Neural Network Tools (II)

Universal Control Structures

Ada Multiprocessor
Real-Time Kernel
Harris RTX 2000™ 16-bit Forth Chip
- 8 or 10 MHz operation and 15 MIPS speed.
- 1-cycle 16 x 16 = 32-bit multiply.
- 1-cycle 14-prioritized interrupts.
- two 256-word stack memories.
- 8-channel I/O bus & 3 timer/counters.

SC/FOX PCS (Parallel Coprocessor System)
- RTX 2000 industrial PGA CPU; 8 & 10 MHz.
- System speed options: 8 or 10 MHz.
- 32 KB to 1 MB 0-wait-state static RAM.
- Full-length PC/XT/AT plug-in (6-layer) board.

SC/FOX VME SBC (Single Board Computer)
- RTX 2000 industrial PGA CPU; 8, 10, 12 MHz.
- Bus Master, System Controller, or Bus Slave.
- Up to 640 KB 0-wait-state static RAM.
- 233mm x 160mm 6U size (6-layer) board.

SC/FOX CUB (Single Board Computer)
- RTX 2000 PLCC or 2001A PLCC chip.
- System speed options: 8, 10, or 12 MHz.
- 32 KB to 256 KB 0-wait-state SRAM.
- 100mm by 100mm size (4-layer) board.

SC32™ 32-bit Forth Microprocessor
- 8 or 10 MHz operation and 15 MIPS speed.
- 1-clock cycle instruction execution.
- Contiguous 16 GB data and 2 GB code space.
- Stack depths limited only by available memory.
- Bus request/bus grant lines with on-chip tristate.

SC/FOX SBC32 (Single Board Computer32)
- 32-bit SC32 industrial grade Forth PGA CPU.
- System speed options: 8 or 10 MHz.
- 32 KB to 512 KB 0-wait-state static RAM.
- 100mm by 160mm Eurocard size (4-layer) board.

SC/FOX PCS32 (Parallel Coprocessor System32)
- 32-bit SC32 industrial grade Forth PGA CPU.
- System speed options: 8 or 10 MHz.
- 64 KB to 1 MB 0-wait-state static RAM.
- Full-length PC/XT/AT plug-in (6-layer) board.

SC/FOX SBC (Single Board Computer)
- RTX 2000 industrial grade PGA CPU.
- System speed options: 8, 10, or 12 MHz.
- 32 KB to 512 KB 0-wait-state static RAM.
- 100mm by 160mm Eurocard size (4-layer) board.

For additional product information and OEM pricing, please contact us at:
SILICON COMPOSERS INC  208 California Avenue, Palo Alto, CA 94306  (415) 322-8763
Features

9 Universal Control Structures
   Kevin Haddock
Forth's execution control can be difficult to learn, and control structures can be an area of concern even for the enlightened. Forth has all the advantages of an interpreted language like BASIC but, again, the control structures get in the way.

One thing that creates difficulty with Forth, compared to C, is eliminating unnecessary tests: If any one of a series of ANDed conditions is not true, there is no need to waste time extracting and evaluating subsequent conditions. This article attempts to provide an integrated solution that is uniform, clean, simple, and Forth-like.

17 Neural Network Words
   Tim Hendtlass
Neural networks are good at certain tasks the human brain is good at, like pattern recognition, often outperforming traditional computing techniques. A neural network is an interconnection of special processing elements that is based on the brain; each element has inputs and outputs, and is termed a "neurone." Neurones are arranged in logical layers, and a collection of such layers comprises a network.

The code in this article allows construction of a class of simulated networks, and their training by the widely used technique called "back propagation." It is portable to many environments, and other training techniques may be incorporated. (The second of two articles.)

24 Ada Multiprocessor Real-Time Kernel
   Hoyt A. Stearns, Jr.
A fully pre-emptive, priority driven, multiprocessing real-time kernel is required in certain applications, and the Ada rendezvous method is simple, straightforward, and elegant. Ada multitasking will work on multiple processors or on a single processor. A task can be local to a particular processor, or can be distributed.

The listing is the kernel for a system with three processors, but the kernel works fine on a single-processor system.

Departments

4 Editorial ...................... Object-oriented Forth, China Forth exams, keeping energized.

5 Letters ......................... Black-belt exhaustion, mastering metacompilation, Forth on the Macintosh, U.K. contest winners (probability of matching birthdays, one-screen full-screen editor), FDE erratum, answer to a dream.

22 Advertisers Index


38 Best of GENie .............. Jef Raskin, Alan Furman, Bill Muench, Guy Kelly, Dean Sanderson, Roy Martens & Glen Haydon, Charles Johnsen.

40-43 reSource Listings ...... FIG, ANS Forth, classes, on-line connections, FIG chapters.
A last reminder: FD has announced a call for papers about object-oriented programming. The closing date is September 16 for prize consideration—see our ad on page 37 for prize information, and refer to the editorial in issue XIII/1 for other details that may be helpful when preparing your paper. (Papers received after the deadline will still be evaluated for publication.)

So, Why Don’t You Write?

FD is also looking for many new articles from its readers to fill the editorial coffers. This is, after all, a reader-written publication—this means you! We are looking for a wide range of topics, as eclectic as our readership. Chances are, if you find something interesting, clever, or challenging, many others will, too. If you aren’t sure, just drop me a note that outlines the idea, and I will try to give meaningful feedback to you. (And check out our “Author Recognition Program,” details of which are reprinted on page 35.)

If you aren’t sure if your idea is really a new one, or can’t remember whose original work it derives from—to say nothing of the times when you wonder whether the blank wall you’ve been staring at has been scaled already by some other Forth explorer—there are some FIG reference tools you might want to get. Look on the FIG Mail Order Form for the “We’re Sure You Wanted to Know…” series. The Forth Bibliography from Rochester also provides Forth reference information from outside the FIG domain, but hasn’t been updated since January 1987. Get a copy anyway for an inspirational look at how much Forth material really has been circulating through the years.

Chinese National Exam of Forth Programmers

China has officially announced a national examination to be held in September. Great importance is often attached to the results of such examinations. As Dr. Ting relates, the Chinese love examinations, a tradition of 2000 years. For your interest, following is a translation of part of the announcement:

To accelerate the application of Forth in various technical fields, to enlarge the ranks of Forth programmers, to raise the expertise in the Forth language, and to satisfy the increasing need for Forth users, the Chinese Scientific Instrumentation Society, the Chinese Software Association, and the Chinese Forth Interest Group will jointly sponsor a National Examination of Forth Programmers on September 8, 1991 in various provinces and cities across China. The sponsors will grant certificates to programmers passing this examination. They will also select the best ten Forth programmers and thirty excellent Forth programmers, based on the examination, to receive special certificates and awards, to be recommended to participate in Forth projects, and to carry out advanced studies abroad.

Subjects of Examination

1. Computer Fundamentals (20%)
   Number systems and conversions
   Internal representations of numbers
   Major components in a computer and their relationships
   Instructions and instruction sets
   Characteristics and functions of registers
   Characteristics and properties of I/O devices
   Data structures (stacks and lists)
   Operating system fundamentals
   Software engineering fundamentals

2. Elementary Knowledge of

(Continued on page 34.)
Letters
Letters to the Editor—and to your fellow readers—are always welcome. Respond to articles, describe your latest projects, ask for input, advise the Forth community, or simply share a recent insight. Code is also welcome, but is optional. Letters may be edited for clarity and length. We want to hear from you!

Black-Belt Exhaustion & Lean, Mean FIG
Dear Phil Koopman, Jr.:

I was sorry to hear that you decided to drop your FIG membership [see FD XII/2]. You are one of the very visible and highly acclaimed Forth programmers. It is a great loss to FIG, not having you continue your membership.

You correctly pointed out that FIG is having financial troubles. The main problem I see is that we have been losing members since 1984, when the membership peaked at about 4,500. The current membership of about 2000 cannot support the operation of FIG using a professional support organization like ADC.

The membership loss has puzzled me all these years. There are more computers, more users, but why is FIG getting leaner and leaner? We have tried many different things to attract more new members, as well as to keep the old ones, but nothing seemed to work. There were substantial improvements on the appearance and substance in FD. We put lots of effort into getting the GEnie Forth RoundTable going. We had gone out of our way to promote Forth to the outside, but to no avail.

Harris put in great amounts of money and energy to promote Forth, and even assembled a very high-powered team of Forth blackbelts, including yourself, to develop the turbo-charged Forth engines. Why did Harris abandon its efforts? I really hope that somebody will tell me what's going on and what we have to do to change the public perception of the inadequacies of Forth and FIG.

FIG has always been an organization of volunteers. The founders of FIG expended great energy to craft fig-Forth, which propelled FIG into a viable association. Other members like Mike Perry, Henry Laxen, Bob Smith, and Tom Zimmer improved on fig-Forth and contributed F83 and F-PC, which have kept FIG going.

I am fully aware of your contributions to Forth, in MVP-FORTH and the WISC technology. However, may I ask what you have contributed to FIG as a member? You complained that there is not a FIG chapter in your area. Every chapter is started by somebody; have you considered starting one yourself? You certainly have the capability, the energy, and the charisma to lead a large group and to teach lots of newcomers. Maybe the reason that FIG is shrinking is due to the exhaustion of leaders like you?

I had not served on the FIG Board of Directors until last year. However, I have regularly attended the business meetings and watched FIG's operations for ten years. One thing I can say for the Board is that every dollar is accounted for and the records are kept accurately. All expenses are openly discussed and must be approved at these meetings. The books are available at the FIG office for members to inspect. Forth Dimensions will also be publishing financial reports. [See the President's Letter in this issue.—Ed.]

I appreciate that you took the trouble to write and tell us what you think of FIG. I fully understand your frustration, and respect your decision not to continue your membership. However, I think the greatest challenge we Forth programmers all face is not just to survive the onslaught of C and Unix, but to pool our resources together and find the niche where Forth can still grow and shine.

Let me repeat an ancient Chinese story to make my point. The King of Tsu met the old man why he was crying. The old man said, "I presented this piece of rock to your grandfather because it contains a large jade. Your grandfather let his jade expert inspect it, and he said, 'It's just a stone.' So, your grandfather cut off my right leg. Then I presented it to your father. Your father had it inspected, and he cut off my left leg. I cannot walk any more. Then Heaven brought you here so I can see you. Here is the rock. It's yours, if you can use it." The King of Tsu ordered his jade worker to saw it open and found the largest and the most beautiful jade ever seen in China. It was from this jade that the First Emperor of China made his imperial seal.

Forth is like this piece of jade in a rock. We know it is the best tool for human beings to control computers. We tried to give it to the world, but what did we get back? Both feet cut off. What can we do? I think all we can do is stick together, keep warm, and keep the pot stirred. If Forth is really that good, it will shine eventually. When? I do not think anybody knows. FIG is the only place where we can keep ourselves warm.

Dr. C.H. Ting
San Mateo, California

Mastering Metacompilation
Dear Editor,

Metacompilation is get-
ting a bum rap.

Recent postings on GEnie, and an item by Dr. C.H. Ting in *Forth Dimensions* ("How Metacompilation Stops the Growth Rate of Forth Programmers," *FD* XIII/1), have re-opened one of the dark closets of Forth: metacompilation.

Metacompilation is probably the most arcane aspect of Forth, and is usually considered the province of the Forth "priesthood." I sympathize. But I think it's a mistake to abandon metacompilation for assembly language source, as Dr. Ting advocates. This is like going back to gas lighting because people get shocked by (and don't understand) electricity.

Education is the answer.

For the last four years, I've been saying that the problem is not that there aren't good Forth metacompilers, but that there isn't any good documentation for metacompilers.

I've come up through this the hard way. My first experience with a metacompiler (cross-compiler) was in 1982, when I tried to use a commercial product. The documentation was incomprehensible, the compiler obscure. Finally, I was able to complete my project by hacking my code onto the kernel supplied with the compiler, and by abandoning any attempt to use defining words, IMMEDIATE words, or vocabularies. (Rather like programming in C, come to think of it.)

My second experience was with a version of Cassady's Metaforth, implemented and modified by a friend of mine. I began to
The breakthrough came on my third attempt: the polyFORTH target compiler. And the reason for the breakthrough is not that the compiler is that much simpler (although it's certainly elegant), but that I was able to attend the polyFORTH advanced course and have someone teach me how metacompilers work. Enlightenment!

I then proceeded to test my understanding of the theory and practice of metacompilers, by writing one of my own. (I had other reasons for this—I needed features that no one else was offering.) But, remembering how much difficulty I had in acquiring the essential concepts, I resolved to keep notes and record my thoughts as I wrote the compiler. I think I may be the first person to document the process of discovery this way.

In December of 1988 I condensed these notes into a one-hour presentation for our local FIG chapter, and was amply rewarded. Even though I delivered it at warp speed, several people came up to me afterward and told me that they understood metacompilation for the first time.

The lesson I learned from giving this talk is this: there's no reason why metacompilers should be hard to learn, use, or understand. The concepts aren't complicated; it's just that everyone who understands metacompilers takes them for granted!

Frank Sergeant, in *Forth Dimensions* XII/6, has started to dispel the mystery of metacompilation, by presenting a clear description of an elegant compiler. Like eForth, Pygmy is being ported to many CPUs in the embed-
MEET THAT DEADLINE!!!

- Use subroutine libraries written for other languages! More efficiently!
- Combine raw power of extensible languages with convenience of carefully implemented functions!
- Yes, it is faster than optimized C!
- Compile 40,000 lines per minute!
- Stay totally interactive, even while compiling!
- Program at any level of abstraction from machine code thru application specific language with equal ease and efficiency!
- Alter routines without recompiling!
- Use source code for 2500 functions!
- Use data structures, control structures, and interface protocols from any other language!
- Implement borrowed feature, often more efficiently than in the source!
- Use an architecture that supports small programs or full megabyte ones with a single version!
- Forget chaotic syntax requirements!
- Outperform good programmers stuck using conventional languages! (But only until they also switch.)

HS/FORTH with FOOPS - The only full multiple inheritance interactive object oriented language under MSDOS!

