheet - cell marker positioning words cl 07/02/84 \ spreadsheet - cell marker positioning words cl 07/02/84

: leftarrow
  \ move cell marker left 1 cell
  \ move cell marker down 1 cell
  \ move cell marker right 1 cell
  \ move cell marker left 1 cell

: col_dis =
  \ cell marker at left of display?
  \ are we at bottom of display?
  \ are we on last row?
  \ are we at bottom of display?

: cur_col @
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: dis_col_change
  \ if correct then one cell
  \ move up one cell
  \ cell marker positioning words
  \ draw new cell marker

: erase_cell_marker
  \ erase current mark
  \ erase cell mark
  \ erase cell mark
  \ erase cell mark
0 \ spreadsheet - cell marker positioning words

\ spread sheet - algebraic functions

25
2 : first_col 0 cur_col ! \ go to column A immediately
3 \ spread sheet - algebraic functions
4 \ loop used to assign values to
5 \ the alphabetic columns
6 \ spreadsheet - cell marker positioning words
7 \ assign_id col_max 0
8 \ loop to 0 and scroll
9 \ do i col_id loop
10 \ col_id
11 \ col_max 4 - cur_col ! \ go to column W immediately
12 \ assign_id
13 \ B C D E F G H I J K L M N O P
14 \ Q R S T U V W X Y Z
15 \ loop used to assign values to
16 \ alphabetic columns
17 \ cur_col to 0 and scroll do
18 i
19 \ col_id
20 \ loop
21 \ cur_col
22 \ spreadsheet - cell marker positioning words
23 \ 07/04/84
24 \ spreadsheet - algebraic functions
25 \ 07/04/84

26
0 \ spreadsheet - cell marker positioning words
1 \ return operand stack position
2 \ left_cols
3 4 0 do left_arrow loop \ move marker left 4 columns
4 \ at a time
5 4 0 do right_arrow loop \ move marker right 4 columns
6 \ at a time
7 \ operand stack + \ top of operand stack
8 \ op_stack dup @ + ; \ store cfa and precedence
9 \ (-- addr)
10 \ opp 4 op_stack +! opp# 2! ; \ return precedence from top
11 \ (cfa prec --)
12 \ prec? opp# @ ; \ ( -- prec)
13 \ top of operand stack
14 \ 1a begin prec? \ end algebraic compilation
15 \ while op> repeat
16 \ pop remaining operands off stk
17 \ forth ; immediate
18 \ and compile then select forth
19 \ vocabulary again

27
0 \ spreadsheet - algebraic functions
1 \ vocabulary algebra
2 \ algebra also definitions
3 \ create high level definition that performs algebraic
4 \ compilation. See text for details of operation
5 \ column_id function assigns n to id at compile time (n --)
6 \ expects row @ on TOS at run time
7 \ subsequent usage of id fetches double value of cell to stack
8 \ word. Creates col ids A-Z
9 \ does> @ cells 2+ \ expect a @ on the TOS and
10 \ 20 i \ pushes the cell value onto
11 \ the parameter stack
12 \ col_id
13 \ column_id high level defining
14 \ word. creates col ids A-Z
15 \ does> @ cells 2+ \ expect a @ on the TOS and
16 \ 20 i \ pushes the cell value onto
17 \ the parameter stack
18 \ col_id
19 \ column_id high level defining
20 \ word. creates col ids A-Z
21 \ does> @ cells 2+ \ expect a @ on the TOS and
22 \ 20 i \ pushes the cell value onto
23 \ the parameter stack
24 \ do> dup rot + rot rot um* rot + ; \ double multiplication
25 \ d/ swap over /mod \ double division
26 \ um/ mod swap drop r ;
27 \ dmod d/ drop 0 ; \ double modulus

28
0 \ spreadsheet - cell marker positioning words
1 \ return operand stack position
2 \ left_cols
3 4 0 do left_arrow loop \ move marker left 4 columns
4 \ at a time
5 4 0 do right_arrow loop \ move marker right 4 columns
6 \ at a time
7 \ operand stack + \ top of operand stack
8 \ op_stack dup @ + ; \ store cfa and precedence
9 \ (-- addr)
10 \ opp 4 op_stack +! opp# 2! ; \ return precedence from top
11 \ (cfa prec --)
12 \ prec? opp# @ ; \ ( -- prec)
13 \ top of operand stack
14 \ 1a begin prec? \ end algebraic compilation
15 \ while op> repeat
16 \ pop remaining operands off stk
17 \ forth ; immediate
18 \ and compile then select forth
19 \ vocabulary again

