$10 Volume XIII, Number 1 May 1991 June

FORTH DIMENSIONS

Symbolic Processing in Forth

Forth for the 90's

Menu Words

C64-matted Source Code

PDE Screen Management

eForth—Portable Forth Model

Contents

Features

Departments

Editorial

I'd like to begin by thanking you for continuing to support this magazine and the Forth Interest Group. With notable exceptions, users groups and special-interest groups have suffered dwindling numbers in recent years. Many have disappeared entirely or are, as the saying goes, mere shadows of their former selves. Whether this is due to the economy, to competition from on-line alternatives, to a societal shift away from grass-roots organizations, or to the collective burn-out of the volunteers who founded and operated such institutions, is a matter for speculation and study.

Fortunately, FIG is enjoying a warm response to this year's reminder notices. Many of you generously have contributed more than the basic membership dues and have expressed encouragement and appreciation of FIG's services. Your kind words are welcome!

It is the intellectual and emotional investment of its members that has kept the Forth Interest Group alive in tough times. Members often support the organization even if they don't always agree with it. Forth itself has always bred controversy and, in some circles, the same is true of FIG; maybe that shouldn't be surprising. But FIG does succeed at providing an arena for this fraternal contention, and most agree that getting good Forth articles and arguments and code into print is a singularly important and useful service. As *FD* enters its thirteenth year of publication, we hope you will join us in the arena—write an article or letter to the editor, review a Forth book, help invigorate the local FIG chapter, or share some Forth code with your fellow readers.

...the intellectual and emotional investment of its members has kept FIG strong.

New Contest $ Announcement $

We enjoyed the stimulation generated by our contest for articles about Forth hardware (see *FD* XI/6), and it is high time we did it again. But we needed a new contest theme—Forth hardware, however you interpret the phrase, is an important subject about which we are *always* looking for more articles. So we asked key Forth vendors for other suggestions and, while a number of tempting topics surfaced, only one garnered a majority opinion.

So it is with pleasure and a sense of anticipation that I announce this call for contest entries on the subject of object-oriented programming. Judges will be instructed to give preference to articles/papers which include code that is educational, illustrative, and/or actually useful, though code is not strictly required. Materials should be submitted in the form of hard copy accompanied by a diskette containing the article, an ASCII (i.e., text without embedded formatting) version of the article, biographical information about the author(s), and any accompanying Forth code. It can also be uploaded to my MARLIN.O e-mail address on GEnie, though hard copy should also be mailed for safety. *For contest entries to be considered for prizes, they must be received by September 16, 1991.*

Prizes, you ask? In addition to publication in *Forth Dimensions*, public recognition, and the admiration of fellow FIG members, the first-place winner will receive $500, second place will earn $250, and third prize will be $100. Kind of like a small, unexpected inheritance...

Feel free to ask the FIG office for a copy of our Writers Guidelines. Following the tradition of our last contest, the theme of this one is purposefully left a bit general. That is done in order to encourage diversity and originality, and to allow room for your particular approach and priorities.

I look forward to receiving your submission!

Calls for Papers—China and Europe

If you don't want to write about object-oriented programming (even if you do), there are at least two other places, at opposite sides of the planet, looking for good Forth technical papers. Jiao Tong University in Shanghai will be the site of a China FORML Conference on the subject of factory automation and industrial applications of Forth. For details, see the ad elsewhere in this issue. The organizers plan a very interesting tour "...off the oft-trodden tourist paths." I also have a letter from Mr. Zeng Jing, announcing a Forth symposium in Beijing

(Continued on page 20.)

Forth Dimensions
Volume XIII, Number 1
May-June 1991

Published by the
Forth Interest Group

Editor
Marlin Ouverson

Circulation/Order Desk
Anna Brereton

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. Fax: 408-286-8988

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 $40/46/52 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."

Letters

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

Go Ahead—Trust the Compiler!

As I sit here writing this, I am wondering if anybody cares any more about someone else's code? Forth code, that is! Since my brief impromptu talk Saturday night, I have had time to consider where I am going and the direction Forth is taking. Yes, this is a late paper which, hopefully, will make it into the FORML Proceedings. As a relative newcomer to Forth, I have been reticent to expose my ignorance about the inner workings of the language (outer workings, too).

Since I am my own client, my code has to satisfy my own needs; and as a Professional Land Surveyor, I have not found commercially available surveying packages to my liking. So if you can't buy it, you have to make it. Most packages are highly protected from the purchaser, not because they are valuable, but because if you found out where the shortcuts were taken, your "garbage-in, gospel-out" mentality would be severely shaken.

I recently attended a non-Forth computer event, where a Windows package was "...so much easier to use" after 500 bug fixes. It appears that computer graphics is now the home crafts hit, replacing needlepoint as m'ladies' pastime. None of these people seem concerned about the possible consequences of having to explain that "It really wasn't my fault, it happened because the software had bugs in it." The world sure looks different when your own personal posterior is on the hot seat.

Occam's Razor is often attributed to Albert Einstein in the press. "Everything should be made as simple as possible, but no simpler."

Go ahead, "Trust the Compiler," put your faith in the obscure. If the graphical user interface is good, then the compiler must be terrific! The GUI sells software; if the compiler suffers, you will get an upgrade offer next year. Why are these mass-marketed consumer products taking over? One reason is that they are buzz words in computer stores. Megabuck promotion and packaging helps allot (pun intended).

Given that Forth is a programmer amplifier, it is apparent that a skilled Forth programmer can get excellent results; but management preference is for entry-level programmers in commodity languages, for the perceived economic and actuarial reasons. Forth is usually the language of last resort, for use when conventional methods prove unproductive or simply can't be bent to fit the requirements.

But what about the company with a couple of in-house Forth programmers and some Forth-based products—what happens when both programmers go skiing and end up in traction?

Imagine this scenario:

"Hello, is this the Forth hiring hall? Yeah, I need a couple of journeyman programmers. Can you send them over this afternoon? We have 30 tons of widgets that have to ship Friday, and the ROM code isn't ready. What! The only guy you've got is on the Ross Ice Shelf and is scheduled to over-winter at Little America, but I can have him if I charter a 747 to pick him up before the blizzard hits? You mean it's that or lay off 600 people? The stockholders will not be pleased..."

MBA-types wake up in the middle of the night and mumble to their spouses over thoughts like these.

Sure, it's preposterous: There isn't a hiring hall.

But maybe there should be! On October 27, the U.S. Congress passed a bill without a hearing which exempts "computer systems analysts, computer programmers, software engineers" and other computer professionals who earn $25 per hour or more from the overtime pay requirements of federal law, on the grounds that these highly paid workers do not need federal protection. These employees will now be able to work extra hours and get paid their regular hourly rates.

So, with all this doom and gloom, why am I still writing in Forth? Because I want something I can understand, that I will maintain, and which is economical of my limited resources. Unlike government employees, I am not rewarded with more employees, a bigger desk, and the other perks of bureaucracy. I am building productivity tools tailored to my own needs and desires, and am even enjoying it.

John Edgecombe
905 S. Jurymast Drive
Oxnard, California 93030

Why am I still writing in Forth—even enjoying it?

Note from a Defector

Dear Marlin,

I'm afraid the time has come for me to give up Forth (and *Forth Dimensions* and FIG). Congenial as Forth is to a left-handed Zen Buddhist Aquarian MIT-graduate phi-

I thought you'd be interested in the mind of a defector.

(Continued on page 21.)

A message from the President of the Forth Interest Group

President's Letter

I need to tell you about the Organization, about the problems and issues, about what is happening right now to solve these issues, how I see the issues, and plans for the future. I can't do this all in one column, so I am going to tell you a little about each in every issue of *FD*. The parts are Organization, Issues, Actions, and Explanations.

Organization

The chart should explain a lot and will be changing over the next year. The empty boxes are for new ideas from you.

While talking to people, I have discovered some misunderstanding about how FIG services are provided. FIG is a volunteer organization and no member of the Board of Directors, no officer, committee chairman, or committee member is paid for that service. Many FIG expenses are paid by individuals and are then reimbursed. We contract only for functions that we cannot accomplish with volunteers because they require equipment or facilities that we do not have or talents and time that are more than we can expect from volunteers. These are, specifically, the editors of *Forth Dimensions* and FORML, and the storefront and mail-order presence of the Association Development Center, who provides the office functions for us.

As the president, I have the privilege of attempting to "lead" this organization. I want to give you a little of my philosophy about leading. As a leader, I cannot lead you in the direction you want to go; I can only lead you in the direction I want to go. If you want to follow, it is my joy to have you with me, but by doing so you give up the privilege of having me listen to you complain that we are not going where *you* want to go. As a guide, my job would be to get ahead of you and help clear the path and light the way, making it easier for you to go in the direction you choose. But the only way I can know where your "ahead" is, is by your input to me about where we should be going. It is your choice but, as the president, I would be just as happy to have the privilege of "lighting the way" for this organization.

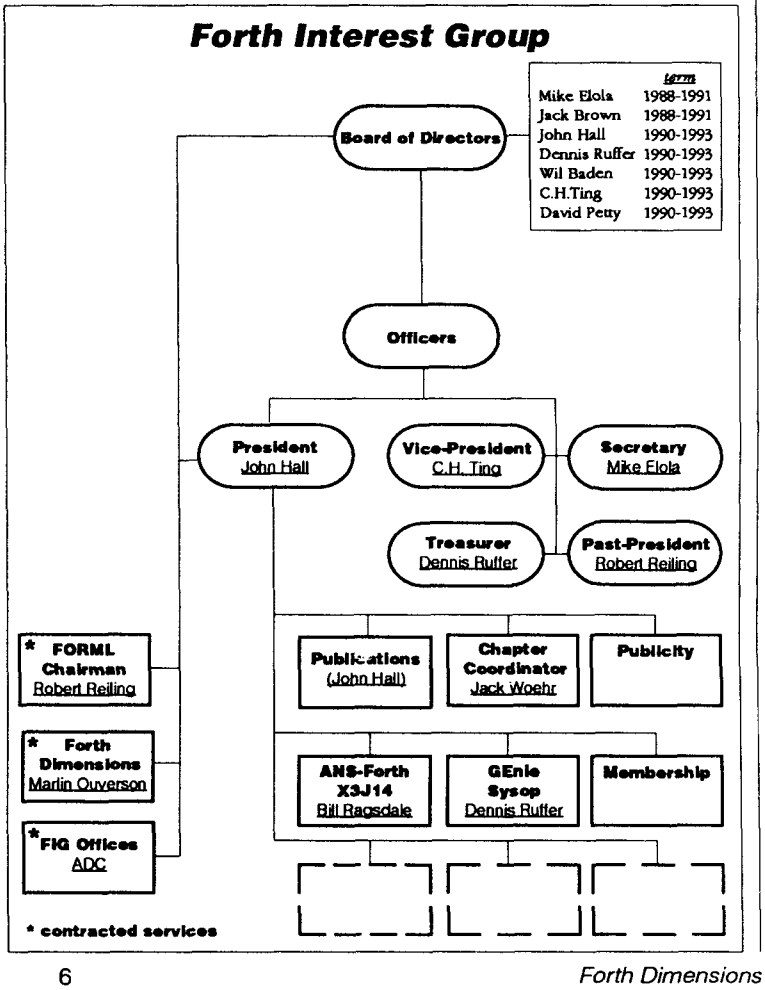
Issues

Here are some of the issues I feel we have to deal with. Some I have felt strongly about and others have been expressed by concerned individuals. They are listed in order of timeliness to discuss. This will change over time as new ones get added.

Forth Interest Group

- Treasurer preparing a review of FIG financial situation for publication in *FD*.
- Silence isn't secret, we could fill *FD* twice over with business meeting minutes and financial statements.
- Dues increases, helping or hurting, necessary?
- Annual billing, why?
- Publicity, public relations, FIG advertisement, mention of FIG in articles, word of mouth.
- Standards, should FIG endorse them?
- Volunteers, must spread

(Continued on page 26.)



Symbolic Processing

Alexander Sakharov
Newton, Massachusetts

Unfortunately, Forth is far from being a general-purpose language in the market; the author would like to see Forth in the position of C in the market. Forth has the potential for this, but it lacks data types and a memory allocation mechanism. Few Forth systems have packages for advanced applications like graphical user interfaces, databases, communications, word processing, etc. These are some of the reasons for Forth's poor position.

