

F O R T H

D I M E N S I O N S

■
ARRAYS IN FORTH

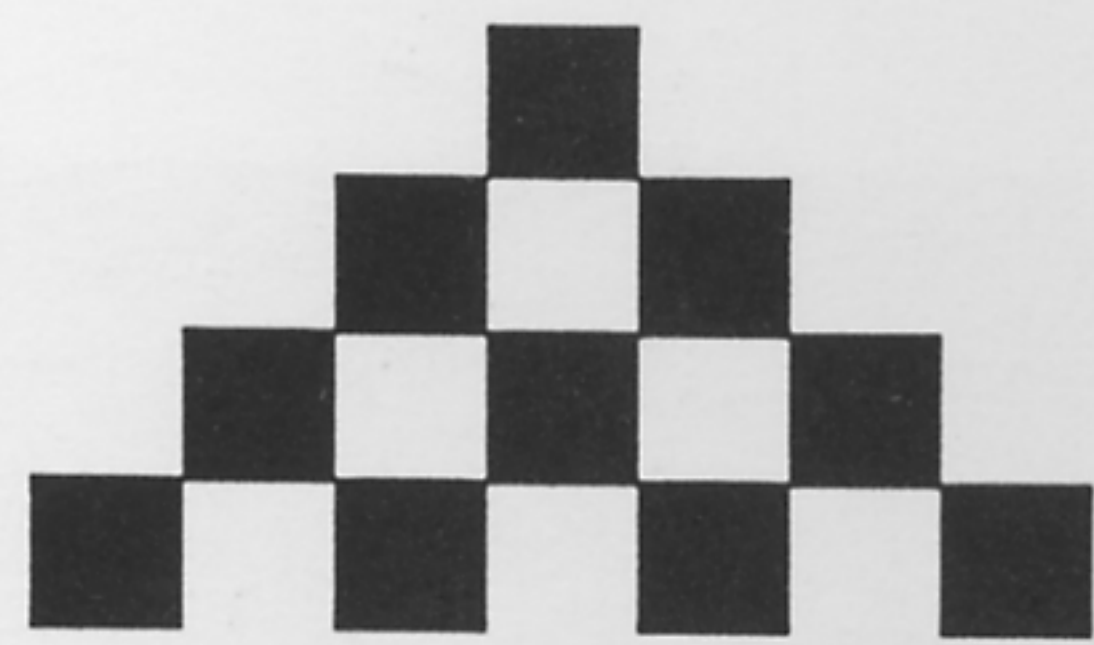
SOVIET FORTH-INFO

FORML XII: FORTH IN INDUSTRY

FORST: A 68000 NATIVE-CODE FORTH (V)

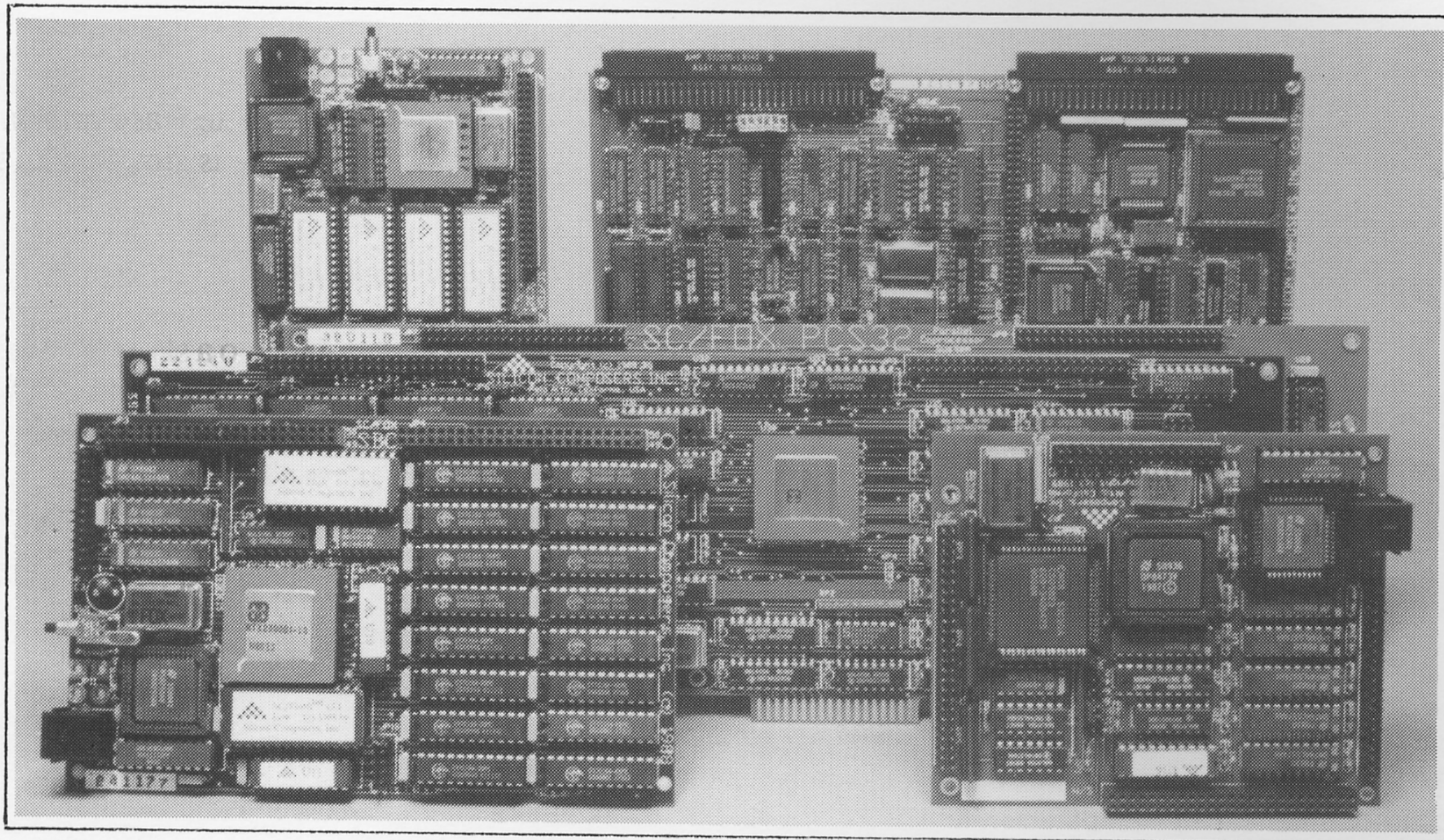
FROM CLEPSYDRAS TO NEURAL NETS

METACOMPILATION MADE EASY
■



SILICON COMPOSERS INC

FAST Forth Native-Language Embedded Computers



Harris RTX 2000tm 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.

SC32tm 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

F O R T H

D I M E N S I O N S

FROM CLEPSYDRAS TO NEURAL NETS - ROBERT GARIAN

6

 A water closet is a simple, on-off feedback control system much like that of neurons. This code models the system underlying a flushing toilet, implementing the logic with embedded computations as a set of production rules, a natural style for building complex causal models.


SOVIET FORTH-INFO - DAVID KIPPING

11

The author visited the Soviet Union as part of a multi-disciplinary delegation. One study site was Forth-Info, a co-op specializing in Forth. Like much in the U.S.S.R., nothing about this "private" company is simple, like profiting in a country with no copyrights!


ARRAYS IN FORTH - LEONARD MORGENSTERN

13

 In Forth, it can be easier to invent something that exactly suits one's needs than to conform to a standard method. But beginners often ask, "Why doesn't Forth have arrays?" Learn the common methods of defining arrays, and then some!

FORST: A 68000 NATIVE-CODE FORTH - JOHN REDMOND

16

 The "late binding" of deferred words lends flexibility to compiled code, but involves run-time decisions. Where these can take place at compile time, try ForST's configurable, new "vectored word." This is the last of a five-part series.


FORML XII: FORTH IN INDUSTRY - NEIL BAWD

20

Regular as the tides but far less predictable, FORML-goers gathered again in Pacific Grove, California last fall. Many of the typically belief-challenging, concept-stretching, water-testing, and otherwise mind-bending technical discussions were also marked by an upbeat mood about Forth's performance in today's marketplace.

PENCIL-AND-PAPER ARITHMETIC - J.J. MARTENS

22

 F83 users get an amusing glimpse at a floating-point experiment that really isn't. It exploits an innocent little word DPL and returns a surprising and interesting result—but it's not the real McCoy!

INTERNATIONAL GENIE ACCESS - DENNIS RUFFER

26

GE Information Services is reaching into more countries, and so is its Forth RoundTable. Access information, sign-up fees, and connect charges are given here, as well as new cost-saving features that affect all users of the network.


FORTH DAY 1990 - C.H. TING

30

Two enthusiastic chapters of the Forth Interest Group teamed up to host a local day-long Forth conference. The best and the brightest held forth to stimulate their audience and to share the latest. Here is a recap of the event.

METACOMPILATION MADE EASY - FRANK SERGEANT

31

 That is, easier to understand and easier to implement. Most languages separate the compiler from its results. Forth is different, its compilers *extend* the current Forth system rather than creating an independent system. This author builds on the work of cmFORTH.

Editorial

4

President's Letter

5

Best of GENIE

24

Advertisers Index

27

Reference Section

28

Author Recognition

38

Welcome New Members

39

FIG Chapters

42—43

EDITORIAL

An important notice to each of you—Beginning with the next issue, your membership in the Forth Interest Group will be handled differently. If you renewed your membership late in previous years, the FIG office would send any issues you might have missed and your membership would still expire at the end of April. The Board of Directors has elected to change that procedure; your membership will now start with the next issue to be printed after your dues are received, and will expire a year later. So be sure to renew on time—the only other way to get our next issue will be to pay the new cover price of \$10.

Along with the cover price, the cost of membership has increased for the first time in five years. The Board of Directors deemed it necessary in order to keep up the standards of quality and service that the Forth community needs so much. You should have received your renewal notice in the mail by now, offering (only to current members) a *limited-time renewal at last year's lower rates*. But that offer must expire before April 1, so be sure to take advantage of it right away. (Or use the mail-order form in the centerfold.)

We earnestly hope you will renew now, and that you will help us to guide and shape the Forth Interest Group this year. And you won't want to miss the exciting, upcoming issues of *Forth Dimensions*, either!

* * *

A new Board of Directors of FIG was installed at the 1990 FORML conference. The new President—and a familiar name to many FIG members—is John D. Hall. We wish him all the best in his position of responsibility. We know he is counting on active participation and lots of input from the membership, but you can read all about that in John's letter elsewhere in this issue.

Robert Reiling, retiring as long-time President and head of the FIG Business Group, will continue directing the FORML conference. His steadying hand has guided FIG through some rocky waters over the years, and we are glad his able direction will still be part of the scene. The entire corps of FIG volunteers and other contributors expresses its col-

lective thanks for Bob's innumerable contributions.

By the way, Mike Perry has agreed to serve again next year as FORML's program chairman. This year's event was a hot ticket, and is recapped in condensed form in this issue by the inimitable Neil Bawd.

* * *

Call for Authors!

We are looking for more Forth authors to share their discoveries, expertise, and opinions with the rest of us. Many of our long-time contributors are finding increasing opportunities to get published in general computing magazines—a welcome reminder that the public is interested in Forth—but we still need to keep *Forth Dimensions*' coffers full of interesting and challenging articles, too. So write to tell us about your latest experiment, how to impress our C-going friends, or about an improvement to a previous article's approach (who has a better interrupt handler or DMA scheme, anyway?). Your work will be in good company, too—we are anticipating new contributions from some of our favorite Forth authors of recent years.

In addition to purely technical writing, we are looking for someone to collect and/or edit Forth-related news items for *FD*. Are you plugged into the working Forth world? Do you know who got hired or promoted, what new product has just been released, which company landed a juicy contract, and how many programmers it really took to screw in that light bulb? We are looking for news sources, too, so even if you don't want to author a column, send your latest scoop to the FIG office or to MARLIN.O on GE Mail. And include your own legible name, address, and telephone number—on the record or off, we have to confirm items to be printed.

Finally, let me restate that *FD* is a reader-created publication without intentional technical biases or covert agendas. We are entirely shaped—every issue—by the letters, articles, and programming styles of our readers. I encourage you to join us in the process of making the next year of *FD* the most relevant, important, and valuable one yet.

—Marlin Ouverson

Forth Dimensions

Published by the
Forth Interest Group
Volume XII, Number 6
March/April 1991

Editor

Marlin Ouverson
Advertising Manager

Kent Safford

Design and Production
Berglund Graphics

Circulation/Order Desk

Anna Breton

Forth Dimensions welcomes editorial material, letters to the editor, and comments from its readers. No responsibility is assumed for accuracy of submissions.

Subscription to *Forth Dimensions* is included with membership in the Forth Interest Group at \$40 per year (\$52 overseas air). For membership, change of address, and to submit items for publication, the address is: Forth Interest Group, P.O. Box 8231, San Jose, California 95155. Administrative offices and advertising sales: 408-277-0668.

Copyright © 1991 by Forth Interest Group, Inc. The material contained in this periodical (but not the code) is copyrighted by the individual authors of the articles and by Forth Interest Group, Inc., respectively. Any reproduction or use of this periodical as it is compiled or the articles, except reproductions for non-commercial purposes, without the written permission of Forth Interest Group, Inc. is a violation of the Copyright Laws. Any code bearing a copyright notice, however, can be used only with permission of the copyright holder.

About the Forth Interest Group

The Forth Interest Group is the association of programmers, managers, and engineers who create practical, Forth-based solutions to real-world needs. Many research hardware and software designs that will advance the general state of the art. FIG provides a climate of intellectual exchange and benefits intended to assist each of its members. Publications, conferences, seminars, telecommunications, and area chapter meetings are among its activities.

"*Forth Dimensions* (ISSN 0884-0822) is published bimonthly for \$34/40/46 per year by the Forth Interest Group, 1330 S. Bascom Ave., Suite D, San Jose, CA 95128. Second-class postage paid at San Jose, CA. POSTMASTER: Send address changes to *Forth Dimensions*, P.O. Box 8231, San Jose, CA 95155."

LETTER FROM THE PRESIDENT

Fellow FIG members, I would like to take this opportunity to thank the members of the Board of Directors for their vote of confidence in electing me as your president. I will try my best to live up to the trust which has been placed in me. I am determined to work diligently to provide the leadership necessary for FIG to continue to improve its services to you its membership, to improve the image of the Forth community, and to expand the awareness of Forth.

I would also like to again thank Robert Reiling for the fine job he has done, and for the many hours of volunteer time he has donated over the past six years serving as president. His work as president inspired many of the improvements we are now beginning to see emerge in the organization. As a result of these efforts, my job will be a little easier, yet I will have to work very hard to fill the shoes he is leaving behind. Thanks again, Bob.

As for myself, I intend to use the President's Letter as a means to communicate to you what is going on at the international FIG Board and at the FIG business meetings. Many decisions are made each month which affect all of you as FIG members and indirectly affect you as Forth users. It is important that you are made aware of the issues and take an *active* role in the decision-making process. You can have an impact on these decisions, preferably by volunteering to serve on or chair a committee dealing with the issues you are most concerned with, or at a minimum by providing feedback to me and the other officers or board members on these issues so that decisions can be made with a first-hand knowledge of what the membership wants out of this organization. This is your organization, ensure that it serves you by getting involved; I guarantee you will get

"...Beyond His Call to Duty"

The Forth Interest Group—being a volunteer organization means that people give of their time and energy to support the organization. It is often expected and too often unrewarded. Yet there are some who give beyond what is expected, some who seem to not be satisfied with the normal, some who seek out the lost and provide help, some who provide for providing's sake without thought of reward.

Each year since 1979, one person from the Forth community is selected to be recognized from the many who are in the category of having "Contributed Beyond their Call to Duty" toward the support of the Forth Interest Group. The group of other so-recognized people, known as *The FIGGY's*, chooses to honor, and therefore drafts into its ranks, the one who in their opinion conspicuously stands out in this category.

The FIG RoundTable on GENie has grown over the last five years and has provided another communication channel besides *Forth Dimensions* between FIG members and others in the Forth community. The difficult and demanding part of providing this form of communication is the job of being a Sysop. The energy and spirit of the Sysop is reflected directly in the energy and spirit of the network. The job on GENie is so demanding that several Sysops are required, and the Sysops of GENie hold one of their own in great esteem.

So, for his tireless efforts as Sysop of the FIG RoundTable on GENie, and for his patience for the last four years as producer and host of the Thursday nights "Figgy Bars" on GENie, *The FIGGY's* have selected and would like to present to you:

1990 FIGGY of the Year

Gary Smith

much more back than you put in.

To start, a new Board of Directors has been formed and has begun its job of reshaping the image of FIG. The board's job is to set the "tone," or as I call it the "image," of the Forth Interest Group; but being the Board of Directors of FIG, they have the absolute final word on its functions and are the highest authority in FIG if you need to appeal to them for action. Call them and pass on to them your ideas. Of the seven, you will certainly find one or several

who will be sympathetic and help champion your ideas.

The Board of Directors is more diverse geographically than it has ever been. Of the seven members, three are from northern California-Silicon Valley; two are from southern California; one from British Columbia, Canada; and one from Boston, Massachusetts. Dr. C.H. Ting and David Petty are new Board members; Wil Baden, Dennis Ruffer, and John Hall have served

(Continued on page 38.)

FROM CLEPSYDRAS TO NEURAL NETS

ROBERT GARIAN - ARLINGTON, VIRGINIA

The ancient American Indians and Africans are known to have used water in crude clocks. They would place a small, leaky canoe or similar object in a bowl of water so that its sinking marked a certain period of time, just like an hourglass. The clepsydra (invented around 250 B.C.) was a bowl or pot, filled with water, that had a column of marks along the inside, so that the water level would act as an indicator of approximate time. The first jewelled timepiece is thought to have been a clepsydra whose orifice was a precious stone with a hole drilled in it.

It is easy ... to build a logic circuit out of flush toilets.

The flush toilet may be a descendant of the clepsydra. The modern toilet is little changed from the ancient Roman ones which had floats and stoppers.

A water closet or toilet system is actually a simple control system (classified as an on-off feedback control system). The parts of a control system are: a sensor (a measuring device), a controller, and a plant (the object that you want to control). The idea is that the measuring device provides the controller with the current value of some plant variable so that it can generate a corrective value to the plant.