Seeing is believing, OOL's really are incredible at simplifying important parts of any significant program. So naturally the theoreticians drive the idea into the ground trying to bend all tasks to their noble mold. Add on OOL's provide a better solution, but only Forth allows the add on to blend parts of any significant program. So and only Forth allows the add on to blend parts of any significant program. So and only

ROBOT, with methods MOVE and

lets define classes BODY, ARM, and

ROBOT, with methods MOVE and

RAISE. The ROBOT class inherits:

INHERIT> BODY
HAS> ARM RightArm
HAS> ARM LeftArm

If Simon, Alvin, and Theodore are robots we could control them with:

Alvin's RightArm RAISE or:
+5 -10 Simon MOVE or:
+5 -20 FOR-ALL ROBOT MOVE

Now that is a null learning curve!

WAKE UP!!!

Forth is no longer a language that tempts programmers with "great expectations", then frustrates them with the need to reinvent simple tools expected in any commercial language.

HS/FORTH Meets Your Needs!

Don't judge Forth by public domain products or ones from vendors primarily interested in consulting - they profit from not providing needed tools! Public domain versions are cheap - if your time is worthless. Useful in learning Forth's basics, they fail to show its true potential. Not to mention being a-l-o-w.

We don't shortchange you with promises. We provide implemented functions to help you complete your application quickly. And we ask you not to shortchange us by trying to save a few bucks using inadequate public domain or pirate versions. We worked hard coming up with the ideas that you now see sprouting up in other Forths. We won't throw in the towel, but the drain on resources delays the introduction of even better tools. Don't kid yourself, you are not just another drop in the bucket, your personal decision really does matter. In return, we'll provide you with the best tools money can buy.

The only limit with Forth is your own imagination!

You can't add extensibility to fossilized compilers. You are at the mercy of that language's vendor. You can easily add features from other languages to HS/FORTH. And using our automatic optimizer or learning a very little bit of assembly language makes your addition zip along as well as in the parent language.

Speaking of assembly language, learning it in a supportive Forth environment turns the learning curve into a light speed escalator. People who failed previous attempts to use assembly language, conquer it in a few hours or days using HS/FORTH.

HS/FORTH runs under MS/DO or PC/DO, or from ROM. Each level includes all features of lower ones. Level upgrades: $295. plus price difference between levels. Source code is in ordinary ASCII text files.

All HS/FORTH systems support full megabyte or larger data & data, and run faster than any 64k limited ones even without automatic optimization -- which accepts almost anything and accelerates to near assembly language speed. Optimizer, assembler, and tools can load transiently. Resize segments, redefine words, eliminate headers without recompiling. Compile 79 and 83 Standard plus F83 programs.

PERSONAL LEVEL $295.

NEW! Fast direct to video memory text & scaled/clipped/windowed graphics in bit bilt windows, mono, cga, ega, vga, all ellipsoids, splines, bezier curves, arcs, turtles; flattest fast pattern drawing even with irregualar boundaries; powerful parsing, formatting, file and device I/O; DOS shells; interrupt handlers; call high level Forth from interrupts; single step trace, decomplier; music; compile 40,000 lines per minute, stacks; file search paths; format to strings. software floating point, trig, transcendental, 18 digit integer & scaled integer math; vars: A B * Associable to 4 words, 1-4 dimension var arrays; automatic optimizer for machine code speed.

PROFESSIONAL LEVEL $399.

hardware floating point - data structures for all data types from simple thru complex 4D var arrays - operations complete thru complex hyperbolics; turnkey, seal; interactive dynamic linker for foreign subroutine libraries; round robin & interrupt driven multitaskers; dynamic string manager; file blocks, sector mapped blocks; x86 & 286.

PRODUCTION LEVEL $499.

Metacompiler: DOS/ROM/微/indirect; threaded systems start at 200 bytes, Forth cores from 2 kbytes; C data structures & struct compiler; TurboWindow-C MetaGraphics library, 200 graphic/window functions, PostScript style line attributes & fonts, viewports.

ONLINE GLOSSARY

$ 45.

PROFESSIONAL and PRODUCTION LEVEL EXTENSIONS:

FOOPS+ with multiple inheritance $ 79.

TOOLS & TOYS DISK $ 79.

ROMULUS HS/FORTH from ROM $ 99.

FORTTRAN translator/mathpak $ 79.

Compile Forth subroutine libraries! Formulas, logic, do loops, arrays; matrix math, FFT, linear equations, random numbers.

When I first started working with Forth, it's execution control was one of the most difficult aspects to learn. The basic words weren't difficult to understand—but I was encountering frustration trying to make my programs fit into the mold those words wanted to impose on them.

After seeing the wisdom of everything else Forth has to offer—simplicity, factorizing, bottom-up design, performance, and economy, to name a few—I see evidence that control structures are an area of concern even for the enlightened few.

Indications of this can be found in the numerous efforts at case statements: Parnas' IT ... TI (FD VI/1), and security-less (e.g., poly-) Forth's gymnastic mixing and matching of incompatible types of control operators by bouncing offset addresses around on the stack at compile time.

I have always claimed that Forth has all the advantages of an interpreted language like BASIC, but here again the control structures get in the way. In BASIC, you can always whip up a quick loop or check the logic of your code with a conditional typed at the command line.

I have seen this interpretative situation addressed to some degree, but to date I have not seen an integrated solution that is uniform, clean, simple, and "Forth-like." (If I can judge what is "Forth-like"; the debate rages on!).

I was discussing the situation with a Forth-wiser friend of mine when I made the first step in stumbling upon the solution. Sometimes the hardest solution to see is the one that is right in front of your nose. (This happens a lot with me, especially when it comes to Forth.)

He said, "You know, Kevin, there really are only two control operators in Forth, in comparison with C, for example, was crafting the code to eliminate unnecessary tests. If any one of a series of ANDed conditions is not true, there is no need to waste any more time extracting and evaluating subsequent conditions. Similarly, if any one in a series of ORed conditions is true, processing can continue in the body of the condition. Since real-time is one of Forths' strengths, how come it falls so flat in this area? Control structures!

While writing my on-line classified ad system in Bourne Shell, I recall having fun seeing how many of my conditional needs could be met with the | | (OR) and && (AND) operators, which would drop out of the current line of script if the previous command returned true or false, respectively. I was pleasantly surprised at how comfortable it was working with this arrangement, although it required a slightly different mode of thought than is conventionally used in programming. I could appease
my old modes of thought by coining the pet names: \texttt{ifso for} \& and \texttt{ifnot for} \}. (Since there was little collision with Forth words, I opted for the single-character versions.) You can use whatever emotional crutch feels the most appropriate to you, or none at all.

It was apparent that just falling out of the line (or colon definition) was inadequate for my needs. Most people aren't willing to factor their definitions to that level, nor should they have to. There was also the question of looping, restarting the execution at some previous point in the code.

Those operators \& and \| marked a point of departure, but there needed to be something (other than \$ and ;) to mark the points of return. In other words, if \& and \| caused you to break out of the execution of the code, based on some condition, where do you land? Other languages have their endifs, thens, line terminators, etc. After serious deliberation, I decided to confisicate \texttt{C}'s block delimiters \{ and \}.

I know, I hear you saying, "Here is just another character trying to make Forth more \texttt{C}-like," but trust me, the icons are where the similarity ends. Years ago, someone suggested they look like profiles of faces. I never took it one step further by saying that, together, they resembled the faces, comedy and tragedy, of live theater.

With this worked out, conditionals \| and \& from which to branch, and layers of balanced control block delimiters, I had discovered how to lay down the \texttt{BRANCH} control ops. The only thing left to determine was how to lay down the \texttt{BRANCH} ops responsible for looping and skipping the \texttt{ELSE} part of a conditional construct.

My solution was to use combinations of one or more carets (\^) and lower-case v's (you can use upper case, if your system converts case automatically) to specify that execution was to float or sink that many control block levels. (Actually, "levels" is probably the wrong word here. You are actually "de-nesting" if you are thinking in terms of \texttt{C}, but what matters is whether you de-nest toward the top or the bottom of the program.)

Let me see if I can bring this all together for you in Figure One. You will notice later that sink 1 (\texttt{v}) is never needed: the default behavior is to fall out of the current block at the end. \& and \| serve the purpose of falling out of the current block before the end, based on a condition.

Some of you may find that a caret (\^) is difficult to enter with your editor, especially if yours is patterned after the Forth, Inc. lineeditor, which uses it as a delimiter. Two alternatives exist. Presented in the order of my personal preference:

- Change your editor's delimiter to something else; my suggestion would be a line feed; this character shouldn't have too adverse an affect on any video updating and, since it's non-printable, you would be highly unlikely to want to enter it into your source code.

Or:

- Change the icon used for the float (\^). You could use the tilde (=), which on most CRT's looks like a bubble, and that could be your memory aid: The more bubbles, the higher it will float.

Another thing you will


\ UCS -- interpretative scanning
: @CHAR ( S - c ) >IN @ BLK @ ?DUP IF BLOCK + C@ ELSE TIB + C@ THEN ;
: ACCOUNT ( S cnt chr - cnt' ) DUP ASCII { = IF DROP 1+ ELSE ASCII } = IF 1- THEN THEN ;
: >NEXT ( S n ) 0< IF 1 ELSE -1 THEN >IN +! ;
: I~ ( S n ) -1 >IN +! DUP 0< IF 1 ELSE -1 THEN BEGIN @CHAR ACCOUNT 2DUP = 0= WHILE OVER >NEXT REPEAT 2DROP ;
: I&| BEGIN @CHAR ACCOUNT ?DUP WHILE -1 >NEXT REPEAT 1 >IN +! ;

\ Universal Control Structures
: \ ( S - a 0 ) STATE @ IF HERE 0 THEN ; IMMEDIATE
: & ( S a - a ) STATE @ IF COMPILE ?BRANCH HERE SWAP , ELSE 0= IF 1 I&| THEN THEN ; IMMEDIATE
: | ( S a - a ) STATE @ IF COMPILE 0= [ COMPILE ] & ELSE IF 1 I&| THEN THEN ; IMMEDIATE
: ) ( S a a ) STATE @ IF BEGIN ?DUP WHILE DUP @ SWAP HERE SWAP ! REPEAT DROP THEN ; IMMEDIATE

\ Universal Control Structures
: ` ( S - a _ ) BL WORD NUMBER DROP ; IMMEDIATE
: ~ ( S a1 a2 a1 a2 ... n ) STATE @ IF COMPILE BRANCH >R SP@ R@ ABS 2* 2* + R> 0< IF HERE OVER @ , SWAP ! ELSE 2+ @ , THEN ELSE I~ THEN ; IMMEDIATE
: ~: ( S n _ ) CREATE , DOES> @ [ COMPILE ] ~ ;
: ~S: ( S e s _ _ _ ... ) DO I ~: IMMEDIATE LOOP ;
5 -4 ~S: vvvv vvv vvv vv ^ ^ ^ ^ ^ ^ ^ ^ ^

\ DUMP
16 CONSTANT WIDE
: .CHAR ( S c ) { [ DUP 32 < | DUP 126 > | VV ] DROP ASCII . } EMIT ;
: .ASC ( S a c ) { OVER C@ .CHAR 1- SWAP 1+ SWAP ?DUP & ^ } DROP ;
: .HEX ( S a c ) { OVER C@ 0 <# # # # # > TYPE SPACE 1- SWAP 1+ SWAP ?DUP & ^ } DROP ;
: .ADR ( S a ) 0 <# ASCII : HOLD # # # > TYPE ;
: .DUMP ( S a c ) { [ KEY? | CR BASE @ >R OVER .ADR 2DUP .HEX SPACE ASC R> BASE ! VV | ABR ] } DROP ;
: DUMP ( S a c ) >R { R@ & [ DUP WIDE MOD ?DUP | WIDE ] R@ MIN R> OVER - >R 2DUP .DUMP + ^ } R> 2DROP ;

@CHAR returns the next char in the input stream
ACCOUNT -- given the current level count and char returns the adjusted count

(Continues.)
order here: the interpreted version just does a string
search for the balancing control
delimiter, so remember: if you use braces—er, faces—
in your words or in comments, always use them in
balance. For instance, you
could name a word (FRED) but try to avoid (FRED un-
less it is to be followed almost immediately with something
like FRED). This is the only
minor un-Forth-like limitation on syntax, but I feel it is
a small price to pay for the
benefit received.

Possible uses for the interpreted version could be
compilation scripts, non-time-
critical routines, memory-
critical programs, debugging
with zero compile time, and
my favorite: a late-binding,
object-oriented system where
methods could exist in
different vocabularies that know
how to deal with their own
data types. You select the
context by determining what
kind of data object you are
dealing with, then process the actual, reusable algorithm
interpretively from disk.
With the old FIG-style hier-
archical vocabularies, you
could essentially build
superclasses.

And now for the moment
you have all been waiting
for: on to the code!

Due to the drastically
different behavior of the
compiled and interpreted
operators, I will describe
them separately, even though
some versions of Forth will
bundle them into the same
state-smart word.

It goes without saying that
all control structures are
compiler directives and, therefore, have precedence
in whatever mechanism is
appropriate for your Forth.

While compiling, UCS
holds a pair of addresses on
the stack for each open con-

> NEXT -- adjusts the input stream pointer based on the
    sign of the given number

I -- adjusts the interpretative pointer to the given control
    block. Negative control block offsets move the adjust the
    pointer forward. Positive or zero backward.

I±I -- adjust the interpretative pointer to the end of the
current control block.

{ -- denotes the start of a control block.

& -- falls out of the current control block if given value is
false

| -- falls out of the current control block if given value is
true

} -- marks the end of a control block. When compiling, this
word resolves all the forward references for this block.

   UCS — Pygmy version.

( Universal control structure 901127 KAH ) COMPILER
: { ( - a 0 ) \ BEGIN 0 ;
: % ( a - a ) COMPILER 0branch HERE da @ - SWAP , ;
: : ( a - a ) COMPILER NOT \ & ;
: ) ( a a ) BEGIN ?DUP WHILE DUP @ SWAP HERE da @ - SWAP !
    REPEAT DROP ;

FORTH

comment:
UCS -- PYGMY VERSION
| opens up a conditional structure. When in doubt use a lot
of these. Make sure to close them. Check your stack after
compiling to make sure you got them all.
& takes an argument off the stack and if false branches to the
end of the current conditional structure. Does a boolean
and but can be thought of as (and pronounced) 'if so'.
| does the same as above except branching when true. Does a
boolean OR. Can be thought of as (and pronounced) 'if not'.
} resolves all the exit conditional branches for this structure
and marks the exit location. These must balance the open
branches above. Check your stack after compiling when not sure

comment:

( Universal control structure 901127 KAH ) COMPILER
: \ (- - a ) 32 WORD NUMBER ;

: % ( a1 a2 a1 a2 ... n _) COMPILER branch
    DUP ABS BEGIN ?DUP WHILE 2SWAP PUSH PUSH 1- REPEAT
    DUP PUSH 0< IF HERE da @ - SWAP , ELSE OVER , THEN
    POP ABS BEGIN ?DUP WHILE POP POP ROT 1- REPEAT ;
UCS -- PYGMY VERSION
allows you to specify the branch direction and number inline while compiling, i.e. -5 branches down 5 levels (not including the one you are in).