Symbolic processing is in the ranks of major applications. For instance, knowledge engineering requires the ability to handle symbolic data. Moreover, the means for symbolic processing are a precursor for most work in artificial intelligence. We propose, and have implemented, a package of Forth words which promote symbolic processing. Primary applications of the package are knowledge-based systems (for instance, expert systems) and systems for symbolic or mixed computations. Since there are no common Forth words supporting list data and memory allocation, our package has been implemented from beginning to end in Forth-83 Standard.

Knowledge is encoded with simple and clear list-notation words.

Any symbolic data (logical or algebraic expressions, graphs, frames, semantic networks, etc.) can be represented by lists. A list notation has been elaborated within Forth. Forth words (: and :) enclose list items. They have the same meaning as (and) in the Lisp notation. Three types of atoms are in use: integers, strings, and slots. Any other types of atoms can be easily added. The content of any integer atom is an integer. The content of any string atom is a pointer to a string. The content of a slot atom is the address of a Forth variable. Advanced symbolic processing (as in knowledge-based systems) requires the means to handle patterns. In our package, slot atoms allow the developer to construct list patterns. These patterns can either be matched against lists or utilized to generate other lists.

The word % constructs an integer atom, taking the top stack value as the content of the atom. The word $ is used in the same manner for strings. The word ^ constructs a slot atom, taking the top stack value as its content; i.e., the top stack value is treated as the address of a Forth variable. Atoms constructed with %, $, or ^ may be items in lists or may stand alone. The word & constructs a list item which is either a list or an atom. It also takes the top stack value as its parameter. All aforementioned words can be used along with other words (in keeping with the Forth philosophy). In order to check the bracket structure of lists and to make the words more efficient, they are implemented as compiling words. Some examples of atoms and lists are given in Figure One.

As usual, lists are represented in memory as sets of connected CONS-cells. A CONS-cell consists of two components. The first is a pointer to a list item. The second is a pointer to the next CONS-cell. Any atom is represented as a CONS-cell. The first element is an atom type that, simultaneously, serves to distinguish atoms from lists. The second element is an atom value. Lists and atoms are represented on the Forth parameter stack

Figure One. Some example atoms and lists.

```
-1 %
" +-*" $
(: 1 % " ABC" $ :)
VARIABLE V 10 % V ! (: V @ & 2 % V @ & :)
VARIABLE X   VARIABLE Y
(: X ^ (: 0 % Y ^ :) :)
(: 10 0 DO I 4 * % LOOP :)
```

Figure Two. Sample rules from demo knowledge bases.

```
: */d (: (: U ^ " -" $ V ^ :) " /d" $ X ^ :) LHS
  (: (: U ^ " /d" $ X ^ :) " -" $ (: V ^ " /d" $ X ^ :) RHS ;

: #5 " FLIES" $ Y/N
  " LAYS_EGGS" $ Y/N
     " BIRDS" $ A-F ;
```

as pointers to their memory representations. Notice that the stack representation of any list or atom differs from 0 (FALSE).

The conventional Lisp functions CAR, CDR, CONS, EQUAL, and MEMBER are useful for programming symbolic transformations and for encoding knowledge and inference procedures. They are implemented as words manipulating the Forth parameter stack. Since atoms are represented as CONS-cells, CAR and CDR can be used to yield an atom type and value, respectively. CONS can construct an atom of its type and value. The following Forth constants elaborate the types of atoms: INT (integer atom), STR (string atom), and VAR (slot atom). NIL is also implemented as a Forth constant. The words INT?, STR?, VAR?, and LST? are Boolean operators. INT?, STR?, VAR?, and LST? test whether the top stack value is an integer atom, a string atom, a slot atom, or a non-empty list, respectively.

To enable arbitrary changing of CONS-cells, the words CAR! and CDR! are introduced. The behave similarly to !. CAR! assigns a new value to the first component of a CONS-cell. CDR! does the same with respect to the second component. Again, these words apply to both lists and atoms.

Two powerful words SB and MT allow the developer to process list patterns. The word MT compares a list or atom to another one. It has two parameters: its top stack parameter may contain slot atoms; the next should not. Any variable may have no more than one occurrence in the first parameter. MT yields a Boolean value showing the result of the comparison. If the parameters match, then any lists or atoms of the second parameter that match slots of the first parameter are stored at the corresponding variables. The word SB has one parameter. It generates a list or atom by substituting lists or atoms assigned to the variables occurring in the parameter as slots.

Our package should be suited for typical Forth applications and, in particular, for real-time systems. Garbage collection is to be avoided in real-time systems; our implementation supports dynamic memory allocation. Some primitives supporting this technique are written in Forth. The word F-L frees all cells of a given list. The word SF-L selectively frees some cells of a list; the cells which do not occur in another list are freed. These two lists are the parameters of SF-L. These words allow the developer to delete garbage at any stage of symbolic processing, thereby avoiding global garbage collection.

Knowledge is encoded with list notation words which are fairly simple and clear. Lisp-like words are used extensively to implement internals of systems. The bottleneck in the development of expert systems and systems for symbolic computations is knowledge engineering. Our words promote knowledge representation. Thus, our words help solve the problems of knowledge engineering.

Two demonstration programs have been implemented with the use of our package. One of them performs symbolic differentiation of algebraic formulae and then simplifies them. The second one implements the animal game. Figure Two contains two rules, one from the knowl dge base of each program.

The words LHS and RHS are used in all rules of the first program. The words Y/N and A-F are used in all rules of the second program. The actions performed by LHS and Y/N include pattern matching. The words RHS and A-F, in particular, substitute atoms or lists for slots. All garbage is deleted when executing these words. It is easy to make the rules in these programs more efficient by constructing lists statically, storing them as Forth constants and using the constants in the rules.

Alexander Sakharov received his M.S. and Ph.D. degrees from Leningrad University, where he worked primarily in compiler development and automatic programming. He began using Forth in 1984—as the implementation language for a functional-language compiler—while involved in a project to build a Forth processor. When he moved to the Soviet National Academy of Sciences' office automation department, he developed artificial intelligence tools in Forth. He is now working in Software Research and Development for Motorola, once again involved in compiler development.

```
( Symbolic Processing in Forth )
( Alexander Sakharov )
( It is assumed that the following words are implemented: )
( TRUE FALSE -ROT RECURSE ?PAIRS )

 ( Auxiliary words )
 : STR=    ( a1 a2 --> f    string=string? )
  COUNT ROT COUNT ROT OVER =
  IF 0 DO OVER I + C@ OVER I + C@ = NOT
         IF FALSE LEAVE THEN LOOP
       IF DROP TRUE ELSE DROP DROP FALSE THEN
  ELSE DROP DROP DROP FALSE THEN ;

 : ?DDFE IF DROP DROP FALSE R> DROP THEN ;


( Constants, variables )
  1 CONSTANT NIL
  10 CONSTANT VAR
  11 CONSTANT INT
  12 CONSTANT STR
  15 CONSTANT OC  ( occupied CONS-cell )
  31 CONSTANT FC  ( free CONS-cell )
  2 CONSTANT VL  ( value length /stack cell size/ in bytes )
 1000 CONSTANT WMS ( number of CONS-cells in working memory)
 HERE WMS VL 2* * ALLOT HERE SWAP
  CONSTANT WMMIN ( working memory min address)
  CONSTANT WMMAX ( working memory max address)

 VARIABLE WMP ( working memory pointer)
 VARIABLE (CNT ( parenthesis counter for lists)

( LISP-like functions )
 : CAR STATE @ IF COMPILE @ ELSE @ THEN ; IMMEDIATE
 : CDR VL + @ ;
 : CAR! STATE @ IF COMPILE ! ELSE ! THEN ; IMMEDIATE
 : CDR! VL + ! ;
( Working memory utilities )
 : GWMC   ( --> ad : get working memory cell )
  TRUE WMS 0
  DO WMMAX WMP @ = IF WMMIN WMP ! THEN
   WMP @ CAR FC = IF OC WMP @ CAR! DROP FALSE LEAVE
                  ELSE VL 2* WMP +! THEN
  LOOP
  IF ." No free working memory"  ABORT  THEN
  WMP @ VL 2* WMP +! ;
 : FWMC  ( ad -->  : free working memory cell )
  FC SWAP CAR! ;
```

```
( LISP-like functions, continued )
 : CONS GWMC SWAP OVER CDR! SWAP OVER CAR! ;
 : INT? DUP NIL = IF DROP FALSE ELSE CAR INT = THEN ;
 : STR? DUP NIL = IF DROP FALSE ELSE CAR STR = THEN ;
 : VAR? DUP NIL = IF DROP FALSE ELSE CAR VAR = THEN ;
 : LST? DUP NIL = IF DROP FALSE ELSE CAR FC SWAP U< THEN ;
 : NIL? NIL = ;
( Generating words )
 : G-C    ( ad1 item --> ad2   : generate CONS-cell )
  OVER CAR OC =
  IF OVER CAR!
  ELSE GWMC SWAP OVER CAR! DUP ROT CDR! THEN ;
 : G-A    ( value type --> ad   : generate atom )
  GWMC SWAP OVER CAR! SWAP OVER CDR! ;


( EQUAL )
 : EQUAL      ( a1 a2 --> f    list=list? )
  OVER OVER =
  IF DROP DROP TRUE EXIT THEN
  DUP INT? IF OVER INT? NOT ?DDFE
      CDR SWAP CDR = EXIT THEN
  DUP STR? IF OVER STR? NOT ?DDFE
      CDR SWAP CDR STR= EXIT THEN
  DUP VAR? IF OVER VAR? NOT ?DDFE
      CDR SWAP CDR = EXIT THEN
  OVER LST? OVER LST? AND NOT ?DDFE
      OVER CAR OVER CAR RECURSE
      IF CDR SWAP CDR RECURSE ELSE DROP DROP FALSE THEN ;



( Atom & List words / run-time )
 : I%     (  ad1 value --> ad2 )
  INT G-A G-C ;
 : O%  INT G-A  (  value --> ad  )
 : I$     (  ad1 value --> ad2 )
  STR G-A G-C ;
 : O$  STR G-A  (  value --> ad  )
 : I(:    (  ad1  --> ad2 ad3 )
  GWMC SWAP OVER G-C SWAP ;
 : O(:  GWMC DUP ;            (    --> ad1 ad2 )
 : (:)) NIL SWAP CDR! ;       ( ad  -->  )
 : I^  VAR G-A G-C ;          ( ad1 value -> ad2 )
 : O^  VAR G-A ;              ( value -> ad )
```

```
( SB, MEMBER )
 : SB        ( ad1 -> ad2 )
  DUP NIL? IF EXIT THEN
  DUP INT? IF INT SWAP CDR CONS EXIT THEN
  DUP STR? IF STR SWAP CDR CONS EXIT THEN
  DUP VAR? IF CDR @ EXIT THEN
  DUP CAR RECURSE SWAP CDR RECURSE CONS ;

 : MEMBER           ( ad1 ad2 -> f )
  BEGIN DUP NIL?
    IF DROP DROP FALSE TRUE
    ELSE OVER OVER CAR EQUAL
      IF DROP DROP TRUE TRUE
      ELSE CDR FALSE THEN
    THEN
  UNTIL ;
( MT, INSTALL )
 : MT         ( ad1 ad2 -> f )
  DUP NIL? IF = EXIT THEN
  DUP INT? IF OVER INT? NOT ?DDFE
             CDR SWAP CDR = EXIT THEN
  DUP STR? IF OVER STR? NOT ?DDFE
             CDR SWAP CDR STR= EXIT THEN
  DUP VAR? IF OVER NIL? ?DDFE
              CDR ! TRUE EXIT THEN
  OVER LST? NOT ?DDFE
    OVER CAR OVER CAR RECURSE
    IF CDR SWAP CDR SWAP RECURSE
    ELSE DROP DROP FALSE THEN ;

 : INSTALL  0 (CNT ! WMMIN WMP !
  WMS 0 DO FC WMMIN I VL 2* * + CAR! LOOP ;
( List notation )
 : %     ( {ad1} value -> ad2 )
  (CNT @ IF COMPILE I%  ELSE COMPILE O%  THEN  ; IMMEDIATE
 : $     ( {ad1} value -> ad2 )
  (CNT @ IF COMPILE I$  ELSE COMPILE O$  THEN  ; IMMEDIATE
 : &     ( {ad1} value -> ad2/value )
  (CNT @ IF COMPILE G-C THEN  ; IMMEDIATE
 : ^     ( {ad1} value -> ad2 )
  (CNT @ IF COMPILE I^  ELSE COMPILE O^  THEN  ; IMMEDIATE
 : (:    ( {ad1} -> ad2 ad3 }
  (CNT @ IF COMPILE I(: ELSE COMPILE O(: THEN
       1 (CNT +! 17 ; IMMEDIATE
 : :)    ( ad ->   )
  17 ?PAIRS
  -1 (CNT +!   COMPILE (:))  ;  IMMEDIATE
```

(Code continued on page 35.)