The chain of events that takes place in a toilet system are automatic: once started, the system cycles and resets itself. In control terms, the float is the sensor or measuring device: its vertical position tells the valve (the controller) when to turn on and when to shut off. The lever, linkage, and

```
\ From clepsydras to neural nets. 10/22/90
\ Written in HS/Forth 3.4, uses coprocessor & CGA
\ R. Garian. Forth Dimensions
\ Requires the GRAPHICS and FLOATVAR extensions
```

DECIMAL FINIT FLOATS

```
( DIMENSIONS ARE IN INCHES AND SECONDS : )
```

```
( Length of tank ) % 19.0 RBOVAR LENGTH
( Width of tank ) % 7.0 RBOVAR WIDTH
( Max level ) % 10.0 RBOVAR H2
( Rest level ) % 9.0 RBOVAR H1
( Min level ) % 2.0 RBOVAR H0
( Full volume ) % 0.0 RBOVAR FULL
( Flapper radius ) % 1.0 RBOVAR R
( Grav. accel. ) % 0.0 RBOVAR G
( Current level ) % 0.0 RBOVAR H
( Area of opening ) % 0.0 RBOVAR AREA
( Elapsed time ) % 0.0 RBOVAR TIME
( Time step ) % 0.0 RBOVAR DT
( Current outflow ) % 0.0 RBOVAR OUTFLOW
( Current inflow ) % 0.0 RBOVAR INFLOW
( Flapper leak ) % 0.0 RBOVAR FLAPLEAK
( Valve leak ) % 0.0 RBOVAR VALVLEAK
( Current volume ) % 0.0 RBOVAR VOLUME
( Time factor ) % 0.0 RBOVAR ALPHA
( Height factor ) % 0.0 RBOVAR BETA
( Finishing time ) % 0.0 RBOVAR FINISH
```

```
( States ) -1 VAR ON
( " ) 0 VAR OFF
( " ) -1 VAR UP
( " ) 0 VAR DOWN
( " ) -1 VAR RISING
( " ) 0 VAR FALLING
( Objects ) 0 VAR FLOAT
( " ) 0 VAR FLAPPER
( " ) 0 VAR VALVE
```

flapper are used to initiate the operational cycle of the toilet system. The cycle is over and the system reset when the valve has shut off. At some point during the filling part of the cycle, it will become possible to force an early cycle.

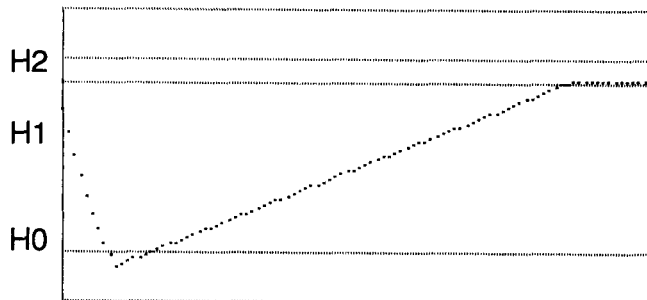
In addition to the primary control system, there is another: the overflow pipe serves as both sensor and controller. The reference level is the height of the overflow pipe, and the correction is made by simply allowing the water above the opening of the pipe to be drained.

There is a strong resemblance between a flushing toilet system and a neuron. I believe that J.Y. Lettvin was the first to notice this analogy. The neuron consists of a cell body and an axon (a tube leading away from the body). Inside the neural membrane, there are positive potassium ions and negative proteins; outside, there are positive sodium and negative chloride ions that create a resting membrane potential of about -70mV. A mechanical, electrical, or chemical stimulus can cause depolarization of the membrane, allowing the temporary cross-flow of sodium and potassium ions. The result is a rapid rise from the negative resting-membrane potential to the positive action potential. This localized change in the membrane potential propagates as the adjacent inactive region's threshold is broken down by the resulting current flow.

Although we shouldn't take the analogy too far, a flush is analogous to a depolarization, and the period after a flush during which it is not possible to initiate a new flush is similar to the refractory period of the neuron during which it must rest and restore the ion balance. The propagation of the action potential is similar to the discharge from the tank system, and the threshold for the action potential is similar to the minimum amount of force required to activate the lever-linkage system that lifts the flapper in the hopper.

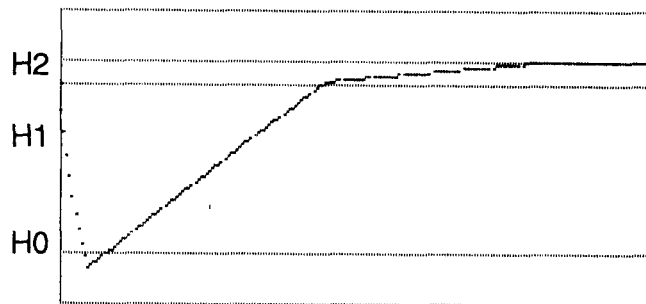
It is easy, although impractical, to build a logic circuit out of a set of flush toilets. We can attach leaky tin cans to the lever or pull-chain system of the toilets, and run pipes from the flapper openings to the cans. Whenever a can fills sufficiently with the discharge from other flushes, another flush is triggered. The logic, which is threshold logic, is the same whether operative in neural nets, flush nets, or gates in electronic circuits.

Time = 100.00000 Volume = 1216.3721
 Flapper is down Valve is off
 Time step = 1.0000000



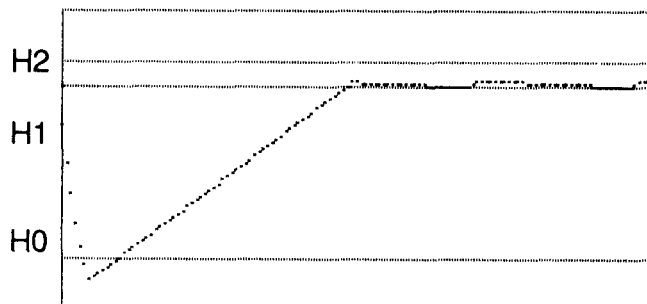
Flapper leak = 0.0000000
 Valve leak = 0.0000000

Time = 200.00000 Volume = 1330.2721
 Flapper is down Valve is off
 Time step = 1.0000000



Flapper leak = .50000000
 Valve leak = 2.0000000

Time = 199.50000 Volume = 1227.9801
 Flapper is down Valve is off
 Time step = 1.5000000



Flapper leak = 1.2000000
 Valve leak = .50000000

Threshold logic is based on a test for inequality. The rule is: Given a number called the threshold, T, if the input from the previous stage adds up to a number greater than T, then send out an output value. By varying T, it is possible to construct a logic gate. For example, if the inputs are measured in units that can be normalized to 0 and 1, and if T is 1, then any unit input will cause a unit output. If T is 2, then two or more inputs will have to deliver unit inputs to cause a unit output. If there are k inputs and $T < k$, then an output will follow any input of T or more units. This last case is sometimes called majority logic if $T = k/2$ when k is even or $T = (k+1)/2$ when k is odd.

The accompanying source code models the basic on-off feedback control system underlying a flushing toilet. The interested reader may want to try his hand at constructing the model before reading further.

The program simulates the mechanical system by implementing the control logic with embedded computations as a set of production rules (condition \geq action). This programming style is natural for building complex causal models because sequencing is less important when most factors act concurrently.

Some of the inaccuracies in the model are due to the fact that it is a discrete model and because it ignores physical properties such as the inertia of the float, variation in water pressure at the valve, the presence of objects in the tank which affect flow, and so on. However, it does account for the varying discharge rate from the flapper opening by applying a special case of Bernoulli's Equation (obtained by Daniel Bernoulli in 1738) called Torricelli's theorem: velocity = $\sqrt{2*G*H}$, for the velocity of water leaving an orifice at a distance H down from the water level.

Experimentation is relatively easy, since new rules can be added at any time to extend or restrict the model. The radical-shaped curves in Figure One show the time behavior of the system under both watertight and leaky conditions. Under leaky conditions, the system tends toward three states: overflowing, underfilling, or oscillating.

```
( Convert G to in/sec*sec )    % 12.0 % 32.0 F* IS G
( Compute BETA--pixels/inch ) % 100.0 % 12.0 F/ IS BETA
( Med Resolution ) C-320 WIPE 17 GROUND ( dark blue background )

: DRAW-GRAPH ( -- )
  50 50 150 270 1 RECT
  % 150.0 H2 BETA F* F- F->S DUP 50 SWAP 270 1 LINE
  % 150.0 H1 BETA F* F- F->S DUP 50 SWAP 270 1 LINE
  % 150.0 H0 BETA F* F- F->S DUP 50 SWAP 270 1 LINE
  8 4 WCURSOR! ." H2"
  10 4 WCURSOR! ." H1"
  16 4 WCURSOR! ." H0"
  20 9 WCURSOR! ." FLAPPER LEAK=" FLAPLEAK F.
  21 9 WCURSOR! ." VALVE LEAK=" VALVLEAK F.
  22 9 WCURSOR! ." PRESS F TO CONTINUE" ;

: INITIAL-CONDITIONS ( -- )
  LENGTH WIDTH F* H1 F* IS FULL
  F=PI R R F* F* IS AREA
  DOWN IS FLAPPER
  OFF IS VALVE
  H1 IS H
  FULL IS VOLUME
  % 0.0 IS TIME
  % 0.01 IS DT
  % 0.0 IS ALPHA ;

: SELECT-PARAMETERS ( -- )
  CR ." Enter simulation time in seconds:"          F#IN IS FINISH
  % 220.0 FINISH F/ IS ALPHA ( pixels/second )
  CR ." Enter amount of flapper leak in cu in/sec: " F#IN IS FLAPLEAK
  CR ." Enter amount of valve leak in cu in/sec: "  F#IN IS VALVLEAK
  CR ." Enter time step: "                          F#IN IS DT ;

\ Velocity of water out of the flapper opening
: VELOCITY ( -- ) G H F* % 2.0 F* FSQRT ;
```

Robert Garian is a technical information and language specialist at the Library of Congress, specializing in Soviet computing. He has a mathematics degree from George Washington University and is the founder of Hibiscus Software and Wordware, a software and information company in Arlington, Virginia. A member of the American Association for Artificial Intelligence, he has written an AI program called Block Solver that automatically rearranges one multistack configuration of blocks into a specified goal configuration under constraints. He has worked on automatic software verification and automatic code generation, and is interested in the simulation of complex systems, cellular automata, and genetic algorithms.


```

: PLOT      ( -- )
  ( row  ) % 150.0 BETA H F* F- F->S
  ( col  ) TIME ALPHA F* F->S 50 + 2 PIX! ;

```

```

\ Assumptions:

```

```

\ .6 is a guess at the drag coefficient for water leaving the opening.
\ 90.0 seconds is the total time required for one cycle.
\ Tank is 19x12x7. Max water level is H2 = 10 = height of overflow pipe.

```

```

: FLUSH      ( -- )
  INITIAL-CONDITIONS
  WIPE

```

```

  SELECT-PARAMETERS

```

```

  WIPE

```

```

  DRAW-GRAPH

```

```

  3 1 WCURSOR! ." TIME STEP=" DT F.

```

```

  PLOT

```

```

  UP IS FLAPPER

```

```

  BEGIN

```

```

  FLAPPER UP   = IF AREA VELOCITY F* % .6 F* IS OUTFLOW THEN

```

```

  FLAPPER UP   = IF FALLING IS FLOAT THEN

```

```

  FLAPPER DOWN = IF RISING IS FLOAT THEN

```

```

  FLAPPER DOWN = IF FLAPLEAK IS OUTFLOW THEN

```

```

  FLOAT FALLING = IF DN IS VALVE THEN

```

```

  VALVE ON     = IF FULL % 90.0 F/ IS INFLOW THEN

```

```

  VALVE OFF    = IF VALVLEAK IS INFLOW THEN

```

```

  H H0 F<=    IF DOWN IS FLAPPER THEN

```

```

  H H1 F>     IF OFF IS VALVE THEN

```

```

  H H1 F<     IF DN IS VALVE THEN

```

```

  H H2 F>     IF VALVLEAK IS OUTFLOW THEN

```

```

  -1         IF VOLUME OUTFLOW INFLOW F- DT F* F- IS VOLUME
             VOLUME LENGTH WIDTH F* F/ IS H PLOT THEN

```

```

\ Update display:

```

```

  1 1 WCURSOR! ." TIME=" TIME F.

```

```

  1 20 WCURSOR! ." VOLUME=" VOLUME F.

```

```

  2 1 WCURSOR! ." FLAPPER IS " FLAPPER IF ." UP " ELSE ." DOWN" THEN

```

```

  2 20 WCURSOR! ." VALVE IS " VALVE IF ." ON " ELSE ." OFF" THEN

```

```

  TIME DT F+ IS TIME

```

```

  TIME FINISH F>

```

```

  UNTIL

```

```

  KEY ASCII F = IF WIPE RECURSE ELSE BW80 BYE THEN ;

```

```

\ Note: in HS/Forth % and the F-words (F#IN,F.,F>,F<=,FINIT,...) relate to
\ floating point numbers. VAR words are fast variables that don't need @.
\ WCURSOR! is used to place the window cursor. and WIPE clears the screen.

```

TETHERBALL IN THE NERVOUS SYSTEM

The October 26, 1990 issue of *Science* contains several articles on the application of site-specific mutagenesis ("systematic changing of amino acids in proteins and studying the consequences") to the understanding of ion channels formed by pores in a nerve's membrane. "Playing tetherball in the nervous system" by Marcia Barinaga (pp. 506-7) describes the results of this research. Building on the work of Clay Armstrong and Francisco Bezanilla in the 1970s, Richard Aldrich, Toshinori Hoshi, and William Zagotta have obtained evidence for an unusual voltage-sensing inactivation method (i.e., a method for stopping the flow of ions through a pore). "Imagine a protein tetherball, dangling by a protein cord on the inside of a nerve cell's outer membrane: while the tetherball floats free, ions pass into or out of the cell; but when the ball pops into the mouth of a pore, the ion flow stops." By stretching and inverting some of the concepts involved, this remarkable molecular mechanism can be made to correspond to the control system of our model if we allow the following similarities: tank:nerve cell and surrounding fluid, ion channel/pore:orifice, voltage-sensing mechanism:float, ball and chain:flapper and chain (here the flapper is normally open instead of closed), ion solution:water. The valve might be likened to the effects of osmotic pressure acting on the ion channel. Thus, simple control systems can be found in astronomical quantities even at the molecular level.

HARVARD SOFTWARES

NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

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 in as an integral part of the language and only HS/FORTH provides true multiple inheritance & membership.

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 s-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 MSDOS or PC DOS, or from ROM. Each level includes all features of lower ones. Level upgrades: \$25. plus price difference between levels. Source code is in ordinary ASCII text files.

All HS/FORTH systems support full megabyte or larger programs & 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 \$299.
NEW! Fast direct to video memory text & scaled/clipped/windowed graphics in bit blit windows, mono, cga, ega, vga, all ellipsoids, splines, bezier curves, arcs, turtles; lightning fast pattern drawing even with irregular boundaries; powerful parsing, formatting, file and device I/O; DOS shells; interrupt handlers; call high level Forth from interrupts; single step trace, decompiler; 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 * IS C compiles 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&7 assemblers.

PRODUCTION LEVEL \$499.
Metacompiler: DOS/ROM/direct/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.
286FORTH or 386FORTH \$299.
16 Megabyte physical address space or gigabyte virtual for programs and data; DOS & BIOS fully and freely available; 32 bit address/operand range with 386.
ROMULUS HS/FORTH from ROM \$ 99.
FFORTRAN translator/mathpak \$ 79.
Compile Fortran subroutines! Formulas, logic, do loops, arrays; matrix math, FFT, linear equations, random numbers.

Shipping/system: US: \$7. Canada: \$19. foreign: \$49. We accept MC, VISA, & AmEx

SOVIET FORTH-INFO

DAVID KIPPING - MENLO PARK, CALIFORNIA

As part of a "People-to-People" program, I visited the Soviet Union from October 9-22, 1990. I was part of a delegation of engineering, political, environmental, and business leaders who met with Soviet counterparts in their various areas of expertise. As a member of the engineering delegation, I visited several Soviet businesses, institutes, and manufacturing facilities. At all of these, there was considerable interest in computing, especially using IBM-compatible personal computers.

One of the businesses that was visited by the engineering delegation was Forth-Info, a Soviet cooperative. They specialize in Forth-based software and hardware. My professional background is in computer software, with current interests in graphical user interfaces for PCs. Previously, I spent seven years involved with Forth-based software, so I feel quite qualified to evaluate the activities of Forth-Info.

Forth-Info Business Structure

Forth-Info is structured as a cooperative business based in Leningrad. Cooperatives are a recent Soviet phenomenon, and are more-or-less like a U.S. private business. The company was started in 1988 by Boris Katsev and his son Sergei Katsev. The company has about 70 employees and associates. The actual number of employees is about 30, with the rest operating on some kind of an independent-contract status. Like all things in the Soviet Union, the arrangements are complex and interlocked. [More information about the company is presented in the accompanying sidebar, which was written sometime in 1989.]

Forth-Info is in the process of designing a building of its own. (It is now in rented and donated quarters.) The building will be five stories (matching adjacent buildings) on land that will be obtained from the government on a 50-year lease. Two floors will be occupied by the Ministry of Education, one floor will be devoted to computer

education, and one floor will contain the offices and laboratories of Forth-Info. One wing will be devoted to about 30 apartments, sufficient for all the staff of Forth-Info. In addition, there will be several spare apartments which are allocated for visiting researchers. Forth-Info hopes to sponsor exchange programs with U.S. businesses and institutions, and apartments for visitors are necessary in the Soviet system.

There is no copyright protection for software in the Soviet Union.

The main source of income for Forth-Info seems to consist of contracts with various ministries for custom software development. One such project is a system for screening blood donors. Another project is a real-time control system for railways, which uses a Soviet fault-tolerant computer. Forth-Info also runs a school for the Ministry of Education that teaches computer programming and operations to grade school and high school students in a hands-on laboratory.

Technical Activities

Although contract programming is the primary source of income, the main interest for the future is standard products. Since there is no copyright protection for software in the Soviet Union, making a profit on standard products is difficult. A six-page product summary gives a description of Forth-Info's activities and offerings. Although I did not test or evaluate all these products, I believe they are real and operating as described. I was able to evaluate some of the products:

Forth Language Description. I in-

spected the documentation for a Forth language product (InfoForth or AstroForth). It has a complete description of language primitives in sufficient detail for an experienced Forth programmer to proceed. The text was in Russian, but all the Forth primitives were in Roman characters and easily understood.

InfoTransHemo. This program for blood screening is apparently in widespread use at Soviet blood donation stations. One of the main difficulties was matching report formats and input requirements to existing procedures and forms. (Probably, the problem was finding out what these procedures and forms were!)

Forth Development System. I got a brief look at available development tools for Forth. The editor, file control, and execution environment seemed quite adequate.

InfoWeaver. This program for textile pattern design was quite impressive. It runs on a PC and uses a full-color graphical interface. It seemed easy to use, and appeared to provide the level of control that a designer would want. Patterns and colors could easily be changed and the results displayed immediately. Response was very fast. The output is a report which gives explicit set-up information for looms. In the Soviet Union, looms are manually set up, but automatic set-up information could be output on paper tape or floppy disk for looms that were so equipped. This product was up to U.S. standards for look, feel, and functionality. Forth-Info provided a demonstration disk of the product.