29
0 \ spreadsheet - algebraic functions
1 \ spreadsheet - algebraic functions
2 \ 07/04/84
3 \ spreadsheet - algebraic functions
4 \ 06/31/84

30
0 \ spreadsheet - algebraic functions
1 \ spreadsheet - algebraic functions
2 \ 07/04/84
3 \ spreadsheet - algebraic functions
4 \ 07/04/84

FORTH Dimensions
34
Volume VII, No. 2
31  0 \ spreadsheet - algebraic functions    \ spreadsheet - input words
    1   \ input row names
    2   \ input row names
    3   \ display command prompt
    4   \ input command prompt
    5   \ display command prompt
    6   \ display command prompt
    7   \ display command prompt
    8   \ display command prompt
    9   \ display command prompt

32  0 \ spreadsheet - algebraic functions    \ spreadsheet - input words
    1   \ right paren
    2   \ right paren
    3   \ right paren
    4   \ right paren
    5   \ right paren
    6   \ right paren
    7   \ right paren
    8   \ right paren
    9   \ right paren

33  0 \ spreadsheet - input words
    1   \ input row names
    2   \ input row names
    3   \ input row names
    4   \ input row names
    5   \ input row names
    6   \ input row names
    7   \ input row names
    8   \ input row names
    9   \ input row names

34  0 \ spreadsheet - input words
    1   \ input cell_data
    2   \ input cell_data
    3   \ input cell_data
    4   \ input cell_data
    5   \ input cell_data
    6   \ input cell_data
    7   \ input cell_data
    8   \ input cell_data
    9   \ input cell_data

35  0 \ spreadsheet - input words
    1   \ input cell_data
    2   \ input cell_data
    3   \ input cell_data
    4   \ input cell_data
    5   \ input cell_data
    6   \ input cell_data
    7   \ input cell_data
    8   \ input cell_data
    9   \ input cell_data

36  0 \ spreadsheet - input words
    1   \ input data to cell
    2   \ input data to cell
    3   \ input data to cell
    4   \ input data to cell
    5   \ input data to cell
    6   \ input data to cell
    7   \ input data to cell
    8   \ input data to cell
    9   \ input data to cell
0 \ spreadsheet - input words  cl 09/21/84 \ spreadsheet - high level commands  cl 07/04/84
1: input_equ \ input equation into dict
2: pos1." Input Cell Equation" \ prompt for equation
3: pos2." Equation: "
4: tib 127 blank \ clear tib
5: ": formula at " \ preasamble to move to tib
6: tib swap caove \ move it to tib
7: tib 13 + dup 127 expect \ get equation to tib
8: span @ + \ pt at end of input
9: "Ja [ cell_ptr 2+ ] literal 2! ; last @ name) cell_ptr !" \ east data ! \ show data in new format
0: -rot swap caove \ move to tib also
1: span @ 70 + @tib ! \ make forth think it all
2: b!k off \in off \ came from the keys
3: algebra \ select algebra vocabulary
4: interpret \ compile equation into dict
5: forth ; \ back to forth vocabulary

0 \ spreadsheet - high level commands  cl 07/04/84 \ spreadsheet - high level commands  cl 07/04/84
1: quit_calc \ exit spreadsheet
2: y/n abort." BYE" ; \ ask again if YES then quit
3: pos1." Column replicate cell data" \ replicate column data
4: new y/n \ ask again if YES clear it
5: if 0 0 cells \ if clear existing spreadsheet
6: row_max col_max bytes/cell \ ask for row
7: ** erase \ erase cells array
8: 0 row_names \ ask again if YES clear it
9: row_max row_name_len * erase \ erase row_name array
0: 0 col_names \ loop
1: col_max col_name_len * erase \ erase col_name array
2: dict_mark perform \ erase all formulas
3: 0 row_disp ! 0 col Disp ! \ set marker to origin
4: dis_screen \ display cleared screen
5: then ; 