Forth for the 90's

Tom Zimmer
Milpitas, California

Yes, I've heard the rumors: Forth is a dead language, even companies that are still using it can't seem to find programmers to maintain their existing body of code. Everybody knows how easy it is to get C programmers—it is, after all, the wave of the present and future.

Of course, the C compilers available for the embedded (microcontroller) market are not very good, but they are bound to get better (someday). In the meantime, you have to do your development in assembly language, which isn't much fun, but we have to have the performance.

Oh, did I mention that embedded applications is a multi-billion dollar market, and growing by more billions every year?

For what, do you suppose, do people want to use even more microcontrollers? We already have computers in almost every imaginable home appliance, how could there be room for growth?

As it turns out, the people who design all these appliances keep thinking up more and better ways to use that processor's power—to enhance functionality, improve ease of use, and maintain high performance and quality. As the uses evolve, more processing power is needed and more complex tasks are performed. The current microcontrollers in VCRs and TVs are mostly four bit, but the market is starting to use eight-bit processors, with 16- and 32-bit processors just around the corner. This is partially due to the need for more processing power, but also due to the inefficiency of the compilers (C) being used to create the applications.

Forth can provide an inherently interactive development environment when the development engineer needs it to test and evaluate hardware. Forth can provide a high-performance, low-cost way to test and debug embedded applications. Development environments for C and assembly typically revolve around an in-circuit emulator (ICE), a multi-thousand-dollar tool you have to purchase *again* every time you switch to a new processor. Clearly a costly, though powerful, solution to the problem of embedded application development. There has to be a better way!

I believe the key (or at least one of them) to further and future growth for Forth is in interactive development systems for microcontrollers—systems that rely heavily on the resources of the host computer to provide editing, compiling, and interactive control of minimal target hardware. But not a system so power hungry that it needs a $25 thousand, 24-megabyte Sun workstation to run a simulation that takes four days to complete. I imagine a system that embodies a minimal monitor function in the target processor board, which is controlled by a powerful AT-class machine to give the user ICE-like power over target operations. I know I'm not the first to think of this, and there must be other people working in this direction, but I don't see many of them.

I have to say "imagine" above, because I have yet to see such a system brought to market. A friend of mine, Mike Mayo is doing things like this with an Intel 80196, providing assembly and (still primitive) Forth source-level debugging of an application using a monitor in the target of less than 1K. I have seen demonstration of it that make the hair stand up on the back of my neck. It is not complete, but the possibilities boggle the mind. I have also heard that Forth, Inc. has something like this, but have not seen it.

There are many people putting mini-Forth kernels on small cards, but they are still doing all their development on the card, which requires much more memory and power than the final application will need. This is fine for small production runs where the cost of the embedded controller can be high, but is usually cost prohibitive for large-market applications where every penny used in the controller cuts tens of thousands of dollars out of the profit. An ideal development system will impose little or no overhead (ROM, RAM, or performance) on

Forth can provide a high-performance, low-cost way to test and debug embedded applications.

the target application, either during development or in the final product. A difficult task perhaps, but not impossible. I am personally working toward this kind of development system, and I hope you will join me in these efforts to push Forth forward.

In the final analysis, the name of the game is, "How do I develop a reliable application in the least time, for the lowest cost per unit, while providing functionality that is a cut above the competition?"

The answer, of course, is "Forth"—at least it can be, with your effort and mine. We must provide tools that improve the efficiency and performance of engineers and programmers working on projects and applications that apply the power of microcontrollers to the challenges of the 90's. Hopefully, it will be easier to do than it is to say.

Tom Zimmer is a Forth veteran who has left a mark on the language. His public-domain F-PC is a luxurious Forth environment whose many developer-oriented features fly in the face of traditional minimalism. It reminds us that not everyone wants to start from a position of austerity, no matter how Zen-like, and it answers many who say, "If only Forth could—" with a resounding, "—it can and it does!"

FIG Board Terms Drawing to a Close

Call for Nominations

The nominating process for the selection of Directors for the 1991 FIG Board of Directors is starting. The candidates elected to the two available director positions will be able to serve a three-year term, with the possibility of reelection thereafter.

To be considered for nomination, or to obtain a nomination by petition, you should carefully read these instructions. The nomination and subsequent election processes take place as prescribed by our by-laws. As the following extract from Article VIII, Section 1 of the by-laws indicates, open elections are made possible by the timely completion of steps stretching over at least a several-month time period.

From the By-Laws...

(a) Nominating Committee. The Board of Directors shall appoint a Nominating Committee composed of at least two Directors to select qualified candidates for election to vacancies on the Board of Directors at least 120 days before the election is to take place. The Nominating Committee shall make its report at least 90 days before the date of the election, and the Secretary shall provide to each voting member a list of candidates nominated at least 60 days before the close of elections.

(b) Nominations by members. Any 25 Members may nominate candidates for directorships at any time before the 90th day preceding such an election. On timely receipt of a petition signed by the required number of Members, the Secretary shall cause the names of the candidates named on it to be placed on the ballot along with those candidates named by the Nominating Committee.

(c) The Corporation shall make available to all nominees, an equal amount of space in *Forth Dimensions* to be used by the nominee for a purpose reasonably related to the election.

(d) Should a petition be received, a ballot process will be provided to the voting membership. Otherwise, the Secretary shall cast a unanimous ballot for the candidates as proposed by the Nominating Committee.

Obtaining a Nomination

The Nominating Committee selects candidates for the ballot. FIG members who wish to become candidates this way should submit a letter requesting consideration by the Nominating Committee (c/o FIG office) before the deadline.

Alternately, a potential candidate shall obtain at least 25 signatures from FIG members and send this petition to the FIG Secretary (c/o FIG office) before the deadline. The names of qualifying candidates are placed directly on the voting ballot.

The deadline for submitting either nominating petitions or letters requesting consideration by the Nominating Committee is June 30, 1991. Send these items to the FIG office at P.O. Box 8231, San Jose, CA 95155.

(If the Secretary does not receive any nominating petitions by June 30, 1991, then the Secretary will cast a unanimous vote for the candidates selected by the Nominating Committee. In such a case, the membership at large will not receive voting ballots.)

The next important date after June 30 of this election year is July 14, if there are any petitions. It is the deadline for candidates to submit their candidate statements so that they can appear in the September-October issue of *Forth Dimensions*.

Ballots will be included in the September-October issue of *Forth Dimensions*, if necessary. The voting ballots must be returned to the FIG office by November 15, 1991. The newly elected directors will assume their duties at the next Board of Directors meeting.

Nominating Petitions must be worded as shown.

The following FIG members certify that they are current FIG members and nominate <candidate-name> to fill the position of Director of the FIG Board of Directors for the term November 1991 through November 1994.

Member Name (Please Print)	Member Signature	Member Number	Date Signed
<name1>	<name1>	<number1>	<date>
<name2>	<name2>	<number2>	<date>
<name3>	<name3>	<number3>	<date>
.	.	.	
.	.	.	
.	.	.	
<name25>	<name25>	<number25>	<date>
.	.	.	
.	.	.	

eForth—A Portable Forth Model

C. H. Ting
San Mateo, California

eForth is the name of a Forth model designed to be portable to a large number of the newer and more powerful processors which are available now and are becoming available in the near future. Originally it was called pigForth; however, because of very strong objections from many experienced Forth programmers, a less provocative name was adopted after much heated debate at the 1990 Rochester Forth Conference.

PIG originally stood for Post-Forth Interest Group, a term used by Mr. Andreas Gopold, a Forth programmer from Germany, to encourage people to think of the future of Forth, in light of recent advancements in computer hardware and software technologies. His feeling was that Forth in its present shape and form will not be useful for programmers in the future, because it does not provide enough support for program development, especially for large and very demanding projects. He thus felt that Forth has to develop in the direction as Tom Zimmer's FPC, Mitch Bradley's ForthMacs, and the Leibnitz he (Goppold) developed. These systems are huge because they incorporate lots of tools and utilities useful in the programming process.

I am stealing the "pig" for a different purpose. As we are marching into the nineties, we are confronting a host of new and very powerful microprocessors of different architectures and designs. The excitement is not the least bit less intense than in the seventies when the first crop of eight-bit microprocessors made their appearance. The Forth Interest Group captured the imagination of the first generation of personal computer enthusiasts by releasing the fig-Forth model riding on six different microprocessors. It is, thus, very interesting to see if we can build a Forth which can be easily ported to many, if not all, of the new and future microprocessors. This is pigForth.

The straightforward answer would have been to use the fig-Forth model and port it to the new processors. The most serious problems are that the fig-Forth model assumed that the host processor had to be an eight-bit processor and that the stack width was 16 bits. There are other problems and limitations in fig-Forth. In addition, the programming environment in which a computer is used has changed drastically over the last 15 years. In the following sections, I shall evaluate these interesting developments in an effort to lay down the ground rules and the fundamental design of eForth, which might give us a good handle on the new processors in the coming decade.

The Changing Time and Environment

In the seventies, personal computers were novelties, totally standalone systems you put on your desk and had them do work for you. (Or, more precisely, you willingly became enslaved to them.) But a personal computer, in order to be useful, had to contain the programming environment. Thus, one needed a CPU with memory, the keyboard, the display monitor, the disk drive, and the operating system including languages and utilities. A personal computer must be self-sufficient to be able to stand alone.

fig-Forth, like all Forth systems at that time, had to be self-sufficient also. It had to be able to support keyboard input, screen and printer output, and the disk drive for mass storage. An editor was always the first application a Forth user did, because that was the only way he could proceed with serious programming.

F83 was a giant step forward, and much human engineering went into it to facilitate programming, including a better editor, more extensive debugging tools, and a smoother interface to the operating system. Nevertheless, F83 was still an environment unto itself, tailored to a single user using a single machine.

The largest change in the eighties was the separation of the processor from the programming environment.

The largest change in the eighties was the separation of the processor from the programming environment, due to the proliferation of standard personal computers, i.e., the IBM-PC and its

Dr. C.H. Ting is respected throughout the Forth community for his many contributions. He edits the "More on Forth Engines" series (see FIG Mail Order Form) and currently is Vice-President of the Forth Interest Group.

Figure One. Personal computer system.

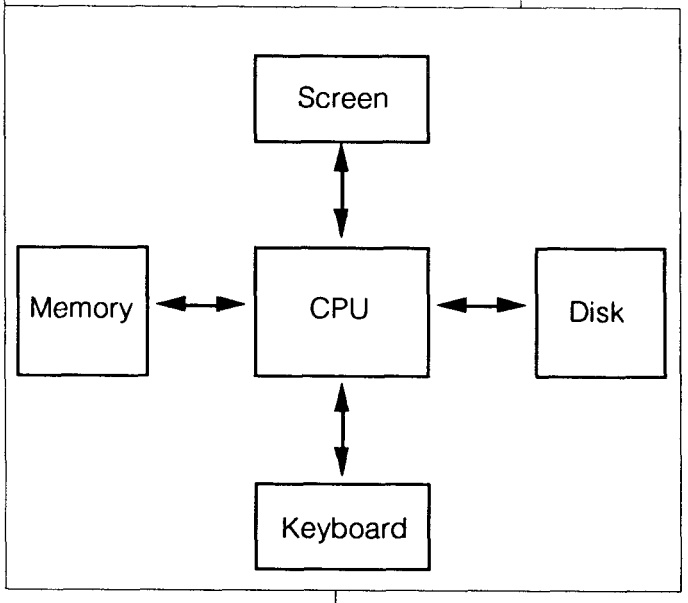
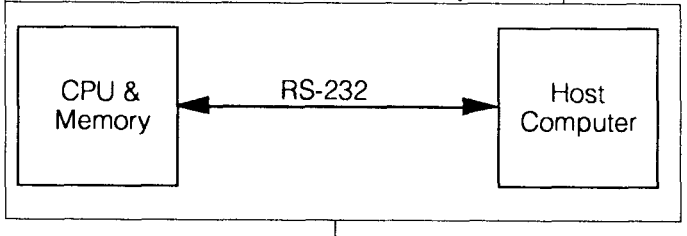


Figure Two. Embedded processor system.



clones. PCs became programming tools for other processors in separate boxes. I do not deny that there is still lots of programming done on PCs, for PCs; however, a very large portion of programming activities for newer and more advanced processors is carried out on host computers separated from the target computers.