Info1816. I was able to spend a fair amount of time with this product. It is a simulator and emulator for the Intel 8048 microprocessor, and runs on a PC interfaced to an in-circuit emulator. The user interface consists of tiled windows for editor, assembler, status, and output information. Programs for the chip can be edited,

assembled, and run in a simulation, or actually executed using the in-circuit emulator. The bulk of the code is not dependent on the specific microprocessor, and Forth-Info is extending it for other Intel processors. This product was up to U.S. standards for functionality and user interface.

I have no doubt that the other products which are described in their literature are of equal quality and professionalism. Forth-Info is aware of the state of the art of U.S. and world developments, and is operating at a similar level.

Computer Education Activities

Forth-Info has an arrangement with the Ministry of Education to provide training in computers for school-age children. A major part of their facility is a large classroom equipped with computers for student use. About 20 students (average age about 12 years) were in the classroom, each with his own computer. By the time we had arrived, formal classes were over and the students were all playing computer games! *Tetris* and a road-racing game seemed to be the most popular.

The computer arrangement was rather interesting. Each student station consists of a system unit, display, and keyboard (Roman and Cyrillic) with BASIC built into ROM. The processor is an 8080 running CP/M, and all the student stations are linked in a local area network to a master (instructor) station. None of the student stations has a floppy disk (or hard disk), but the instructor station had floppy disks and a printer. The system seems to work quite well, because 12-year-olds had no trouble getting their favorite games running.

Apparently, the education and training activities of Forth-Info are quite important to the company and to the Ministry of Education. This explains the close connection with the ministry with respect to Forth-Info's building plans.

Soviet PCs Really Exist!

Forth-Info was the last stop on our two-week tour of businesses in the Soviet Union. PCs are in great demand and are very expensive, since they are all imported and paid for with hard currency. Most of the units we saw seemed to be manufactured in the Far East (Hyundai and Epson were two brands I recognized). The cost of a PC is equivalent to about ten years' salary for a senior professional, which explains the Fort Knox-type security of all computer

rooms.

Previously, we had visited a Soviet computer factory and suggested that they design and manufacture PC clones. They said it was not possible because components (chips, floppy drives, and hard disks) are not available and, besides, Soviet computers are not reliable. At other businesses and institutes, we were again told that the Soviet Union does not manufacture any PCs.

After visiting the student training room at Forth-Info, we went into another room which contained several machines of the same type. I asked what they were, and was told that they are Soviet-made PCs. In one corner, a woman was using one of them to prepare a document (in Russian) using a word processor.

The machines consist of a small desktop box (about 14 inches wide) for the system unit and a similarly sized box for the two five-inch floppy disks. There was no hard disk in evidence. A standard monochrome monitor and keyboard (Roman and Cyrillic) completed the configuration. The de-

sign is about five years old and the machines we saw were several years old. There seems to be no effort to produce a new design, or much interest in manufacturing and distributing this model. The criticisms of this Soviet PC were that it had some compatibility problems, was not reliable, and was not up to date. Several people said they would prefer to have no computer at all, rather than use the Soviet PC for development. They rejected the concept of using the cheap and available Soviet PC for initial development and using an expensive, imported PC to check compatibility and final functionality.

Forth-Info Business Goals

Forth-Info is interested in publishing their products in the U.S., and of particular promise are InfoWeaver and Info1816. They recognize that the manual and packaging must be brought up to American standards, and that they will need an American publisher and distributor. Our delegation offered to assist them in locating these resources.

Forth-Info

The cooperative Forth-Info started its activities at the Leningrad University in early 1988. Programmers of the highest qualification from various Leningrad research enterprises are working here. Forth, a fourth-generation language, and hardware-software problems are their common area of interest.

Forth-Info is now developing Forth-based commercial software for personal computers, mini- and microcomputers, OEM. Forth systems for all computers that are widely used in the U.S.S.R. have been created by members of the cooperative. Software development systems and applications packages in Forth have been developed and are on sale. A threaded-code processor, executing Forth code at the hardware level, is now being designed as a single chip.

Forth-Info also propagates Forth and teaches people. Its members participated in creating the first two Forth manuals published in Russian and in the U.S.S.R. A translation of Leo Brodie's *Starting Forth* and *Thinking Forth* is being prepared for publication under the editorial supervision of some members of the cooperative.

Members of the cooperative give lectures on Forth to a wide range of programmers, paying special attention to programmers from industry. At the end of 1989, the cooperative planned to organize an all-Union conference of Forth users in the U.S.S.R. Its aim is to exchange information and experience about Forth applications and to start organizing an association of Forth users. Foreign participants were also expected.

The chairman of Forth-Info is B. Katsev. Leading specialists and authors of Forth software are I. Agamizian, A. Astonovsky, S. Katsev, V. Kirillin, A. Klubovich, N. Nozdrunov, V. Patryshev, and S. Tovstolugsky. There are 28 specialists on the staff of Forth-Info, 11 with Ph.D. degrees; 40 more work on contract.

Forth-Info will support any contacts with the Forth community, both in the U.S. and in Europe.

Forth-Info Co-op, P.O. Box 279, 10a, O.Koshevoy St., 197198, Leningrad, U.S.S.R. Telephone (812)-233-3410.

ARRAYS IN FORTH

LEONARD MORGENSTERN - MORAGA, CALIFORNIA

A natural question beginners often ask is: Why doesn't Forth have features that are standard in other languages, for example, arrays? The answer is, Forth is so facile at creating new data types that it is often easier to invent something that exactly suits one's needs than to force a program to conform to some arbitrary standard. The upcoming ANSI standard for Forth will not include specifications for arrays, which is as it should be, in my opinion. An exploration of examples will explain why.

It is important to recognize that Forth arrays are not the same as the abstract arrays of mathematics, on the one hand, or the arrays of conventional computer languages, on the other. In mathematics, an array is simply a set of numbers arranged in a pattern. Conventional computer languages add an action: an index is accepted and a quantity is returned. Arrays in Forth, as a rule, return the address where the data resides, but can return the data itself or perform an action; and arrays that do other things are easy to assemble, as we will see.

The following way of defining arrays is economical and fast, and is so popular that many Forth programmers mistakenly believe that it is standard:

```
: ARRAY ( n -- ) ( i -- a )
  CREATE 2* ALLOT DOES>
  SWAP 2* + ; ( 1 )
```

At defining time, two bytes of memory are allotted for each element of the array; at run time, an index *i* is used to compute an address *a* where data can be fetched, stored, or otherwise manipulated. The index and the address are simple—that is, they each consist of one single-precision integer. Each element occupies two bytes in dictionary space, and the array has one dimension. There is no range checking. I cite these features in detail because any or all of

them can be altered, thereby broadening the definition.

A narrower definition is used by F-PC:

```
: ARRAY ( n -- ) ( -- a )
  CREATE 2* ALLOT ; ( 2 )
```

This allots the same amount of space as the previous definition, but at run time takes no index, always returning the address of the origin. My prejudice is that this is not an array, but rather a work area, or buffer. But to avoid semantic controversies, I will use the terms *indexed* and *non-indexed* for the two kinds of arrays.

Many programmers mistakenly believe it is standard.

An Array with Elements of Any Length

The first definition above is restricted to elements of length 2 in one dimension. The word `-BYTE-ARRAY` (Screen One) expands this to arrays with elements of any length in one or two dimensions. Two examples are presented: a one-dimensional array named `FOO1` with 200 six-byte elements, and a two-dimensional array named `FOO2` with 200 12-byte elements arranged in ten rows and 20 columns. `FOO2` takes a compound index consisting of two integers, one for each dimension. The order of the two is the same as that used to declare the array. Therefore, `i1` ranges from 0-9 and `i2` from 0-19.

I like the English-like syntax of `-BYTE-ARRAY`. A purist might complain, but similar constructs are used a lot in Forth, and the results can be elegant. The elegance is not always free: here, there is a

cost in run-time speed. The word `2-DIMENSION` is a dummy with no action, included for parallelism. This sort of thing is sometimes contemptuously called "syntactic sugar," suggesting that it is pleasant but unessential and even harmful. Actually, it is like giving a box of fine chocolates to someone you love: the benefits far outweigh the cost.

Fields

When using an array with long elements, it is useful to subdivide them into fields. In Screen Two, a defining word `FIELD` creates a class of Forth words to reference these. For instance, if each element of `FOO1` (Screen One) represents a node of a binary tree, the six bytes can be subdivided into three fields: a right pointer, a left pointer, and data. The words `.LEFT`, `.RIGHT`, and `.DATA` provide access to these. The "dot" syntax is copied from Pascal. These are not just names: as with all Forth words, they have an action, which in this case is to increment the preceding address. For example, `3 FOO1 .RIGHT` specifies the address of the right pointer of the third element of `FOO1`.

Arrays of Bits

We next proceed to describe something more complex: an array of bits. It uses a non-indexed array of bytes named `MASKS` (Screen Three) which contains powers of two, from one to 128. `MASKS` is initialized by the useful technique of "comma-ing," which is often superior to defining an empty array and then filling in the values.

Locating a bit requires two integers—a mask and a RAM address—representing a compound address which can be used directly by Forth words that manipulate bits, such as `CSET`, `CRESET`, and `CTOGGLE`. The address is computed by using the sequence:

\ Screen 1. A versatile array definer.

```
: -BYTE-ARRAY ( r c n -- ) ( r c -- addr; or c -- addr)
CREATE      DUP 2OVER , , , ROT 1 MAX * * ALLOT
DOES>      DUP 2+ @ 0= IF 0 -ROT THEN      ( Fix stack if 1-dim)
           >R SWAP R@ @ * + R@ 4 + @ * R> 6 + + ;

: 1-DIMENSION ( c -- 0 c)    0 SWAP ;
: 2-DIMENSION ;

\ Examples
  200 1-DIMENSION 6 -BYTE-ARRAY F001
10 20 2-DIMENSION 12 -BYTE-ARRAY F002
\ Note that there must be a space before the first hyphen in
\ -BYTE-ARRAY but not before the second.
```

8 /MOD (8 n -- rem quot)

The quotient is added to the base address of the array, and the remainder is used to fetch the mask from BYTES.

Using a bit array to manage an imaginary lighting system is illustrated in Screen Three. Other examples of arrays that return compound addresses are those situated in virtual memory (i -- ofs blk) or in expanded or extended memory (i -- seg ofs).

Some Random Thoughts

The indexed arrays discussed up to this point return an address, which may be simple or compound. Thus, they do what Forth variables do, and may properly be regarded as "VARIABLE-like." If the contents were returned instead, the designation "VALUE-like" would be appropriate. Arrays that contain actions (CFAs) are widely used, and are called "execution vectors" or, sometimes, just "vectors." And, of course, a Forth array can contain a mixture of these types.

It is usual to construct arrays by means of defining words, but this is a convenience and not a necessity. The array MASKS was created without one. A defining word is essential when several structures of the same kind are wanted, but also can be useful when defining a solitary example.

For simplicity, I have omitted range checking from the examples. Almost always, this is a necessity. It may be omitted only if the index cannot possibly be out of range, as when a keystroke is masked by 255 AND and is used to index into an array

of 256 elements. (A programmer wounded by a past encounter with Murphy might check anyway.) Forth words are more efficient and easier to debug if their parameters are guaranteed correct. The word that does the checking and takes corrective action should, as a rule, precede the array word and not be part of its definition.

Forth vs. Other Languages

Recently, I came across the following statement in a textbook: "Each language provides a limited number of elementary data types." I have become so accustomed to the power and flexibility of Forth's CREATE ... DOES> that it was a shock to recall how limited other languages can be. In Forth, as we have seen, new data types can be created and abandoned at the convenience—or even at the whim—of the programmer. True, some Forth constructs are so useful that they have become official or *de facto* standards (CONSTANT and VARIABLE, for example) but it is more usual for a programmer to define what is needed in the most convenient way. As always, freedom has its costs, one of which is a greater need for documentation. When you declare an array in BASIC, you describe its purpose, but you do not have to describe what it is or how it works, as you must in Forth.

Another source of subtle complications is the fact that defining words not only allocate memory for their contents, they also specify the action they will perform with or on those contents. In effect, the simplest Forth program incorporates object-oriented features. Niklaus Wirth claimed that algorithms + data structures = programs,

and he thought so highly of the slogan that he made it the title of a book. The foregoing discussion suggests that, in Forth at least, there is no clear separation between data and the algorithm.

Leonard Morgenstern is a retired pathologist and computer hobbyist. His interest in Forth goes back over ten years. Currently, he is a sysop of the Forth RoundTable on the GENie network. His son, David Morgenstern, is also an author on computer-related subjects.

\ Screen 2. Defining subfields within array elements.

```
VARIABLE FIELD-OFFSET
: FIELD ( len -- )
  CREATE FIELD-OFFSET @ , FIELD-OFFSET +!
  DOES> @ + ;

FIELD-OFFSET OFF
2 FIELD .LEFT           \ Define 3 subfields, each of length 2
2 FIELD .RIGHT
2 FIELD .DATA
```

\ Screen 3. A bit array.

```
CREATE MASKS 128 C, 64 C, 32 C, 16 C, 8 C, 4 C, 2 C, 1 C,
\ In creating a bit array, len is the ceiling of the number
\ of bits divided by 8.

: BIT-ARRAY ( len -- ) ( i -- m a )
  CREATE ALLOT
  DOES> SWAP 8 /MOD MASKS + C@ -ROT + ;

\ Creating a light controller for 24 lights. Uses a
\ vocabulary named LIGHTS because we are redefining ON and OFF

3 BIT-ARRAY LIGHT           \ 3 bytes = 24 bits
VOCABULARY LIGHTS
LIGHTS DEFINITIONS
: ON ( n -- ) LIGHT CSET ;   \ Set the bit
: OFF ( n -- ) LIGHT CRESET ; \ Reset the bit
: ON? ( n -- f ) LIGHT C@ AND 0<> ; \ Test the bit
: ALLON ( -- ) 0 LIGHT 3 127 FILL DROP ; \ Turn on all bits
: ALLOFF ( -- ) 0 LIGHT 3 0 FILL DROP ;

FORTH DEFINITIONS
\ EXAMPLES
LIGHTS           \ Set the vocabulary
ALLOFF
12 ON           \ Turn on Light number 12
```

FORST: A 68000 NATIVE-CODE FORTH

JOHN REDMOND - SYDNEY, AUSTRALIA

Deferred words are well established in Forth, and they provide a simple means of achieving forward referencing and vectored execution. The concept was originally developed for indirect-threaded code systems, and it implicitly makes use of the headers which are present at run time.

To use a simple example:

```
DEFER GREET
: NICE
  ." Good morning" ;
` NICE IS GREET
```

When GREET is executed, we get the "Good morning" message, of course. But, because GREET is DEFERred, we can change its behavior:

```
: GROAN
  ." You again!" ;
` GROAN IS GREET
```

All this is obvious—but what if we incorporate GREET into a higher-level word?

```
: GREETING
  GREET <further code> ;
```

Now the run-time behavior of GREET-ING will depend on the state of GREET when GREETING is executing. Even if GREETING is compiled when GREET is NICE, it will not stay friendly if it is made to GROAN.

This late binding of behavior lends great flexibility to compiled code but, because it involves decision making at run time, it may not always be appropriate. For those cases where we want the vectoring to take place at compile time (early binding), ForST provides a new class of configurable word—a vectored word:

```
VECTOR SAY-HELLO
```

The Compilation Record

Before compiling a normal word, the ForST compiler looks up its code address (as an offset in the CFA) and fetches its optimization flags (from the FFA). In these, it has all that is necessary to compile a call to the code, or to expand it with optimization. A deferred or vectored word also needs this information to specify completely its present status. To this end, its PFA points to an eight-byte compilation record:

```
bytes 0-3:  offset address of actual
             code
bytes 4-7:  FFA corresponding to this
             code
```

(This arrangement of fields was chosen so that code which uses ` (tick) and PERFORM will operate in the usual way.)

When a deferred or vectored word is first defined, its compilation record is initialized to point to a NOOP. When it is subsequently modified (by IS), the record is altered:

```
HEAD GREET IS SAY-HELLO
```

HEAD returns the CFA of a word. Its use (rather than `) allows access to the necessary header fields of GREET and separates the activity of the vector word from the header of the vectored word. This header need not even continue to exist! (Note: in ForST, ` still returns the address of code or data in the usual way.)

Because they have no directly bound code, when a number of VECTOR words are defined in sequence, their compilation records will be located in adjacent memory with no intervening breaks. This array, a *state table*, is a very concise summary of their present behavior and, as such, can be saved to another array—or initialized from one—by simple block moves.

Arithmetic States

There are far too many words in Forth, and there is sense in trying to avoid further unnecessary proliferation. When a floating-point package is available, for instance, a number of new words appear to become necessary, e.g., FNUMBER, F., F+, F-, F*, F/, FMOD, FNEGATE, and FABS. They can be avoided at the user level by setting up the familiar number words (NUMBER, ., +, etc.) as vectored words, with their compilation records in a state table. REALS and INTEGERS can then be used as immediate words to manipulate its contents at compile time. This approach of overloading existing words makes it possible to write very clean code (Figure One).

Separating Compile-time and Run-time Activities

ForST enables the programmer to create completely independent application code. To do this most effectively, it is necessary to separate compile-time code and execution code—to unthread (or unbind) some of the support we take for granted in a Forth system.

The compilation of string literals is a good illustration of the way compilation and execution functions are bound up:

```
: ."
  COMPILER (." )
  , " ; IMMEDIATE
```

When ." is encountered at compile time, it has an immediate action of compiling (.") and then enclosing a counted string in the compiled code. The roles of (.") and , " are quite different, and take place at different times. There is, therefore, no reason for them to have a fixed relationship in code space.

The real difficulty with this definition of ." is that the run-time code, (."), must


```

\ Demonstration code to illustrate the use of
\ arithmetic states and vectoring of . + and number input.

\ The code will print out sines of angles at 0.05 radian
\ intervals, waiting for a key stroke after each step.
\ The infinite loop is ended by pressing <ESCAPE>.
\ Note that local constants and addresses are permitted
\ because they occupy no code space.

```

macros

decimal integers

```

: sines { 1 reg angle 27 constant escape }
  reals
  cr 0 to angle
  begin
    angle . angle sin . cr
    angle 0.05 + to angle
    integers
    key escape = if leave then
  again ;

```

\ Simple trigonometric functions

decimal reals

```

355 113 / constant pi
pi 2 / constant pi/2
pi 2 * constant 2pi

```

```

: (sin) { 1 arg angle 2 locals square term }

```

```

  angle to term
  angle angle * to square
  angle term square * to term
  term -0.1666667 * + term square * to term
  term 0.00833333 * + term square * to term
  term -0.0001984 * + term square * to term
  term 0.0000028 * + ;

```

```

: sin { 1 arg angle 1 local sign }

```

```

  angle 0< to sign
  angle abs to angle
  angle 2pi >
  if angle 2pi mod to angle then
  angle pi >
  if angle pi - to angle sign not to sign then
  angle pi/2 >
  if pi angle - to angle then
  angle (sin) sign if negate then ;

```

(Continued on next page.)