0 \ spreadsheet - high level commands  cl 07/03/84 \ spreadsheet - high level commands  cl 07/04/84
1: mode \ set auto calculation mode
2: pos1." Set auto calculation mode" \ prompt operator
3: pos2." Normal=0 or Auto=-1: 
4: key ascii 1 = \ get response
5: if true \ set mode_flag
6: else false \ accordingly
7: then if cur_row ! \ if ok store it
8: else false \ if cur_row ! \ if ok store it
9: then 0 col_disp ! 0 row_disp ! \ place cell marker at origin
1: dis_screen \ show new screen
2: perform_calc \ force calculations
3: execute formulas then
4: dis_data ; \ show result
5: then ;

FORTH Dimensions 36  Volume VII, No. 2
Availability of Spreadsheet Source Code and F83

To save some rather tough typing, the source code for this program is available postage paid from the author for $25 (6 Sutherland Place, Manitou Springs, Colorado 80829). It is, however, only available on 5.25" disk format for MS-DOS or PC-DOS. The public-domain F83 program should be available from No Visible Support Software, Mike Perry, 1125 Bancroft Way, Berkeley, California 94702.

This is an incredible implementation of Forth–83. Everyone even remotely interested in Forth should buy a copy of it. The entire Forth community should give thanks to Laxen and Perry for donating such an excellent piece of software to the public domain.

The Mirth Dimension

2SWAP DROPY

With apologies to Lewis Carroll.

'Twas SYSOUT and the SFILL toves
Did SRW in the wabe;
All FOUTPUT were the borogoves,
And the BOOT COLD outgrabe.

"Beware the 2SWAP DROP my son!
The ROLLS that bite, the DOs that catch!
Beware the DLIT LOOP and shun
The NORETCOND Bandersnatch!"

He took his OUTER PICK in hand!
Long time the PCOND PORT he sought.
So rested he by the IF ELSE tree,
And FLUSH PMODE in thought.

And as in PIXBLIT thought he stood,
The 2SWAP DROP, with ROTS of flame,
Came TONED through the /MOD wood
And < BUILDS DOES > as it came!

One, two! One, two! And through and through
The OUTER PICK went snicker-snack!
He left it WARNING and with its head
He went CPATH back.

"And hast thou slain the 2SWAP DROP?
Come DI + my QUSER boy!
A TYPED day! Callooh callay!"

He RANDOM in his joy.

'Twas SYSOUT and the SFILL toves
Did SRW in the wabe;
All FOUTPUT were the borogoves,
And the BOOT COLD outgrabe.

—Wayne Cox
Rochester Forth Conference 1985

Forth programmers, project managers, vendors and evaluators gathered in Rochester, New York in June at the annual Forth conference held there. The directors and staff of the Institute for Applied Forth Research assembled a broad spectrum of presentors who gave us a clear profile of the expertise being brought to bear in today’s Forth community. From applications on the space shuttle and automation of an entire airport, to object-oriented programming, to putting Forth in hardware, the papers were intelligent, sometimes witty, and occasionally mind boggling.

Elizabeth Rather of Forth Inc. spoke on the subject of developing and implementing a large-scale application. Titled “Fifteen Programmers, 400 Computers, 36,000 Sensors and Forth,” the talk described work done to automate and fully integrate an international airport in the Middle East. Major applications such as safety, security, climate control, personnel, power distribution, runway lighting, etc., all reside in one large system, with largely common subsystems providing facilities to each. Forth lends itself well to this method of using modular common factors to serve in diverse areas.

Rather described the months of strategy, planning and testing required by the project. Much of the work was done in Alabama, where the team worked in two-weeks-on, one-week-off shifts. A huge hangar-like room was used to house a crowded maze of cables and interconnected machines — each programmable from any of the others — that represented only about a third of the actual site equipment.

After describing the size and complexity of this task, the speaker went on to offer her observations about the requirements for success with large projects: knowledgeable management, and direction; professional skill in all parts of the working team; good communication between the team and management, and between management and the client; the discipline to adhere to strict standards and conventions; and responsibility. She believes all of these points to be crucial, and that the importance of each increases with the overall size of the task.