The separation of the programming environment from the target processor had a great impact on the design of eForth—which I presume will go into the target processor, not the host computer. This is also of great advantage to the design of eForth, because eForth does not have to include features supported by the host computer, and eForth's requirements can be greatly simplified. In a sense, eForth is addressing the needs of embedded processors, the latest buzzword in the microprocessor world.

A PC-Based Development Platform

The microcomputer for which fig-Forth was designed can be shown schematically as in Figure One. It contained a CPU, some memory, and it had to talk to a keyboard, a monitor, and a disk.

The microcomputer eForth has to deal with is much simpler, as shown in Figure Two. It is a CPU with some memory, perhaps with some peripherals which cannot be specified. It is connected to the outside through an RS-232 cable. This RS-232 line may not be used in real applications, but it is connected to a host computer during programming, testing, and debugging. In certain applications, the RS-232 umbilical cord may never be severed, so that the host can download specific code to the target, to perform specific tasks depending on different circumstances. Many of the new generation of instruments are designed using this capability.

Software for PC-Based Development

Because of the proliferation of personal computers, it makes sense to use them as the interface to the latest crop of CPUs. To enhance our ability to use these new CPUs, a version of Forth is required that can easily be made to run on any one of them. There are many tradeoffs to consider in designing such a universal Forth, yet its ability to be ported to many different CPUs remains the primary concern.

There are also many models of Forth which we can draw upon in order to take advantage of the wisdom accumulated over the last twenty years in the Forth community. These Forth models include fig-Forth by Bill Ragsdale, cmForth by Chuck Moore, FCode by Mitch Bradley, and bForth by Bill Muench.

The following is a list of requirements for the universal eForth model. Note that the editor, file server, and other utilities are provided in the host system, so the target CPU (running eForth) does not have to provide these services.

- A small set of kernel words, which is machine dependent. Only this set of words is rewritten for a specific CPU. A minimum-kernel word set encourages porting eForth to new CPUs.
- A high-level word set built on the kernel words. The high-level definitions must be portable to all target CPUs, including eight-, 16- and 32-bit machines.
- The only I/O words are `KEY`, `EMIT`, and `?KEY` because the only I/O device in the target is the RS-232 port.
- Image upload and download use `EXPECT` and `TYPE` built from `KEY` and `EMIT`.
- `EVALUATE` is used to interpret terminal input and downloaded source code.
- Source code must be provided in the MASM form. Metacompilation is undesirable because it is an obstacle to acceptance outside of the Forth community. (See sidebar.)
- Two source files are needed for each CPU: a kernel file containing CPU-specific code and a universal high-level-code file.
- Total number of words is between 150 to 200. Mechanisms must be included for extension word sets.

Seeding the eForth Model

The ideal of eForth has been tossed around for a year now. It has been discussed extensively at the Silicon Valley FIG Chapter, in the last few FORML sessions of the chapter meetings. To help decide how to implement eForth, various models were evaluated, including fig-Forth, F83, cmForth, LaForth, ZenForth, Fcode, the ANS Standard basis, and bForth. Recently, the consensus favored bForth, whose principal author is Bill Muench. It has a very small core of machine-specific words, and also has most of the features desired in eForth.

I took the model and

implemented it on an IBM-PC in MicroSoft Assembler (MASM) format. The MS-DOS-equipped personal computer was chosen because it is such standard gear, and MASM was chosen because it, also, is readily accessible. MASM and a PC are within the reach of every serious programmer.

The MASM source listings we produced have become the boilerplate for subsequent implementations of eForth. As such, this first implementation defines what the eForth model is in more profound ways than any other documentation. Furthermore, MASM should be the tool used to create other eForth implementations, rather than a metacompiler, as explained in the sidebar accompanying this article.

MASM was designed to assemble programs specifically for the 80x8x class of processors. However, it can be used to produce code for other types of processors after some hand-assembled code is inserted via the use of special MASM data-handling commands (DB, DW, DD, DQ, and DT).

eForth is not really designed for the PC, but for the CPUs likely to be used in embedded applications systems. Nevertheless, the MASM listings can be used to create a working eForth system when assembled on 80x8x systems running DOS.

Because a PC version of eForth has been created first, a large number of programmers already have the equipment needed to use it. The resulting Forth system is tiny, with a code area of six Kbytes and a name area of two Kbytes. Using MASM and running the executable object files it can produce, the

(Continued on page 30.)

How Metacompilation Stops the Growth Rate of Forth Programmers

Metacompilation is a trap. Forth seems to be easy to use and master until you try to do metacompilation. Conceptually, metacompilation is straightforward, because a Forth system contains all the necessary ingredients to recompile itself or a target system using a different CPU. However, there are many tricky loose ends which have to be tightened before you can succeed in the metacompilation. Only the very persistent Forth programmers, with great effort, can overcome the frustration of metacompilation to attend the Tao of Forth.

Most Forth systems were built by Enlightened Forthians after they attended Tao and closed the loop on metacompilation. To prove that they indeed had the Tao of Forth, they might even include the metacompiler in their systems or sell it as an extra-cost option. Forth systems produced through metacompilation are elegant, appearing to be very tight, yet impossible to understand fully. Forth and Forth's metacompiler are like chicken and egg, you cannot tell which comes first. You cannot understand one without understanding the other first. The most likely outcome is that you leave it without understanding either. This is precisely the syndrome which stops the growth rate of Forth programmers.

fig-Forth broke this self-locking mechanism by offering source code in regular assembly languages, not in Forth. Writing Forth in assembly is clumsy, the source listing looks much more complicated than Forth written in Forth. However, written in assembly, fig-Forth hid nothing from the user. The process of building a Forth system can be understood completely by programmers fluent in the assembly languages of the released fig-Forth models.

The immense success and popularity of fig-Forth clearly showed that complete knowledge of a Forth system could be, and has been, appreciated by a large segment of assembly programmers.

Following fig-Forth, and after the Forth-83 Standard was established, [Laxen & Perry's] F83 was released. Although F83 was much superior than fig-Forth in its capabilities and its integrated environment, with many utilities built into the system, it did not penetrate the programming community outside the existing Forth society. The most significant stumbling block was that F83 was built by metacompilation. Although the complete source code of F83 system and the metacompiler are published, very few Forth programmers fully understood the system. How could we expect programmers off the streets to make use of F83? Failing to penetrate deeply into the assembly programmer's community, it is not unexpected that FIG membership dwindled since 1985. We are preaching to the choir and forget that if the Forth community is to grow, the new Forthians must be converted from non-Forth users. Did we provide means for their conversion?

fig-Forth defeated this mechanism by offering source code in regular assembly languages.

eForth counters the trend towards metacompiled Forths, along with their implementation complexities. Its reduced breadth is compensated for by the PC itself and by the PC's operating system. With the possibility of off-loading certain housekeeping tasks, such as file support, the eForth approach offers the kind of benefits associated with both minimal Forths and fat Forths. This helps provide a development environment equipped to meet the needs of programmers who have to deal with the CPUs of the 1990's, because it is based upon the tools and environments of the 1990's.

Very powerful CPUs are marching into the marketplace. Every one of them cries out for a good Forth implementation so that users can fully realize their capabilities in number crunching, digital signal processing, high-speed telecommunication and networking, etc.

To make it very easy to port eForth to the new CPUs, eForth must be as simple as possible. Its portability is enhanced through having very few machine-code routines. Nevertheless, talented implementors are free to optimize the model's otherwise high-level code, thereby making effective use of the specialized capabilities of a given CPU.

—Dr. C. H. Ting

Menu Words

John Edgecombe
Oxnard, California

This utility owes its inspiration to Frans Van Duinen's article (*FD* VII/2), referenced in Screen Zero of the accompanying code. It may come as a surprise to others, as it did to me, that the word .X" (Screen Five), essentially identical to .AX" in Frans' code, uses [COMPILE] to compile a non-immediate word. This takes some thought, and it is informative to check out the actions of ['], ', COMPILE, and [COMPILE] in your system to see the differences.

MASTER (Screen 11) is laid down in the dictionary as shown in Figure One (in MicroMotion's direct-threaded Z80 Forth.

Everything looks like a normal colon definition until you get to (.X"), which is laid down by the ticked CFA of the word to be executed when this menu item is selected.

That is followed by a counted string. The character after the count is the key that selects this menu item, followed by its description.

Thus, SLAVE1 (Screen Nine) will look like this on the screen:

```
SLAVE2 MENU
A -> ITEM 4
B -> ITEM 6
C -> SLAVE 1
```

When loading SLAVE2, notice that SLAVE1 isn't compiled yet, so the deferred word SLAVE.1 is substituted and later vectored to SLAVE1. Thus, you can go anywhere from anywhere!

BALLOT^ (Screen Two) executes as a top-of-menu-stack @ word, while the 14 bytes allotted to it (totalling 16) function as a stack for holding the CFAs of menus. The bottom-of-stack is seeded by START with the CFA stored in the deferred word ITINERARY. BALLOT^FILL spreads the seed throughout the stack in order to initialize it. Push and pop functions are handled by >BALLOT and BALLOT>. >BALLOT will not overwrite the seed, so stack overflow affects intermediate menus but not the root menu; and underflow is harmless.

In BALLOT (Screen Seven), the choice of keystroke values may be system dependent. The exit key was originally a null, but that is non-portable to CP/M systems because the OS filters out nulls so that BALLOT never sees them. A little interactive fiddling with KEY . illuminated the situation. EXIT is a programming aid to let you get out of the menu loop without rebooting. (DISPLAY.BALLOT) and (DISPLAY.RESTORE) can be left unvectored to scroll the screen. They can be vectored to DARK to home the cursor and clear the screen. Fancy menu boxes are left to the readers to design.

Why is TOUR a separate word? BALLOT^ IP! could be substituted inside BALLOT, but the hint in the name TOUR would be lost and the growth of the return stack would not be obvious. IP! could be renamed IP_PUSH because it causes return-stack growth. The R> DROP in >BALLOT (Screen Three) pops the leftover CFA, circumventing return-stack growth. The code may have system-dependent side effects: SEE goes on a wild goose chase.

The "help" words in Screen Eight are an obvious candidate for customization. The simple demonstration in Screens Nine through 11 allow you to get some feel for the menus. After backtracking four levels, the average user is likely to use Esc to jump back to the root menu.

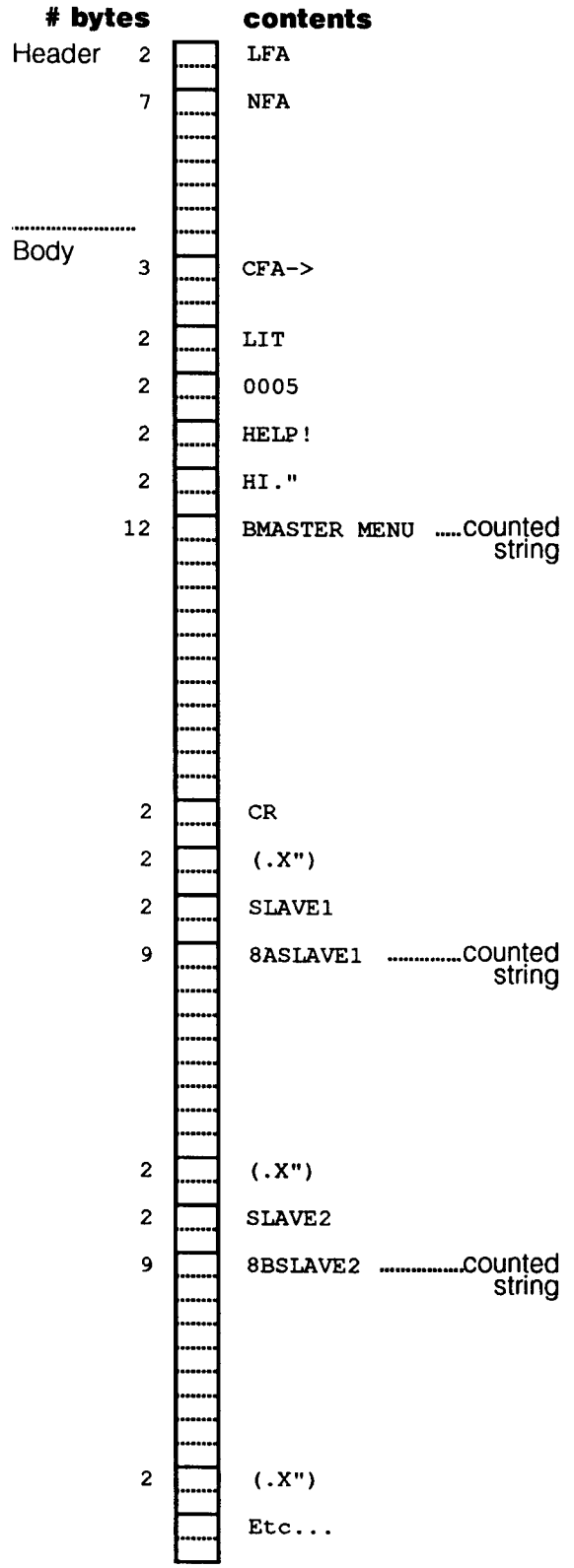
I hope the reader finds this a useful addition to Forth.