Figure One.



NGS FORTH

A FAST FORTH,
OPTIMIZED FOR THE IBM
PERSONAL COMPUTER AND
MS-DOS COMPATIBLES.

STANDARD FEATURES INCLUDE:

- 79 STANDARD
- DIRECT I/O ACCESS
- FULL ACCESS TO MS-DOS FILES AND FUNCTIONS
- ENVIRONMENT SAVE & LOAD
- MULTI-SEGMENTED FOR LARGE APPLICATIONS
- EXTENDED ADDRESSING
- MEMORY ALLOCATION CONFIGURABLE ON-LINE
- AUTO LOAD SCREEN BOOT
- LINE & SCREEN EDITORS
- DECOMPILER AND DEBUGGING AIDS
- 8088 ASSEMBLER
- GRAPHICS & SOUND
- NGS ENHANCEMENTS
- DETAILED MANUAL
- INEXPENSIVE UPGRADES
- NGS USER NEWSLETTER

A COMPLETE FORTH
DEVELOPMENT SYSTEM.

PRICES START AT \$70

NEW ◀ HP-150 & HP-110
VERSIONS AVAILABLE



NEXT GENERATION SYSTEMS
P.O. BOX 2987
SANTA CLARA, CA. 95055
(408) 241-5909

(Figure One, continued.)

```
: cos pi/2 + sin ;

: tan dup sin swap cos / ;

decimal integers ( default state)
```

```
d_array: bsr create
         bsr over
         bsr comma
         bsr mult
         bsr allot
         bsr xdoes ;end of compile-time
code
comp rec: dc.l offset (runtime) ;address of run-time
code
         dc.l 0 ;default FFA (no mac-
ros)
runtime: rts ;noop code unless de-
fined after DOES>
```

Figure Two.

```
: array { 2 regargs #rows #columns }
create #rows w, #rows #columns * allot
does> { 3 regargs row column address }
address inc w@ column * row +
address + ( return address) ;
```

Figure Three-a.

```
: d_array { 2 regargs #rows #columns }
create #rows w, #rows #columns * allot
does> ;

: doarray { 3 regargs row column address }
address inc w@ column * row +
address + ( return address) ;
```

Figure Three-b.

be defined before the immediate word, `."`, can be defined. In other words, the compilation code is compiled *after* the run-time code and, if we wish to generate free-standing application code, we are forced to include in it code which has absolutely no run-time role.

This difficulty can be avoided by the following:

```
VECTOR (".")
: ."
  DELAY (".")
  , " ; IMMEDIATE
```

The internal code of `."` is now a little more complex, but specification of the address and macro characteristics of `."` can now be delayed until `."` is needed in a definition. (Note that another new word, `POSTPONE`, has essentially the same action as `COMPILE`.)

When `COMPILE` is used in the definition of `."`, it fetches the CFA and FFA of `."` and uses them to compile into `."`. When `DELAY` is used, on the other hand, a pointer to the compilation record is compiled, together with appropriate code to access it. The record is outside the application code, and fetching of its contents is delayed until `."` is used in another definition (by this time, it should have been initialized by a word in the application code).

Delayed Defining Words

For all their elegance, words using `CREATE ... DOES>` are traditionally the most problematic in their binding of compile-time and run-time activities. This can be avoided by giving them vectored behavior. To use a *Starting Forth* example:

```
: ARRAY (#rows #columns --)
CREATE OVER , * ALLOT
DOES> (row col -- address)
DUP @ ROT * + + CELL+ ;
```

Using a delayed `DOES>`, we can write:

```
: D_ARRAY
CREATE OVER ,
* ALLOT DOES> ;
```

At this point, the code for `D_ARRAY` looks something like that in Figure Two. (`ARRAY` would differ by having the run-time code directly after the compilation record.) Later in the source code, we might write:

```
8 8 D_ARRAY BOARD
< more code>
```

```
: DOARRAY
  DUP @ ROT
  * + + CELL+ ;
```

```
HEAD DOARRAY IS D_ARRAY
<more code>
```

```
: HIGHER WORD
  5 3 BOARD
  <more words> ;
```

In doing this we have used D_ARRAY to define BOARD without having specified its run-time action.

BOARD, like all constants, variables, addresses, and other words defined with CREATE ... DOES>, is an immediate word in ForST. When it is used in the definition of HIGHER_WORD, it compiles the address of its own data area, then fetches the contents of a compilation record (embedded in ARRAY) and incorporates a call to, or a macro expansion of, the appropriate code (DOARRAY). If the contents of the compilation record had not been defined at this stage, a NOOP would have been compiled.

In this way, D_ARRAY, a defining word, can be kept out of application code, and its run-time component initialized (at compile-time) from within the application code.

The approach also allows the programmer to vary the action of BOARD, say during a debugging session, but a more important advantage is that the coding elegance of CREATE ... DOES> constructs can be used in application code without the penalty of including the defining code.

Clearer and more efficient definitions of ARRAY and D_ARRAY, using register variables, are given in Figures Three-a and Three-b. Note that the ARGS or REGARGS for the run-time code must include the implicit address on top of the stack.

John Redmond is an Associate Professor of Organic Chemistry at Sydney's Macquarie University. He is a "...sometimes-evenings-when-I-have-time programmer" who welcomes letters from FD readers: 23 Mirool Street, West Ryde, NSW 2114, Australia.



**1991
ROCHESTER
FORTH CONFERENCE
ON
AUTOMATED INSTRUMENTS**

*June 18 -21, 1991
University of Rochester*

For more information, contact:

**Lawrence P. Forsley
Conference Chairman
Institute for Applied Forth Research, Inc.
70 Elmwood Avenue
Rochester, NY 14611 USA
(716) 235-0168 • (716) 328-6426 fax**

**E-Mail: GENieL.Forsley
Compuserve ...72050,2111
Internet72050.2111@COMPUSERVE.COM**

FORML XII

NEIL BAWD - GOAT HILL, CALIFORNIA

The Twelfth FORML Conference was held at Asilomar, Thanksgiving weekend, 1990. The theme of the conference was "Forth in Industry," and the presentations gave ample evidence that Forth is doing very well in this arena. Forth's forté is being able to do things which no other language can do, given the particular environment.

The conference began with GUY KELLY describing nine projects representative of some of his experiences with Forth in industry. GUY GROTKE expanded with a few of his experiences, and RAY ST. LAURENT, from New Brunswick, demonstrated that Forth is equally potent north of the border. JOHN WATERMAN, V.P. of Forth, Inc., spoke on event management in process control. KEN BUTTERFIELD showed how he uses Forth at Los Alamos to control serial devices. ANDREW SCOTT, in an impromptu talk, told how he renamed Forth to be used as a language for testing communication-line protocol.

Forth is for professionals doing real applications in real time.

Some of these applications are quite glamorous, others less so, but all showed that Forth has strength in dedicated applications unmatched by any other language. Forth is a language for professionals doing real applications in real time.

Forth is a language for engineers. Dr. RICHARD TURPIN told how he used Forth as the language of design in a microprocessor systems design course at a California college, and Dr. RICHARD HASKELL matched it with his account of the use of Forth in a required engineering course in a Michigan college. In other presentations, Dr. Haskell

Report by EDWIN ALB at FORML XII

At the 1989 FORML conference, an unofficial unFORML contest was held by the participants and their guests.

*"Complete the following sentence:
Programming is like Sex Because..."*

The contest was conducted by the aptly named NEIL BAWD, and the judge was an expert in the subject—his wife. The contestants, except for the winners, were kept anonymous. The following entries were received.

*This entry was rejected by the judge because "eunuchs" was misspelled:
It isn't much fun to do with unix.*

*This entry was rejected because it was unsigned:
Programming is not like sex because I can still program.*

The Women's Entries

You never know when to stop.

Many users are satisfied without documentation.

Programming is not like sex because sex can't work with software.

Even with the best preparation, all efforts lead to frustration if the hardware is out of order.

The women's winning entry:

It's never finished. ANNE EDGECOMB

The Men's Entries

The first three entries are from the same person.

I can never get enough of it.

My wife doesn't want me to do either.

When you do it professionally, it's not as much fun.

After three days without it, life becomes meaningless.

Satisfaction is guaranteed with a steady relationship, patient endurance, and lots of physical effort.

When I can't get to sleep I have to do one or the other.

Experience should improve the experience; when this is not the case one of the partners should change.

Safety is always rewarded—but then so is indulgence.

It keeps us up late at night making little extensions.

Most of us use more than just three fingers.

When you make a mistake you end up supporting it for years.

The men's winning entry:

You can hear about it, you can talk about it, you can read about it, you can even watch it done by experts, but even with all the fumbling and mess, it's still more satisfying to get personally involved with it. NICK GROSSMAN

described the successful use of Forth for handicapping horse races, and the tribulations of porting eForth to the 68000.

All other computer languages, compared to Forth, are languages of bondage and discipline. FORML celebrates Forth's freedom. Dr. LEONARD MORGENSTERN showed how to implement safety nets for error recovery and yet another object program in Forth. DEAN SANDERSON's discussion of managing dictionaries as packages was awarded first prize. MIKE ELOLA showed how to incorporate generic stack operations. C. H. TING explained eForth—the model, design, and implementation.

Forth is a language for research and development. S.Y.TANG uses it for approximate rational arithmetic, JOHN WAVRIK for the study of group theory, FRANK SERGEANT to investigate the use of a three-key keyboard for text input, ANDREW McKEWAN spoke on his use of Forth for 6805 development on a budget and a multi-tasker for the 68HC11, TOM ZIMMER gave his thoughts on Forth for the 90's.

Someone attending said that FORML has become the forum for standardizing Forth, and ANSI X3J14 the laboratory for modifying Forth. MITCH BRADLEY's report on the progress of ANS Forth gave reassurance that the ANSI effort will ultimately be successful—even his proposal for stream I/O in Forth seemed fair and reasonable.

The reason that I personally prefer FORML to other computer conferences is that there is always something to take home and try myself. Presenters do not all just say, "This is what I did." They say, "This is what I did, this is how I did it, and how you can do it too." This year, besides much of the material above, there was GLEN HAYDON's self-teacher for musical ear training, and ROB CHAPMAN's technique for an efficient hashing method.

As good as the papers were, what makes FORML great is what happens between the sessions. This cannot be explained; it has to be experienced. The paradox of the conference is that it is exhilarating and exhausting. It is something for the entire family. If you bring some or all of your family, you will see them only late at night and/or early in the morning; but you all will be so busy that they won't miss you and you won't miss them.

The success of the conference was due, as always, to the organizational genius of

(Continued on page 25.)

Proceedings of the 1990 Rochester Forth Conference

Over 70 papers on the state of the art in Forth and threaded interpretive languages, including comparisons of C, ADA and Forth for embedded systems, and eleven papers from the Soviet Union.

- ◆ **ShBoom on ShBoom: A Microcosm of Hardware and Software Tools**
Mr. Charles Moore, *Computer Cowboys*
- ◆ **Using Forth to Analyze and Debug Kernel-less Embedded Systems**
Mr. Darrel Johansen, *Orion Instruments, Inc.*
- ◆ **Active Messages and Passive Objects: An Object Oriented View of Forth**
Mr. Rod Crawford, *MPE, Ltd.*
- ◆ **The Forth System Behind VP-Planner: Designing for Efficiency in the Face of Complexity**
Dr. Kent Brothers, *Stephenson Software*
- ◆ **The Future of Forth in Astronomy**
Dr. Arne Henden, *Ohio State Univ.*
- ◆ **Ada and Forth: How Do They Stack Up?**
Dr. James D. Basile, *Long Island Univ.*

JFAR

Volume 6 Number 2

- ◆ **The Cost of User-Friendly Programming: MacImage as Example**
Ferren MacIntyre, *Univ. of Rhode Island*
- ◆ **Little Universe: A Self-referencing State Table**
Karl-Dietrich Neubert, *Physikalisch-Technische Bundesanstalt, Berlin, FRG*
- ◆ **A FORMula TRANslator for Forth**
J.V. Noble, *University of Virginia*
- ◆ **A Generalized EXIT**
Carol Pruitt, *University of Rochester*
- ◆ **Strings, Associative Access, and Memory Allocation**
N. Sointseff, *McMaster University*

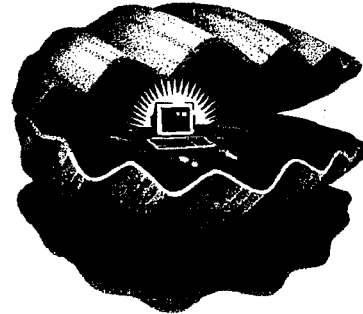
Volume VI Subscriptions

	Individual	Corporate
USA	\$60.00	\$145.00
Canada/Mexico	\$65.00	\$145.00
Europe/Asia	\$75.00	\$160.00

1990 Rochester Conference Proceedings
Embedded Systems \$30 plus \$5 S/H.

1990
ROCHESTER
FORTH
CONFERENCE

EMBEDDED
SYSTEMS



JUNE 12 - 16TH, 1990
UNIVERSITY OF ROCHESTER

THE JOURNAL OF
FORTH
APPLICATION AND
RESEARCH

Volume 6 Number 2 1990

User-friendly Programming
FORMula TRANslator for Forth
Strings and Things

Send name, full address and phone number. Check or money order in US funds, or, VISA/MC # and exp. date to:

Institute for Applied Forth Research
70 Elmwood Avenue
Rochester, NY 14611 USA

(716) 235-0168 • (716) 328-6426 fax

Email: GEnie L.Forsley
CompuServe 72050,2111
Internet 72050.2111@
compuServe.com

PENCIL-AND-PAPER ARITHMETIC

J.J. MARTENS - KAUKANA, WISCONSIN

The Idea for WHATS-IT.BLK came while I was reviewing an out-of-print book.* In a sense, the program shows that Forth can work with decimals, including long division, in much the same way we used to do it with pencil and paper. Let me explain.

Assuming that you have looked over the code and have access to F83, let's go directly to MULTIPLY, the first of the three high-level words in screen five. Before entering anything, it's important to be aware of:

The Game Rules

1. All numbers must be entered with a decimal point somewhere.
2. Theoretical limits for data input and output are 2,147,483,647 to -2,147,483,648. For practical use, the limits are $\pm 999,999,999$.
3. When multiplying: the number of digits in the multiplicand *plus* the number of digits in the multiplier must not exceed nine.
4. When dividing: the dividend must have as many or more places to the right of the decimal as does the divisor.
5. When adding: All numbers must have the same number of digits to the right of the decimal. If any are short, fill with zeroes.

The penalty for violating rule #1 is a system crash. Other penalties are usually less severe. Caution is recommended.

We are now ready for a trial run.

Decimal Multiplication

ok 2 MULTIPLY <enter>

At the application's prompts (there are two), enter 263.42 and 12.4 then follow the

succeeding prompts with .1468 and .463 and the result should look like this:

```
Enter either number first
ENTER NUMBER 263.42 * 12.4 =
  3266.408
ENTER NUMBER .1468 * .463 =
  .0679684
```

Note that in the first example, there are two digits to the right of the decimal in the multiplicand (.42) and one in the multiplier (.4). Therefore, according to the rule for pencil-and-paper arithmetic, the product will have $2 + 1 = 3$ digits to the right of the decimal (.408). In the second example, the program pads the product with a zero in order to get seven digits to the right of the decimal.

Decimal Division

ok 2 DIVIDE <enter>

At the application's prompts (again, there are two), enter 679.684 and 14.68 then follow succeeding prompts with 7461.70000 and 56.83 and the result should be:

```
Enter dividend first
ENTER NUMBER 679.684 / 14.68 =
  46.3
ENTER NUMBER 7461.70000 /
  56.83 = 131.298
```

In the first example, there are three digits to the right of the decimal in the dividend (.684) and two is the divisor (.68). Since we are now dividing instead of multiplying, according to the rule for pencil-and-paper arithmetic, the program will subtract instead of add, putting $3 - 2 = 1$ digit to the right of the decimal in the quotient (.3). The second example illustrates adding a zero to the dividend to satisfy game rule #4, and

three more to add three extra places to the quotient.

Decimal Addition and Subtraction

When adding and subtracting decimals, the pencil-and-paper method puts numbers one below the other, with the decimal points aligned, and proceeds with implied zeroes in the empty places to the right of the decimal point. The ADDMODE computation uses actual zeroes.

ok ADDMODE <enter>

At the application's prompt (which reappears after each entry), enter 1254.6200, 129.3600, 26375.8000, 4.7530, 18.9654, -83.0000, and 0. successively, and the result should look like this:

```
ADDMODE
      1254.6200
      129.3600
  26375.8000
      4.7530
      18.9654
      -83.0000
TOTAL  27700.4984
```

* *Handbook of Business Mathematics*, Wm. R. Minrath. D. Van Nostrand Co., Inc. 1959.