~ lays down an inline branch (for positive numbers) or forward (for negative numbers) the given number of levels.

's and v's are shorthand for forward and backward branches.

(Interpretive universal control structures) FORTH

: ACCOUNT ( cnt chr - cnt') { { DUP ' = & DROP 1+ vv } ' } = & 1- } ;

: NEXT ( n ) { { 0< & 1 vv } -1 } >IN +! ;

: \- ( n ) -1 >IN +! { { DUP 0< & 1 vv } -1 } { @CHAR ACCOUNT { 2DUP = | OVER >NEXT ^ } } 2DROP ;

(Interpretive Universal Control Structures)

: & ( n ) { | 1 { @CHAR ACCOUNT ?DUP & -1 >NEXT ^ } 1 >IN +! } ;

(The compiled version)

The top one points to a null-terminated thread of forward branch offset cells (the ones compiled right after the BRANCH and OBRANCH ops) for the current control block. The second one holds the address of the beginning of the control block, to resolve the backward branches.

Comedy () just leaves the current dictionary address and the initial 0 on the stack.

Ifso (&) compiles the conditional branch op OBRANCH, pushes the address of the offset cell following it (HERE), then compiles (comma's) the previous (possibly null) forward reference link there. This adds the next link in the forward reference chain.

Ifnot (i) is self-explanatory, just compiling an op to invert the test for ifso.

All the floaters and sinkers are just user interfaces to a word that digs down in the compile-time stack, either to link into the appropriate forward thread or to resolve a backward branch.

Tragedy () just tosses the control block start address left by comedy, then loops through the thread, resolving the forward branches until terminated by the null link.

The interpreted versions of comedy and tragedy do nothing except act as targets for character scanning.

In an actual installation, the character-scanning primitives (consisting of the words @CHAR, ACCOUNT, >NEXT, and \-, and even the loops in & and i) should be in machine code. I will leave it as an exercise to you, the reader, to work this out. Suffice it to say the code just scans the input stream pointer across the text until it finds the appropriate level of imbalanced open or closed braces. Also, you will note in
the Pygmy version, this gives us our first live example of compiled UCS in action. I also leave it as an exercise to re-code the compilable part of UCS for metacompiling (and even your Forth nucleus, if you get so inspired). I'm not saying a Forth should not have the traditional control structures, just that it might be a more elegant solution defining them and Forth in terms of UCS, rather than the other way around.

Included in the listings are some quick examples: a case statement in the form of what could be a key/command loop, both in interpreted and compiled form; a memory dump utility; and a simple utility to create memory headers that will go out and interpretively load blocks, saving memory and compilation time (also with possible object-oriented implications as mentioned earlier).

As a final note, remember to keep an eye on your stack depth changes before and after compiling. The UCS have, and need, no compiler security. Too many comedies will grow your stack and too many tragedies will shrink it. Minimalists should already be familiar with this technique.

I hope you will have as much fun playing with, and improving upon, the UCS as I did discovering them. Enjoy!

: | ( n) { & 1 { @CHAR ACCOUNT ?DUP & -1 >NEXT ^} 1 >IN +!} ;
: > ^ 0 \~ ; : ^^ 1 \~ ; : ^ ^ ^ 2 \~ ;
: ^ ^ ^ 3 \~ ; : ^ ^ ^ ^ ^ 4 \~ ;
: vvv -1 \~ ; : vvv vvv -2 \~ ;
: vvv vvv -3 \~ ; : vvv vvv vvv -4 \~ ;

comment:

UCS -- PYGMY VERSION
These have basically the same meaning and usage as the compilable ones
comment;

( \ \REM \COMP ED: and SCAN)

DEFER \n
: \COMP CREATE BLK @, POP DROP DOES> @ LOAD ;
: \REM >IN @ 64 / 1+ 64 * >IN ! ;
' \REM IS \n
: ED: ( _) ' 3 + @ EDIT ;
: SCAN [ ' ] \COMP [ ' \ 1+ ] LITERAL !
THRU [ ' ] \REM [ ' \ 1+ ] LITERAL ! ;

comment:

UCS -- PYGMY VERSION
\ during loading acts as a interpretative comment marker and
during scanning compiles the following word which loads
the affected screen\n\COMP is the version that compiles the loading word\n\REM is the comment (REMark) marker

Initially set the vector to remark

ED: <name> edits the screen for the loading word <name>
SCAN acts like thru except scanning and creating loading words

comment:

( Compiled sample key input/case loop )

: TEST
{ 
  CR ." ENTER A LETTER:" KEY
  { DUP 27 = & ." BYE " vvv }
  { DUP 'A = & ." ALPHA " vv }
  { DUP 'E = & ." EDWARD " vv }
  { DUP 'I = & ." IDA " vv }
  { DUP 'O = & ." OCEAN " vv }
  { DUP 'U = & ." UNION " vv }
}

September 1991 October 14 Forth Dimensions
CR BEEP DUP EMIT ." IS NOT A VOWEL! "
} DROP ^
} DROP ;

comment:
UCS EXAMPLE -- PYGMY VERSION
An example of a compiled case structure
hit escape to quit

comment;

DUMP ) 16 CONSTANT WIDE
: .CHAR ( c ) { { DUP 32 < | DUP 126 > | vv } DROP . } EMIT ;

: .ASC ( a c ) { SWAP C@+ .CHAR SWAP 1- ?DUP & ^ } DROP ;

: .HEX ( a c ) { SWAP C@+ 0 <# # # #> SPACE SWAP 1- ?DUP & ^ } DROP ;

: .ADR ( a ) 0 <# ': HOLD # # # # > ;

: .DUMP ( a c ) ?SCROLL CR BASE @ HEX PUSH OVER .ADR 2DUP .HEX SPACE .ASC POP BASE ! ;

: DUMP ( a c ) PUSH { I & { DUP WIDE UMOD ?DUP | WIDE } I MIN POP OVER - PUSH 2DUP .DUMP + ^ } POP 2DROP ;

comment:
UCS EXAMPLE -- PYGMY VERSION
WIDE returns how many bytes to dump per line
.CHAR prints the given char if printable else prints an ellipse.
.ASC prints the given count number of chars at the given addr
.HEX prints c hex bytes at the given addr
.ADR prints the given address
.DUMP prints one line of the dump for the given addr and count
DUMP dumps c bytes of memory at the given addr.

comment;

\ TEST interpreted sample key input/case loop
{ CR ." ENTER A LETTER:" KEY
{( DUP 27 = &." BYE " vv v
{( DUP 'A = &." ALPHA " vv }
{( DUP 'E = &." EDWARD " vv }
{( DUP 'I = &." IDA " vv }
{( DUP 'O = &." OCEAN " vv }
{( DUP 'U = &." UNION " vv }

CR BEEP DUP EMIT ." IS NOT A VOWEL! "
} DROP ^
} DROP ;
(Pascal, C, C++, MacApp, Assembler), MDS Assembler, Language Systems Fortran, MacFortran, Prograph, Smalltalk, Think C, TLM Pascal, True Basic, Lisp, Modula2, Simula, MacForth (still on sale), Mach2 Forth (no more advertised), Neon (become public domain), MasterForth (not supported for a long time), Pocket Forth, Hypercard, and... as well as the many dBase-like languages like Omnis, 4th Dimension, etc.—sure the Mac "speaks many languages," but the environments are "oh, so different."

Macintosh programmers are busy generating standalone applications and adhering rigidly to Apple's programming guidelines. Forth is not concerned with that. (How could it be?)

The majority of Forth code exchanged within the Forth community in journals like Forth Dimensions and the Journal of Forth Application Research doesn't deal with standalone applications, Forth community lost its last meeting platform. Whatever happened to the MacForth user group? The last newsletter I got was sometime in late 1986!

The Forth community is, naturally, busy debating Forth. And it is mostly machine independent.

The Forth compiler and interpreter is a powerful tool, in itself, that is always present. This is perhaps the major and most important difference between Forth and any other language, and is immensely hard to explain to non-Forthers. A lot of Forth's power lies precisely in its accessibility: the ability to extend the compiler and interpreter, to add to it, to use or abuse it (refer to Defector's reference to anarchy) within new definitions. The fact is that the processor is there for you to use, directly from an interactive console or via some prior compilation.

On the Mac, we are expected to program so that, ultimately, the Forth layer will not be present. No more definable words, no more vocabularies, no more Forth interpreter! Some of Apple's programming guidelines also affect the available Fortths directly. For example: not to use any of the 68000 trap mechanisms, and not to use in-line variables in code segments (due to future instruction caching). Words like

(Continued on page 30.)
Neural Network Words

Tim Hendtlass
Hawthorn, Victoria, Australia

The Last of Two Parts

In part one of "Neural Network Words," a number of words were developed to build and train neural networks. Words are now presented to save a network layer to disk. Most networks will learn from real life data rather than from an artificial example, such as the XOR relationship used in part one. The advantages of scaling the real life input and output values to suitable internal values for the neurones used was also demonstrated in part one.

Working out scaling factors is tedious, at best, and words are now presented to read a data file, calculate the required scaling factors, and save these to disk. Finally, a general-purpose word to train a network is presented. It is customised to a particular application by assignments made to four deferred words. An example that uses these words is given.

Words to copy a network layer to and from disk
SAVE-LAYER
This word saves a neural layer to disk. The address of the layer to save must be on the stack on entry, and the name of the file to create must immediately follow SAVE-LAYER on the current input line. If a file by this name already exists, the user is asked for permission to overwrite it.

READ-LAYER
This word restores a neural layer from disk. The layer must have been created before the word is called. The address of the layer to copy to must be on the stack on entry, and the name of the file to read must immediately follow READ-LAYER on the current input line. No check is made to see that the layer being read to has the same dimensions as the layer on disk.

More complicated the relationship to be learned, the more neurones will be needed.

Errata
In the first part of this article (FD XIII/2), we erroneously printed an incorrect figure. In that issue, the Forth file printed as Figure Eight (beginning on page 26) actually contains words related to disk I/O, which is discussed in this installment. Please refer to it for that code. In turn, the file NNDemo.SEQ, which should have been printed last time as Figure Eight, appears hereafter. We apologize to the author and any readers who were inconvenienced.

The author also wishes to note that in DISKNN.SEQ (the file which appeared last time as Figure Eight), he used a couple of outdated word names in the opening comment: NL->DISK instead of SAVE-LAYER, and DISK->NL instead of LOAD-LAYER.
—Ed.

Words to read a data file
Data files used by this software are simple ASCII files, with one set of input values and one set of output values per line. No explicit provision is made in this version for comments, although anything on a line after the specified number of inputs and outputs will be skipped over. The variable N-IN specifies the number of inputs to the whole network, the variable N-OUT the number of outputs from the whole network. READ# reads one number from the current input and scales it into internal form.

The convenient word PROCESS-FILE saves information about what we were doing and moves up one handle on the handle stack. It then opens a new file, processes it, closes it, and returns to what we were doing when the PROCESS-FILE word was encountered. PROCESS-FILE must be followed directly by the name of the file to use, and processing continues immediately after the filename. The actual processing done is controlled by the deferred word (PROCESS-FILE) which can be set to anything at all. SET-SCALES and TEACH-NETWORK (see below) differ mainly in what (PROCESS-FILE) is equated to.

Scaling words
SET-SCALES reads the file specified immediately following this word. It finds the maximum and minimum input and output numbers,
and works out the scaling factor to scale the real life input and output values to the range from -0.5 to +0.5, which suit the non-linear transform in use. After the scale factors have been set, the words SCALE-IN and SCALE-OUT# scale an input or output number. The word UNSCALE-OUT# returns an output from internal representation to real life units. The scaling factors can be saved to disk and restored from disk with SAVE-SCALES and READ-SCALES, respectively. Each of these words must be immediately followed by the name of the file to use.

The generic training word

TEACH-NETWORK is the top generic training word. It expects the number of training pairs to be on the stack on entry, but if the stack is empty it defaults to 1000 cases. It must be followed by the name of the data file to use. It does not establish the scale factors—these must have been just calculated, or a previously generated set must have been read from disk.

TEACH-NETWORK contains four deferred words, and these must be pointed to the routines for the specific task in hand. The deferred words, and their functions, are:

DO-INIT
This may do any initialization of disk, display, or anything else required.

FORWARD-WORD
This performs one evaluation pass of the network, updating the internal activation of each output. It expects the inputs, already scaled, on the stack on entry. Once a network is trained, this is the basic word for using it in recall mode.

TRAIN-WORD
Given a set of scaled outputs on the stack, this word must calculate the errors in the outputs and back-propagate these to the network inputs, updating the weights as it goes. It leaves nothing on the stack.

DISPLAY-INFO
This is a word to display any information about the state of the network that may be desired. It is called after each training example.

With these words, TEACH-NETWORK will train for the specified number of cases or until a key is pressed. Figure One shows the code for the words described above. [See Errata on page 17; the Figure One referred to here was printed as Figure Eight in the last issue.—Ed.]

Example

Figure Two shows an example that uses TEACH-NETWORK, and Figure Three provides a suitable data file. Once again, the exclusive-or relationship is used so that the file is short, but any other could be substituted. The more complicated the relationship to be learned, the more neurons will be needed. If too few are used, the network either will not stabilise or will learn only the general features of the relationship and fail to learn the detail. Of course, this ability to generalize may be of considerable interest in some situations. The error display is very simple, just printing the error for the four examples per line; since there are four possible examples, only this provides a simple way to see how the network is behaving. Alternatively, a simple four-graph plotting routine would make trends easier to observe.

Conclusion

The words presented here extend those in part one. Together, they provide a toolkit for constructing non-linear feed-forward networks and for training them by back propagation. They demonstrate again the power of Forth to produce a special language for a task. Following the approach used here, words could simply be produced to implement other types of neural layers, such as self-organising Kohonen layers. By standardising on passing data between layers on the stack, networks of arbitrary complexity can be readily constructed, trained, and used.