(Code begins on page 20.)

Do we dare tell them that Forth isn't a vitamin, it's an aphrodisiac?

John Edgecombe is a Professional Land Surveyor. He uses Forth for 32-bit math and computational geometry on eight-bit processors, while avoiding floating point.

Figure One. Dictionary structure of MASTER.



Epilog

There will be attrition in Forth as there is in life, but where are the new births in Forth?

A student soon learns that original thoughts don't conform to multiple-choice answers. Students are encouraged to regurgitate, but not to think; the universal refrain is, "I hate word problems." Real-world problems aren't trivial. Facts do not stand alone, they must be interwoven into the fabric of the intellect, just as a good cook seasons the evening meal.

Forth is a gourmet language. Canned languages can be very convenient, like the frozen dinner: just heat and eat. It's not a feast, but it doesn't take a gourmet chef's skills, either.

If you describe a gourmet meal to someone who has never indulged, they may reply, "Who needs it?" or "I just had my checkup, and the doctor didn't find any vitamin Forth deficiency symptoms."

Do we dare to tell these people that Forth isn't a vitamin, it's an aphrodisiac?

(Editorial, from page 4.)

to be held in October 1991. He invites authors to send their work for inclusion; articles must contain fewer than 5,000 words and must be received by August 1. (Send papers or write for details from Mr. Zeng Jing, Office for Soliciting Contributions, China Management Software College, Xue Yuan Road, Ji Men Li 100088, District of Hai Dian, Beijing, China.)

Also in October of this year, the seventh EuroFORML conference is to be held at the Casino of Mariánské (Marienbad), Czechoslovakia. This year's conference will again focus on Forth in real-time applications. To be included in the proceedings, papers must be received by September 23. *Early registration is advised.* (For information, contact Marina Kern or Klaus Schleisiek-Kern, Uhlenhorster Weg 3, D-2000 Hamburg, Germany; phone +49 40 2296441 or fax +49 40 2297205.)

P.S. About our Facelift...

Content is primary, but form also communicates: a publication's design significantly contributes to how the public feels about its message. And while *FD*'s previous design served us well, we came to need more flexibility and efficiency on our pages and a more contemporary tone. So this issue unveils a fresh new design for *Forth Dimensions*. It will evolve in coming issues, but I invite comment on what you see here.

Desktop publishing and telecommunications long have helped us to achieve good results within our resources. Their advent a few years ago weaned us from the then-usual outside typesetting service; today's increasingly sophisticated tools are assisting us to streamline the production process further in ways that bring new economy without loss of quality. We hope you agree!

—Marlin Ouverson
Editor

```
SCR# 0
\ Menu words                                                  5DEC90JWE

Menu words for heirarchical menu tree traversal with a 7 level
lookback stack featuring return to master menu on stack
underflow.

.X" is a modification of .AX" from Frans Van Duinen's
article "Menus in forth", FORTH DIMENSIONS, Vol. VII No. 2,
pg 15-19.

: PERFORM  @ EXECUTE ;
```

```
SCR# 1
\ Menu words                                                  5DEC90JWE

ONLY FORTH DEFINITIONS ALSO   DECIMAL

   2 7  THRU    \ MENU WORDS
     8  LOAD    \ HELP WORDS
   9 11 THRU    \ DEMONSTRATION
```

```
SCR# 2
\ Menu words                                                  5DEC90JWE
DECIMAL

DEFER  (DISPLAY.BALLOT)    ' NOOP IS (DISPLAY.BALLOT)
DEFER  (DISPLAY.RESTORE)   ' NOOP IS (DISPLAY.RESTORE)
DEFER  (HELP)              ' NOOP IS (HELP)
DEFER  ITINERARY
0 VALUE BALLOT^            14 ALLOT
VARIABLE OLDX
VARIABLE .X~

: HI."      COMPILE +HILITE  [COMPILE] ."  COMPILE -HILITE ;
  IMMEDIATE

: HITYPE    COMPILE +HILITE  COMPILE TYPE  COMPILE -HILITE ;
  IMMEDIATE
```

```
SCR# 3
\ Menu words                                                  5DEC90JWE

: >BALLOT  ( --)  R> R> R> DROP  SWAP >R
  [ ' BALLOT^ >BODY DUP ] LITERAL
  [ DUP 2+ ]              LITERAL 12 CMOVE>  LITERAL ! ;

: BALLOT>  ( --)  BALLOT^  3 -  ( ->BODY)  OLDX !
  [ ' BALLOT^ >BODY DUP 2+ ]  LITERAL LITERAL  14 CMOVE ;

: BALLOT^FILL  ( --)  [ ' BALLOT^ >BODY DUP 2+ ] LITERAL
  LITERAL  14 CMOVE> ;
```

FIG MAIL ORDER FORM

NEW

HOW TO USE THIS FORM: Please enter your order on the back page of this form and send with your payment to the Forth Interest Group.
Most items list three different price catagories: USA, Canada and Mexico / Other countries Surface Mail / Other countries Air Mail
Note: Where only two prices are listed, Surface Mail is not available.

FORTH DIMENSIONS BACK VOLUMES

The six issues of the volume year (May-April)

Volume 1 Forth Dimensions (1979/80) 101 - $15/16/18
Introduction to FIG, threaded code, "TO" variables. fig-Forth.

Volume 2 Forth Dimensions (1980/81) 102 - $15/16/18
Recursion, file naming, Towers of Hanoi, CASE contest, input number word set, 2nd FORML report, FORGET, VIEW.

Volume 3 Forth Dimensions (1981/82) 103 - $15/16/18
Forth-79 Standard, Stacks, HEX, data base, music, memory management, high level interrupts, string stack, BASIC compiler, recursion, 8080 assembler.

Volume 4 Forth Dimensions (1982/83) 104 - $15/16/18
Fixed-point Trig, Fixed-point square root, fractional arithmetic, CORDIC algorithm, interrups, stepper motor control, source screen documentation tools, recursion, recursive decompiler, file systems, Quick Text Formatter, romable Forth, Indexer, Forth-83 Standard, teaching Forth, Algebraic Expression Evaluator.

Volume 5 Forth Dimensions (1983/84) 105 - $15/16/18
Computer graphics, 3-D animation, double precision math words, overlays, recursive sort, a simple mulit-tasker, meta-compilation, voice output, number utility, menu-driven software, vocabulary tutorial, vectorerd execution, data acquisition, fixed-point logarithms, Quicksort, fixed-point square root.

Volume 6 Forth Dimensions (1984/85) 106 - $15/16/18
Interactive editors, anonymous variables, list handling, integer solutions, control structures, debugging techniques, recursion simiphores, simple I/O words, Quicksort, high-level packet communications, China FORML.

Volume 7 Forth Dimensions (1985/86) 107 - $20/22/25
Generic sort, Forth Spreadsheet, control structures, psuedo-interrupts, number editing, Atari Forth, pretty printing, code modules, universal stack word, polynomial evaluation, F83 strings.

Volume 8 Forth Dimensions (1986/87) 108 - $20/22/25
Interrupt driven serial input, data base functions, TI 99/A, XMODEM, on-line documentation, dual-CFA's, random numbers, arrays, file query, Batcher's sort, screenless Forth, classes in Forth, Bresenham line-drawing algorithm, unsigned division, DOS file I/O.

Volume 9 Forth Dimensions (1987/88) 109 - $20/22/25
Fractal landscapes, stack error checking, perpetual date routines, headless compiler, execution security, ANS-Forth meeting, Computer Aided Instruction, local variables, transcendental functions, education, relocatable Forth on 68000.

Volume 10 Forth Dimensions (1988/89) 110 - $20/22/25
dBase file access, string handling, local variables, data structures, object-oriented Forth, Linear Automata, standalone applications, 8250 drivers, serial data compression.

Volume 11 Forth Dimensions (1989/90) 111 - $20/22/25
Local variables, graphic filling algorithms, 80286 extended memory, expert systems, Quaternion Rotation calculation, multiprocessor Forth, Double-entry bookkeeping, binary table search, Phase-Angle Differential Analyzer, sort contest.

Volume 12 Forth Dimensions (1990/91) 112 - $20/22/25
Floored division, stack variables, embedded control, Atari Forth, optomizing compiler, dynamic memory allocation, Smart RAM, extended precision math, interrupt handling, neural nets, Soviet Forth, arrays, meta-compilation.

NEW

FORML CONFERENCE PROCEEDINGS

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

1980 FORML PROCEEDINGS 310 - $30/31/40
Address binding, Dynamic Memory Allocation, Local Variables, Concurrency, Binary Absolute & Relocatable Loader, LISP, How to manage Forth Projects, N-Level File System, Documenting Forth, Forth Structures, Forth Strings.

1981 FORML PROCEEDINGS 311 - $45/48/55
CODE-less Forth Machine, Quadruple Percision Arithmetic, Overlays, Executable Vocabulary Stack, Data Typing in Forth, Vectored Data Structures, Using Forth in a Classroom, Pyramid Files, BASIC, LOGO, Automatic Cueing Language for Multi-Media, NEXOS - a ROM Based Multi-tasking Operating System

1982 FORML PROCEEDINGS 312 - $30/31/40
Rockwell Forth Processor, Virtual Execution, 32-bit Forth, ONLY for Vocabularies, Non-IMMEDIATE Looping words, NUMBER Input Wordset, I/O Vectoring, Recursive Data Structures, Programmable Logic Compiler.

1983 FORML PROCEEDINGS 313 - $30/32/40
Non-Von Neuman Machines, Forth Instruction Set, Chinese Forth, F83, Compiler & Interpreter Co-Routines, Log & Exponential Function, Rational Arithmetic, Transcendental Functions in Variable-Precision Forth, Portable File System Interface, Forth Coding Conventions, Expert Systems.

1984 FORML PROCEEDINGS 314 - $30/33/40
Forth Expert Systems, Consequent-Reasoning Inference Engine, Zen Floating Point, A Portable Graphics Wordset, 32-bit Forth, HP71B Forth, NEON-Object Oriented Programming, Decompiler Design, Arrays and Stack Variables.

1985 FORML PROCEEDINGS 315 - $30/32/40
Threaded Binary Trees, Natural Language Parsing, Small Learning Expert System, LISP, LOGO in Forth, Prolog Interpreter, BNF Parser in Forth, Formal Rules for Phrasing, Forth Coding Conventions, Fast High-Level Floating-Point, Forth Component Libaray, Forth & Artificial Intelligence, Electrical Network Analysis, Event Driven Multi-tasking.

REISSUED

1986 FORML PROCEEDINGS 316 - $30/32/40
Threading techniques, Prolog. VLSI Forth Microprocessor, Natural Language Interface, Expert System Shell, Inference Engine, Multiple Inheritance System, Automatic Programming Environment.

1987 FORML PROCEEDINGS 317 - $40/43/50
Includes papers from '87 euroFORML Conference. 32 bit FORTH, neural networks, control structures, AI, optimizing compilers, hypertext, Field and Record Structures, CAD Command Language, Object Oriented Lists, Trainable Neural Nets, Expert systems.