It would appear that those same ingredients contributed to the success of the Forth programmers and scientists who worked on experiments performed aboard space shuttle missions in November 1981 and October 1984. Dr. Henry Harris of Pasadena’s Jet Propulsion Laboratory described his work with JPL and the Johnson Space Center. The three-year project focused on instrument control and, in the end, was beset by in-orbit hardware problems that would have been insurmountable if it were not for Forth’s interactivity.

JPL chose to use Forth for this project because of its adaptability to a wide range of needs and conditions. They employed intense number-crunching, graphics and sophisticated arrays of geometrical attitudes, comprising something like a megabyte of compiled code. The team implemented elements of LISP and Prolog in their LMI Forth system, which was then used to develop a context-sensitive editor and critical constraint checking to define permissible instrument movements.

Dr. Harris asserted that, because of its modularity, Forth excels in large programs. It is good for jobs with critical deadlines (such as a launch date) and where interactive control can help meet changing conditions (as when it was used to save the shuttle experiment during hardware failures). Harris then presented three myths that have been dispelled by recent accomplishments: that Forth is only good for small programs; that no good software has ever been written in Forth; and that if a program is larger than 64K, it isn’t right. By way of one example, JPL’s Forth on an IBM-XT more accurately predicted the space shuttle’s position than did the mainframe at Johnson Space Center.

All told, the Rochester Forth Conference provided a platform for about sixty speakers to address an audience of nearly two hundred. Several described their experiences with creating significant extensions to Forth for purposes of education, artificial intelligence and adding object orientation, for example. It became apparent that Forth is a natural seedbed for the best features of contemporary computer languages, which can be easily implemented in Forth without the possible restrictions of the environment in which they originate. Charles Duff discussed NEON, an object-oriented language making waves in the greater Macintosh community; Pierre Moreton presented HPORTH, an “English-like business application language”; and Arnold Epstein intrigued listeners with his MAGIC/L, which incorporates a Pascal-like syntax for ease of maintainence.

For an afternoon session, the conference divided into about a dozen working groups, special-interest gatherings ranging in size from six to twenty members. Topics included education, Forth under Unix, robotics, image processing and artificial intelligence, state machines, Forth in hardware, standards and others. Brief reports from the groups indicated a variety of progress, with at least one working group determining to continue research and to report back at next year’s conference.

One evening was devoted to a tour of the University of Rochester’s Laboratory for Laser Energetics, with whose cooperation the conference was held. For two hours, conference attendees inspected the Forth-controlled fusion research facility, whose multiple-beam laser can be fired at half-hour intervals. LLE workers were kind enough not only to tolerate the presence of more than one hundred visitors crowding an evening shift, but to answer many questions and generally humor us.

The lecture program concluded with
a presentation by Lawrence Forsley, conference chairman. His paper on “nth-Order Defining Words” discussed clearly a subject that is somewhat abstract for many, the definition of words that will define other words. Although limited by time, Larry also managed — with the help of an assistant — to demonstrate programming on the fly, and the difficulties of debugging an overhead projector. Needless to say, the audience was delighted.

The final day consisted of a number of well-attended exhibits, including Fort Inc., Miller Microsystems, Forth machines from Novix and Metaforth, the Institute for Applied Forth Research, the Forth Interest Group and Dash-Find Assoc., a Forth-specific employment referral service. Concurrent tutorials attracted members of the community as well as conferees.

In addition to the caliber of presentations, integral to the success of the Rochester conference was the inspiration and efficiency of its organizers. Congratulations and thanks go to Lawrence P. Forsley, conference chairman; Thea Martin, conference coordinator; and to Jonathan Ross and Sherry Snyder of the institute. Their friendly and patient efforts were a reward to all who attended.

—Martin Ouverson

For details of the approximately sixty presentations, watch for the published Proceedings of the 1985 Rochester Forth Conference.
Ultra-Fast Text Search Using Probabilistic Dictionaries

John James
Santa Cruz, California

Suppose you have a large database of text files, such as the documents in a lawyer's office, perhaps one or 100 megabytes or more in dozens or hundreds of separate files. You want to search the whole database for any files which contain certain words, phrases, or AND and OR combinations of words and phrases. And you want the search to be very fast, almost instantaneous.

I haven't implemented this procedure, and don't know of anyone who has. So it's impossible to know for sure how well it will work. Despite the uncertainty, this tutorial seems worth publishing because it illustrates important data structures and concepts as well as offering the promise of important practical benefits.