**Figure Three.** A small sample data file (XOR.DAT).

```
0 0 0 \ input input output
1 0 1 \ input input output
0 1 1 \ input input output
1 1 0 \ input input output
```
Figure Two. An example using TEACH-NETWORK (NNDEMO2.SEQ).

```forth
2 n-in ! 1 n-out ! \ define number of network inputs and outputs
1 3 layer TOP
  top initialize \ top (output) layer
3 2 layer BTM
  btm initialize \ bottom (input) layer
clipping off

: --> ( -- )
  btm compute \ use stack data and evaluate the bottom layer
  btm get-outputs \ load bottom's output to stack
  top compute \ use bottom layer outputs, evaluate top layer
;

' --> IS FORWARD-WORD

: <-- ( On On-1 .... On-- )
  top calc-load-errors \ calculate and load the errors
  s0.3 top train \ train top layer, learning rate of 0.3
  btm load-errors \ load back propagated errors
  s0.8 btm train \ train bottom layer, learning rate of 0.8
  n-in @ 0 do 2drop loop \ lose back-propagated errors @ network inputs
  1 icount +! ; \ increment iteration count

' <-- IS TRAIN-WORD

: GET-ERROR ( n -- S# ) \ get nth output UNSCALED error
  dup get-oeadr 2@ \ get the internal error
  rot get-intact 2@ s'fn(x) s/ \ convert to external form
  o-s 2@ s/ \ and unscale it
;

: .ERROR \ show errors each set of inputs
  icount @ 1- 4 /mod swap 0= if \ time for a new error line?
    crlf ." Set # " 1+ 4 .r ." - errors "
  else drop then 1 get-error s. \ print abs error
;

'.ERROR IS DISPLAY-INFO

: MISC-INIT 0 icount ! ;

' MISC-INIT IS DO-INIT

\ EXAMPLE
  set-scales xor.dat \ ( Conversion scales computed ) crlf
  save-scales xor.sca \ ( Scales saved) crlf
  500 teach-network xor.dat \ ( Network teaching terminated)
  top save-layer xortop.lyr \ ( Top layer saved) crlf
  btm save-layer xorbtm.lyr \ ( Bottom layer saved) crlf
```
\ Four examples of simple artificial neural networks. Tim January 1990
\**********************************************************************************************************************************************************************************
comment:
These examples all use simple non-linear feedforward networks and are trained
by back propagation. For simplicity, they use very simple inputs. Examples
2 to 4 all learn to implement the exclusive or relationship, which practically
would never be generated this way! Needs BASICNN to have been loaded, the
line below will check and load this file if it is not already loaded.
comment:
\**********************************************************************************************************************************************************************************
( ==> ) NEEDS BASICNN.SEQ ( <=== )
\**********************************************************************************************************************************************************************************
new program

\ Example One, single layer learning two relationships.
1 2 layer TEST \ build network with 2 inputs and 1 output
: EX1 ( -- ) \ the word that puts example 1 together
  test initialize \ initialize layer
  1 icount ! \ initialize iteration counter
  clipping off \ clipping not required
  begin
    crlf ." Pass " icount @ .
    S0.5 S0.5 \ first use inputs of 0.5 and 0.5
    test compute \ do one forward pass
    test get-outputs \ output to stack
    2dup ." Output " s. \ print result we got
    S-0.5 2swap d- \ calculate error, result should be -0.5
    test load-errors \ load the error
    S0.8 test train \ update weights using a learning rate of 0.8
    2drop 2drop \ only 1 layer, backpropagated errors not needed
    S-0.5 S1 \ second use inputs of -0.5 and 1
    test compute \ do one forward pass
    test get-outputs \ output to stack
    2dup ." Output " s. \ print result we got
    S0.5 2swap d- \ calculate error, result should be 0.5
    test load-errors \ load the error
    S0.8 test train \ update weights using a learning rate of 0.8
    2drop 2drop \ don’t need the backpropagated errors
    1 icount +! \ bump the iteration counter
    key? if key drop exit then \ keep going until we get bored
    again

\ Example Two, two layers, XOR, standard input
1 3 layer test2 \ 1 output, 3 inputs The top layer
3 2 layer test1 \ 3 outputs, 2 inputs The bottom layer
defer docases \ the difference between examples 2 3 and 4
defer top-layer
defer bottom-layer
: One-pass ( correct-output input2 input1 -- error )
  test1 compute \ forward process the bottom layer
  test1 get-outputs \ get its outputs ready for the next layer
  test2 compute \ forward process the top layer
  test2 get-outputs \ get its outputs ready for the error calc
  2dup s. d- \ print actual output, compute error
load into test2 ready for training
update layer 2 weights, learning coeff 0.3
load into test1 ready for training
and in layer 1, learning coeff 0.8
lose errors backpropagated to inputs

: Ex2-docases
  S0 S0 S0 one-pass \ do 0 0 case
  S1 S0 S1 one-pass \ do 0 1 case
  S0 S1 S1 one-pass \ do 1 1 case
  S1 S1 S0 one-pass \ do 1 0 case

: (Ex2-4)
  1 icount ! \ initialize iteration counter
  clipping off \ clipping not required
  begin
    crlf " Outputs "
    docases
    ." after pass " icount @
    1 icount +!
    key? \ user want anything?
    if
      crlf top-layer .layer \ if so show them the layers
      crlf bottom-layer .layer
      key upc ascii Q = \ they want to quit?
      if exit \ if so get out of here
      else begin key?
        until key drop crlf \..until they tell us that we may go on
      then
      then
    again
  ;
  
  \ Example Three, two layers, XOR, scaled inputs and outputs
  : Ex3-docases
    S-0.5 S-1 S-1 one-pass \ do scaled 0 0 case
    S0.5 S-1 S1 one-pass \ do scaled 0 1 case
    S-0.5 S1 S1 one-pass \ do scaled 1 1 case
    S0.5 S1 S-1 one-pass \ do scaled 1 0 case
  
  \(Continued\ on\ next\ page.)
Example Four, two layers, XOR, special geometry, scaled inputs and outputs

1 3 layer test4  
1 2 layer test3

MOD-ONE-PASS (correct-output input1 input2 -- error)

4dup test3 compute  
copy the inputs, use one copy on test3

S test3 get-outputs  
get the output from the bottom layer

test4 get-outputs  
other copy + test3 output = input to test4

test4 compute  
get the actual output from the top layer

2dup s. d-  
print actual output, compute error

test4 load-errors  
load the error into the top layer

S test4 train  
update weights in top layer

> r 2drop 2drop > r>  
lose errors propagated back to the inputs

test3 load-errors  
load the error at output of test3

S test3 train  
now update bottom layer

2drop 2drop  
lose errors backpropogated to inputs

Ex4-docases

S-0.5 S-1 S-1 mod-one-pass  
do scaled 0 0 case

S0.5 S-1 S1 mod-one-pass  
do scaled 0 1 case

S-0.5 S1 S1 mod-one-pass  
do scaled 1 1 case

S0.5 S1 S-1 mod-one-pass  
do scaled 1 0 case

Ex4

test3 initialize  
initialize our...
test4 initialize  
...two layers

[!] test3 is bottom-layer
[!] test4 is top-layer
[!] ex4-docases is docases

(ex2-4)

Advertisers Index

FORTH Dimensions

FORML ........................................ 44
Forth Interest Group ................ 35, 37
Harvard Softworks .................... 8
Journal of Forth Application & Research ................ 6
Miller Microcomputer Services ...... 22
Next Generation Systems ............ 16
Offete Enterprises .................. 34
Silicon Composers .................. 2
A message from the President of the Forth Interest Group

President’s Letter

Organization, Issues, Actions, and Explanations.

Organization

Business Group
Summary of recent business meeting minutes for July 21, 1991 (full minutes of the Business Group are available upon request):

Membership dues (correction): In the last FD, I stated that the dues for the new student membership were $24 per year. That was one of the figures suggested, however the correct figure that was approved by the business group was a student membership of $18 per year.

ADC Reports: There are now 1511 FIG members to date. This is an increase from last year at this time. This turn-around has been a long time in coming, and I expect the upward trend to continue.

Board of Directors
The nomination process for the Board of Directors has taken place. According to the FIG by-laws, a nomi-

(Continued on page 26.)

Who would complain if Forth were required in their project?

Forth Interest Group
Statement of Change in Financial Position
April 30, 1990 to April 30, 1991

<table>
<thead>
<tr>
<th></th>
<th>$4/30/90</th>
<th>$4/30/91</th>
<th>Change*</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ASSETS:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current Members</td>
<td>2106</td>
<td>1831</td>
<td>-275</td>
</tr>
<tr>
<td>Current Assets:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Foothill Bank, Money Market</td>
<td>15,925.38</td>
<td>24,865.91</td>
<td>8,940.93</td>
</tr>
<tr>
<td>Foothill Bank, Checking</td>
<td>636.43</td>
<td>700.14</td>
<td>63.71</td>
</tr>
<tr>
<td>Pending Foreign Clearing</td>
<td>-102.00</td>
<td>51.67</td>
<td>153.67</td>
</tr>
<tr>
<td>Returned Checks Pending</td>
<td>0.00</td>
<td>72.00</td>
<td>72.00</td>
</tr>
<tr>
<td>FORML, Money Market</td>
<td>15,894.11</td>
<td>16,916.46</td>
<td>1,022.35</td>
</tr>
<tr>
<td>FORML, Checking</td>
<td>1,926.01</td>
<td>1,236.64</td>
<td>-689.37</td>
</tr>
<tr>
<td>Total Current Assets:</td>
<td>34,279.93</td>
<td>43,842.82</td>
<td>9,562.89</td>
</tr>
<tr>
<td>Inventory:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory at cost</td>
<td>34,147.53</td>
<td>26,601.17</td>
<td>-7,546.36</td>
</tr>
<tr>
<td>Total Inventory:</td>
<td>34,147.53</td>
<td>26,601.17</td>
<td>-7,546.36</td>
</tr>
<tr>
<td>Other Assets:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deposit, United Parcel Service</td>
<td>200.00</td>
<td>200.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Second Class Postal Account</td>
<td>156.31</td>
<td>192.41</td>
<td>36.10</td>
</tr>
<tr>
<td>Accounts Receivable</td>
<td>3,166.20</td>
<td>2,491.41</td>
<td>-1,031.10</td>
</tr>
<tr>
<td>Total Other Assets:</td>
<td>3,522.51</td>
<td>2,491.41</td>
<td>-1,031.10</td>
</tr>
<tr>
<td>TOTAL ASSETS:</td>
<td>71,949.97</td>
<td>72,935.40</td>
<td>985.43</td>
</tr>
<tr>
<td><strong>LIABILITIES:</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sales Tax due</td>
<td>33.89</td>
<td>35.58</td>
<td>1.69</td>
</tr>
<tr>
<td>FD Dues Alloc to future months FD</td>
<td>37,487.30</td>
<td>41,518.51</td>
<td>4,031.21</td>
</tr>
<tr>
<td>TOTAL LIABILITIES:</td>
<td>37,521.19</td>
<td>41,554.09</td>
<td>4,032.90</td>
</tr>
<tr>
<td>Financial Reserve:</td>
<td>34,428.78</td>
<td>31,381.31</td>
<td>-3,047.47</td>
</tr>
</tbody>
</table>

*Increase = +
Decrease = -
Ada Multiprocessor Real-Time Kernel

Hoyt A. Stearns, Jr.
Phoenix, Arizona

A fully preemptive, priority driven, multiprocessed real-time kernel is required in certain applications (process control, in this case). After evaluating several schemes, I decided that the Ada rendezvous method is simple, straightforward, and elegant. Ada multitasking will work on any configuration of multiple processors that can communicate, or on a single processor. A single task can be localized to a particular processor, or can be distributed among many.

The listing is the kernel for a system with three processors. In the actual implementation, one of the three processors is of a completely different type, which is handled by having two code fields in each Forth word. All processors share the same memory.

Having a processor of a different type added some complications in the listing not normally necessary—such as the constant WS for word size—since the processors were of different addressability.

The kernel works fine on a single-processor system, in which case all references to semaphores may be removed. Memory is allocated from the top down, starting

at the contents of variable RAMPTR. First the system variables are allocated, then task control blocks (TCB’s), as tasks are registered to the system with REGIS. Each TCB has three link fields, one for each processor, so a task may be registered to be scheduled among one to three processors.

Although this kernel implements the Ada rendezvous system, the syntax is, of course, different.

The Ada selective-wait structure is invoked in this system with the word SELECT (S bit_mask timeout_value entry_# TCB_addr - msg... status) where timeout_value is how many clock ticks to wait for a message on one of the open entries (ADA DELAY). The entry_# return parameter is which entry the message came in on, or the total number of entries on this task if it times out. The &msg is where to get data if this is an

Ada : in message, or where to copy the data if it is an Ada :out message.

The sender parameter is the address of the caller’s TCB, and is used as a parameter to the word RELEASE, which is equivalent to the Ada END_SELECT construct.

Ada entry calls are implemented here with the word CALL($msg...timeout_value entry_# TCB_addr - msg... status) where timeout_value is how many clock ticks to wait for receipt of the message before returning with #call_expired status. The other parameters are which

Ada’s multitasking works on multiple processors that can communicate, or on a single processor.
Improper structuring of tasks in a system may easily result in deadlocks. The word DEDLOK may be invoked periodically to break any deadlocks by returning the parameter #BROKEN to one of the CALL’S involved. This shouldn’t be necessary in a properly programmed embedded system.

There should be a MASTER task—registered to only one of the processors in the system—which registers the rest of the tasks, then calls slave tasks on the other processors, which start up their own schedulers.

Interrupts were handled in the target system by revectoring NEXT in the variable NP (next pointer) on a particular processor to high-level code to post an IXMIT message, then revectoring NEXT back to normal. NEXT may also be vectored for tracing.

There is an ancillary task, SYSSTAT, which takes a snapshot of the entire system and prints a report of task status, message queues, run-time statistics, etc.—but that is for another article.