1988 FORML PROCEEDINGS 318 - $24/25/34
Human interfaces, Simple Robotics Kernel System, MODUL Forth, Language topics, hardware, Wil's workings & Ting's philosophy, Forth hardware applications, ANS Forth session, Future of Forth in AI Applications.

1989 FORML PROCEEDINGS 319 - $40/43/50
Includes papers from '89 euroFORML. PASCAL to Forth, extensible optimizer for compiling, 3-D measurement using object-oriented Forth, CRC polynomials, F-PC, Harris C cross-compiler, modular approach to robotic control, RTX recompiler for on-line maintenance, module, trainable neural nets.

1990 FORML PROCEEDINGS 320 - $40/43/50
NEW
Forth in Industry, Communications Monitor, 6805 Development. 3-Key Keyboard, Documentation Techniques, Object Oriented Programming, Simplest Forth Decompiler, Error Recovery, Stack Operations, Process Control Event Management, Control Structure Analysis, Systems Design Course, Group Theory using Forth

1988 AUSTRALIAN PROCEEDINGS 380 - $24/25/34
Proceedings from the first Australian Forth Symposium held May, 1988 at the University of Technology in Sydney. Subjects include training, parallel processing, programmable controllers, Prolog, simulations & applications.

BOOKS ABOUT FORTH

ALL ABOUT FORTH, 3rd ed., June 1990, Glen B. Haydon 201 - $90/92/105
An annotated glossary of most Forth words in common usage, including F-79, F-83, F-PC, MVP-Forth. Implementation examples in high-level Forth and/or 8086/88 assembler. Useful commentary is given for each entry.

THE COMPLETE FORTH, Alan Winfield 210 - $14/15/19
A comprehensive introduction including problems with answers (Forth79)

F83 SOURCE, Henry Laxen & Michael Perry 217 - $20/21/30
A complete listing of F83 including source and shadow screens. Includes introduction on getting started.

FORTH, APPLICATIONS IN ENGINEERING AND INDUSTRY
NEW
John Matthews (hard cover only) 218 - $66/68/76
This book introduces hardware engineers to control of microprocessor-based products through Forth programming. How to add to the conventional assembly language environment to facilitate development, factory test, & product maintenance is showed. F83 & fig-Forth are compared. The design of a simple controller on which many of the examples given can be reproduced exactly, together with support software for a PC, are given in the appendices.

FORTH: A TEXT AND REFERENCE 219 - $31/32/41
Mahlon G. Kelly & Nicholas Spies
A textbook approach to Forth with comprehensive references to MMS-FORTH and the 79 and 83 Forth Standards.

FORTH ENCYCLOPEDIA, Mitch Derick & Linda Baker 220 - $30/32/40
A detailed look at each fig-FORTH instruction.

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

FORTH NOTEBOOK II, Dr. C. H. Ting 232a - $25/26/35
Collection of research papers on various topics as image processing, parallel processing and miscellaneous applications.

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

LIBRARY OF FORTH ROUTINES AND UTILITIES, James D. Terry 237 - $23/25/35
Comprehensive collection of professional quality computer code for Forth; offers routines that can be put to use in almost any Forth application, including expert systems and natural language interfaces.

MASTERING FORTH, 2nd Edition, Anita Anderson & Martin Tracy 240 - $22/23/28
A step-by-step tutorial including each of the commands of the Forth-83 International Standard; with utilities, extensions and numerous examples.

OBJECT ORIENTED FORTH, Dick Pountain 242 - $28/29/34
Implementation of Data Structures. First book to make object orientated programming available to users of even very small home computers.

STACK COMPUTERS, THE NEW WAVE 244 - $62/65/72
Philip J. Koopman, Jr. (hard cover only)
Presents an alternative to Complex Instruction Set Computers (CISC) and Reduced Instruction Set Computers (RISC) by showing the strengths and weaknesses of stack machines. (hard cover only)

STARTING FORTH, 2nd Edition, Leo Brodie 245 - $29/30/38
In this edition of Starting Forth, the most popular and complete introduction to Forth, syntax has been expanded to include the new Forth '83 Standard.

TOOLBOOK OF FORTH 267 - $23/25/35
(Dr.Dobb's) Edited by Marlin Ouverson
Expanded and revised versions of the best Forth articles collected in the pages of Dr.Dobb's Journal.

TOOLBOOK, V.1 & DISK (MS-DOS) 267a - $40/42/50

TOOLBOOK OF FORTH, V.2, (Dr. Dobbs) 268 - $30/32/40
Complete anthology of FORTH programminmg techniques and developments, picks up where V.1 left off. Topics include programming windows, extended control structures, design of a FORTH target compiler and more.

TOOLBOOK, V.2 & DISK (MS-DOS) 268a - $46/48/56

REFERENCE

ANS X3J14 BASIS DOCUMENT 306 - $15/16/20
Current - August 1990
Working document of the X3J14 ANS Forth Committee, changes frequently, but useful as a working tool.

FORTH 83-STANDARD 305 - $15/16/18
Authoritative description of 83-Standard Forth. Reference, not instruction.

SYSTEMS GUIDE TO fig-FORTH 310 - $25/28/30
C. H. Ting - 2nd Edition, 1989
Hows and Whys of the fig-Forth Model by Bill Ragsdale, Internal structure of fig-Forth system.

BIBLIOGRAPHY OF FORTH REFERENCES 340 - $18/19/25
3rd Edition, January 1987
Over 1900 references to Forth articles throughout computer literature.

F-PC USERS MANUAL, 2nd Edition, V3.5 350 - $20/21/27
Users Manual to the public domain Forth system optimized for the IBM-PC/XT/AT computer. A fat, fast system with many tools.

F-PC TECHNICAL REFERENCE MANUAL 351 - $30/32/40
A must if you need to know the inner workings of F-PC.

Contributions from the Forth Community

The FIG "Contributions From the Forth Community" disk library contains author submitted donations, generally including source, for a variety of computers on their respective disk formats. The files usage is determined by the author as Public Domain, Shareware, or use with some restrictions. This library does not contain "For Sale" applications.
To submit your "Contributions", send them to the FIG Publications Committee.

The cost is per disk: $6/9 for one (1)
$25/28 for any five (5)

FLOAT4th.BLK V1.4 Robert L. Smith C001 - (1)
Software Floating-Point for fig, Poly, 79-STD, 83-STD Forths. IEEE Short 32-bit, Four standard functions, Square Root and Log. **IBM.**

NEW **Games in Forth** C002 - (1)
Misc Games, GO, TETRA,Life, ... Source. **IBM**

F83 V2.01, Mike Perry & Henry Laxen C100 - (1)
The newest version that has been ported to a variety of machines. Editor, assembler, decompiler, meta-compiler. Source and shadow screens. Manual available separately (see items 217 & 235). Base for other F83 applications. **IBM, 83.**

F-PC V3.53, Tom Zimmer C200 - (5)
A full Forth system with pull-down menus, sequential files, editor forward assembler, meta-compiler, floating point. Complete source and Help files. Manual for V3.5 available separately (see items 350 & 351). Base for other F-PC applications. Hard disk recommended. **IBM, 83.**

F-PC TEACH V3.5, Lessons 0-7 Jack Brown C201a - (2)
Forth classroom on disk. First seven lessons from Jack Brown of BC Institute of Technology on learning Forth. **IBM, F-PC.**

VP-Planner Float for F-PC, V1.01 Jack Brown C202 - (1)
Software floating point engine behind the VP-Planner spreadsheet. 80-bit (temporary-real) routines with Transcendental Functions, NUMBER I/O support, vectors to support numeric coprosessor overlay and user NAN checking. **IBM, F-PC.**

F-PC Graphics V4.2f, Mark Smiley C203a - (3)
The latest versions of a number of new graphics routines including CGA, EGA and VGA suppport, with numerous improvements over earlier versions created or supported by Mark Smiley. **IBM, F-PC.**

PocketForth V1.4, Chris Heilman C300 - (1)
Smallest complete Forth for the Mac. Access to all Mac functions, files, graphics, floating point, macros, create stand-alone applications and DA's. Source and manual included. **MAC**, based on fig & *Starting Forth*

NEW **Yerkes Forth V3.3** C350 - (2)
Complete Object Oriented Forth for the Mac. Object access to all Mac functions, files, graphics, floating point, macros, create stand-alone applications. Source, tutorial, assembler and manual included. **MAC**

JLISP V1.0, Nick Didkovsky C401 - (1)
LISP interpreter invoked from Amiga JForth. The nucleus of the interpreter is the result of Martin Tracy's work. It has been extended to allow the LISP interpreter to link to and execute JForth words. It can communicate with JForth's ODE (Object Development Environment). **AMIGA, 83.**

PYGMY V1.3, Frank Sergeant C500 - (1)
A lean, fast Forth with full source code. Includes full screen editor, assembler, and meta-compiler. Up to 15 files open at one time. **IBM.**

KForth, Guy Kelly C600 - (3)
A full Forth system with windows, mouse, drawing, and modem packages. Source code and documentation included. **IBM, 83.**

NEW **ForST**, John Redmond C700 - (1)
Forth for the Atari ST. Source and documentation included. **Atari ST.**

ACM - SIGFORTH

The ACM SIGForth Newsletter is published quarterly by the Association of Computing Machinery, Inc. SIGForth's focus is on the development & refinement of concepts, methods and techniques needed by Forth professionals.

Volume 1 Spring 1989 900 - $6/7/9
F-PC, Glossary utility, Euroforth, SIGForth '89 Workshop summary (real-time software engineering), Intel 80x8x.

Volume 1 Summer 1989 901 - $6/7/9
Metacompiler in cmForth, Forth exception handler, string case statement for UF/Forth.

Volume 1#3 Fall 1989 902 - $6/7/9
1802 simulator, tutorial on multiple threaded vocabularies.

Volume 1#4 Winter 1989 903 - $6/7/9
Stack frames, duals: an alternative to variables, PocketForth.

Volume 2#2 December 1990 904 - $6/7/9
BNF Parser, Abstracts 1990 Rochester Conf, F-PC Teach

Volume 2#3 March 1991 904 - $6/7/9
Tethered Forth Model, Abstracts 1990 SIGFORTH Conf

NEW **1989 SIGForth Workshop Proceedings** 931 - $20/21/26
Software Engineering, Multi-tasking, interrupt driven systems, object oriented Forth, error recovery & control, virtual memory support, signal processing.

MISCELLANEOUS

T-SHIRT "May the Forth Be With You" 601 - $12/13/15
(Specify size: Small, Medium, Large, Extra-Large on order form)
White design on a dark blue shirt.

POSTER (BYTE Cover) 602 - $5/6/7

FORTH-83 HANDY REFERENCE CARD 683 - FREE

WE'RE SURE YOU WANTED TO KNOW SERIES

NEW

***Forth Dimensions*, Article Reference** 151 - $4/5
A listing of Forth articles by keyword from *Forth Dimensions* Volumes 1 thru 12, 1978 thru 1991

FORML, Article Reference 152 - $4/5
A listing of Forth articles by keyword, author, and date from the FORML Conference Proceedings, 1980 thru 1989

MORE ON FORTH ENGINES

- $15/16/18

Volume 10 January 1989 810
RTX reprints from 1988 Rochester Forth Conference, object oriented cmForth, lesser Forth engines.

Volume 11 July 1989 811
RTX Supplement to *Footsteps in an Empty Valley*, SC32, 32 bit Forth engine, RTX interrupts/utility.

Volume 12 April 1990 812
ShBoom Chip architecture and instructions, Neural Computing Module NCM3232, pigForth, Binary Radix Sort on 80286, 68010, and RTX2000.

Volume 13 October 1990 813
PALs of the RTX2000 Mini-BEE, EBForth, AZForth, RTX2101, 8086 eForth, 8051 eForth.

fig-FORTH ASSEMBLY LANGUAGE SOURCE

Listings of fig-FORTH for specific CPUs and machines with compiler security and variable length names. - $15/16/18

fig-FORTH INSTALLATION MANUAL 501
Glossary model editor - we recommend you purchase this manual when purchasing the source code listings below.