Background: The Probabilistic Dictionary

Used in some spelling checkers, the probabilistic dictionary is a most interesting data structure, and one which opens new approaches to a number of problems. I'll explain it first with an example. Suppose a spelling checker needs a dictionary of about 40,000 English words but only has 100K bytes to store them in RAM and can't afford the speed penalty of keeping the words on disk, which could require a separate disk access for each word being checked. A probabilistic dictionary can comfortably store the words in two-and-a-half bytes each, no matter how long the actual words are.

To build the dictionary, start by clearing all the 100K bytes (800K bits) to zero. Now, to add each word to the dictionary, use ten different "hashing" functions (see below for explanation of hashing) on the ASCII strings which represent the word. Each hashing function will select one of the bits, by computing a number between zero and 799,999 from the ASCII string. Set each of the ten selected bits to "1" — whether or not it had been set before.

After all 40,000 English words have been entered in this way, somewhat fewer than 400,000 bits — half of the 800,000 available — will have been set. Now, to look up a word, compute the ten hash functions on it and see if all ten bits are set. If even one of the bits is not set, then the word is definitely not in the dictionary. If all ten of the bits are set, the word is probably in the dictionary. But with each bit having about a fifty percent probability of being set anyway, there is about one chance in a thousand of a false hit, meaning that a word not in the dictionary would be "found." For a spelling program, this represents a one-in-a-thousand chance that a misspelled word would not be detected, a risk which may be acceptable. Hence the name "probabilistic dictionary."

Background: Hashing

The point of "hashing" is to take a value, such as an English word or a lengthy part number, and transform it into an arbitrary number which is a more suitable key for indexing into a table. In the example, we need to transform an English word into a number from 0 to 799,999. Of course, different words may transform into the same number (a "collision"); but we try to select a computational procedure such that the numbers don't tend to bunch up, so there won't be too many collisions.

A common hashing computation is to divide by an appropriate number, throw away the quotient and take the remainder. In the example above, this divisor would be a little less than 800,000, so that the remainder will be the right size to select one of the bits. The divisor should be a prime number; for more background on picking a good one, see Knuth. In the above example, ten different divisors would be used to select the ten bits in the table of 800,000. Note that before division, the English word whose ASCII string will be the dividend should be left-justified or rearranged in some other way; otherwise, for short words the dividend would be less than 800,000 and all ten divisions would give the same remainder. Picking a good hashing function can be tricky.

Dictionary for Text "Search"

Now let's use the probabilistic dictionary for another purpose — text searching. The trick is to prepare a separate, small dictionary for each file in the database. All these little dictionaries might fit into RAM or in a small file on disk. Then, to search for a word in the entire text database, just look it up in all of the dictionaries. Only one pass through the dictionaries is enough, even for a complex search involving many words, phrases, and AND and OR combinations of them.

Different kinds of condensed dictionaries could be used, not only the probabilistic dictionary suggested here. We suggested this one because it is compact, easy to implement and very efficient, especially when many dictionaries must be searched for the same words.

How big should each of the dictionaries be? A typical text document of about 10,000 words will have only about 1500 distinct words; a 3000 word dictionary would accommodate much more than a 20,000 word document. Two-and-a-half bytes per word gives 7.5K bytes for each document's dictionary. And note that each dictionary can actually hold more than 3000 words and in fact will never overflow, although performance degrades gradually if too many words are added. The actual dictionary size can be selected by the end user, although for any given database of text files, it's easier to keep all dictionaries the same size than to have larger ones for large documents.

If 500K of RAM is available, over sixty dictionaries of 7.5K each could be kept there for faster searches, although it wouldn't hurt much to keep these dictionaries as a file on disk, since a search would need only one pass through that file. Each dictionary should also contain the name of the file.
which it represents. Then the search through the dictionaries could yield a list of file names. These files would then be searched word by word (full text search), both to eliminate the few false hits resulting from the probabilistic nature of the dictionary, and also to locate the words or phrases sought, in case the user wanted to examine them in context.

Miscellaneous Hints

(1) Logical AND and OR searches are easily handled, but NOT presents a problem because it changes false hits, which are tolerable, into false misses, which are not. So use the full text search before excluding any file from consideration.