### Ada listing.

**Scr # 1**

```ada
0 \ load screen real time kernel
1
2 decimal
3 variable ramptr sp0 @ 2000 - ramptr ! \ top of tcb space
4
5 (s d addr--)
6 : d+! dup r> 2@ d+ r> 2!;
7 : umin (s n1 n2--n3) 2dup u> if swap then drop ;
8
9 2 constant ws
10 4 constant dws \ word size=addr increments/word
11
12 vocabulary kernel kernel definitions
13 2 28 thru
14
15 \ forth definitions
```

**Scr # 2**

```ada
0 \ has063091
1 : access dup create , does> @ + ;
2 : sys dup create , does> @ ramptr @ + ;
3
4 \ receiver definitions
5 1 constant open 2 constant refus 128 64 + constant pused
6 64 constant used 32 constant rsemaphore
7 0 access rlist ws + access rstate ws + constant rcvrsz
8
9 \ task definitions
10 1 constant selbt 2 constant endobt 4 constant delbt
11 128 constant stkbt 64 constant suspbt 32 constant tsemaphore
12 16 8 or constant proc_lock
13 0 constant #call-ok 2 constant #broken
14 1 constant #refused 3 constant #call_expired 0 constant id
15
```

**Scr # 3**

```ada
0 \ has111690
1 0 sys event
2 dws - sys clock dws - sys start_time
3 ws 3 * - : up0 [ dup ] literal Id + ramptr @ + ;
4 ws - ramptr @ + constant first_tcb_top
5
6 0 up0 !
7
8 (s bits addr--): set tuck @ or swap ! ;
9 : reset tuck @ swap not and swap ! ;
10
11 (s word true false--flag)
12 : logic over or -rot xor and 0= ;
13
14 : up@ up0 @ ; : up! up0 ! ;
```

(Code continues.)
nating committee consisting
of two members of the Board
of Directors (John Hall and
Dennis Ruffer) was ap-
pointed. After a search for
candidates, three were se-
lected to fill the vacant po-
positions. They are Mike Elola,
current board member, Nick
Solntseff, southern Ontario
FIG Chapter; and Jack
Woehr. Since there were
insufficient write-in peti-
tions (25) for any other can-
didate, an election by ballot by
the membership is not required.
A unanimous vote for the
nominees will be cast by
the Secretary at the annual
meeting of the Board of
Directors in November.
These candidates will sub-
mitt personal statements in
the November-December
issue of *Forth Dimensions*.

**Issues, Actions and
Explanations**

Treasurer has prepared
review of FIG financial situa-
tion. The treasurer has
prepared, and the Board has
approved, the financial
statement presented in this
*FD*. In explanation, April 30
is the end of FIG’s fiscal
year and the figures reflect
those dates. The form is
similar to a “Statement of
Change in Financial Position”
changed to reflect the
year-ends for 1990 and 1991
with comparison. This form
shows Assets and Liabilities,
and the differences.
Most categories are clear,
with the exception of the
"FD Dues Alloc. to future
months *FD" and "Financial
Reserve." "FD Dues Alloc to
future months *FD" is money
received by FIG, budgeted
to complete a member’s year
of *Forth Dimensions* (this is
the amount FIG is obligated
to put aside to produce fu-

Scr # 4

```plaintext
0 \ low level
1 \ message definitions and call_frame definitions
2 0 access mlink ws + access dest access cdest
3 ws + access dest_rcvr access centry
4 ws + access sender access cdelay dup constant csize
5 ws + dup constant mdata constant msize
6 0 : link dup up0 <> if [ rot dup ] literal
7 + id + [ -rot ] then ;
8 ws 3 * + access nrcvr ws + access priority ws + access tstate
9 ws + access maxim ws + access actime dws + access urp
10 ws + access usurp ws + access usurp0 ws + access urp0
11 ws + access up ws + access unwp ws + access totime dws +
12 access spmin ws + access rpmmin ws + access splim ws +
13 access rplim ws + access tname 2 dws * + access rcvrs
14 ws + access _dp
15 ws + constant tcbase_size
```

Scr # 5

```plaintext
0 \ user functions
1 2 ($ rcvr_addr--) \ mlink must be 0 before free
3 : rsem_clr rstate rsemaphore swap reset ;
4 ($ rcvr_addr--) \ wait for semaphore, then set it
5 : ?rsem rstate begin dup @ rsemaphore and 0= until
6 rsemaphore swap set ; \ must be done under lock
7 : idlock ( semaphore to lock a processor's task list );
8 : idunlock ;
9 : di ( disable interrupts on this processor) ;
10 : ei ( enable interrupts on this processor) ;
11
12 : ?tsem ( test and set a task linking semaphore) ;
13 : tsem_clr ( clear tsemaphore) ;
14 : sys_atks ( set stack pointers to system area) ;
15 : 'next ( address of "next" in target for NP, the next vector);
```

Scr # 6

```plaintext
0 \ has062991
1 (S up -- f, true if activation time)
2 : ?actime actime 2@ ( di) clock 2@ ( ei) d- nip 0< ;
3
4 (S start_addr number item_size--item_size, +or- end a start a)
5 : <b dup >r * dup >r over + 1 r> ?negate -r > r -rot swap ;
6
7 (S up offset -- size end start) \ setup loop over receivers
8 : rcvrs over rcvrs + swap nrcvr @ rcvrsz <b ;
9
10 (S up bit_mask--)
11 : sbclr swap 0 rstate
12 : rbnds do over i reset dup +loop 2drop ;
13 (S rcvr_addr--flag, open with msg)
14 : ?msg dup dup >r ?rsem dup rstate @ open and 0<>
15 : swap rlist @ and r> rsem_clr ;
```
Scr # 7
0 \ has062991
1 (S data up--)
2 : tpush usp dup @ ws - dup rot ! ! ; \ push parm to up task
3
4 (S up rcvr_addr--rcvr_number)
5 : ra>r# swap rcvrs = rcvrsz / ;
6
7 (S up rcvr_number--rcvr_addr)
8 : r#>ra rcvrsz * swap rcvrs + ;
9
10 (S rcvr_addr--)
11 : rcv_msg dup dup ?rsem rlist dup di @ tuck @ swap ! ei
12 -1 swap ! ( rec'd) on rsem_clr ;
13
14 (S up rcvr_number--rcvr-addr)
15 : rcv>msg rlist @ ;

Scr # 8
0 \ has062991
1 (S rcvr_addr--)
2 : mark_used rstate pused swap set ;
3
4 (S msg addr--)
5 : clr_sender_delay sender @ tstate delbt swap reset ;
6 (S &tstate--tstate flag) \ True if not...
7 \ ...suspended, tsem=0, stk ok
8 : ? runnable dup @ swap ?tssem over [ stkbt suspbt or ] literal
9 and or 0= ;
10 (S --up')
11 : robind up@ dup link @ swap priority @ over priority @
12 <> if drop up0 @ then ;
13
14 : setnxt (S up--) tsemaphore over tstate ! di up! ei
15 clock 2@ start_time 2! ;

Scr # 9
0 \ has112590
1 (S rsize end_addr rcvr_addr--rsize end_addr rcvr_addr',false)
2 : find_msg >r >r r0 swap false swap r> r> do i ?msg if
3 nip i swap leave then dup +loop -rot ;
4
5 (S up msg addr--)
6 : msgexi 2dup sender @ swap tpush 2dup mdata + swap tpush
7 dest_rcvr @ swap tpush ;
8
9 (S up rcvr_addr--)
10 : gotsy over open sbclr dup mark_used dup rcv>msg
11 swap rcv_msg dup clr_sender_delay msgexi ;
12
13 (S up--data)
14 : tpop usp dup @ @ ws rot +! ;
15

Forth Dimensions 27

Public relations: For the last several months, the focus of the Business Group has been on publicity for Forth and FIG. There are several lines of attack that we are preparing to take. In the past, our approach has been to try to encourage individual programmers to look at Forth and, when they do, to use FIG as a resource. We are shifting the emphasis of exposure toward what I call, for lack of a better term, the "mid-level manager"—the people who control the direction of projects, who have some control of a budget, and who want to get the most product for the money they control. The emphasis will be on the innovation that is going on in Forth, the fact that there are people ready and available to help with their projects, and the wealth of information that is already available. These are the exact people about whom I have heard the complaint, "If only these people understood what could really be done with Forth, my life would be a lot easier." These are the people that I have heard complain, "I would like to use Forth, but where am I going to find enough of the right people to produce and maintain my project?" Two specific types in this group, to give you a better feel of
what I mean by mid-level manager, are project leaders, at all levels of industry; and researchers in R&D organizations, whether universities or R&D divisions of larger companies.

How do we entice them? The primary way will be with articles in all appropriate trade publications about the innovative ways that Forth is being used to solve current problems. Highlighted will be individuals or teams or applications or Forth vendors' products or standards or places where Forth already is used but the fact is generally unknown. The emphasis will be on innovative ideas and innovative people!

Second, by volume we let them know that they have always been surrounded by Forth and that the idea of using it is not novel—in fact, that it is required and that there have always been people near them who are willing to help.

Third, that FIG is one of their conduits for information about Forth, whether it is for literature, training, people, or direction toward vendor products. We will emphasize all the Forth resources available to make them and their projects successful.

How do we do all this? This is a multi-path attack, and one of the places it starts is with you!

1. If you will soon be writing a technical article about your project, you will see an article in the next *FD* about the style and approach that FIG would like you to add to your article. Horace Simmons has been compiling a list of the publications that will be the vehicles for this endeavor.
2. If you are on a project that is new and interesting, and you are interested in letting others know and can write about the technical parts, we will find a writer that can reshape it into an article that will complement the technical aspects with general-interest aspects of Forth and will help find an appropriate place to get it published.

3. As articles are published, we will coordinate Forth- and FIG-related advertisement to be placed in those issues along with the articles.

Why are we emphasizing mid-level managers? They are the people who specify or can direct the technical details of a project. They are the people who put together a project team. They are the people who would most likely pale at new innovation without justification. They are the people who make a project succeed or fail. They are the people we have failed to reach. Who of us would complain if Forth were required in your project?

I am always available for comments.

---John Hall
415-535-1294
JDHALL on Genie
CREATE, CREATE DOES>, and, in some Forths, also VARIABLE, will need to be recoded. As of today, if you are using Forth and you want your application to be compatible with future hardware and system software, you have to refrain from using some very natural Forth mechanisms. This is all very un-Forth-like!

Forth could and should, if only some more people bothered, evolve on the Macintosh. There is no reason why the vast programming utilities of MPW, its many tools, its great multi-scrollbar editor, could not become available to the Forth environment. I myself am depending more and more on the MPW environment and, yet, it still lacks the one thing taken for granted in any Forth environment: interactivity—the kind you get by being able to compile small entities and immediately execute and debug them.

I think that if Forth has proved to be a disappointment on the Mac, one has only oneself to blame. With Forth's simple and open architecture, the compiler is not the limit, the user is.

Conrad Weyns
Bjerkebakken 62D
0756 Oslo 7
Norway

U.K. Contest Winners

Dear Marlin,

I have enclosed two of the winning entries in the Forth Programmer competition [see "Letters," last issue—Ed.], along with some comments from the entrants.

---

Scr # 16

```
0 \ has062991
1
2 (S stack_size pgm_size #rcvr up--next_up)
3 : set_stacks dup >r swap
4 r#>ra dup r> swap r> r + 80 + tuck + 140 + dup r8 - 1+ r8
5 swap erase tuck dup r8 urp0 ! r8 urp ! r8 splim ! 100 - dup
6 r8 rplim ! 20 - dup r8 usp !
7 r8 usp0 ! r> _dp + r> swap ! ;
```

Scr # 17

```
0 \ Link in a new task has111390
1 (S up0--&predecessor)
2 : find_predecessor dup >r begin dup link @ dup r8 <> while
3 nip repeat r> 2drop ;
4
5 (S &pred up0 &new &link--&pred up0 &new &link link flag)
6 : pri?< 2dup link @ dup priority @ rot priority @ u< ;
7
8 (S &pred up0 &new &link link--)
9 : do_links rot tuck link ! swap dup >r link !
10 @ swap dup r> = if 2drop else link ! then ;
11
12 (S &new up0 up0--)
13 : link task dup @ ?dup if link next else over dup
14 2swap pri<?, not if begin nip pri<?, over r8 = or
15 until then r> drop do_links ;
```

Scr # 18

```
0 \ New task linking cont'd. has111390
1
2 (S &new up0--)
3 : link_task dup @ ?dup if link_next else over dup
4 link ! ! then ;
5
6
7 \ Tcb linking test words
8
9 (S priority--)
10 : doit here swap , 0 , up0 link_task ;
11
12 (S --)
13 : sl up0 @ dup >r begin cr dup u. link @
14 dup dup u. r8 = key? or until r> 2drop ;
```
As an update to the report I sent, I would like to confirm that the discussions with IBM were successful, and they will be hosting our next London meeting. Incidentally, they have been most generous in supplying not only space in their South Bank Lecture Theatre, but also a buffet meal. Perhaps other FIG Chapters would be interested to know that there is some benefit to be made from approaching large corporations with regards to meetings. We are currently trying to encourage them to attend euroFORML.

The FANS1 project is proceeding apace, and initial circuit diagrams have been drawn up for our planned processor board. We have actually settled on a 6309 processor, which is a CMOS 6809. This was chosen for its low cost and availability, and for the orthogonality of its instruction set and direct support for two stacks. It is possible that, in the future, we will produce a second board using a stack processor, very possibly the German device designed by Klaus Schleisiek-Kern, incidentally the organizer of this year's euroFORML.

Yours sincerely,
Gordon Charlton
Events Secretary, FIG-UK
31 Pikestone Close
Hayes, Middlesex UB4 9QT
Great Britain

I have been using Forth since 1978. Most of the professional work I have done has been embedded software in traffic monitoring equipment for an outfit called

Forth Dimensions 31 September 1991 October
Golden River. More recently, I wrote an implementation of Forth-83 for the Transputer. I started working on the Matching Birthdays problem as a result of Gil's blatant provocation in asserting it was a formidable calculation. My first attempt performed the entire calculation on the stack and ran to six screens of source code. My general thoughts about Forth are that it is a good thing.

Big Number Arithmetic and the Probability of Matching Birthdays

In the editorial of Forth-write 56, Gil gave the formula for calculating the probability of finding at least one pair of matching birthdays in a group of \( n \) people as \( (1 - P) \), where \( P \) is found by multiplying all the integers in the range \( (366 - n) \) to 365 inclusive, then dividing by 365 \( n \) times.