1802	513 - March 81	**9900**	519 - March 81
6502	514 - September 80	**Apple II**	521 - August 81
6800	515 - May 79	**IBM-PC**	523 - March 84
6809	516 - June 80	**PDP-11**	526 - January 80
8080	517 - September 79	**VAX**	527 - October 82
8086/88	518 - March 81	**Z80**	528 - September 82

MEMBERSHIP IN THE FORTH INTEREST GROUP

The Forth Interest Group is a world-wide, non-profit, member-supported organization with over 1,500 members and 40 chapters. FIG membership includes a subscription to the bi-monthly magazine, *Forth Dimensions*. FIG also offers its members an on-line data base, a large selection of Forth literature and other services.

Cost is $40.00 per year for USA & Canada surface mail; $46.00 Canada air mail; all other countries $52.00 per year. No sales tax, handling fee, or discount on membership.

When you join, your first issue will arrive in four to six weeks, subsequent issues will be mailed to you every other month as they are published - six issues in all. You will also receive a membership card and number which entitles you to a 10% discount on publications from FIG. Your member number will be required to receive the discount, so keep it handy.

Dues are not deductable as a charitable contribution for U.S. federal income tax purposes, but may be deductable as a business expense where applicable.

FORTH INTEREST GROUP

P.O. BOX 8231 SAN JOSE, CALIFORNIA 95155 (408) 277-0668 (408)286-8988(FAX)

Name _____

Company _____

Street _____

City _____

State/Prov. _____ Zip _____

Country _____

Daytime phone _____

OFFICE USE ONLY

By_____ Date_____ Type_____

Shipped by_____ Date_____

UPS USPS XRDS

Wt._____ Amt._____

BO By _____ Date_____

Wt._____ Amt._____

Item #	Title	Qty.	Unit Price	Total
	*MEMBERSHIP →			SEE BELOW

☐ CHECK ENCLOSED (Payable to: Forth Interest Group)

☐ VISA ☐ MasterCard

Card Number _____

Expiration Date _____

Signture _____

($15.00 minimum on all VISA/MasterCard orders)

Sub-Total	
10% Member Discount Member # _____	
Sub-Total	
**Sales Tax (CA only)	
Mail Order Handling Fee	$3.00
☐ New ☐Renewal *Membership $40/46/52	
* Enclosed is $40/46/52 for 1 full years dues. This includes $34/40/46 for Forth Dimensions.	

PAYMENT MUST ACCOMPANY ALL ORDERS

MAIL ORDERS
Send to:
Forth Interest Group
PO Box 8231
San Jose, CA 95155

PHONE ORDERS
Call 408/277-0668 to place credit card orders or for customer service.
Hours:
M-F,9am-5pm pst.

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

POSTAGE & HANDLING
Prices include shipping. A $3.00 handling fee is required with all orders.

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

**** CALIFORNIA SALES TAX BY COUNTY**
6.25%: Sonoma; 6.5%: Fresno, Imperial, Inyo, Madera, Monterey, Orange, Riverside, Sacramento, San Benito, Santa Barbara, San Bernardino, and San Joaquin; 7%: Alameda, Contra Costa, Los Angeles, San Diego, San Mateo, San Francisco, Santa Clara, and Santa Cruz; 6%: Other counties.

(Letters, from page 5.)

losopher Macintosh software developer who runs a natural foods store and who lives at a growth center in the Montana Rockies, it has not revealed an easy path to financial solvency. Besides, the Mac listens better when I speak Pascal.

Forth may find its niche in embedded control devices, but its virtues as a general programming language—compactness, speed, interactivity, flexibility (anarchy)—have become old-fashioned indeed, and are frequently superseded by mainstream languages in more fully evolved development environments with vastly larger user communities.

I'm not trying to convince anyone, but I thought you'd be interested in the mind of a defector with close to nine years of using Forth. If nothing else, it's been fun—and that may be the only thing that counts.

Sincerely,
Laughing Water
1090 Helena Avenue
Helena, Montana 59601

```
SCR# 4
\ Menu words                                                5DEC90JWE

: UKEY  ( char -- CHAR)
  KEY 127 AND 96 OVER < OVER 123 < AND IF 95 AND THEN ;

: START ( --) [ ' ITINERARY >BODY ] LITERAL @ ?DUP
  IF    >BODY [ ' BALLOT^ >BODY 14 + ] LITERAL ! \ seed it
        BALLOT^FILL \ spread seed
        ITINERARY \ do it
  ELSE  HI." ITINERARY NOT VECTORED"  BEEP
  THEN ;

SCR# 5
\ Menu words                                                5DEC90JWE

: (.X")
R> ( IP@)  DUP @ ( cfa)  DUP ROT  2+  COUNT \ (cfa cfa string)
2DUP + >R ( IP!) \ IP -> IP+$
-ROT TUCK C@  DUP EMIT \  ( cfa len addr cfa char)
PAD COUNT 2DUP 3 + SWAP 1- C! + TUCK C! 1+ ! \ ( cfa len addr)
." -> " 1+ SWAP 1- ROT OLDX @ = \ ( string f)
 IF .X~ ON  HITYPE
 ELSE         TYPE
 THEN  CR ;

: .X"
COMPILE (.X")  [COMPILE] ' , ," ; IMMEDIATE


SCR# 6
\ Menu words                                                5DEC90JWE

: TELLER  ( char true|false)
  UKEY PAD COUNT + 2DUP C! >R \ ( char)
  PAD 2- BEGIN 3 + 2DUP C@ = UNTIL DUP R> = \ ( char pad+ f)
   IF DROP DUP 13 = .X~ @ AND
      IF DROP ." DEFAULT " CR FALSE
      ELSE TRUE
      THEN
   ELSE 1+ @ OLDX !  EMIT CR FALSE
   THEN ;

: TOUR  BALLOT^ >R ;

SCR# 7
\ Menu words                                                5DEC90JWE

: BALLOT  ( --)  >BALLOT
  BEGIN  PAD OFF .X~ OFF  (DISPLAY.BALLOT)
         CR TOUR  TELLER  (DISPLAY.RESTORE)
         IF DUP 27 = IF DROP BALLOT^FILL ELSE
            DUP 32 = IF DROP BALLOT>     ELSE
            DUP 63 = IF DROP (HELP)      ELSE
                31 = IF      EXIT        ELSE
                     HI." INVALID SELECTION" CR  BEEP
                     THEN THEN THEN THEN
         ELSE OLDX PERFORM
         THEN
  AGAIN ;

SCR# 8
\ Menu words                                                5DEC90JWE

VARIABLE HELP#

: HELP!  HELP# ! ;

: HELP CR CR HELP# @ ." See help #" . CR KEY DROP ;

' HELP IS (HELP)
```

```
SCR# 9
\ Menu words
: ITEM1 ." THIS IS ITEM 1 "  10000 0 DO NOOP LOOP ;
: ITEM2 ." THIS IS ITEM 2 "  10000 0 DO NOOP LOOP ;
: ITEM3 ." THIS IS ITEM 3 "  10000 0 DO NOOP LOOP ;
: ITEM4 ." THIS IS ITEM 4 "  10000 0 DO NOOP LOOP ;
: ITEM5 ." THIS IS ITEM 5 "  10000 0 DO NOOP LOOP ;
: ITEM6 ." THIS IS ITEM 6 "  10000 0 DO NOOP LOOP ;

DEFER SLAVE.1

: SLAVE2 BALLOT 1 HELP!
  HI." SLAVE2 MENU" CR
  .X" ITEM4 AITEM 4"
  .X" ITEM6 BITEM 6"
  .X" SLAVE.1 CSLAVE 1" ;


SCR# 10
\ Menu words

: SLAVE1  BALLOT 3 HELP!
  HI." SLAVE1 MENU" CR
  .X" ITEM5 AITEM 5"
  .X" SLAVE2 BSLAVE 2"
  .X" ITEM6 CITEM 6" ;


: SLAVE3  BALLOT
  HI." SUB MENU SELECTOR" CR
  .X" SLAVE1 ASLAVE 1"
  .X" SLAVE2 BSLAVE 2"  ;



SCR# 11
\ Menu words


: MASTER  BALLOT 5 HELP!
  HI." MASTER MENU" CR
  .X" SLAVE1 ASLAVE 1"
  .X" SLAVE2 BSLAVE 2"
  .X" ITEM1 CITEM 1"
  .X" ITEM2 DITEM 2"
  .X" ITEM3 EITEM 3"
  .X" SLAVE3 FSLAVE 3"  ;

' SLAVE1 IS SLAVE.1

' MASTER IS ITINERARY   START
```

Advertisers Index

Add and Delete Screens in PDE

Walter J. Rottenkolber
Visalia, California

After modifying my Kaypro II with larger disk drives, I was finally able to crank up my Laxen and Perry F83 files. I rapidly became disenchanted by the standard, arcane line editor, and longed for the simplicity of Laxen's full-screen editor as implemented on my fig-FORTH by Bob Bumalla. So it was with great hope that I greeted Frans Van Duinen's PDE (*Forth Dimensions* XI/2), since it was based on Laxen's.

The real advantage of PDE, I soon found, is the editor's seamless integration of screen-handling and debugging utilities. In small memory systems such as my Kaypro II (64K), screens enable you to deal with monster source files without having to scan through acres of text. But adding or deleting screens has been awkward and is, too often, avoided by using spaghetti-like screen-load lists. In PDE, ADD-SCR! and DEL-SCR! were to provide the luxury of doing so with a single keystroke.

However, I soon discovered they had a number of rough edges.

1. After adding or deleting a few screens, the *source* and *shadow* screens became de-synchronized, so that the shadow screens referred to the wrong source screens.

2. Adding or deleting multiple screens led to numbers of duplicate screens that caused no end of confusion.

3. As the file filled up, it was easy to add screens and find that the end screens had shifted off into oblivion.

In the accompanying code, I present an extended set of words to correct these problems. Screen Zero outlines the improvements.

Screens One through Nine contain the routines for the PDE version. YESNO? and CONFIRM in Screen One are from PDE and are here to help implement a standalone version. I added the CR in CONFIRM to help in screen text presentation. Many of the other words may seem the same as in PDE, but be warned: most have been redefined. ADD-SCR! and DEL-SCR! are assigned to a control or function key, so one keystroke from within the editor does it. You return to the same, numbered screen.

Screen 10 is a guide to implement a non-PDE version, using ADD-SCR and DEL-SCR. The primary difference is the need to supply the base screen number to these words. You will also need the variable SCR—which will hold that value for several of the routines—and the CONVEY words from the Forth-83 utilities.

Screen 11 is an example of how the WIPE routines clear a duplicate screen (format suggested by an Atari fig-FORTH), at least in the first two lines. Normally, the rest of the screen is blanked out. I've included my (?ERROR) and (WHERE) words, as their original incarnations would crash my system, a problem you might be facing. With these words, a LOAD error now results in a beep, followed by entering the editor, typing out the error message in the second window, displaying the screen, and positioning the cursor after the offending word.

The promise of PDE as an all-purpose development environment has been fulfilled.

Using the new ADD-SCR! and DEL-SCR from within PDE is simple. Before you begin, though, the variably SHDW must be set ON if using shadow screens, and OFF if using straight source screens. This is important for the correct routines to be chosen. The TOG-SHDW key on my system displays the setting on the status line, and makes switching easy. To do this, you need the variable OLD-SHDW and, also, follow the lead of Mode in the STUS word.

Pressing the appropriate function key from within the screen causes the screen number to be checked. No adds or deletes are allowed from the shadow screens. This forces you to look at the source screens before manipulating them.