(2) Searches are fast. The hash functions need only be computed once, no matter how many dictionaries need to be searched, so the time to compute them should be imperceptible. And for any single word or AND of words, a single bit mask can be prepared for high-speed test against the dictionaries. A quick lookup in a small table specially prepared for the word being sought will usually show that that particular letter is not contained anywhere in the word; and in that case, the intermediate character positions don’t need to be examined at all, since they could not possibly be part of the word being sought. Many other refinements have been developed, since text searching has such practical importance.

(3) The suggested 7.5K-byte dictionary size requires hash-function divisors around 60K. If your system doesn’t have an unsigned division available, consider getting one bit by another method, such as adding the ASCII letters of the English word together and taking the last bit of the sum. But note that this bit will be the same for all ten hash values of a given word.

(4) Phrases can be handled as an AND of the words, with false hits excluded later in the full text search.

(5) This method won’t find parts of words. But common prefixes and suffixes can be treated. For example, to avoid missing plurals of words, just drop the final “s” or “es” before entering or searching for any word. Some false hits will occur, but they will be resolved during the full text search, at a modest cost in performance. Let users know that irregular plurals must still be searched explicitly, with an AND.

(6) In the full text search, you can do better than looking at every letter. For example, when you are searching text for a word, look at the last-letter position in the text first. A quick lookup in a small table specially prepared for the word being sought will usually show that that particular letter is not contained anywhere in the word; and in that case, the intermediate character positions don’t need to be examined at all, since they could not possibly be part of the word being sought. Many other refinements have been developed, since text searching has such practical importance.

(7) Include each file’s last-modification date and time with its dictionary. Then the search program can examine the directory of files, to automatically regenerate the dictionary for any new or changed text file. This mechanism will also generate the dictionary initially, the first time the program is run.

References

1. Bentley, Jon, “A Spelling Checker,” Communications of the ACM, May 1985, Vol. 28, No. 5. This “Programming Pearls” article includes background on text documents and on several methods of checking spelling, and has good references for further research.


References

1. Bentley, Jon, “A Spelling Checker,” Communications of the ACM, May 1985, Vol. 28, No. 5. This “Programming Pearls” article includes background on text documents and on several methods of checking spelling, and has good references for further research.

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

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

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

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

• ARKANSAS
Central Arkansas Chapter
Twice Monthly: 2nd Sat., 10 a.m.,
& 4th Wed., 7 p.m.
Call Gary Smith
501/227-9763

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

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

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

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

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

Bay Area Chapter
Silicon Valley Chapter
Monthly, 4th Sat.,
FORML 10 a.m., FIG 1 p.m.
ABC Christian School Aud.
Dartmouth & San Carlos Ave.
San Carlos
Call John Hall 415/532-1115
or call the FIG hotline:
408/277-0668

Stockton Chapter
Call Doug Dillon
209/931-2448

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

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

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

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

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

• GEORGIA
Atlanta Chapter
Call Ron Skelton
404/393-8764

ILLINOIS
Cache Forth Chapter
Call Clyde W. Phillips, Jr.
Oak Park
312/385-3147

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

Fox Valley Chapter
Call Samuel J. Cook
312/879-3242

Rockwell Chicago Chapter
Call Gerard Kustolek
312/885-8092

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

Fort Wayne Chapter
Monthly, 2nd Wed., 7 p.m.
Indiana/Purdue Univ. Campus
Rm. B71, Neff Hall
Call Blair MacDermid
219/749-2042

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

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

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

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

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

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

MICHIGAN
Detroit Chapter
Monthly, 4th Wed.
Call Tom Chrapkiewicz
313/562-8506

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

• MISSOURI
Kansas City Chapter
Monthly, 4th Tues., 7 p.m.
Midwest Research Inst.
Mag Conference Center
Call Linus Orth
816/444-6655

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

• NEVADA
Southern Nevada Chapter
Call Gerald Hasty
702/452-3368

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

• NEW MEXICO
Albuquerque Chapter
Monthly, 1st Thurs., 7:30 p.m.
Physics & Astronomy Bldg.
Univ. of New Mexico
Call Rick Granfield
505/296-8651