To obtain four decimal places using integer arithmetic, we must multiply by 10,000 and subtract the result from 10,000. All the multiplication must be performed before the division, giving rise to very big intermediate values. It is only necessary to multiply and divide by single-length numbers, so the algorithms for "short multiplication" and "short division" are used; the product or quotient is written, cell-by-cell, back to the same memory locations that held the corresponding cells of the multiplicand or dividend.

It is interesting to compare the results of this program with those which Gil obtained using the Monte Carlo method. The big num-

```fith
Scr # 22
0 \ has111590
1
2 \( S \) up--
3 : enable dup >r begin dup tstate suspb t swap reset
4 link @ dup r@ = until r> 2drop ;
5
6 \( S \) ****
7 \( S \) --
8 : startup id " MASTER" drop 1- find 0= abort" No Master!"
9 255 0 30 0 2 first_tcb_top
10 regis MASTER
11 up@ tpush up@ enable sched ;
12
13
14
15
```

```fith
Scr # 23
0 \ has112090
1
2 \( S \) receiver_specifier 'set or 'reset bit--
3 : sr_all rcvr rot up@ 0 rstate rbnds rot drop do
4 3dup 1 and * i rot execute u2/ rcvrsz +loop 2drop drop ;
5
6
7
8
9
10
11
12
13
14
15
```

```fith
Scr # 24
0 \ has112090
1
2 \( S \) delay_parameter 0|selbt --
3 : setdel swap dup if dup < if >r up@ actime 2@ r> -1 d-
4 else 0 clock 2@ d+ then up@ actime 2! delbt then
5 or up@ tstate set ;
6
7 \( S \) rcvr_specifier--
8 : open_rcvr ['] set open sr_all rcvr ;
9
10 \( S \) receiver_specifier delay_specifier -- &sender &data rcvr#
11 : select
12 enter 2 trash_parms selbt setdel open_rcvr sched ;
13
14 \( S \) delay_specifier -- Dclock
15 : delay enter 1 trash_parms 0 setdel sched ;
```
Scr # 25
0 \ functions  has063091
1.
 2 (S sender--flag) \ true if msg not from interrupt (ixmit)
 3 : ~ix_msg tstate @ [ delbt not ] literal <> ;
 4
 5 (S sender--) \ ADA end of critical section
 6 : release enter 1 trash_parms dup ?-ix_msg if #call_ok
 7 over tpush then tstate endobt swap reset sched ;
 8
 9 (S rcvr_specifier--)
10 : busy
11 enter 1 trash_parms ['I set refus sr_all_rcvr sched ;
12
13 : unbusy
14 enter 1 trash_parms ['I reset refus sr_all_rcvr sched ;
15
Scr # 26
0 \ Ixmit  has063091
1
 2 (S entry_# dest_task &msg--status) \ send msg frm interrupt
 3 : ixmit >r 2dup swap r#>ra dup ?refused
 4 if r> 2drop 2drop #refused
 5 else rot r# dest_rcvr ! ( dest_rcvr)
 6 swap r# dest ! ( dest) r#
 7 [ 0 tstate 0 mlink - ] literal - r#
 8 sender ! ( pseudo-sender) r> m_link #call_ok then ;
 9
10 \ Message structure: link, dest, dest_rcvr#, 
11 pseudo_sender, data... mlink is set
12 to -1 and ~delbt (fffb) when rec'd. When released
13 endobt is cleared (fff9).
14 The sender field is set so that the 
15 mlink field appears as tstat

Scr # 27
0 \ break  has062991
1 (S up--up f)
 2 : ?breakable dup dup usp @ mdata - mlink @ 1+ swap
 3 tstate @ endobt delbt logic and 0<> ;
 4
 5 (S up|false--up'|false)
 6 : >dest
 7 dup if ?breakable swap usp @ mdata - dest @ and then ;
 8 (S up-- up'|false)
 9 : bchase dup dup >r begin over >dest 2dup <> dup
10 if drop rot drop false -rot over r> link @ dup >r <>
11 over and then while rot drop swap
12 repeat 2drop r> drop ;
13
14 : ppad (S --addr) _dp @ 80 + ;

---

Forth Dimensions 33
September 1991 October

One-Screen
Full-Screen Editor

Our company, M.A.S.S., operates as a bespoke software house working with a wide variety of projects, 90% of which are now Forth based.

Between jobs, I get involved in what can loosely be called R&D. This can range from developing software tools (tinkering with Forth), looking at ways to improve efficiency (making a case for the latest hardware), and exploring software techniques (playing games).

Seriously, though, we started to move over to Forth in about 1985, after studying the language (during R&D) and comparing it to the languages used at the time, i.e., BASIC, Pascal, and assembler. The benefits gained came slowly, at first, but then at an accelerating rate as experience was gained and the various "pennies dropped."

We tried several commercial versions of Forth, with varying levels of success, before developing homegrown versions. This gives us absolute control to customize as required. Forth makes you greedy!

One of the many benefits Forth gives us is the ability to very quickly produce prototypes. While our competitors are busy drawing up impressive-looking flowcharts and "t specs," we are demonstrating some sort of prototype to which the client
usually can relate. This has worked very well for us. (As a point of interest, we have had over 12,000 Forth applications distributed worldwide.)

Another R&D session looked long and hard at C. I obviously missed something. All that syntax!

Yet another period of R&D produced the one-page full-screen editor. We were working on text manipulation at the time, so I lifted some of the code and worked on minimizing it.

I would not have believed that any sort of screen editor would have fit into 1K of source code—"packed jumble" or not—especially as the editing facilities offered are quite respectable. The average words per definition worked out to fewer than six, and only an improvement to the word TYPE makes any significant improvement to performance.

I found the competition a worthwhile exercise and, indeed, have imposed the one-screen restriction on several pieces of code since. This approach forces a re-think and a re-work of just about every word used. Perhaps Gordon Charlton has unwittingly opened up a whole new programming technique? I cannot, however, recommend the regular use of the style used in my example! [See Figure Two on page 36.]

—Mike Lake

PDE Erratum

Dear Mr. Ouverson,

In my article on PDE screen management ("Add and Delete Screens in PDE,

(Continued on page 37.)

September 1991 October

Scr # 28

0 \  has063091
1 \ Break a task deadlock
2 (S up_to_resume--flag, true if broken)
3 : break _ dup bchase ?dup if
4 over cancel_msg
5 usp @ [ 0 dest_rcvr mdata - ] literal + @ over tpush
6 #broken over tpush
7 tstate endobt swap reset true else drop false then ;
8
9 (S --)
10 : dedlok up0 @ dup >r begin dup break drop link @ dup
11 r@ = until r> 2drop ;
12
13
14
15

(Editors, from page 4.)

Forth (50%)
Program design
Stack operations
Postfix notation
Forth compilation (definitions, conditionals, loops)
Number types and operations
Constants, variables, arrays
Dictionary structures
I/O operations
Control structures

3. Forth Programming (20%)
4. Forth and Professional English (10%)

Conscientious Computing

Omni magazine's May 1991 issue contains a guide to energy-efficient behavior, something many technophiles appreciate as a kind of self-preservation. Among their suggestions related to office work: use machines with controls that keep them from running at full power during work lulls; save your laser printers for output that demands their quality output (they require ten times as much energy as other printers); use a fax without a thermal print mechanism; use laptop computers, which are more efficient than desktop models; use smaller screens; use recycled paper; and carpool or, better yet (take my word for it), telecommute.


—Marlin Ouverson

Editor

eForth Implementations

2104 eForth Model and 8086 Implementation, Bill Muench and C. H. Ting—$25.00
2105 8051 eForth, C. H. Ting—$25.00
2107 68000 eForth, Richard H. Haskell—$25.00
2108 32-Bit 8086 eForth, R. VanNorman—$25.00
2109 Z80 eForth, Ken Chen—$25.00
2110 Subroutine Threaded eForth for 68K, 68HC11 and 8086, Richard Haskell—$25.00

New F-PC Releases

2102 F-PC V3.55 and TCOM, Tom Zimmerman—$25.00
2106 The Forth Course, Richard H. Haskell—$25.00
2111 F-PC User Contributions, 80 files—$25.00

Others

1002 Forth Notebook, C. H. Ting—$25.00
1003 Inside F83, C. H. Ting, 3rd Edition—$25.00
1011 Forth Notebook, Volume 2, C. H. Ting—$25.00
2103 LaForth, LaFarr Stuart and R. L. Smith—$25.00
4004 Indeliko RTX2001A Kit—$150.00
4008 RTX2001AJC-8 Chip—$68.00
4113 More on Forth Engines, Volume 14—$15.00
9006 Neural Computing Module NCM3232—$85.00

Offete Enterprises, 1306 South B Street
San Mateo, CA 94402 (415) 574-8250

Forth Dimensions
Attention Forth Authors!

AUTHOR RECOGNITION PROGRAM

To recognize and reward authors of Forth-related articles, the Forth Interest Group has adopted the Author Recognition Program.

Articles
The author of any Forth-related article published in a periodical or in the proceedings of a non-Forth conference is awarded one year's membership in the Forth Interest Group, subject to these conditions:

a. The membership awarded is for the membership year following the one during which the article was published.
b. Only one membership per person is awarded in any year, regardless of the number of articles the person published in that year.
c. The article's length must be one page or more in the magazine in which it appeared.
d. The author must submit the printed article (photocopies are accepted) to the Forth Interest Group, including identification of the magazine and issue in which it appeared, within sixty days of publication. In return, the author will be sent a coupon good for the following year's membership.
e. If the original article was published in a language other than English, the article must be accompanied by an English translation or summary.

Letters to the Editor
Letters to the editor are, in effect, short articles, and so deserve recognition. The author of a Forth-related letter to an editor published in any magazine except Forth Dimensions is awarded $10 credit toward FIG membership dues, subject to these conditions:

a. The credit applies only to membership dues for the membership year following the one in which the letter was published.
b. The maximum award in any year to one person will not exceed the full cost of the FIG membership dues for the following year.
c. The author must submit to the Forth Interest Group a photocopy of the printed letter, including identification of the magazine and issue in which it appeared, within sixty days of publication. A coupon worth $10 toward the following year's membership will then be sent to the author.
d. If the original letter was published in a language other than English, the letter must be accompanied by an English translation or summary.
Figure Two. One-screen full-screen editor.

------

ONE SCREEN FULL SCREEN EDITOR

Mike Lake

0 VARIABLE KT 512 ALLOT : Y R@ @ C/L / ; : X R@ @ C/L MOD ;
: RANGE OVER + DUP 0< OVER 1023 > OR 0= IF SWAP THEN DROP ;
: MOV R@ @ SWAP RANGE R# ! ; ; LEFT -1 MOV ; ; RIGHT 1 MOV ;
: UP -64 MOV ; ; DOWN C/L MOV ; ; HOME 0 R# ! ; ; >E C/L X - ;
: LO SCR @ (LINE) ; ; AD 0 L0 DROP R# @ + ; ; LAD Y L0 ;
: Y L0 OVER AT L0 TYPE ; ; .L Y Y .L ; ; CEOL AD >E BLANKS .L ;
: ED 16 0 DO I Y.L LOOP CR SCR ? ; ; CRLF X MINUS R# +1 DOWN ;
: CL LAD BLANKS .L ; ; >S UPDATE SCR +! .ED ; ; N 1 >S ;
: PUT LAD PAD SWAP CMOVE ; ; !SCR HOME AD 1024 BLANKS .ED ;
: GET PAD LAD CMOVE .L ; ; TOGG KT 1 TOGGLE ; ; P -1 >S ;
: CLR BL AD >E 1- 2DUP OVER 1+ ROT ROT CMOVE + C! .L ;
: !K DUP AD KT @ IF DUP DUP 1+ >E 1- CMOVE C! .L ELSE C! DUP

THE CONFINES OF ONE SCREEN RESULTS IN THE PACKED JUMBLE ABOVE

EMIT THEN RIGHT ; ; DEL LEFT CLR ; ; UNDO EMPTY-BUFFERS .ED ;
: SET CFA SWAP 2* KT + 2+ ! ; ; SETS 256 0 DO 1' :K SET LOOP ;
: GO BEGIN X Y AT KEY DUP 2* KT + 2+ @ EXECUTE 0< UNTIL ; SETS
: ED FLUSH SCR : HOME .ED GO UPDATE ; ; TAB 8 X OVER MOD - MOV ;

CORE WORDS

---

KT ( ...ADDR) FIRST 2 BYTES=FLAG 256*2 BYTES FOR KEY ACTIONS
X ( ...N) X POSITION OF CURSOR
Y ( ...N) Y POSITION OF CURSOR

RANGE ( CURSOR N ...CURSOR) LEAVES CURSOR+N OR ORIGINAL CURSOR POS
MOV ( N ... ) MOVES CURSOR BY N
>E ( ...N ) CHARACTERS FROM CURSOR TO END OF LINE
LO ( ... ADDR C/L) ADDRESS OF START OF EDITING LINE N
AD ( ... ADDR) ADDRESS OF CURSOR
LAD ( ... ADDR C/L) ADDRESS OF START OF CURRENT LINE
Y.L ( ...N) LOCATE AND PRINT LINE N
.L ( ... ) PRINT CURRENT LINE
.ED ( ... ) PRINT EDITING SCREEN
>S ( N ... ) MOVE ON N SCREENS

GO ( ... ) MAIN KEY LOOP ;
ED ( N ... ) INVOKES EDITOR USING SCREEN N
SET ( N PFA ... ) SET KEY N TO PERFORM WORD ACTION

SETS ( ... ) SETS ALL KEYS TO DEFAULT... (STORE KEY PRESS)

---

WORDS TO ASSIGN TO REQUIRED KEY PRESS

---

CLR ******************** = CLEAR CHARACTER UNDER CURSOR
DEL * ALL THESE WORDS ARE * = CLEAR CHARACTER TO LEFT OF CURSOR
CEOL * ASSIGNED TO KEYPRESSES.* = CLEAR FROM CURSOR TO END OF LINE
CL * ADDING EXTRA ACTIONS * = CLEAR LINE
:SCR * IS A DOODLE. A NEW * = CLEAR EDITING SCREEN
UNDO * ACTION HAS THE KEY * = RESTORE EDITING SCREEN TO ORIGINAL
N * AVAILABLE ON THE STACK * = NEXT SCREEN
P * AND MUST LEAVE A VALUE * = PREVIOUS SCREEN
PUT * ON THE STACK. A MINUS * = PUT CURRENT LINE TO PAD
GET * VALUE SIGNALS TO * = GET CURRENT LINE FROM PAD
TOGG * EXIT THE EDITOR. * = TOGGLE INSERT/OVERWRITE MODES
!K ******************** = STORE KEY PRESS (DEFAULT ACTION)
MINUS = FORTH WORD USED TO EXIT EDITOR

LEFT RIGHT UP DOWN HOME CRLF TAB = MOVE CURSOR

---

INSTRUCTIONS .. THE ABOVE EXPECTS THE FOLLOWING COMMON FORTH WORDS.
-CMOVE ( ADD1 ADD2 COUNT ...) AS CMOVE BUT FROM HIGH TO LOW MEMORY.
AT ( X Y ... ) LOCATE CURSOR AT X Y.
LOAD THE SCREEN AND ASSIGN ACTIONS TO THE REQUIRED KEY PRESSES.
13 ' CRLF SET 8 ' TAB SET 27 ' MINUS SET ETC.
THE CONFINES OF ONE SCREEN RESULTS IN THE PACKED JUMBLE ABOVE
HOWEVER DEFINITIONS ARE VERY SMALL AND EASY TO FOLLOW.
in my PDE uses a non-zero \texttt{BLK} as a flag that the error occurred during a load, so it enters the editor. This causes the system to crash.