J.J. Martens ran the family business for nearly three decades, then spent several self-employed years until his 'practical retirement.' His interest in Forth and subsequent purchase of a Jupiter Ace computer (and more equipment later), was aroused in 1983 by Popular Computing, which he calls "...a good magazine, how extinct."

```

0
0 12-5-90JJM \ Double number keyword: D/ 11-21-90JM
1 WHATS-IT.BLK is really an experimental floating point
2 that exploits an innocent little F83 word DPL (decimal point
3 location). The result is surprising and interesting but it's
4 not the real McCoy.
5
6 The FORTH involved is a LAXEN-PERRY 2.1.0 F83.COM file
7 over an MS-DOS 2.11.22 COMMAND.COM.
8
9 Screens 2 and 3, which together constitute a key
10 component, are a slightly altered version of that by Zafer
11 Esser M. D. originally appearing in FORTH DIMENSIONS Vol. 7
12 No. 4 Pg. 5.
13
14 J. J. Martens : D/ (S d1 d2 --d3) D/MOD 2SWAP 2DROP ; \ key word #2
15

1
0 \ Nuts and bolts -- Load screen 11-26-90JM \ Floating point utility 11-19-90JM
1 : D0<> (S --) D0= NOT ; \ useful - not in dictionary
2 : 2+! (S d a--) DUP >R 2@ D+ R> 2! ; \ ditto
3
4 : EN (S --) ." ENTER NUMBER " ;
5 : INPUT (S --d) QUERY BL WORD NUMBER ; \ stack a dbl len no.
6 : ENTERNBR (S --d) EN INPUT ;
7 : WASH (S --) -LINE 13 EMIT ; \ clear clutter
8 : TEST (S d--d f) 2DUP D0<> ;
9 : EENF (S --) ." Enter either number first" ;
10 : EDF (S --) ." Enter dividend first" ;
11
12 2VARIABLE TOTAL \ holds ADDMODE total
13 : TOT00 (S --) 0 0 TOTAL 2! ; TOT00
14
15 2 5 THRU \ OK will now load the source code.

2
0 \ Double number keyword: D* 11-21-90JM \ Utilities - HIGH LEVEL WORDS 12-5-90JJM
1
2 : DM* ( d1,u--dprod)
3 DUP ROT * ROT ROT UM* ROT + ;
4
5 : D* (S d1 d2 -- dprod) \ key word #1
6 2DUP 0 0 D< >R DABS DABS >R >R 2DUP R> DM* R> ROT ROT \
7 >R >R DM* 32768 DM* 2 DM* R> R> D+ R> IF DNEGATE THEN ;
8
9 : DM/MOD (S d1 u--drea dquot)
10 SWAP OVER /MOD >R SWAP UM/MOD >R 0 R> R> ;
11
12
13
14
15

3
: D/MOD (S d1 d2--drea dquot)
2DUP 0 0 D< >R 2DUP DABS SWAP DROP
IF R0
IF >R >R DNEGATE R> R> DNEGATE
THEN SWAP >R /MOD R> SWAP >R 0 R0 S>D D* D- R> S>D
ELSE R0
IF >R >R DNEGATE R> R>
THEN DROP ABS DM/MOD
THEN R>
IF >R >R DNEGATE R> R>
THEN ;

4
VARIABLE FDPL 0 FDPL ! \ floating decimal point location
: FDPL! (S --) DPL @ FDPL ! ; \ store floating decimal pt
: FDPL+! (S --) DPL @ FDPL +! ; \ add to floating decimal pt
: FDPL-! (S --) DPL @ NEGATE FDPL +! ; \ subtract from "
: (F.) (S d --a l) TUCK DABS \ convert dbl len no to a string
<# FDPL @ 0 ?00 # LOOP ASCII . HOLD #S ROT SIGN #> ;
: F. (S d n--) (F.) TYPE SPACE ; \ output as a signed double
\ number with a trailing space
: F.R (S d n--) >R (F.) R> OVER - SPACES TYPE ; \ output
\ as a signed double number
\ n spaces right justified

5
: PROCEED (S d f --) \ utility word
IF FDPL! 2DUP TOTAL 2+! 24 F.R TRUE
ELSE 2DROP FALSE THEN ;
: .TOTAL (S --) TOTAL 2@ ." TOTAL" 17 F.R ; \ utility word
HIGH LEVEL WORDS
: MULTIPLY (S n--) EENF CR \ Floating point multiplication
0 DO ENTERNBR FDPL! ." * " INPUT FDPL! D* ." = " F. CR
LOOP ;
: DIVIDE (S n--) EDF CR \ Floating point division
0 DO ENTERNBR FDPL! ." / " INPUT FDPL! D/ ." = " F. CR
LOOP ;
: ADDMODE (S --) \ Calculator style addition
BEGIN CR ENTERNBR WASH TEST PROCEED WHILE
REPEAT .TOTAL TOT00 CR ;

```

Printed with File PR6SCR.54

BEST OF GENIE

GARY SMITH - LITTLE ROCK, ARKANSAS

News from the *Genie Forth RoundTable*—Last issue we recapped guest conferences with Robert Smith, “Floored Division and Floating Point”; Phil Koopman, “Stack Machines”; Charles Curley, “A Minimalist View of Forth”; and John D. Hall, “The Business of FIG.”

In this issue, we will continue our romp through guest conference memory lane by checking back on visits with Elizabeth Rather, “X3J14 Chair Discusses the ANS Forth Effort”; Steve Sarns and Steve Wheeler, “Vesta Technologies and Embedded Forth”; Dr. John Wavrik, “Teaching Forth”; Jim Tittsler and Allan Pratt of Atari Product Development, “Hardware Prototyping with Forth”; and Tom Almy from Tektronix, Inc.’s engineering, “User Interface for non-Forth Users.”

Elizabeth D. Rather (January 1990)
President of FORTH, Inc.
Chair of ANS Forth Technical Committee

In the fall of 1986, about a dozen people met at FORTH, Inc. to start an effort to produce an ANSI standard for Forth. They included three major vendors (competitors!), users from large companies (IBM, MCI, etc.), and individual consultants/experts (including Chuck Moore). All of us agreed that an ANSI standard would improve acceptance of the language, improve our ability to hire programmers or get jobs (depending on which side of the fence you’re on!), and make it easier to move programs from one Forth to another.

The ANSI process operates under very explicit, detailed rules designed to ensure broad participation in the development and review of any standard. Anyone can be a member of the TC who has the time and money (mostly for travel—the fees are minor in comparison), and you don’t need to be a member to submit proposals or

review our work. When we think we’re done, our draft standard will be published for a four-month public review period. We’re required to respond in writing to all comments, and if we make changes as a result there will be an additional two-month review. This cycle will repeat as many times as necessary. At the end, our governing body will review both our standard and the process. Not only must the standard be clear (etc.), but the process must be proven to be open and fair.

We welcome proposals from anyone. The most useful are specific proposals to make explicit changes to BASIS. Less-specific comments are also welcome and will be taken under advisement, but are harder to act on and thus less effective.

A standard should not try to innovate, improve, or extend a technology. Insofar as possible, we try to identify “common practice” and articulate it. Some compromise is inevitable, particularly when there are conflicts in common practice.

Steve Sarns, Steve Wheeler (February 1990)
Vesta Technology, Inc.

Sarns: Five years ago, I founded Vesta in my living room. I was the only employee. I built the prototype of the product we now vend as the SBC88. The unit used Forth and had a Forth in ROM. I typed it in from the FIG listings, but made it ROM-able, and at the same time was debugging the wire-wrap board.

Radio Electronics published an article in March 1984 that featured the board controlling “smart homes.” That three-month series basically launched Vesta Technology. In fact, we still get product inquiries based on those articles!

Nowadays, about half our business is custom engineering, and we sell five off-

the-shelf single-board computers with a variety of ROM language support. Some have C, some have BASIC, but all have Forth, since that is our in-house development language for custom projects.

Wheeler: We have applications running all the way from the SBC88 monitoring a couple of digital input bits, to networked boards monitoring several analog and digital inputs in real time while communicating with the user via LCD and keypad on one board, with all boards using a multi-drop RS485 network to talk to each other. The typical application anymore seems to be one board with a few analog inputs, a number of digital inputs and outputs, an LCD and keypad for user interface, and talking RS-232 to some kind of logging station.

Dr. John Wavrik (March 1990)
University of California - San Diego
Mathematics Department

Dr. Wavrik's topic, "Teaching Forth," quickly centered on how an effective standard can improve the ability to teach Forth and make the language more credible. Whether you agree or disagree with some of its points and conclusions, this transcript should be required reading for anyone involved in teaching or standardizing Forth.—Gary

My assigned topic is “Teaching Forth,” and I’d like to talk about why Forth is a very good language for work in mathematics, and discuss how to teach it. Having a language taught is essential for its spread. It also has desirable side effects like establishing notational conventions and writing styles (so that users of the language can understand each others’ programs).

Forth is a very good language to teach. It can be understood in terms of how it

works—this is a powerful handle for learning anything. The only people who really have problems with it are those who think that a language must be defined by a BNF. Forth is very different from other languages—but not hard to learn if approached on its own terms.

Forth is also a good language to use in teaching since so much can be made visible which other languages hide. It can be used to help students gain insight into how computers represent data structures and implement algorithms. It can even provide an understanding of how conventional languages work.

I would be remiss, however, if I failed to mention that having Forth taught depends a great deal on whether the Forth community makes it teachable. A language cannot be taught if it is made unteachable.

Here is how to make a language unteachable:

1. Have no university-level texts.
2. Make it impossible for anyone to write a university-level text (see 3 & 4).
3. Make the language unstable, e.g.:
 - a. change the language's standards every few years, and
 - b. split the language into multiple dialects.
4. Make the language non-portable.
5. Have no pool of (portable) basic application packages.
6. Make it impossible for such a pool to be produced (see 3 & 4).
7. Fail to provide support to education.

**Allan Pratt, Jim Tittsler (April 1990)
Atari Product Design**

Tittsler: Here at Atari, we have been using Forth to “bring-up” and debug new computer and peripheral hardware for many years. It has a few significant advantages over other methodologies:

- The interactive, incremental nature of Forth allows the user to explore new hardware quickly and at the same time build on already-known components.
- Basic Forth literacy can be achieved quickly by hardware engineers whose goal is hardware verification, not software development.
- The amount of hardware that has to be correctly functioning to support a “significant” development-and-test environment is quite limited.

I consider Forth a reasonable *tool* for a large class of problems—but it is just one tool of many I may choose to use.

Pratt: I have a little less to say—I take over when Jim has brought up the hardware. I don't use Forth any more, but I guess I'm (in)famous for CFORTH, a small, portable Forth interpreter with the kernel written in C, not assembly. If you're using CFORTH, you should probably stop. Get a real Forth for your machine from the nice folks at FIG or someplace (like this library), because CFORTH is so weird it'll warp you for life. On the other hand, it *can* help you grok the very low-level stuff in a Forth interpreter if you understand C and not assembly.

**Tom Almy
Electrical Engineer, Tektronix, Inc.**

I have a deep concern with keeping Forth so that the “average” person (one who is not computer literate) can make use of the PRIDE CAD program without going nuts or having to get hold of me for each little problem. I also have taught Forth classes to over 100 people, and have seen where they have difficulties. Major changes in my system, some of which are being adopted:

- No screen files.
- Better number formats.
- “To” variables.
- All words, even control structures, work in the interpret state so that one doesn't need to be concerned about being in a colon definition.
- Graceful aborts on errors.
- Easy access to host OS.

The net result is that the users (mainly scientists with little background with programming) are able to write simple Forth programs and issue commands without problems.

The result is that these users have no fear of using Forth. Managers, of course, still do but they are happy with the results. Namely, the designers are able to create masks for ICs, hybrids—whatever their job—in short time, using a program that runs on a relatively inexpensive system, and with little training.

* * *

I have to believe that the insights offered in these guest conferences are the “stuff” that can make a significant difference in the way Forth is perceived and used. I will grant that interactive conferences can be trying, because they lack the intimacy one-on-one conversations provide. On the other hand, when was the last opportunity you had to talk to a Steve Sarns about an embedded Forth application, or to gain an instructor's keen perspective from Mahlon Kelly or John Wavrik? I suspect your only such opportunities are via the GENIE Forth RoundTable guest conferences, and if you failed to participate in these your chance is probably forever lost. Won't you join us in a forthcoming one and discover for yourself what you have missed?

* * *

Upcoming guests on GENIE's Forth RoundTable:

March 21	Roy Martens and Glen Haydon, “The Status of MVP in Today's Market”
April 18	Charles Johnsen, “Mutable Instruction Set Computers”
May 16	Bradford Rodriguez, “ANSI: Ask Not for Standards Improvement”
June 20	Jan Shepherd, “The Forth Interest Group”

(Continued from page 20.)

BOB REILING, his hard work, and the hard work of MIKE PERRY and JAN SHEPHERD.

Canada was represented by four attendees and a guest, Australia by RODNEY YOUNG, and Russia by IGOR AGAMIRZIAN.

In the final hour of the conference, MARTIN TRACY, WIL BADEN, GUY GROETKE, MITCH BRADLEY, and CHUCK MOORE gave their impromptu thoughts about the future of Forth. They differed in many things but they all believe that Forth has a mature, strong, and immortal life ahead.

— NEIL BAWD

INTERNATIONAL GENIE ACCESS

DENNIS RUFFER - D.RUFFER ON GENIE

Despite some of the difficulties being experienced within ForthNet, the GENIE leg is alive and expanding. In fact, GENIE is running at its lowest cost in history. Since the sign-up costs and user fees have changed recently, I will explain below what the latest information is. If you are not interested in this information, please disregard it. For those of you who are not totally set against capitalism, I've got some good news for you.

Whatever method you use to sign up, drop by and say hello in the Forth RoundTable (type FORTH at any main menu). My GE Mail address is D.RUFFER, drop me a note if you have any problems.

U.S.A.

To sign up, have your modem call 800-638-8369 using 300/1200/2400 baud, seven bits, no parity, and half duplex (i.e., local echo). After you connect, type HHH pausing slightly between the H's. GENIE will respond with a U# prompt. If it does not, hit return and it will. At the U# prompt, type XJM11849,GENIE and hit return. You will be taken into the sign-up module that will ask for your credit information (Visa, MasterCard, Discover, American Express, or direct checking withdrawal) and will tell you the closest access number. Your account should then be validated for your use with 24 hours. If you have any problems, call 800-638-9636.

Sign-up costs

Sign-up	\$ 0.00
Manual	12.95 optional
Postage	2.00 optional
Net Cost	\$ 0.00 or \$14.95

Plus, GENIE offers a money-back guarantee! We're inviting users to try GENIE for a month. After the first month, if they're not

satisfied, we will refund the monthly subscription fee.

Effective 10/1/90, we implemented a mandatory monthly subscription fee of \$4.95 per month. After extensive research, users (both ours and our competition's) have indicated that they would prefer unlimited access to select services for a set monthly fee. The most broadly accepted price point is \$4.95 per month. Thus, the introduction of GENIE's unlimited access services—during non-prime (local time) hours—which include over 100 products and services.

GENIE Services which do not incur additional charges:

News, Weather, Sports (top stories), Travel reservation information (including EasySabre), Closing Stock Quotations (U.S. markets only), Electronic encyclopedia, GE Mail (message send-and-receive only), Leisure & Professional BB's (just the BB, no RTC or SL's), Classic GENIE Games (single player games, not MP), Shopping Services, Free subscription to LiveWire (ten times per year, up from six times)

Canada

(All access numbers listed are tri-speed.)

Sign-up only:	800-387-8330
Vancouver	604-253-8429
Toronto	416-858-1230
Calgary	403-232-6121
Montreal	514-333-1117

The sign-up procedure and Client Services telephone number is the same as for the U.S.A. (See above.)

Japan

For information about signing up in Japan, use the following number to reach their Client Services or write them at the

following address:

Telephone: 03-452-9800
NEC VAN Marketing Division
Mita Kokusai
BIL. 5F 1-4-28
Mita, Minato-Ku Tokyo
Japan 108
Att'n: Yutaka Fujii

The current sign-up user number and password in Japan is XJM11719,GENIE and there is an 8000 yen sign-up fee.

GENIE via Public Data Networks

GENIE can be accessed from any of the following countries by using the state-owned Public Data Networks:

Australia, Belgium, Denmark, Finland, France, Hong Kong, Ireland, Italy, Mexico, Netherlands, New Zealand, Norway, Philippines, Portugal, Singapore, South Africa, South Korea, Spain, Sweden, Taiwan, United Kingdom.

In order to access GENIE through the above-listed countries, one must follow the procedures listed below (there is no sign-up fee):

1. Connect to the local PDN (public data network) by joining the local PDN service.
2. Enter the following information all on one line:
3136 (GE Information Services' DNIC) 9 (indicates PAD service) 00 (indicates asynchronous service)

Thus, enter:

3136900

or:

31369000000000

if the PDN requires the full X.121 address. Contact the PTT representative if there are any questions as to the correct addressing structure. GE Information Services will distinguish the access speed (300 bps or 1200 bps) automatically.

3. Follow the MARK*NET Service sign-on prompt U#= with the traditional user number and password:
XJM11997,PDN
4. After the signup, a startup package will be mailed. This includes a contract, which we need signed and returned before the account can be validated. If you choose, you can FAX the signed contract. (The fax's number appears on the contract.)

Europe

As with the PDN countries, there is no sign-up fee for GENie Europe. A manual is not included, but may be ordered for \$28.95 (which includes shipping and handling). For information and local sign-up procedures, call or write:

GE Information Service GmbH
Robert-Bosch Strasse 6
5030 Huerth-Efferen, Germany
49 (2233) 6091
Telex: 8 882 420

Hourly Connect Charges

	Prime	Non-Prime
U.S.A.	\$18	\$6
Canada	C\$25	C\$8
Japan	5400	5400 (yen)
PDN countries*	\$20	\$8
Europe	\$27	\$18

*Prices are in addition to the PDN charges. Prime time is in effect from 8 a.m. to 6 p.m. Eastern time (U.S.A.) on weekdays (i.e., Monday through Friday). Non-prime time is all other hours, including GENie holidays.

Total control with **LMI FORTH**TM

**For Programming Professionals:
an expanding family of compatible, high-performance, compilers for microcomputers**

For Development:
Interactive Forth-83 Interpreter/Compilers for MS-DOS, OS/2, and the 80386

- 16-bit and 32-bit implementations
- Full screen editor and assembler
- Uses standard operating system files
- 500 page manual written in plain English
- Support for graphics, floating point, native code generation

For Applications: Forth-83 Metacompiler

- Unique table-driven multi-pass Forth compiler
- Compiles compact ROMable or disk-based applications
- Excellent error handling
- Produces headerless code, compiles from intermediate states, and performs conditional compilation
- Cross-compiles to 8080, Z-80, 8088, 68000, 6502, 8051, 8096, 1802, 6303, 6809, 68HC11, 34010, V25, RTX-2000
- No license fee or royalty for compiled applications

LMI Laboratory Microsystems Incorporated
Post Office Box 10430, Marina del Rey, CA 90295
Phone Credit Card Orders to: (213) 306-7412
FAX: (213) 301-0761

ADVERTISERS INDEX

Forth Interest Group	44
Harvard Softworks	10
Journal of Forth Application & Research	21
Laboratory Microsystems	27
Miller Microcomputer Services	29
Next Generation Systems	17
Silicon Composers	2

REFERENCE SECTION

Forth Interest Group

The Forth Interest Group serves both expert and novice members with its network of chapters, *Forth Dimensions*, and conferences that regularly attract participants from around the world. 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

Board of Directors

Robert Reiling, President (*ret. director*)
Dennis Ruffer, Vice-President
John D. Hall, Treasurer
Wil Baden
Jack Brown
Mike Elola
Robert L. Smith

Founding Directors

William Ragsdale
Kim Harris
Dave Boulton
Dave Kilbridge
John James

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

ANS Forth

The following members of the ANS X3J14 Forth Standard Committee are available to personally carry your proposals and concerns to the committee. Please feel free to call or write to them directly:

Gary Betts
Unisyn
301 Main, penthouse #2
Longmont, CO 80501
303-924-9193

Mike Nemeth
CSC
10025 Locust St.
Glenn Dale, MD 20769
301-286-8313

Andrew Kobziar
NCR Medical Systems Group
950 Danby Rd.
Ithaca, NY 14850
607-273-5310

Elizabeth D. Rather
FORTH, Inc.
111 N. Sepulveda Blvd., suite 300
Manhattan Beach, CA 90266
213-372-8493

Charles Keane
Performance Packages, Inc.
515 Fourth Avenue
Watervleat, NY 12189-3703
518-274-4774

George Shaw
Shaw Laboratories
P.O. Box 3471
Hayward, CA 94540-3471
415-276-5953
David C. Petty
Digitel
125 Cambridge Park Dr.
Cambridge, MA 02140-2311

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 link**)
Call GENie local node, then type M710 or FORTH
SysOps: Dennis Ruffer (D.RUFFER), Leonard Morgenstern (NMORGENSTERN), Gary Smith (GARY-S)
- MACH2 RoundTable
Type M450 or MACH2
Palo Alto Shipping Company
SysOp: Waymen Askey (D.MILEY)

BIX (ByteNet)

For information, call 800-227-2983

- Forth Conference
Access BIX via TymeNet, then type j
forth
Type FORTH at the : prompt
SysOp: Phil Wasson (PWASSON)
- LMI Conference
Type LMI at the : prompt
Laboratory Microsystems products
Host: Ray Duncan (RDUNCAN)

CompuServe

For information, call 800-848-8990

- Creative Solutions Conference
Type !Go FORTH
SysOps: Don Colburn, Zach Zachariah, Ward McFarland, Jon Bryan, Greg Guerin, John Baxter, John Jeppson
- Computer Language Magazine Conference
Type !Go CLM
SysOps: Jim Kyle, Jeff Brenton, Chip Rabinowitz, Regina Starr Ridley

Unix BBS's with forth.conf (ForthNet links and reachable via StarLink node*

9533 on TymNet and PC-Pursuit node casfa on TeleNet.)