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

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

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

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

• OHIO
Athens Chapter
Call Israel Urieli
614/594-3731

Cleveland Chapter
Call Gary Bergstrom
216/247-2492

Cincinnati Chapter
Call Douglas Bennett
513/831-0142

• ROCKFORD
Rockford FIG Chapter
Monthly, 1st Mon., 7 p.m.
Call David Tubbs
815/886-0545

• SOUTH CAROLINA
Spartanburg Chapter
Monthly, 1st Mon., 7 p.m.
Call Samuel Adair
864/735-8967

• TENNESSEE
Knoxville FIG Chapter
Monthly, 1st Wed., 7 p.m.
Call Bill Seal
865/575-1996

Tennessee Valley FIG Chapter
Monthly, 1st Mon., 7 p.m.
Dillon McNay
1-2448

• VERMONT
Rutland FIG Chapter
Monthly, 1st Mon., 7 p.m.
Call Richard Newell
802/828-5151

• WASHINGTON
Seattle FIG Chapter
Monthly, 4th Wed., 7 p.m.
Central Library
Call John Forsberg
206/325-0108
Dayton Chapter
Twice monthly, 2nd Tues., & 4th Wed., 6:30 p.m.
CFC 11 W. Monument Ave.
Suite 612
Dayton, OH
Call Gary M. Granger
513/849-1483

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

• OREGON
Greater Oregon Chapter
Monthly, 2nd Sat., 1 p.m.
Health Tech. Bldg., OSU Tech.
Call Gary M. Granger
513/849-1483

• PENNSYLVANIA
Philadelphia Chapter
Monthly, 4th Sat., 10 a.m.
Drexel University, Stratton Hall
Call Melonie Hoag
215/895-2628

• TENNESSEE
East Tennessee Chapter
Monthly, 2nd Tues., 7:30 p.m.
800 Oak Ridge Turnpike, Oak Ridge
Call Richard Secrist
615/693-7380

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

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

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

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

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

• VERMONT
Vermont Chapter
Monthly, 2nd Tues., 7:30 p.m.
Vergennes Union High School
210, Monkton Rd.
Vergennes, VT
Call Don VanSycle
802/388-6698

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

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

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

• WISCONSIN
Lake Superior FIG Chapter
Call Allen Anway
715/394-8360

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

• FOREIGN
• AUSTRALIA
Melbourne Chapter
Monthly, 1st Fri., 8 p.m.
Contact Lance Collins
65 Marin Road
Glen Iris, Victoria 3146
03/29-2600

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

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

• CANADA
Nova Scotia Chapter
Contact Howard Harawitz
227 Ridge Valley Rd.
Halifax, Nova Scotia B3P 2E5
902/477-3665

Southern Ontario Chapter
Quarterly, 1st Sat., 2 p.m.
General Sciences Bldg.
Rm. 312
McMaster University
Contact Dr. N. Solntseff
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S 4K1
416/525-9140 ext. 3443

Toronto FIG Chapter
Contact John Clark Smith
P.O. Box 230, Station H
Toronto, ON M4C 3J2

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

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

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

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

• IRELAND
Irish Chapter
Contact Hugh Doggs
Newton School
Waterford
051/7657 or 051/74124

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

• JAPAN
Japan Chapter
Contact Toshihito Inoue
Dept. of Mineral Dev. Eng.
University of Tokyo
7-3-1 Hongo, Bunkyō 113
812-2111 ext. 7073

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

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

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

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

FIGGRAPH
Call Howard Pearlmutter
408/425-8700
Seventh Annual
Forth National Convention
September 20–21, 1985
Hyatt Rickeys, Palo Alto
4219 El Camino Real, Palo Alto, California

Forth Elements Conference Program
EARTH • AEROSPACE • FIRE • WATER
Major Forth conference covering Earth resources, aerospace projects, fire and fusion in industry, and water hydroprojects. Learn about these successful Forth applications during the two-day conference.

Convention preregistration $10.00; or $15.00 at the door.
Banquet $32.00

Special Convention room rates are available at Hyatt Rickeys.
Telephone direct to Hyatt reservations by calling 800 228-9000 and request special Forth Interest Group rates for September 20th and 21st.

More information ................. Call FIG Hotline 408 277-0668
Be a conference speaker ............ Call FIG Hotline 408 277-0668
Exhibitor information ............... Call FIG Hotline 408 277-0668

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