However, in setting up the Forth command line, \texttt{QUIT} early on resets \texttt{BLK} to zero, so an error there displays the error message normally.

You can solve the problem at several levels.  
1. Simplest is to redo the \texttt{SAVE-SYSTEM} from the Forth command line. This ensures that \texttt{BLK} is zero.  
2. You can rewrite \texttt{START} to include \texttt{0 BLK} ! before calling \texttt{DEFAULT}. It’s easy to patch in, as \texttt{BOOT} is a deferred word. Then, future additions to the program using this load screen to save the system won’t cause problems.  
3. Ultimately, correcting \texttt{DEFAULT} to reset \texttt{BLK} and recompiling would guarantee that this gremlin won’t return to haunt you.

Yours truly,  
Walter J. Rottenkolber  
P.O. Box 596  
Visalia, California 93279

\textbf{Answer to a Dream}  
Dear Sir,  

In answer to John G. Derrickson’s letter requesting a low-cost, better Forth (“Dreaming That It’s Forth,” \textit{FD XIII/2}), I would like to recommend Upper Deck Forth, available from Upper Deck Systems (P.O. Box 253342, Escondido, California 92026).

I have found it to be exceptionally fast and easy to use. It is a text-based system and incorporates such features as mouse control for the full-screen editor, direct access to DOS functions from the terminal input stream, full memory usage, disassembler, assembler, and support for generating headerless, turnkey applications during compilation, so determination of the exact location of the interrupt routine was made more difficult. This was solved by readjusting the code within the interrupt routine after compiling.

I highly recommend this MS-DOS Forth system.

Sincerely,  
Glen F. Ingle  
1585 Samedra Street  
Sunnyvale, California 94087

\begin{center}$\$\text{Contest Announcement}\$\$

\textbf{Call for Papers!}

\textit{Forth Dimensions} is sponsoring a contest to encourage authors of articles about Forth and \textit{“Object-Oriented Programming”}

\begin{center}
\textbf{1st prize:} \quad $500  
\textbf{2nd prize:} \quad $250  
\textbf{3rd prize:} \quad $100
\end{center}

Deadline: September 16, 1991
See editorial in issue XIII/1 for details!

\end{center}

\textbf{I would not have believed that any sort of screen editor would have fit into 1K of source code.}
In the February column I promised not to delay so long between recaps of guest conferences. Before making good on that promise, I want to call your attention to the improved appearance of the GENie Forth RoundTable. Topics and entire categories are much cleaner. The rat's nests of messages, many of which my ForthNet message ports created, are largely gone. All this can be attributed to our newest SysOp, Elliott Chapin. Elliott came on board knowing he would have to pick up after the likes of me, so he is either one heck of a good guy or a glutton for thankless jobs. I happen to believe the "nice guy" theory will stand the test of time.

Here, in his own words, is Elliott's introduction: "I was born in 1942 in New York City, just five days before my better known half-nephew, Harry. A couple of years later, my father James Chapin, an established artist, moved the family out to rural northwest New Jersey. Now I have a family of my own in Toronto. I studied math at Princeton (starting my acquaintance with computers there) and Columbia in the 60s. Since then, I have found various kinds of work in the arts and education, but I also drive a taxi when necessary. I am enjoying the learning opportunities available on the GENie's Forth RoundTable, while doing "slash and stash" messagebase edits as the latest assistant SysOp." (Note: ELLIOTT.C is Elliott's e-mail address.)

Welcome to the funhouse, Elliott.

It is true that revisits with our guests in this column never capture the intimacy of the Real-Time Conference, where you pose the question, usually sparked by another comment, but they do serve to remind attendees of poignant exchanges and to let non-participants know what they missed. Transcripts of the complete conferences are archived in the GENie Forth RoundTable's Software Library 1. Please make a note, as you scan this column, to download those you find of value, because the conferences as presented in this column are reduced to the guests' opening remarks.

Guest conferences reviewed here include Jef Raskin, "What Happened to the Cat?" Alan Furman, "ACM SIGForth Update"; Bill Muech, "Embedded with eForth"; Guy Kelly, "Forth and Industry"; Dean Sanderson, "Addressing Management Concerns" (regarding use of Forth); Roy Martens and Glen Haydon, "MVP is Alive and, well..."; Charles Johnsen, "Mutable Instruction Set Computers."

Jef Raskin, creator of the Mac and Canon Cat, asked "What happened to the Cat?" in reference to the marketing failure of the Canon Cat. This initial discussion rapidly gave way to a more thorough look at what the interface between computer and man should be—certainly an area, it can be safely argued, in which Jef Raskin is one of the ranking authorities, if not the authority. 10/17/90

Jef Raskin: I am pleased to be here, and I think that it might be best to just start with questions. When I see the directions the questions take, I can make more extended comments.

Cool CATJAX: First I were a Programmer... Now I am a Project Engineer... Jef, how long do I have to wait before I start having Ideas? :-)

Jef Raskin: Thank you Mr. Cat. My real interest these days is in interfaces. For example, windows are dumb, icons wrong, and mouses a nuisance.

Dennis Ruffin: Stick to that heresy, let me ask how/why did you start the Mac project?

Jef Raskin: I can't stand the usual hand-to-mouse existence. The real question is why we have to bother with all those "features" when we are trying to get something done. I started the Mac project in 1979. I have learned something in the intervening decade. In those days I was (correctly, I think) inspired by the work at PARC. But one mustn't confuse "better" with "good." The Mac was better, but one can go a lot better.

WilBaden: Please tell us what you think happened to the CAT.

Jef Raskin: It was, I have been told, a victim of internecine warfare within Canon and a lack of marketing support. I am sure of the second hypothesis.

Gary Smith: There was excitement here after John Bumgarner was in conference with Steve Roberts about the possibility of a portable, personal Forth laptop then bing0 the CAT is on sale in Service Merchandise. That's a lot of smoke and mirrors. Any further comment?

Jef Raskin: They are, by the way, still available. A guy, David Wing, in San Diego has kept track of them. Canon did not know what it had, and tried to sell it through its electronic typewriter division. This was a mistake. Then they tried selling it for twice the design price. That was a mistake. We did make some prototype portables, I have one here and it works (and runs Forth), but our Vulture Capitalists had lost their stomachs (and minds).

Dr. Alan Furman, independent software and electronics consultant, presented an
update on a sister organization, SIGForth ACM. As a principle of the group's genesis, Dr. Furman was more than qualified to do so.
11/15/90

I should begin by acknowledging the tremendous contribution of George Shaw, who has been the driving force in SIGForth from the beginning. My main role was as a crusader, in late 1988, for some kind of organization that would specialize in professional issues and the commercialization of Forth. FIG's constituency is a mixture of professionals and hobbyists. And many of those professionals didn't care who admired Forth; they were in a position of having discretion over programming language. Whenever you have a group of people together, you have to stick to things that they are all interested in at the same time, which in FIG's case is Forth technology and sharing of ideas.

I will close with some exciting news. The Forth community in Leningrad would like to put on a conference either Fall 1991 or Spring 1992, which will be run in cooperation with SIGForth.

Bill Muench, president of Ontologic and coauthor of eForth, discussed eForth, the .asm Forth kernel for the 90's. 12/13/90

eFORTH is derived from my commercial version bFORTH. Both are ANSI Forth subsets. I designed eFORTH for ease of implementing new systems, to easily fit in 8K bytes, to have a minimum of machine code and some debugging tools. My debug tools are TX!, the transmit primitive, S, DUMP, and WORDS. I use a simple metacompiler rather than MASM, which means I first write an assembler. That is how I get to know the processor.

After the new system is running, I optimize it, first loading the assembler, then new code words. When working, they are added to the kernel, repeat 'til done.

I use my host file server, bHOST, to upload source text files to the target system. MASM was chosen by Dr. C.H. Ting as a vehicle to communicate eFORTH. It is commonly available and on a widely used platform, the PC. But any assembler will do. Possible alternatives are spreadsheets or word processors with macros.

Guy Kelly expanded on his November 1990 FORML theme, "Forth in Industry." Guy has a vast array of Forth-driven industrial projects to his credit to support his views that Forth is an excellent platform for such endeavors. 1/17/91

Experiences as a Forth producer, user, and teacher convince me that it's almost impossible to get a professional programmer to use Forth and almost impossible to discourage a professional engineer from using Forth.

Now that Laxen, Perry, and Zimmer have handled the "high end" and Dr. Ting is attacking the "low end," how about the "Forth as a hardware development tool" area?

Dean Sanderson, software engineer with FORTH Inc., discussed "Addressing Management Concerns over use of Forth as an Applications Platform." 2/21/91

Because of the power that Forth has given us (to keep projects small and quick), we have been able to avoid learning what others have had to about software development. Those who have grown up with Forth do management by intuition. We have trouble communicating with those who've been successful using Fortran, C, or assembler. It's as if we speak different languages. As projects escalate, we find we have not killed the dragon, only maimed him. As we ready for battle, we find our pride has left us with few new weapons.

For Forth to survive as a respected language, it must prove its adaptability and change enough to support the concerns of management. These include: Integration, Maintenance, Documentation, Declining cost, Q.A., Configuration, and Scheduling.

Though we've started late, we can survive by capitalizing on what others have learned.

This conference's invited co-guests were Roy Martens, president of Mountain View Press, with Glen Haydon, author of MVP Forth. Glen discussed the fate of MVP in the 1990s market. 3/4/91

Thank you for inviting Mountain View Press. We are alive, but changes are in store. Roy has sold his home and is moving to San Francisco. He is taking an apartment April 1. We have agreed for me to take over Mountain View Press. As some of you know, Phil Koopman, has taken a new job which requires him to divest himself of WISC Technologies. Epsilon Lyra is my company. I am bringing WISC Technologies in as a division. Roy and I agree to doing the same with Mountain View Press.

I plan on carrying the public-domain versions of Forth which I included in my new edition of All About Forth: fig-FORTH, MVP-FORTH, F83, and F-PC. I will also carry available documentation.

We will also carry professional implementations from vendors who will make appropriate distributor's agreements. For the present, we will keep the same mailing address and phone numbers. We look forward to continuing our support of Forth users.

Charles Johnson, President of MISC, Inc. (Mutable Instruction Set Computer, formerly Minimum Instruction Set Computer) was our invited guest. Charles was joined by Dr. David Fox; MISC's Software Engineer as the two discussed this new silicon engine. 4/18/91

Thanks for the welcome. I wanted to speak about the Silicon Palimpsest this evening. That is the processor without a fixed instruction set. MISC originally stood for Minimum Instruction Set Computer. It was a tiny company (and still is), set up to create a Forth stack engine.

We wanted to do something different from Novix. We wanted a processor for embedded control, not desktop computing. That guided our efforts and our business. Today we have changed our name to Mutable Instruction Set Computer, Inc, because we have an even better idea. By using FPGA (field-programmable gate arrays), we believe we can create a processor with a mutable instruction set. The advantage of a mutable instruction set is that custom instructions can be designed for improved performance.

—Gary Smith

GARY-S on GENie
The Forth Interest Group serves both expert and novice members with its network of chapters, *Forth Dimensions*, mail-order services, and on-line activities. For membership information, or to reserve advertising space, contact the administrative offices:

Forth Interest Group  
P.O. Box 8231  
San Jose, California 95155  
408-277-0668  
Fax: 408-286-8988

Board of Directors  
John Hall, President  
C.H. Ting, Vice-President  
Mike Elola, Secretary  
Dennis Ruffer, Treasurer  
Wil Baden  
Jack Brown  
David Petty  
Dennis Ruffer

Founding Directors  
William Ragsdale  
Kim Harris  
Dave Boulton  
Dave Kilbridge

**In Recognition**

Recognition is offered annually to a person who has made an outstanding contribution in support of Forth and the Forth Interest Group. The individual is nominated and selected by previous recipients of the "FIGGY." Each receives an engraved award, and is named on a plaque in the administrative offices.

1979 William Ragsdale  
1980 Kim Harris  
1981 Dave Kilbridge  
1982 Roy Martens  
1983 John D. Hall  
1984 Robert Reiling  
1985 Thea Martin  
1986 C.H. Ting  
1987 Marlin Ouverson  
1988 Dennis Ruffer  
1989 Jan Shepherd  
1990 Gary Smith

**Forth Instruction**

*Los Angeles*—Introductory and intermediate three-day intensive courses in Forth programming are offered monthly by Laboratory Microsystems. These hands-on courses are designed for engineers and programmers who need to become proficient in Forth in the least amount of time. Telephone 213-306-7412.
On-Line Resources

To communicate with these systems, set your modem and communication software to 300/1200/2400 baud with eight bits, no parity, and one stop bit, unless noted otherwise. GEnie requires local echo.