- WELL Forth conference
Access WELL via CompuserveNet or 415-332-6106
Fairwitness: Jack Woehr (jax)
- Wetware Forth conference
415-753-5265
Fairwitness: Gary Smith (gars)

PC Board BBS's devoted to Forth (ForthNet links*)

- British Columbia Forth Board
604-434-5886
SysOp: Jack Brown
- Grapevine
501-753-8121 to register
501-753-6389
StarLink node 9858
SysOp: Jim Wenzel
- Real-Time Control Forth Board
303-278-0364
StarLink node 2584 on TymNet
PC-Pursuit node coden on TeleNet
SysOp: Jack Woehr

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
- Drama Forth Board
512-323-2402
StarLink node 1306 on TymNet
SysOps: S. Suresh, James Martin, Anne Moore

Non-Forth-specific BBS's with extensive Forth libraries

- DataBit
Alexandria, VA
703-719-9648
PCPursuit node dcwas
StarLink node 2262
SysOp=Ken Flower
- The Cave
San Jose, CA
408-259-8098
PCPursuit node casjo
StarLink node 6450
SysOp=Roger Lee

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 link*)
United Kingdom
0905 754157
SysOp: Jon Brooks
- Sky Port (ForthNet link*)
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 Dalt, 104-106, Entlo.
4-5
08024 Barcelona, Spain
+ 34 3 2103355 (voice)
+ 34 3 2147262 (modem)
SysOps: Jesus Consuegra, Juanma Barranquero
barran@nexus.nsi.es (preferred)
barran@nsi.es
barran (on BIX)

This list was accurate as of February 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!lrark!glslrk!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.*

**MAKE YOUR SMALL COMPUTER
THINK BIG**

(We've been doing it since 1977 for IBM PC, XT, AT, PS2, and TRS-80 models 1, 3, 4 & 4P.)

FOR THE OFFICE — Simplify and speed your work with our outstanding word processing, database handlers, and general ledger software. They are easy to use, powerful, with executive-look print-outs, reasonable site license costs and comfortable, reliable support. Ralph K. Andriat, author/historian, says: "FORTHWRITE lets me concentrate on my manuscript, not the computer." Stewart Johnson, Boston Mailing Co., says: "We use DATAHANDLER-PLUS because it's the best we've seen."

MMSFORTH System Disk from \$179.95
Modular pricing — Integrate with System Disk only what you need:

FORTHWRITE - Wordprocessor	\$99.95
DATAHANDLER - Database	\$99.95
DATAHANDLER-PLUS - Database	\$99.95
FORTHCOM - for Communications	\$49.95
GENERAL LEDGER - Accounting System	\$250.00

mmsFORTH

MILLER MICROCOMPUTER SERVICES
61 Lake Shore Road, Natick, MA 01760
(508/653-8138, 9 am - 9 pm)

FOR PROGRAMMERS — Build programs FASTER and SMALLER with our "Intelligent" MMSFORTH System and applications modules, plus the famous MMSFORTH continuing support. Most modules include source code. Forren Macintosh, spreadsheet, says: "Forth is the language that microcomputers were invented to run."

SOFTWARE MANUFACTURERS — Efficient software tools save time and money. MMSFORTH's flexibility, compactness and speed have resulted in better products in less time for a wide range of software developers including Ashton-Tate, Excelsior Technologies, Lindbergh Systems, Lockheed Martin and Space Division, and NASA-Goddard.

MMSFORTH VLSI System Disk from \$179.95
Needs only 2M RAM compared to 12K for BASIC, C, Pascal and others. Convert your computer into a Forth virtual machine with sophisticated Forth editor and related tools. This can result in 4 to 10 times greater productivity.

Modular pricing — Integrate with System Disk only what you need:

EXPERT-2 - Expert System Development	\$69.95
FORTHCOM - Enable data transfer	\$49.95
UTILITIES - Graphics, 8087 support and other facilities.	

and a little more!

THIRTY-DAY FREE OFFER — Free MMSFORTH GAMES DISK worth \$39.95, with purchase of MMSFORTH System. CRYPTOQUOTE HELPER, OTHELLO, BREAK-FORTH and others.

Call for free brochure, technical info or pricing details.

Sponsored by the Silicon Valley and North Bay FIG Chapters

NORTHERN CALIFORNIA FORTH DAY

C.H. TING - SAN MATEO, CALIFORNIA

November 17, 1990
10:00 a.m. - 5:00 p.m.

Presentations

ANSI Forth Standard Review

John Rible and ANSI X3J14 members

Basis 14 will be released shortly. It is expected that the final draft will be completed in the January meeting next year. John specifically clarified the concepts of standard systems and standard programs in the context of ANS Forth Standard.

Catch-Throw Error Handling

Mitch Bradley

200,000 Forth systems have been shipped by Sun Microsystems, in the form of Forth boot ROMs. All Sun Sparc stations have Forth in them, and all S-Bus peripherals are required to have Forth on board for booting their drivers. Mitch claimed that S-Bus will be the bus of choice for computerized hardware.

CATCH-THROW requires only ten lines of code for two Forth words. It is postfix and requires no immediate actions. Fully nestable, no hidden run-time code, implementable in high level, and no metacompilation implications. Usage and code examples are also presented.

Safety Net Error Handling

Leonard Morgenstern

This is another way to provide error handling in Forth systems. You use SET-NET to initialize a net, which is a six-byte array. FALL allows you to jump to a net when an error condition is asserted. It is nestable if additional words like PUSH-NET and POP-NET are provided. It is also usable as coroutine, about which people disagreed on the definition and functionality.

Critique of eForth

Mike Elola

Mike reviewed the design of eForth. Its main feature is the separation of the user interface from the system functions. The user interface is left out of eForth, because it can be handled much better by the host computer. Only the system functions are needed in eForth residing in a target CPU. It is something CPU board manufacturers might be interested in. If it is packaged and promoted properly, it may become a salable product.

80196 Target Forth

Mike Mayo

Mike was asked to repeat his presentation from the October meeting. He presented many interesting diagrams showing the functional parts in the 80196 Target Compiler, relating to code development, target code testing, and debugging tools. He keeps a ten-byte record for every byte compiled into the target, so that the host can support single-instruction-stepping and source-level debugging in real time.

YAOOP—Yet Another Object Oriented Program

Leonard Morgenstern

Only three Forth words are needed: CLASS, METHOD, and ATTRIBUTE. CLASS defines an object with a number of execution methods, which can be selected at run time. It is very simple and highly portable. The only drawback is that it does not support full inheritance. The full paper was presented at the Asilomar FORML Conference.

Transputer Forth

Bob Barr

The Forth for transputer is based on an F83-68000 implementation. A special board was designed to support a solid-state disk emulator at Mass Memory Technology. It takes advantages of the 4K internal memory, 32-bit data and address space, 20

MHz clock rate, and the four high-speed serial communication lines.

SC/Forth for RTX and SC32

George Nicol

George reviewed the product line of Silicon Composers. SC/Forth has been implemented across all products which run different CPUs, like the NC4000, RTX2000, and SC32. The portability of a single language model greatly eased product introductions and software development. The newest product is a VME single-board computer based on the SC32 processor.

Mechanizing Power/Mod Computation

Alfred Tang

Alfred described a unique method to reduce the computational load of raising a big number to a very high power, and taking the modulus of the product. This method can be very effective in building hardware for the public key encryption-description system based on the RSA algorithm. The math is fully disclosed.

Programming Contest

Contestants were given a text file containing Psalm 119, and were asked to count the number of words, the number of unique words, and to produce an alphabetically sorted list of all the unique words. The file is about 16K. There are about 1500 words and 500 unique words in it. The Most Honorable Cowboy Chuck Moore presided as judge.

Six braves entered the contest. Mitch Bradley brought a formidable Sparc1 station fully loaded. He used Unix shells and pipes to finish the counting and sorting in eight minutes, but was asked to go back and write a program to do the job. Andrew McKewen completed the counting and sorting in 42 minutes, and Mitch completed his in 54 minutes. Andrew was awarded a

(Continued on page 41.)

METACOMPILATION MADE EASY

FRANK SERGEANT - SAN MARCOS, TEXAS

Metacompilation prowls the narrow alleys of Forth, lurking in the shadows, scaring the children. Does the monster really exist or is it just a legend? (Eerie music.) It's nearly midnight, wanna go out and look for it, huh? Get the flashlight, a fifth of Scotch, a 12 gauge shotgun and come with me. Don't be afraid. Here, let me hold the shotgun. No, no, after *you*.

The flashlight is all we need. Metacompilers can be monsters, but shining a little light on them should dispel the terror. We'll take a look at them in general and also discuss a specific, simple version from Pygmy Forth. But first...

cmFORTH's novel and elegant method has just two vocabularies.

What Is A Compiler?

A compiler is a program that *lays down* code and data that matches your source code. It is usually thought of as a translator that converts source code into machine code. It might be more helpful, though, to think of it as something that steps through memory, byte by byte, filling in whatever should be put there.

Most languages have a compiler. Forth has *many* compilers.

The most basic compilers in Forth are C, ("C-comma") and , ("comma"). They take numbers from the stack and lay them down at the next free byte or word in the dictionary (and then increment the dictionary pointer). : ("colon") is the most commonly used compiler in Forth. It uses C, and , to lay down bytes and words into the dictionary as it compiles new definitions.

```

16 CONSTANT TMAX-FILES    ( allow room in tgt for 15 files, )
                          ( but MUST be a power of 2 )

2 1 - CONSTANT TNB        ( set number of disk buffers )

VARIABLE RAM              ( next system variable will go )

VARIABLE H' $8000 ,       ( relocation amount )
                          ( 1st cell is tgt's DP & )
                          ( 2nd cell is tgt's offset )

$8000 $2000 0 FILL        ( initialize target space to zeroes )

$8000 H' !                ( start target dictionary at $8000 )

 3 13 THRU                ( load the metacompiler extensions )

17 80 THRU                ( load the kernel )

PRUNE                     ( relink the target dictionary )

(                          ( switch to target dictionary )

$8100 HERE SAVEM KERNEL.COM

)                          ( switch back to host dictionary )

```

Figure One. Load screen for metacompiling a new kernel.

These variables hold the address of the corresponding target run-time routines. These are needed by the metacompiler so it can compile the proper code. As each run-time routine is defined in the target, its address is stored into its corresponding variable.

```

VARIABLE TVAR ( variable)      VARIABLE TLIT ( literal)
VARIABLE TCOL ( docol)         VARIABLE TBRA ( branch)
VARIABLE TOBR ( zero branch)   VARIABLE TEXIT ( exit)
VARIABLE TFOR ( for)           VARIABLE TNEXT ( next)
VARIABLE TARR ( array)         VARIABLE TABORT ( abort")
VARIABLE TDOT ( dot")         VARIABLE TNULL

```

Figure Two.

The curly braces switch back and forth between the host and target spaces, by swapping the dictionary pointers and relocation factors. This allows the regular comma, C-comma, and HERE to access both the host and target dictionaries.

```

: { ( -) dA @ HERE H' 2@ H ! dA ! H' 2! ;
: } ( -) { ;

```

Figure Three.

TCREATE creates a header for a word in the target dictionary. It lays down a two-byte link field that points to the previous word's link field. Then it lays down a three-byte jump to the run-time routine for VARIABLE. This new word's link field address is stored in the vocabulary head (CONTEXT @ HASH).

Defining words other than VARIABLE will un-compile those three bytes (with -3 ALLOT etc.) and lay down a jump to their own run-time routine.

```

HEX
: TCREATE ( -)
( 2byte link, counted name, & 3 byte jump to targets var)
( Meta's TVAR holds var's addr as soon as we know it)
HERE 0, 20 WORD ( cur.lfa cur.nfa )
CONTEXT @ HASH ( lfa nfa vocab )
2DUP ( cur.lfa cur.nfa vocab cur.nfa vocab )
@ ( cur.lfa cur.nfa vocab cur.nfa prev.lfa)
SWAP ( cur.lfa cur.nfa vocab prev.lfa cur.nfa)
2 - ( back up) ( cur.lfa cur.nfa vocab prev.lfa cur.lfa)
! ( cur.lfa cur.nfa vocab)
SWAP ( cur.lfa vocab cur.nfa)
C@ ( cur.lfa vocab len)
1+ ALLOT ( comma in the entire name field)
! ( make vocab point to this new word's link field )
TVAR @ LJMP, ( lay down 3byte jump to dovar) ;

```

Figure Four.

forget (-) Smudges the name field of the most recently defined (target) word, making it temporarily un-findable in the dictionary. This is rarely needed.

CONSTANT (n -) Defines a constant in the target dictionary. Its value cannot be used directly during metacompilation.

VARIABLE (-) Defines a variable in the target dictionary.

ARRAY (a -) (n -) (n is a word, not byte, index)
Defines an array of 16-bit numbers in the target dictionary.

DEFER (-) Defines a word in the target dictionary that can be changed to execute different words by the word IS.

IS (pfa -) Stores the address a into the following DEFER'd word in the target dictionary.

```

HEX
: forget ( -) CONTEXT @ HASH @ 2 + DUP C@ 20 XOR SWAP C! ;
: CONSTANT ( n -) TCREATE -3 ALLOT
  BX PUSH, #, BX MOV, NXT, ; ( use "in-line" constants )
: VARIABLE ( -)
  ( RAM @ CONSTANT 2 RAM +! for ROMing)
  TCREATE 0, ;
: ARRAY ( a -) ( n -) ( n is a word, not byte, index)
  TCREATE -3 ALLOT TARR @ LJMP, , ;
: DEFER ( ) ( ... ) TCREATE -3 ALLOT 0 #, AX MOV, AX JMP, ;
: IS ( pfa -) dA @ - ' 1+ ! ;

```

Figure Five.

This is usually done in a number of steps, with the help of various compiling words, each of which does its part. For example, IF starts an IF ... THEN structure by laying down a conditional branch. THEN completes the branch. VARIABLE and CONSTANT are other familiar Forth compilers. Ultimately, they all rely upon , and C, to get those bytes laid down.

How Forth Is Different

Most other languages completely separate the compiler itself from the result of the compiler. You take source code, feed it into the compiler, and get object code. The object code is later executed independently of the compiler. Forth is different in that, *usually*, its compilers *extend* the current Forth system rather than creating an independent system. Your new definitions are added to the definitions that are already present.

What Is Metacompilation?

Simply put, metacompilation is the process of making Forth do what other languages do all the time: generate an independent system. The fancy name probably contributes a lot to the confusion. When you extend the dictionary in Forth you call it compilation. When you generate an independent system with Forth you call it metacompilation. (In most other languages, the only thing you *can* do is generate an independent system.)

Come on, there's got to be more to it than that, you say? Well, okay. The "meta" means self-referencing. In other words, when what you compile is *another Forth system*, it's called metacompilation. Using the same logic, if you used a C compiler to compile another C compiler you could also call it metacompilation. (As an aside, you don't often hear of that, do you? No! Forthers are almost the only people generally willing to build their own systems. Of course, some might say we're the only ones who need to.)

Target Compilation

Well, the above explains where "meta" originated, but it isn't the full story. Another term, "target compilation" complicates things slightly. What if you want to generate an independent system that is *not* a Forth system, perhaps a short utility program? In that case, you might call it target compilation instead of metacompilation.

What if you use your Forth system to generate independent code designed to run on a different processor? Its "target" is the other processor, so clearly this would be target compilation. But what if that independent system, designed to run on the other processor, is another Forth system? Is it target compilation, metacompilation, or meta-target-compilation? I don't think this leads anywhere, so I suggest we just use the word metacompilation for all cases where an independent system is generated.

Other Definitions

By target system or dictionary, we mean the result of the metacompilation. By host system or dictionary, we mean our regular Forth system. By the way, in cmFORTH and Pygmy, \ is used instead of [COMPILE]—it does not indicate a comment. PUSH and POP are used instead of >R and R>.

Uses of a Metacompiler

Here are the main uses I can think of for a metacompiler:

1. Use your current Forth system to make a slightly changed version of itself to run on the current hardware.
2. Use your current Forth system to make a new (perhaps drastically different) Forth system to run on the current hardware.
3. Use your current Forth system to make a new Forth system to run on a different processor or operating system.
4. Use your current Forth system to make a standalone utility or application to run on your current hardware.
5. Use your current Forth system to make a standalone utility or application to run on different hardware.

Problems

There are only two main problems that metacompilers must solve: logical versus physical addresses, and arranging and searching the dictionary so the right word is found at the right time.

Logical vs. Physical Addresses

Whereas the regular compiler merely extends the dictionary, the metacompiler

These words re-link the target dictionary so that the correct values will be present in the link fields.

SCAN (lfa - lfa) Follows the link fields backwards to find the next one that is in the target dictionary space.

TRIM (lfa new-lfa - new-lfa) Converts the previous link field address from a physical address to a logical address and stores it in the current word's link field, then unsmudges the current word's name field in case it is smudged.

CLIP (voc-head -) Goes through all the words in a vocabulary, TRIMing each one, then links the first word to the null word.

PRUNE (-) CLIPs each of the target's vocabularies.

```
: SCAN ( lfa - lfa ) @ BEGIN DUP 1 $8000 WITHIN WHILE @ REPEAT ;
: TRIM ( lfa new-lfa - new-lfa ) DUP PUSH dA @ - SWAP ! POP
  DUP 2 + DUP C@ $DF AND SWAP C! ( unsmudge ) ;
: CLIP ( voc-head - ) DUP BEGIN DUP SCAN DUP WHILE TRIM REPEAT
  DROP TNULL @ dA @ - SWAP ! @ , ;
: PRUNE ( - ) ( 8 HASH CLIP 6 HASH CLIP
  TNULL @ OFF ( zero out its link field ) { ;
```

Figure Six.