**GEnie**
For information, call 800-638-9636
- Forth RoundTable (ForthNet*)
  Call GEnie local node, then type M710 or FORTH
  SysOps: Dennis Ruffer (D.RUFFER), Scott Squires (S.W.SQUIRES), Leonard Morgenstern (NMORGENSTERN), Gary Smith (GARY-S)
- MACH2 RoundTable
  Type M450 or MACH2
  Palo Alto Shipping Company
  SysOp: Waymen Askay (D.MILEY)

**BIX (ByteNet)**
For information, call 800-227-2983
- Forth Conference
  Access BIX via TymNet, then type J forth
  Type FORTH at the : prompt
  SysOp: Phil Wasson (PWASSON)
- LMI Conference
  Type LMI at the : prompt
  LMI products
  Host: Ray Duncan (RDUNCAN)

**Unix BBS's with forth.conf**
(ForthNet* and reachable via StarLink node 9533 on TymNet and PC-Pursuit node calan on TeleNet)
- WELL Forth conference
  Access WELL via CompuserveNet or 415-332-6106
  Fairwitness:
  Jack Woehr (jax)

**PCBoard BBS's devoted to Forth**
(ForthNet*)
- British Columbia Forth Board
  604-434-5886
  SysOp: Jack Brown
- Grapevine
  501-753-8121 to register
  501-753-6859
  StarLink node 9858
  SysOp: Jim Wenzel

**Other Forth-specific BBS's**
- Laboratory Microsystems, Inc.
  213-306-3530
  StarLink node 9184 on TymNet
  PC-Pursuit node calan on TeleNet
  SysOp: Ray Duncan
- Knowledge-Based Systems
  Supports Fifth
  409-696-7055
- Drumia Forth Board
  512-323-2402
  StarLink node 1306 on TymNet
  SysOps: S. Suresh, James Martin, Anne Moore
- WELL Forth conference
  Access WELL via CompuserveNet or 415-332-6106
  Fairwitness:
  Jack Woehr (jax)

**Non-Forth-specific BBS's with extensive Forth libraries**
- DataBit
  Alexandria, VA
  703-719-9648
  PC-Pursuit node dcwas
  StarLink node 2262
  SysOp: Ken Flower
- Real-Time Control Forth Board
  303-278-0364
  StarLink node 2584 on TymNet
  PC-Pursuit node coden on TeleNet
  SysOp: Jack Woehr

**International Forth BBS's**
- Melbourne FIG Chapter
  (03) 809-1787 in Australia
  61-3-809-1787 international
  SysOp: Lance Collins
- Forth BBS JEDI
  Paris, France
  33 36 43 15 15
  7 data bits, 1 stop, even parity
- Max BBS (ForthNet*)
  United Kingdom
  0905 754-157
  SysOp: Jon Brooks
- Sky Port (ForthNet*)
  United Kingdom
  44-1-294-1006
  SysOp: Andy Brimson
- SweFIG
  Per Alm Sweden
  46-8-71-35751
- NEXUS Servicios de Informacion, S. L.
  Travesera de Dalit, 104-106, Entlo. 4-5
  08024 Barcelona, Spain
  + 34 3 210355 (voice)
  + 34 3 2147262 (modem)
  SysOps: Jesus Consuegra, Juanma Barraquero
  barrans@nexus.nii.es (preferred)
  barrans@nii.es
  barrans (on BIX)
- Unnent!ddi1!irark!glsrk!gars

This list was accurate as of August 1991. If you know another on-line Forth resource, please let me know so it can be included in this list. I can be reached in the following ways:

Gary Smith
P. O. Drawer 7680
Little Rock, Arkansas 72217
Telephone: 501-227-7817
Fax (group 3): 501-228-9374
GEnie (co-SysOp, Forth RT and Unix RT): GARY-S
Usenet domain.: uunet!ddi1!irark!glsrk!gars

*ForthNet is a virtual Forth network that links designated message bases in an attempt to provide greater information distribution to the Forth users served. It is provided courtesy of the SysOps of its various links.*
# FIG Chapters

The Forth Interest Group Chapters listed below are currently registered as active with regular meetings. If your chapter listing is missing or incorrect, please contact Anna Brereton at the FIG office's Chapter Desk. This listing will be updated regularly in Forth Dimensions. If you would like to begin a FIG Chapter in your area, write for a "Chapter Kit and Application."  

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

## U.S.A.

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

- **ALASKA**  
  Kodiak Area Chapter  
  Ric Shepard  
  Box 1344  
  Kodiak, Alaska 99615  

- **ARIZONA**  
  Phoenix Chapter  
  Kodiak, Alaska 99615  
  4th Thurs., 7:30 p.m.  
  Arizona State Univ.  
  Memorial Union, 2nd floor  
  Dennis L. Wilson  
  (602) 381-1146

- **CALIFORNIA**  
  Los Angeles Chapter  
  4th Sat., 10 a.m.  
  Hawthorne Public Library  
  12700 S. Grevillea Ave.  
  Phillip Wasson  
  (213) 649-1428  

  North Bay Chapter  
  2nd Sat.  
  12 noon tutorial, 1 p.m. Forth  
  2055 Center St., Berkeley  
  Leonard Morgenstem  
  (415) 376-5241

  Orange County Chapter  
  4th Wed., 7 p.m.  
  Fullerton Savings  
  Huntington Beach  
  Norris Jesung  
  (714) 842-3032

  Sacramento Chapter  
  4th Wed., 7 p.m.  
  1708-59th St., Room A  
  Bob Nash  
  (916) 487-2044

  San Diego Chapter  
  Thursdays, 12 Noon  
  Guy Kelly  
  (619) 454-1307

- **COLORADO**  
  Denver Chapter  
  1st Mon., 7 p.m.  
  Emprise Corp., Marietta  
  Don Schrader  
  (404) 428-0813

- **FLORIDA**  
  Orlando Chapter  
  Every other Wed., 8 p.m.  
  Herman B. Gibson  
  (305) 855-4790

- **GEORGIA**  
  Atlanta Chapter  
  3rd Tues., 7 p.m.  
  Emprise Corp., Marietta  
  Don Schrader  
  (404) 428-0813

- **ILLINOIS**  
  Cache Forth Chapter  
  Oak Park  
  Clyde W. Phillips, Jr.  
  (708) 713-5365

  Central Illinois Chapter  
  Champaign  
  Robert Illyes  
  (217) 359-6039

- **INDIANA**  
  Fort Wayne Chapter  
  2nd Tues., 7 p.m.  
  1/F Univ. Campus  
  871 Neff Hall  
  Blair MacDermid  
  (219) 749-2042

- **IOWA**  
  Central Iowa FIG Chapter  
  1st Tues., 7:30 p.m.  
  Iowa State Univ.  
  214 Comp. Sci.  
  Rodrick Eldridge  
  (515) 294-5659

  Fairfield FIG Chapter  
  4th Day, 8:15 p.m.  
  Gurdy Leete  
  (515) 472-7782

- **MARYLAND**  
  MDFIG  
  3rd Wed., 6:30 p.m.  
  JHU/ APL, Bldg. 1  
  Parsons Auditorium  
  Mike Nemeth  
  (301) 262-8140 (eves.)

- **MASSACHUSETTS**  
  Boston FIG  
  3rd Wed., 7 p.m.  
  Bull HN  
  300 Concord Rd., Billerica  
  Gary Chanson  
  (617) 527-7206

- **MICHIGAN**  
  Detroit/Ann Arbor Area  
  Bill Walters  
  (313) 731-9660  
  (313) 861-6465 (eves.)

- **MINNESOTA**  
  MNFIG Chapter  
  Minneapolis  
  Fred Olson  
  (612) 588-9532

- **MISSOURI**  
  Kansas City Chapter  
  4th Tues., 7 p.m.  
  Midwest Research Institute  
  MAG Conference Center  
  Linus Orth  
  (913) 236-9189

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

- **NEW JERSEY**  
  New Jersey Chapter  
  Rutgers Univ., Piscataway  
  Nicholas G. Lordi  
  (908) 952-2062

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

- **NEW YORK**  
  Long Island Chapter  
  3rd Thurs., 7:30 p.m.  
  Brookhaven National Lab  
  AGS dept., bldg. 911, lab rm. A-202  
  Irving Montanez  
  (516) 282-2540

  Rochester Chapter  
  Monroe Comm. College  
  Bldg. 7, Rm. 102  
  Frank Lanzafame  
  (716) 482-3398

- **OHIO**  
  Columbus FIG Chapter  
  4th Tues.  
  Kal-Kan Foods, Inc.  
  5115 Fisher Road  
  Terry Webb  
  (614) 878-7241

  Dayton Chapter  
  2nd Tues. & 4th Wed., 6:30 p.m.  
  CFC  
  11 W. Monument Ave. #612  
  Gary Ganger  
  (513) 849-1483

- **PENNSYLVANIA**  
  Villanova Univ. Chapter  
  1st Mon., 7:30 p.m.  
  Villanova University  
  Dennis Clark  
  (215) 860-0700

- **TENNESSEE**  
  East Tennessee Chapter  
  Oak Ridge  
  3rd Wed., 7 p.m.  
  800 Oak Ridge Turnpike  
  Richard Secrist  
  (615) 483-7242

- **TEXAS**  
  Austin Chapter  
  Matt Lawrence  
  PO Box 180409  
  Austin, TX 78718

  Dallas Chapter  
  4th Thurs., 7:30 p.m.  
  Texas Instruments  
  13500 N. Central Expwy.  
  Semiconductor Cafeteria  
  Conference Room A  
  Warren Bean  
  (214) 480-3115

  Houston Chapter  
  3rd Mon., 7:30 p.m.  
  Houston Area League of  
  PC Users (HAL-PC)  
  1200 Post Oak Rd.  
  (Galleria area)  
  Russell Harris  
  (713) 461-1618
VERMONT
Vermont Chapter
Vergennes
3rd Mon., 7:30 p.m.
Vergennes Union High School
RM 210, Monkton Rd.
Hal Clark (802) 453-4442

VIRGINIA
First Forth of Hampton Roads
William Edmonds
(804) 898-4W

Potomac FIG
D.C. & Northern Virginia
1st Tues.
Lee Recreation Center
5722 Lee Hwy., Arlington
Joseph Brown
(703) 471-4403

INTERNATIONAL
AUSTRALIA
Melbourne Chapter
1st Fri., 8 p.m.
Lance Collins
65 Martin Road
Glen Iris, Victoria 3146
03/889-2600
BBS: 61 3 809 1787

Sydney Chapter
2nd Fri., 7 p.m.
John Goodsell Bldg., RM LG19
Univ. of New South Wales
Peter Tregeagle
10 Binda Rd.
Yowie Bay 2228
02/524-7490
Usenet:
tedr@usage.csd.unsw.oz

BELGIUM
Belgium Chapter
4th Wed., 8 p.m.
Luk Van Loock
Lariksdrift 20
2120 Schoten
03/658-6343

HOLLAND
Holland Chapter
2nd Wed., 7-9:30 p.m.
N. Alta. Inst. of Tech.
Tony Van Muyden
(03) 486-6666 (days)
(03) 962-2203 (eves.)

Southern Ontario Chapter
Quarterly: 1st Sat. of Mar., June, and Dec. 2nd Sat. of Sept.
Genl. Sci. Bldg., RM 212
McMaster University
Dr. N. Solntseff
(416) 525-9140 x3443

ENGLAND
Forth Interest Group-UK
London
1st Thurs., 7 p.m.
Polytechnic of South Bank
RM 408
Borough Rd.
D.J. Neale
58 Woodland Way
Morden, Surrey SM4 4DS

CANADA
Forth-BC
1st Thurs., 7:30 p.m.
BCIT, 3700 Willingdon Ave.
BBY, Rm. 1A-324
Jack W. Brown
(604) 596-9764 or
(604) 436-0443
BCFB BBS (604) 434-5886

Northern Alberta Chapter
4th Thurs., 7-9:30 p.m.
N. Alta. Inst. of Tech.
Tony Van Muyden
(403) 486-6666 (days)
(403) 962-2203 (eves.)

SPECIAL GROUPS
Forth Engines Users Group
John Carpenter
1098 Villa St.
Mountain View, CA 94041
(415) 950-1256 (eves.)

ITALY
FIG Italia
Marco Tausel
Via Gerolamo Forni 48
20161 Milano

JAPAN
Japan Chapter
Toshio Inoue
University of Tokyo
Dept. of Mineral Development
Faculty of Engineering
7-3-1 Hongo, Bunkyo-ku
Tokyo 113, Japan
(81)3-3812-2111 ext. 7073

REPUBLIC OF CHINA
R.O.C. Chapter
Ching-Tang Tseng
P. O. Box 28
Longtan, Taoyuan, Taiwan
(03) 4798925

SWEDEN
SweFIG
Per Alm
46/8-929631

SWITZERLAND
Swiss Chapter
Max Hugelshofer
Industrieberatung Ziberstrasse 6
8152 Opfikon
01 810 9289

"IBM will be hosting the next London meeting of FIG-U.K."

ITALY
FIG Italia
Marco Tausel
Via Gerolamo Forni 48
20161 Milano

JAPAN
Japan Chapter
Toshio Inoue
University of Tokyo
Dept. of Mineral Development
Faculty of Engineering
7-3-1 Hongo, Bunkyo-ku
Tokyo 113, Japan
(81)3-3812-2111 ext. 7073

SEPTEMBER/OKTOBER 1991
FORML CONFERENCE

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

Following Thanksgiving, November 29–December 1, 1991

Asilomar Conference Center
Monterey Peninsula overlooking the Pacific Ocean
Pacific Grove, California U.S.A.

Theme: Simulation and Robotics

Papers are invited that address relevant issues in the development and use of Forth in simulation and robotics. Virtual realities, robotics, and graphical user interfaces are topics of particular interest. Papers about other Forth topics are also welcome.

Attendees are invited to enter a robot in a robotics contest where the robot solves a puzzle.

Mail abstract(s) of approximately 100 words by September 1, 1991 to FORML Conference, Forth Interest Group, P.O. Box 8231, San Jose, CA 95155.

Completed papers are due November 1, 1991.

Conference Registration

Registration fee for conference attendees includes conference registration, coffee breaks, and notebook of papers submitted, and for everyone rooms Friday and Saturday, all meals including lunch Friday through lunch Sunday, wine and cheese parties Friday and Saturday nights, and use of Asilomar facilities.

Conference attendee in double room—$350 • Non-conference guest in same room—$200 • Children under 17 years old in same room—$140 • Infants under 2 years old in same room—free • Conference attendee in single room—$450

Forth Interest Group members and their guests eligible for ten percent discount on registration fees.

Register by calling the Forth Interest Group business office at (408) 277-0668 or writing to: FORML Conference, Forth Interest Group, P.O. Box 8231, San Jose, CA 95155.