Some regular words need to be re-named so we can still get to them after we have defined metacompiler versions of them.

```
FORTH' ( - ) Sets CONTEXT to the host's FORTH vocabulary.
COMPILER' ( - ) Sets CONTEXT to the host's COMPILER vocabulary.
\' ( - ) Compiles the following word in the input stream from the host's COMPILER
  vocabulary.
:' ( - ) Starts a colon definition in the host's FORTH vocabulary.
dA@- ( a - a' ) Converts a target physical address to a logical address.

: FORTH' FORTH ;
: COMPILER' COMPILER ;
COMPILER : \' \ \ ; FORTH
: : ' : ;
: dA@- dA @ - ; ( this is used often )
```

Figure Seven.

LITERAL (n -) Compiles a 16-bit literal into the target.

] () (-) is the metacompiler's colon compiler. It looks up the next word from the input stream and does something with it. If the word is in the host's COMPILER, the word is executed. Else, if the word is in the target's FORTH, the word is compiled. Else, if the "word" is a valid number in the current base, it is compiled as a literal. Else, an error is reported. Sequence is everything. This approach, plus the ability to switch between host and target spaces with { and }, is the secret to the simplicity of the metacompiler.

```
COMPILER
: LITERAL ( n - ) TLIT @ ,A , ;
FORTH

: ] BEGIN 4 -' ( restrict execution to host's COMPILER)
  IF 6 -FIND ( restrict finding to target's FORTH )
    IF NUMBER \ LITERAL
      ELSE ,A
      THEN
    ELSE EXECUTE
    THEN
  AGAIN ;
```

Figure Eight.

\ (-) compiles the following word from target's COMPILER.

Metacompiler words to create target looping and branching structures within target colon definitions:

BEGIN (- a)	Saves the address to branch back to.
UNTIL (a -)	Compiles a conditional branch to address a.
AGAIN (a -)	Compiles an unconditional branch to address a.
THEN (a -)	Completes a branch started by IF ELSE etc.
IF (- a)	Starts a conditional branch.
WHILE (a - a a)	Starts another conditional branch.
REPEAT (a a -)	Completes a BEGIN WHILE structure.
ELSE (a - a)	Starts the false part of an IF statement.
FOR (- a)	Starts a FOR NEXT loop.
NEXT (a -)	Completes a FOR NEXT loop.

```
COMPILER
: \ 8 -' ABORT" ?" ,A ; ( F83's [COMPILE] )
: BEGIN ( - a ) HERE ;
: UNTIL ( a - ) TOBR @ ,A ,A ;
: AGAIN ( a - ) TBRA @ ,A ,A ;
: THEN ( a - ) HERE dA @ - SWAP ! ;
: IF ( - a ) TOBR @ ,A HERE 0 , ;
: WHILE ( a - a a ) \ ' IF SWAP ;
: REPEAT ( a a - ) \ ' AGAIN \ ' THEN ;
: ELSE ( a - a ) TBRA @ ,A HERE 0 , SWAP \ ' THEN ;
: FOR ( h - ) TFOR @ ,A \ ' BEGIN 0 , ;
  ( performs u times instead of u+1 times )
: NEXT ( h - ) DUP \ ' THEN 2 + TNEXT @ ,A ,A ;
FORTH
```

Figure Nine.

needs to lay down code somewhere else, usually referred to as the target space or target dictionary. The address where this code is laid down is *not* the address where it will sit later, when it is executed. For example, if you are generating code that will later be loaded at address zero, you might build it at address \$8000 (or wherever you enough free memory to hold it).

When it is an *address* that you are laying down, you have two things to consider: its physical address (where it is now, at compile time) and its logical address (where you plan for it to be later, when it actually executes). Suppose we wanted to compile the address of the following 16 bits into address \$8000. The physical address of the word is \$8002. But, later executed, what's at \$8000 will be at \$0000 (and what's at \$8002 will be at \$0002); we must store \$0002 into address \$8000. Thus, the metacompiler needs a method of converting between logical and physical addresses.

Search Order

When you want to compile a definition into the target space, where do you look for the words, and what about duplicates? Should the word be compiled or executed? Suppose you want to define the following word:

```
: DO-NOTHING ( n - n )
  DUP SWAP DROP ;
```

Let's take the words left to right. You don't want to use the host colon that compiles into the host dictionary. Instead you want to find and execute the metacompiler version. Then, you want to find DUP, SWAP, and DROP, not in the host dictionary, but in the target dictionary. And you want to compile them rather than execute them. For the semi-colon, the metacompiler version from the host must be found and executed.

There are various tricks for looking in the right place at the right time. It can also be very simple. Getting this search order right is the heart of the metacompiler. Now let's look at a simple way of solving these problems.

cmFORTH and Metacompilation

Charles Moore invented a beautifully simple metacompiler in his cmFORTH for the Novix. I used this general approach in Pygmy Forth for MS-DOS machines.

The words { and } (left and right curly

braces) switch between the regular dictionary and the target space, e.g.,

```
{ : ABC DUP SWAP DROP ;
: CDE ABC ABC ;
: FGH ABC CDE ;
}
```

would compile those three words into the target space and then return the dictionary pointer back to the regular dictionary.

Rather than having separate pointers for the two spaces, and two versions of HERE (perhaps HERE and THERE), the curly braces switch the value stored at H (the dictionary pointer, called DP in some Forths). By handling the change at this low level, the need for separate versions of comma, C-comma, and HERE disappears.

Relocation of addresses is handled by the words dA and ,A. dA ("delta-address") is a system variable that holds the relocation factor. It is used by ,A ("comma-address") both in the regular Forth system (when dA has a value of zero) and when metacompiling (when dA has a value of \$8000).

The above takes care of the simple problem of compiling code at one address to run later at a different address. The more difficult problem is search order: how do we find the correct word to execute or compile?

In some metacompilers this gets very complex, with complicated switching of vocabularies and testing for whether we are in the target dictionary or not. Fortunately there is a simpler method.

Charles Moore solved the problem (and some others) in cmFORTH with the novel and elegant method of having just two vocabularies: COMPILER and FORTH. Words in COMPILER are immediate. All the other words are in FORTH. When interpreting, words are looked for only in FORTH. When compiling, COMPILER is searched first. If the word is found, it is executed (it is an "immediate" word); else FORTH is searched and the address is compiled. New words are linked into either FORTH or COMPILER depending on the value of CONTEXT. Thus, there is no IMMEDIATE word. Thus, you can have two versions of a word with the same name, one in each of the vocabularies. One only works when compiling and the other only when interpreting. For example, you can have the word ." in COMPILER that compiles a

```
ABORT" ( -) ( flag-)      Compiles following error message.
."      ( -) ( -)        Compiles following string to be typed later.
[']     ( -)             Compiles pfa of following word from target's FORTH
                        as a 16-bit literal.
FORTH   ( -)             Sets CONTEXT to target's FORTH.
COMPILER ( -)            Sets CONTEXT to target's COMPILER.
: ( -)      Starts a colon definition in target dictionary. Note that it lays down
                        a jump (it's faster) and not a call.
; ( -)      Ends a colon definition in target dictionary.

HEX
COMPILER
: ABORT" TABORT @ ,A 22 STRING ;
: ."     TDOT  @ ,A 22 STRING ;
: [']    TLIT  @ ,A ;
FORTH

: FORTH 6 CONTEXT ! ;
: COMPILER 8 CONTEXT ! ;

: : TCREATE -3 ALLOT TCOL @ LJMP,
      ( lay down 3byte jump to docol) forget ] ;

COMPILER'
: ' ; forget POP DROP TEXTIT @ ,A ; ( must be the last colon)
                                      ( def in the metacompiler)

FORTH'
```

Figure Ten.

string for later printing, and the word ." in FORTH that prints the string right now. This eliminates the need for the abomination . (, which some Forths use to print a string while interpreting.

The vocabulary heads immediately precede the variable CONTEXT. Thus, in Pygmy Forth, CONTEXT 2 - is the address of the vocabulary head for FORTH and CONTEXT 4 - is the address of the vocabulary head for COMPILER. Since the Novix is a cell-addressed machine, instead of a byte-addressed machine, the offsets in cmFORTH are 1 for FORTH and 2 for COMPILER, rather than 2 and 4 as in Pygmy.

cmFORTH uses just those two vocabularies, both for regular and meta-compilation. To make my life a little easier with Pygmy, I added two more. It has a FORTH and COMPILER for the host dictionary and another set of FORTH and COMPILER for the target dictionary. Thus, offsets from CONTEXT of 2 and 4 point to the host FORTH and COMPILER voc-heads and 6

and 8 point to the target FORTH and COMPILER voc-heads. This is not strictly necessary, but makes it simpler to follow what you are doing and to control exactly where you are searching. These extra two vocabularies are used only when metacompiling.

Look at the code for INTERPRET in Figure Thirteen. It only searches host's FORTH. If the word is found it is executed. No problem, and no confusion with words of the same name that have been compiled into the target dictionary. This gives us access to our regular Forth system for saving memory images to disk, for poking around in memory with DUMP, for editing, examining the stack, etc.

The same search mechanisms such as -FIND, ', and -' that are used with the host dictionary work fine without any changes with the target dictionary.

Look at the code for] (the metacompiler's colon compiler) in Figure Fourteen. Here we explicitly control the search. Within a colon definition we first search the host's COMPILER. This finds

Start of the Pygmy Forth Kernel. 6 and 8 are the CONTEXT values for the target's FORTH and COMPILER vocabularies.

boot (-) is where the execution of the PYGMY.COM file starts. It is made headerless by the -7 ALLOT. It initializes the two stack pointers and jumps to the word reset.

\$ (-) is the "null" word. The dollar sign is not its real name. Its real "name" has a length of zero! It is named \$, then the -2 ALLOT erases the dollar sign and length of 1. The 0 C, changes the length to zero. When null is executed it exits an endless loop. This happens, for example, when the input stream is exhausted.

```
HEX
6 HASH OFF
8 HASH OFF

( ( to target)

100 ALLOT ( first 256 bytes are reserved for DOS)
-7 ALLOT ( align pfa of BOOT to $0100 )

FORTH ( sets context to 6 )

CODE boot ( for now leave stacks & everything in one 64K seg)
FE00 #, BP MOV, ( initialize return stack)
FE00 #, SP MOV, ( initialize parameter stk)
0 #, AX MOV, ( addr of reset - patch it later)
AX JMP, ( jump to "reset") END-CODE

HERE TNULL ! ( following is null word that will get renamed)
CODE $ -2 ALLOT 0 C, SWITCH, SI POP, SWITCH, NXT, END-CODE

HERE dA @ - RAM !
2A TNB 1+ 2* + ALLOT ( leave room for system variables)
```

Figure Eleven.

the words IF, ELSE, THEN, BEGIN, WHILE, REPEAT, ABORT", and such. In other words, the immediate words. These words are the metacompiler versions that are defined in the host's dictionary. These words are *executed*. They affect the compilation, usually laying down special run-time target code. They are not confused with the host's regular versions of these same words, even though they are also in host's COMPILER because the metacompiler versions were defined after the regular versions. Any of the regular versions of words in either host's FORTH or COMPILER that might be needed later must be renamed. See Figure Seven for an example of renaming FORTH, COMPILER, /, and : so we can access them later, after the metacompiler versions with the same names are defined. Note that with this system, only those four need to be renamed.

If the word is not found in host's COMPILER, then we look in target's FORTH. If found, its address is compiled into the tar-

get dictionary. Thus, there is no danger of finding the host's version of that word by mistake. If the word is not found in either host's COMPILER or in target's FORTH, then an attempt is made to convert it to a number and compile it as a literal. If that fails, an error message is displayed.

Let's take a closer look at how the defining words such as IF, ELSE, LITERAL, ABORT", etc. work. They must be defined in host's COMPILER so they will execute during the definition of a target word to lay down a target run-time word that hasn't been defined yet. How do we handle this impasse?

There are two general approaches. The first is to sprinkle these defining words throughout the target code. For example, IF must lay down the code for 0branch. We could switch to the target dictionary and define 0branch and then switch back to the host dictionary and define the metacompiler version of IF that uses it. This is a very workable method, but I prefer to

group all the metacompiler definitions together in one place and then group all the target definitions in another, rather than interleaving them.

The second approach, which is used in Pygmy, is to define variables to hold the addresses of those run-time words (Figure Two). Then we can go ahead and define IF, for example, to fetch the contents of the variable T0BR and lay it down. At the time we define IF, the address of the run-time routine 0branch is not known. Then, in the target definitions, as soon as we define 0branch we store its address into T0BRA. Of course, either way we must not use the word IF in a target definition until after 0branch has been defined! Figures Five, Nine, and Ten show how the defining words use these variables. Figure Twelve shows the target definitions for the run-time words var, 0branch, and branch and how their related metacompiler variables TVAR, T0BR, and TBRA get initialized.

cmFORTH has it a little easier. It doesn't call run-time routines, such as 0branch and lit. Instead, these are laid down as in-line machine code. Thus, they *are* known at the time IF, LITERAL, etc. are defined.

As the target image is being compiled, the link fields contain physical addresses so the host system can find target words. When finished, before saving the target image to disk, these link addresses must be converted to the correct logical addresses by applying the relocation factor from dA to them. SCAN, TRIM, CLIP, and PRUNE in Figure Six take care of this re-linking and a few other housekeeping details.

Using It

Actually using the metacompiler in PygmyForth is very easy. Just type 1 LOAD (Figure One) and a new kernel will be generated automatically. Screen One loads the metacompiler and then loads the source code for the Pygmy kernel, then re-links the target dictionary and saves the image to disk as a .COM file. At this point you can return to DOS with BYE and run the new kernel. All the code for the metacompiler, the kernel, and the editor and assembler is in the single file PYGMY.SCR.

Pygmy Forth (for MS-DOS machines) is available on disk from FIG or by downloading from GENIE or other bulletin boards. Look for version 1.3. If you haven't

Examples of target run-time code words.

lit (- n) is the run-time routine for a literal. It pushes the following 16-bit word to the data stack.

array (n - a) is the run-time routine for an array. It pushes the address of the nth 16-bit entry of the array to the data stack.

var (- a) is the run-time routine for a variable. It pushes the address of the variable to the data stack.

Obranch (n -) If the top of stack is non-zero, the following in-line address is skipped. If the top of stack is zero, a jump to the following in-line address is taken. "zero branch"

branch (-) An unconditional jump to the following in-line address is taken.

Following is how a variable is compiled into the dictionary:

```
LINK,NAME,JMP<var>,VALUE
 2   ?   3   2      (# of bytes in each field)
```

```
CODE lit ( -n)  HERE TLIT !
    BX PUSH,    ( push TOS to SOS)
    AX LODS,    ( ax <-- [IP], IP++ )
                ( get in-line value, not addr)
    AX BX MOV,  ( to TOS)
    NXT,
END-CODE

CODE array ( n -a)  HERE TARR ! ( nth word index into array )
    3 #, AX ADD,  ( jump over 3 byte JMP)
    AX BX XCHG,
    0 [BX] BX MOV,
    1 #, AX SHL,  ( multiply by 2 to addr nth word)
    AX BX ADD,  ( now TOS holds addr of nth word of array)
    NXT,  END-CODE

CODE var  HERE TVAR !
    BX PUSH,    ( push TOS to SOS)
    3 #, AX ADD,  ( jump over 3 byte JMP)
    AX BX MOV,  ( put that addr in TOS)
    NXT,  END-CODE

CODE Obranch  HERE TOBR !
    AX LODS,  BX BX TEST,
    0=, IF, AX SI MOV, THEN,
    BX POP,
    NXT,      END-CODE

CODE branch  HERE TBRA !
    0 [SI] SI MOV,  NXT,  END-CODE
```

Figure Twelve.

(Figures continued on page 41.)

tried metacompiling, this is an easy way to experiment with it. The first step is to type 1 LOAD without making any changes, to get a feel for the process. Then start making minor changes to the kernel, such as the number of disk buffers or the default colors. As your confidence and experience with it grows, you can tackle more adventurous changes.

Well, we shined the light nearly everywhere and found no monster to fear. Let's put on the safety on the shotgun and finish that bottle of Scotch.

Frank Sergeant is a hardware and software consultant specializing in business and real-time systems. He is the author and implementor of Pygmy Forth.

(Continued from page 5.)

as Directors before; and Mike Elola and Jack Brown are continuing Directors. The first meeting of the Board was held at FORML, where new officers of the corporation were elected and the Board decided to increase the membership fees to \$40. The Board's next meeting was planned for January in southern California. [After press time for this issue. —Ed.]

The officers—John Hall, President; C.H. Ting, Vice-President; Mike Elola, Secretary; and Dennis Ruffer, Treasurer—have started to reshape the functions of FIG. The officer's job is to carry out the directions of the Board of Directors, serving as leaders in carrying out those directions and performing the day-to-day functions of running the corporation. One of the President's duties is to preside at the Board of Directors meetings and the monthly business meeting; another is to appoint committees or individuals to perform needed jobs or to explore the implications of potential useful services to our members. Call me with ideas!

The monthly business meeting (held the Tuesday evening before the fourth Saturday) is where the decisions of the day-to-day functions get made. The members of the Business Group consist of the officers and other individual members who are interested in the business of FIG. The Business Group is open to all members, each of whom—on all decisions of the Business Group—can participate equally. Call the FIG business office. Tell us you are coming!

In the next *Forth Dimensions*, I want to explore issues that I feel need to be discussed. In the meantime, with your ideas, call me!

—John Hall
415-535-1294
(after 7:00 p.m. Pacific time)

Attention Forth Authors!

AUTHOR RECOGNITION PROGRAM

To recognize and reward authors of Forth-related articles, the Forth Interest Group (FIG) has adopted the following 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.

WELCOME NEW MEMBERS

The Forth Interest Group welcomes the new members who joined us during the past year, including:

Akademii Nauk SSSR Biblioteka
Fejer Megyei Vizmuvek
Hiradastechnika Szovetkezet
IAP/PCA Satellite Library
Maryland Procurement Office
Mixer Systems, Inc.
RAS Systems, Inc.
Schweitzer Sortiment
WFT Electronic Systems
Steven Allard, W. Tombling Ltd.
Eric C. Anderson
Casey T. Bazewick, Jr.
Ray Bell
Joseph H. Boutwell
Robert W. Brown
Paul Bunyan
Matthew Burke, Computer Science Dept.
David Cabana
Mark Christensen
Lee Cobum
E. Coleman
Val J. Cooper
Bernard Cosell
Jason Cunliffe, Mighty Dimension Inc.
Federico Daneri
David P. Dille, Controlsoft Research
Aaron C. Dutton, Computer Sciences Corp.
David Erbas-White
David Ferrara
Chris Frank
Gabriel Fugulin
Richard Gallehr
David Garcia
Remy Gentis
Charles A. Gray
Emilio Guarise
David C. Haas, Owens-Corning
Tech. Center
J. Harhellier
Tim Heaps
Daniel P. Hill
Michael, Hillerstrom, Dansk
Data Elektronik
Peter Hiltz
Charlie Holloway, H&N Instruments, Inc.
Andrew P. Houghton
Mark Humphries
Art Hyde, Florida Drum Co.
Brian Jackson
Woo-Jin Jang, Photone Research Ctr.
Hyun Jeong Tae, Korea Advanced
Energy Research Inst.
William R. Jones
Kun Katsumata, Nictrix Corp.
Martin Kees
Robyn L. King, NASA
David F. Klink
Aratt Klinsong
Chung-Cheng Li Kong, Ta Tong
Book & Magazine Service
Choonu Lee, Photone Research Center
John Longton
Peter Lu
Jacob Maier

Leningrad, USSR
Szekesfehervr, HUNGARY
Budapest, HUNGARY
Riyadh, SAUDI ARABIA
Ft. George Meade, MD
Pewaukee, WI
Campbell, CA
Munchen, GERMANY
Denver, CO
Spaloing, Lincs, ENGLAND
Beavercreek, OH
San Francisco, CA
Buffalo, NY
Atlanta, GA
Medina, OH
Rio Linda, CA
Pullman, WA
Tampa, FL
Lindon, UT
Lakewood, CO
Los Gatos, CA
Monmouth, OR
Lexington, MA
New York, NY
Genova, ITALY
St. Louis, MO
Baltimore, MD
El Toro, CA
Santa Rosa, CA
Garland, TX
Montreal, PQ, CANADA
Boston, MA
Staten Island, NY
Wilmington, MA
Phoenix, AZ
Milano, ITALY
Granville, OH
Eupen, BELGIUM
Agoura Hills, CA
Vancouver, WA
Hjallerup, DENMARK
Lake Orion, MI
Newark, OH
Columbus, OH
Daly City, CA
Pine Bluff, AR
Cleveland, MS
Seoul, KOREA
Choone-Nam, KOREA
Morgantown, KY
San Jose, CA
Pasco, WA
Greenbelt, MD
Fort Collins, CO
Chicago, IL
Taipei, Taiwan
Seoul, KOREA
Brooklyn, NY
Occidental, CA
Toronto, ON, CANADA

Mark Malmros
Elliott R. Marsh, IBM
Robert Mayer
Richard G. McCord, Conpane
Industries, Inc.
Edwin Miller
Mike Mills
Fred Minetto
Larry Minthorn, University of Iowa
J.T.P. Moule
Paul Mrofczak
Patrick Newton, nView Corp.
Adorjan Noemi
J. Ozegetic
Clinton B. Petry II
Clyde Phillips, Access Express
Brian Piercy, Harris Semiconductor
Abdur Rahmaan, Noble Science Library
R.M. Reynolds
Franco Ricci
Thomas Ritchey, Ritchey Engineering
David Ritchie
William Robinson
Joel Rubin
Scott T. Schad
Boris Schlensky
Conrad Schlundt
Tom Shaw
Delwin P. Stevens
Robert Suess
Brian Sutton, Citrus Park Chiropractic
Alain Theroux, Logiciel Plus
V.H. Vinerts
Richard C. Wagner, Naval Air
Development Center
Steven A. Wallace
Manfred Wolffe
Charles R. Wollitz
Mark Wood, RSS, Inc.
Edward Woodman
Bill Zimmerly
Patrick Dixon
Christian Beauregard
J. Klir, MCSL
Peter Spencer
Loy Spears
Anthony Blazej, IBM
Jack Rausch
Giorgio Kourtis
Thor-Bjom Bladh
Fred Temple
Edward Nass
Gavin Craig
Hubert M. Schmitter
Peter Hauser
William Higinbotham,
Brookhaven National Lab
Richard Reichstadt
Robert Nace
Patrick Ness
William H. Madden
Paul Schumann
Michael Hubbard
Sandy Warshaw, Optisys

Washington Crossing, PA
Endicott, NY
Quakertown, PA
Louisville, KY
Bellaire, TX
Cincinnati, OH
Temple Hills, MD
Iowa City, IA
Isle of Man, BRITISH ISLES
Warren, OH
Newport News, VA
Budapest, HUNGARY
Split, YUGOSLAVIA
Placentia, CA
Oak Park, IL
Melbourne, FL
Tempe, AZ
Geneva, NY
Firenze, ITALY
Chester Springs, PA
Eagle, ID
Chicago, IL
San Francisco, CA
Tulsa, OK
Palo Alto, CA
The Colony, TX
Philadelphia, PA
Anaheim, CA
Indianapolis, IN
Tampa, FL
Sherbrooke, PQ, CANADA
Newark, CA
Revere, PA
Knoxville, TN
Berlin, GERMANY
Buckeye, AZ
Costa Mesa, CA
Newbury Port, MA
St. Louis, MO
Hastings on Hudson, NY
Loretteville, PQ, CANADA
St. Albans, Herts, ENGLAND
Bothell, WA
Placentia, CA
Milford, CT
Adelphi, MD
Genoa, ITALY
Lund, SWEDEN
Plattsburgh, NY
Plainview, MN
Mtn. View, CA
Jefferson, LA
Taegerwilen, SWITZERLAND
Upton, NY
Fairmont, MN
Pleasant Hill, CA
Rapid River, MI
Livonia, MI
Mesquite, TX
Boulder, CO
Long Beach, CA

Gary Richardson
Jim Polstra, Orbital Sciences Corp.
R.W. Fergus, Astute Solutions
John Spencer
Ferenc Kish, Robway Safety Systems
John Ott
John Svae
Javier Campos, Universidad de Zaragoza
A. Foord
J.J. Kent
Bill Stoddart, Teesside Polytechnic
John Dillon

Steve Jacobs, Iomega Corp.
Rick Powers, Phoenix Inst. of Technology
Gary Keen, War Veteran's Homes
Philip W. Y. Keung
Raymond N. Thornton
Chris Mayne, Bemoulli Optical
Mike Budaji
David Zethmayr
Thomas Schossig
John Tse, Chrisma Technology
John L. Brown
Robert Drap
B.A. Manook
Randy Black, SCI Technology, Inc.
Luther Huffman
Bill Wood
William B. Bunch
David Vaughn
Roberto Dell'Acqua
Sarah Cornel, Mitel Corp.
Edward Tylka
Lee Robertson
Harry L. Spencer, Jr.
Robert Kennedy
A.E.F. Coenen, Sweelinck Conservatory

Owen Ransen
Robert F. Orenstein
Richard Silkowski
Bruce Raymond
John Hall
Yukinobu Miki
Yale Univ. Medical School
Peter Peterson, Mykro/P.S.I.
Stephen Pierotti
Salil Donde
Hogan Law, Synoptics
Communications, Inc.
John W. Klay, M.D.
Kenneth Perusek
Pablo Casado
Kenneth Frost, OEM Controls
Bruce Bales
Brian Rogoff
Bob Burchsted, Aeon Systems

Alameda, CA
Boulder, CO
Lombard, IL
San Jose, CA
Thebarton SA, AUSTRALIA
Bremen, IL
NORWAY
Zaragoza, SPAIN
Malvern, Worcs, ENGLAND
QLD, AUSTRALIA
Middlesbrough, Clevid, ENGLAND
Stratford On Avon, Worcs,
ENGLAND

Roy, UT
Phoenix, AZ
Mackay, QLD, AUSTRALIA
Aberdeen, HONG KONG
Hackensack, NJ
Boulder, CO
Cleveland, OH
LaGrange, IL
Gotha, GERMANY
SINGAPORE
Oakland, CA
Mission Viejo, CA
Swindon, Wilts, ENGLAND
Huntsville, AL
Dayton, OH
Crystal, MN
Dallas, TX
Austin, TX
Milano, ITALY
Kanata, ON, CANADA
San Diego, CA
Sandy, UT
Huntsville, AL
Elmhurst, IL
Amsterdam,
THE NETHERLANDS
Milano, ITALY
Metuchen, NJ
Poughkeepsie, NY
Beavercreek, OH
Wandwick, NSW, AUSTRALIA
Ibaraki, JAPAN
New Haven, CT
Salt Lake City, UT
Broomfield, CO
Goose Creek, SC
Mtn. View, CA

Pittsburgh, PA
Solon, OH
Pamplona, SPAIN
Shelton, CT
Benton, KS
Menlo Park, CA
Billerica, MA

Nick Janow
Bob Bruns
Michael Dorais
Forrest Schornberg, USS/POSCO
Alan Tabur, NSW
Peter Blasberg, Boehringer
Mannheim GmbH
Gabriel Marro Gros
Dr. Pfuller, FORTech Software GmbH
Saman Samimi
U.S. Remote Controls
Jose E. Korneluk, Jr.
David Dan, Intel Tech. Far East Ltd.
Colin Connery
Olaf Meding, Amtelco
B.M. Shann
Bengt Karlstrom, Mactec Control
Deiter Weller
David Aitkon, PC Systems Corp.
Robert W. Curry
Garth Wilson
John J. Deering
J.G. Woyka

Dr. Howard Shapiro MDPC
Robert A. Hall
William Hickey, Fox Theaters
Donald Adnam
Rob Miller, Greenlee Textron
Dosatron S.A.
Kouji Hori
Greg Schmidt
Fabien Tuizat, Atelier de la Forge Royale
Raymond LaPorte
Jan Michaels
Knut Johnson
Chih-Yu Chao, Altech Controls
Paul Chivers
Alfred Oliphant
Ralph Stockhausen
Greg Bentz
Robert Thompson
Michael Krell
Cesar Clavell, Naval Ocean Systems Center
Doug Baron
Robert Koponen
Paul O. Lloyd, Microsemi Corp.
Gerald Childers
Gary Camer
Brian Howell, Shenandoah Products
Klaus Malzahn
John F. Dodds
Richard Carleton
Martin Marietta Library
Michael Hosmer
Ryon Root
Knut Johnson

Vancouver, BC, CANADA
Campbell, CA
San Diego, CA
Antioch, CA
Dahlgren, VA
Mannheim, GERMANY
Zaragoza, SPAIN
Rostock, GERMANY
Parramatta, NSW, AUSTRALIA
San Francisco, CA
W. Palm Beach, FL
Taipei, Taiwan
Mt. Vernon, NY
Madison, WI
Freeland, Oxford, ENGLAND
Kristianstad, SWEDEN
Springfield, OR
Melville, NY
Rochester, NH
Whittier, CA
Tucson, AZ
Haddington, E. Lothian,
ENGLAND
West Newton, MA
Los Gatos, CA
Taft, CA
Ottawa, ON, CANADA
Rockford, IL
Latresne, FRANCE
Saitama, JAPAN
Richmond Heights, OH
Paris, FRANCE
Vancouver, BC, CANADA
Berlin, GERMANY
Sylmar, CA
Houston, TX
Bateman, WA, AUSTRALIA
Trinidad, CA
Milwaukee, WI
Richmond Hill, ON, CANADA
Hattiesburg, MS
Charlottesville, VA
San Diego, CA
Indianapolis, IN
Lakeland, FL
Orange, CA
East Holden, ME
Jamesville, NY
Cottonwood, AZ
Duesseldorf, GERMANY
Newbury, OH
Augusta, GA
Piketon, OH
Orem, UT
Glendora, CA
Sylmar, CA

(Continued from page 30.)

set of FORML Proceedings, and Mitch got an RTX2001A chip. Both winners agreed to present their programs in our December meeting.

Chuck Moore's Annual Fireside Chat

Chuck revealed his newest Forth engine design, MuP20. It is a 20-bit processor with several coprocessors around it to handle I/O and memory transactions. The coprocessors include a DMA processor, a video processor, a UART, and three input lines for a three-key keyboard. An internal clock will allow it to run up to 200 MHz.

Chuck also conceded that Forth has never been successful in the mass-consumer market. He will probably concentrate on developing highly customized products using his special CAD system as leverage.

In the question-answer period, Chuck also commented on error recovery ("don't"), writing video drivers ("shorter and shorter"), disk error ("turn the computer off and walk away"), memory ("forget disks, use RAM"), files ("yesterday's problem"), the three-key keyboard ("most natural human interface"), etc.

Banquet at Ritz Chinese Seafood Restaurant

(Authentic Chinese Banquet)

Ten-course dinner with cold plate, shark fin soup, walnut prawns, beef in nest, fresh green vegetable, crab in sauce, steamed fish, sweet desert, etc., over lively conversations about computers, politics, the economy, and miscellaneous subjects.

(Continued from page 37.)

These are the regular Pygmy words that work in both the host and target dictionaries. There is no need for special metacompiler versions of them.

```
-' ( u - here t | pfa f) gets the next word from the input stream and looks it
up in vocabulary u.

' ( - pfa) Looks up next word in input stream in active vocab. "tick"

INTERPRET ( blk# offset -) Until the input stream is exhausted, step through it,
looking up the words in FORTH and executing them
if found or trying to convert them to a number.

: -' ( n - here t | pfa f) 32 WORD SWAP -FIND ;

: ' ( - pfa) CONTEXT @ -' ABORT" ?" ;

: INTERPRET ( blk# offset -)
>IN 2! ( )
BEGIN 2 ( ie search FORTH) -'
( here true | pfa false)
IF NUMBER ELSE EXECUTE THEN
AGAIN ;
```

Figure Thirteen.

These are regular Pygmy words, most of which work unchanged in both the host and target dictionaries. Special metacompiler versions of LITERAL and] are needed, however.

```
ALLOT ( n -) Moves the dictionary pointer H by n (either forward or backward).

, ( n -) Stores 2-byte number n into the next free location in the dictionary and
advances H past it. "comma"

C, ( c -) Stores 1-byte number c into the next free location in the dictionary and
advances H past it. "C-comma"

,A ( a -) Commas a 16-bit address into the dictionary. It is relocated by the value
in dA.

COMPILE ( -) Compile the following word at this word's run-time.

LITERAL ( n -) Compile n so that at run time it will be pushed to the data stack.
It is DEFER'd.

] ( -) This is the heart of the colon compiler. It gets the next word from the input
stream. If the word is in the COMPILER vocabulary it is executed. Else, if the
word is in the FORTH vocabulary it is compiled. Else, if the word is a valid
number it is compiled as a literal. Else, error.
```

Figure Fourteen.

FIG CHAPTERS

The FIG 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 in each issue of *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
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**
3rd Sat.
12 noon tutorial, 1 p.m. Forth
2055 Center St., Berkeley
Leonard Morgenstern
(415) 376-5241

Orange County Chapter
4th Wed., 7 p.m.
Fullerton Savings
Huntington Beach
Noshir 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

Silicon Valley Chapter
4th Sat., 10 a.m.
Applied Bio Systems
Foster City
(415) 535-1294

Stockton Chapter
Doug Dillon (209) 931-2448

- **COLORADO**
Denver Chapter
1st Mon., 7 p.m.
Clifford King (303) 693-3413

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

Tampa Bay Chapter
1st Wed., 7:30 p.m.
Terry McNay (813) 725-1245

- **GEORGIA**
Atlanta Chapter
3rd Tues., 7 p.m.
Emprise Corp., Marietta
Don Schrader (404) 428-0811

- **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.
I/P Univ. Campus
B71 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) 472-7077

- **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 Lordi
(201) 338-9363

- **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
Laboratory
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**

Cleveland Chapter
4th Tues., 7 p.m.
Chagrin Falls Library
Gary Bergstrom
(216) 247-2492

• **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

• **OREGON**

Willamette Valley Chapter
4th Tues., 7 p.m.
Linn-Benton Comm. College
Pann McCuaig (503) 752-5113

• **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.
Sci. Appl. Int'l. Corp., 8th Fl.
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
Clif Penn (214) 995-2361

Houston Chapter
3rd Mon., 7:30 p.m.
Houston Area League of PC
Users
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) 898-4099

Potomac FIG

D.C. & Northern Virginia
1st Tues.
Lee Recreation Center
5722 Lee Hwy., Arlington
Joseph Brown
(703) 471-4409
E. Coast Forth Board
(703) 442-8695

Richmond Forth Group

2nd Wed., 7 p.m.
154 Business School
Univ. of Richmond
Donald A. Full
(804) 739-3623

• **WISCONSIN**

Lake Superior Chapter
2nd Fri., 7:30 p.m.
1219 N. 21st St., Superior
Allen Anway (715) 394-4061

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
Lariksdreef 20
2120 Schoten
03/658-6343

Southern Belgium Chapter

Jean-Marc Bertinchamps
Rue N. Monnom, 2
B-6290 Nalinnes
071/213858

• **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.)

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, Surry SM4 4DS

• **FINLAND**

FinFIG
Janne Kotiranta
Arkkitehdinkatu 38 c 39
33720 Tampere
+358-31-184246

• **GERMANY**

Germany FIG Chapter
Heinz Schnitter
Forth-Gesellschaft e.v.
Postfach 1110
D-80 Unterschleissheim
(49) (89) 317 3784
e-mail uuap:
secretary@forthev.uucp
Internet:
secretary@admin.FORTH-
eV.de D-8044 Unterschleis-
sheim
(49) (89) 317 3784
Munich Forth Box:
(49) (89) 725 9625 (telcom)

• **HOLLAND**

Holland Chapter
Vic Van de Zande
Finmark 7
3831 JE Leusden

• **ITALY**

FIG Italia
Marco Tausel
Via Gerolamo Forni 48
20161 Milano

• **JAPAN**

Tokyo Chapter
3rd Sat. afternoon
Hamacho-Kaikan, Chuoku
Toshio Inoue
(81) 3-812-2111 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

SPECIAL GROUPS

• **NC4000 Users Group**
John Carpenter
1698 Villa St.
Mountain View, CA 94041
(415) 960-1256 (eves.)

HURRY! For a limited time you can...

SAVE 50% UP TO

Offer good ONLY to current FIG members...

As you may know, the Forth Interest Group's Board of Directors has authorized an increase in membership fees. This was needed in order to meet the rising cost of important member services like *Forth Dimensions* and our mail-order operations. **But until April 1, 1991 current members can still renew at last year's rate.**

FIG depends entirely on the support of people like you, and every member's fee goes to keeping our doors open and the presses rolling. Without our magazine, source code listings, and other publications, the world would lose its major source of Forth education and information. So, please, take a moment right now to fill out the attached renewal form—your membership expires with *FD* issue XII/6!

Two more reasons to renew NOW...

Other recent changes at FIG make it even more important that you renew on time. First, members who renew late will no longer automatically receive any issues they missed—they must order them separately as back issues, subject to availability, *and at the new cover price of \$10*. From now on, each FIG membership will last for twelve months from the day it is received at the FIG office, no longer linked to the *Forth Dimensions* calendar year.

**Three ways to get
Forth Dimensions
next year!**

\$60 if purchased separately (6 issues x \$10 each)

\$40* to members after April 1, **saving 1/3!**

\$30** to members renewing before April 1, **saving 1/2!**

* \$46 Canada, \$52 other countries. ** \$36 Canada, \$42 other countries.

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

Second Class
Postage Paid at
San Jose, CA