Next, a request for the number of screens to add or delete appears in the second window. I did this as multiple individual adds or de-

(Continued on page 35.)

```
     0
0 .\  Add/Delete Screens Words
1
2 ####  Words to Add and Delete Screens from a File  ####
3
4
5       Derived from PDE202, but improved to:
6
7       1. Maintain synchronization between source
8          and shadow screens.
9       2. Check for adequate space to Add screens, and
10         enlarge file if necessary.
11      3. Blank duplicate screens and mark them as 'Spare'.
12
13      Words are designed to be used from within PDE,
14      but can be modified for pure command-line use.
15
```

```
     1
0 .\ YESNO? CONFIRM SHOW SHOW? TOG-SHOW OLD-SHOW
1 : YESNO?       ( addr len -- f )
2    TYPE  ." (Y/N)? "
3    BEGIN  0KEY  DUP  EMIT  0DF AND
4       DUP ASCII Y  = SWAP ASCII N  =  \ -)  f(=Y) f(=N)
5       OVER  OR  0=  WHILE  DROP BEEP  BS EMIT  REPEAT ;
6
7 : CONFIRM      ( addr len --  f  )
8     0W2  YESNO? CR 0= IF R> DROP THEN  ;
9
10   VARIABLE SHOW      SHOW ON
11   VARIABLE OLD-SHOW  OLD-SHOW OFF
12
13 : SHOW?  ( -- f ) SHOW @ ;
14   \ Fetches Shadow flag
15 : TOG-SHOW  ( -- )  SHOW DUP @ NOT SWAP ! ;
```

```
     2
0 .\  WIPE  WIPES  WIPE-SCR  WIPE-SCR+SHOW
1
2 : WIPE  ( scr# -- )
3    \ Blanks scr., marks it 'Spare', & puts '\S' in line #1.
4    BLOCK DUP )R C/SCR BLANK
5    " .\  Spare" R@ SWAP MOVE " \S" R> C/L +
6    SWAP MOVE UPDATE ;
7
8 : WIPES  ( from-scr# to-scr# -- )
9    2DUP ( IF SWAP THEN ?DO I WIPE LOOP ;
10
11 : WIPE-SCR  ( scr# -- )
12    DUP HOPPED @ + WIPES ;
13
14 : WIPE-SCR+SHOW  ( scr# -- )
15    DUP WIPE-SCR CAPACITY 2/ + WIPE-SCR ;
```

```
     3
.\  EMPTY?  MORE?  IS-SHOW?

: EMPTY?  ( scr# -- f )
   BLOCK [ C/L 2+ ] LITERAL +
   [ C/SCR C/L - 2- ] LITERAL -TRAILING NIP 0= ;

: MORE?  ( -- f )
   \ Checks if more file screens (TRUE) are needed.
   FALSE CAPACITY SHOW? IF 2/ THEN DUP HOPPED @ -
   ?DO DROP I EMPTY? NOT IF TRUE LEAVE THEN FALSE LOOP ;

: IS-SHOW?  ( -- f )
   \ Check if scr# a shadow, and set flag TRUE if so.
   SHOW? IF SCR @ CAPACITY 2/ 1- > ELSE FALSE THEN ;
```

```
     4
.\  CONVEY+  CONVEY-

: CONVEY+  ( scr1 scr2 -- ) HOPPED @ - CONVEY ; \ moving up

: CONVEY-  ( scr1 scr2 -- )                    \ moving down
   SWAP HOPPED @ - SWAP  CONVEY ;              \ Add neg #
```

```
     5
.\  MORE-SCR  MORE-SCR+SHOW  ENUF-SCR?

: MORE-SCR  ( -- )
   HOPPED @ MORE ;

: MORE-SCR+SHOW  ( -- )
   CAPACITY DUP 2/ DUP ROT 1- HOPPED @
   2* MORE CONVEY+ WIPE-SCR ;

: ENUF-SCR?  ( -- f )
   MORE? IF " Increase File Size to Insert Screens?"
   YESNO? CR IF SHOW? IF MORE-SCR+SHOW ELSE MORE-SCR THEN TRUE
   ELSE ." Add Screen Aborted!" CR FALSE THEN ELSE TRUE THEN ;
```

```
      6
0 .\  ((ADD-SCR+SHDW))  ((ADD-SCR)) (A-S+S) (A-S)
1
2 : ((ADD-SCR+SHDW))  ( scr# -- )
3     DUP [ SHADOW ] )SHADOW CAPACITY
4     DUP )R 1- CONVEY+ R) 2/ 1- CONVEY+ ;
5
6 : ((ADD-SCR))  ( scr# -- )
7    CAPACITY 1-  CONVEY+ ;
8
9 : (ADD-SCR+SHDW)  ( -- )
10    SCR @ DUP ((ADD-SCR+SHDW)) WIPE-SCR+SHDW ;
11
12 : (ADD-SCR)  ( -- )
13    SCR @ DUP ((ADD-SCR)) WIPE-SCR ;
14
15
```

```
      7
0 .\  ADD-SCR  ADD-SCR!
1
2 : ADD-SCR ( n -- )
3     ABS 1 MAX HOP MARK-UPD
4     ENUF-SCR? IF SHDW? IF (ADD-SCR+SHDW)
5     ELSE (ADD-SCR) THEN THEN ;
6
7
8 : ADD-SCR! ( -- )
9     @W2  IS-SHDW? IF ." Cannot Add Screens from Shadow" CR
10    ELSE  ." # Screens to Add: " IN@  CR @=
11    IF BEEP DROP ELSE DUP IF ADD-SCR
12    ELSE DROP THEN THEN THEN QUICK-INIT ;
13
14
15
```

```
      8
0 .\  OK#DEL? (DEL-SCR+SHDW)  (DEL-SCR)
1
2 : OK#DEL?  ( -- f )
3     HOPPED @ ABS SCR @ + 1- CAPACITY SHDW?
4     IF 2/ THEN ( DUP NOT
5     IF ." # Delete Screens Too High" CR THEN ;
6
7 : (DEL-SCR+SHDW)  ( -- )
8     SCR @ DUP CAPACITY DUP )R 2/ 1- CONVEY-
9     [ SHADOW ] )SHADOW  R@ 1- CONVEY-
10    R) 2/ WIPE-SCR+SHDW ;
11
12 : (DEL-SCR)  ( -- )
13    SCR @ CAPACITY DUP )R 1- CONVEY- R) WIPE-SCR ;
14
15
```

```
      9
.\  DEL-SCR  DEL-SCR!

: DEL-SCR  ( n -- )
   ABS 1 MAX NEGATE HOP " Delete screen" CONFIRM
   OK#DEL? IF SHDW? IF
   (DEL-SCR+SHDW) ELSE (DEL-SCR) THEN THEN ;

: DEL-SCR!  ( -- )
    @W2  IS-SHDW? IF ." Cannot Delete Screens from Shadow" CR
    ELSE  ." # Screens to Delete: " IN@ CR @=
    IF BEEP DROP ELSE DUP IF DEL-SCR
    ELSE DROP THEN THEN THEN QUICK-INIT ;
```

```
      10
.\  ADD-SCR  DEL-SCR

: ADD-SCR ( n scr# -- )
    SCR ! ABS 1 MAX HOP
    IS-SHDW? IF ." Cannot Add Screen from Shadow" CR
    ELSE ENUF-SCR? IF SHDW? IF (ADD-SCR+SHDW)
    ELSE (ADD-SCR) THEN THEN THEN ;

: DEL-SCR  ( n scr# -- )
   SCR ! ABS 1 MAX NEGATE HOP " Delete screen" CONFIRM
   IS-SHDW? IF ." Cannot Delete Screen from Shadow" CR
   ELSE OK#DEL? IF SHDW? IF
   (DEL-SCR+SHDW) ELSE (DEL-SCR) THEN THEN THEN ;
```

```
      11
.\  Spare
\S
: (ERROR)   ( adr len --   ) adr len scr# cpos )
     BLK @ IF BLK @ >IN @ WHERE  ELSE SPACE TYPE SPACE
     THEN SP0 @ SP! PRINTING OFF QUIT   ;

: (?ERROR)   ( adr len f -- )
    IF  (ERROR)  THEN  2DROP  ;          \ Assumes (ERROR) QUITs

' (?ERROR) IS ?ERROR

\  : !CPOS  ( cpos -- )  &CURS ! ;
: (WHERE)       ( adr len scr# cpos -- )
    BEEP !CPOS SET-SCR OTH-INIT @W2 TYPE QUICK-INIT (V) ;

' (WHERE)  IS  WHERE    \ Link into ABORT
```

(Continued from page 6.)

the responsibility and the authority. It is not true that quality is equal to the price you pay for it , the quality of services to FIG by volunteers is in fact much greater than anyone could afford to pay.
- Chapters, are they functioning, the future, help from the center?
- Conventions, Forth Day—expand to national?
- *Forth Dimensions* changes necessary? Changes in direction?
- Publications okay?
- FORML okay? New direction, education?
- Time for a new fig-Forth?
- Time for the sun to set on FIG?

Forth
- Standards
- Time is passing us by while we improve Forth, applications?

Actions and Explanations

Financial review for publication.

Two years ago the FIG Board of Directors took up the question of a public statement in *Forth Dimensions* about the financial situation of FIG. It was decided then that it was desired, but the final form needed to be reviewed later by the Board. To those of you who are interested, in the next issue of *Forth Dimensions* and in every July/August issue of *FD*, FIG will publish a review of the finances of FIG over the previous year.

Silence isn't secret.

Because there has not been a lot of discussion of FIG business in the past in *Forth Dimensions* doesn't mean that FIG has been keeping any secrets. If anything, we have wanted to keep from boring you. Many people read *FD* for its technical content. However, enough of you have wanted to know what is going on internally that I will make it my business to try to keep you informed. I hope I am starting in that direction with this column.

Dues increases.

For Volume XII of *FD*, FIG's income from membership dues and advertising in *FD* was 52% of income while the expense for the same volume was 54%. The difference in the past has always been covered by sales of books and reserve that had been accumulated over earlier years. In 1990, that translated to a *FD* cost of $35.21 per member per year. In 1991, we will attempt to reduce this, by consolidating paid jobs, to $32.20 per member per year; but an expected loss of members due to the membership increase to $40 in 1991–1992 will probably offset the cost reduction.

Is all of this helping or hurting? I can't see how a dues increase can ever help unless it is sufficient to exceed the losses anticipated due to the increased dues. This increase will help in the short run of two to three years, but we must eventually go one of two ways. We will be required to increase our membership to reduce the cost per membership or admit that we are an elite technical group and increase the membership dues radically to serve the few who are willing to support these type of services.

Enough have wanted to know what is going on internally that I will make it my business to keep you informed.

Annual billing.

The tradition of *FD* has been as a technical journal and we have looked at ourselves as a valuable reference for Forth programmers. How could anyone using Forth not want to have a complete reference to the current thinking of the Forth community, i.e., the current volume year? When new people find us, won't they want to know what we have been doing? Yes and no. Some have said they want to join and to continue to see what we are doing over the coming year. Our previous policy was that, when someone joined FIG in mid-year, we would send them back issues of the current volume.

The new policy, called "Annual Billing," will be similar to a magazine subscription. A person's membership will begin when they join and they will receive the next six issues of *FD*, regardless of the current volume year.

Publicity.

If we are ever to become more than an organization that is talking to itself, we must publicize the fact that we exist and are here to help with Forth issues. We have tried over the past, the most notable being the October 1980 issue of *BYTE* magazine. We have not had many publicity successes since then. We have attempted in several ways, but I think that even though I was involved in the attempts, I would characterize them as timid, inefficient, and ineffective. We must make a much bolder attempt to find our kindred non-member spirits. The focus of discussion of several of the FIG Business Group's meetings will be publicity and all related topics. I want people to look at FIG as the place to go for information about Forth and the Forth community, but they have to know we exist!

I am always available for comments.

—John Hall
415-535-1294 (voice)
or on GEnie as JDHALL

HARVARD SOFTWORKS

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 PCDOS, 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

Sixty-formatted Source Code

Hank Wilkinson
Greensboro, North Carolina

Glen Haydon's "Formatting Source Code" (*FD* X/6) interested me in implementing his word set. Left as an exercise to the reader was moving the ASCII text file from disk to memory. A simple loop allows using Forth error screens along with Glen's FILECOMPILE word set.

My implementation of fig-FORTH 6502 by W.F. Ragsdale is modified to run on Berkeley Softworks' GEOS for the Commodore 64. (GEOS is the desktop metaphor; i.e., programs are run by clicking their icons with a mouse pointer.) The word processor GeoWrite allows all the structures useful for good documentation. Fonts, tabs, emphasis, and margins all may be changed to format text in a pleasing manner. Drawings also may be inserted in documents.

GeoWrite on the '64 doesn't have the facility to save your document as a stripped ASCII file, with pictures, font, and emphasis structures removed. Nor does the '64 have enough memory to contain all of a large document.

A simple page-profiling routine revealed that a 3K buffer would easily hold the largest page I had written on my '64. Forth's method of screen handling gave me an algorithm for implementing text-loading facilities for my own system. Instead of screens, text pages would be retrieved and loaded.

The challenge on my '64 was incorporating text files seamlessly with the fig-FORTH screen system. By constructing a word to initialize the error screens and return a "safe" address, text files and error screens coexist.

Reality is, Forth's screen-loading mechanism can't handle it.

Refer to Figure One. Notice that Forth will not pass the BEGIN ... WHILE ... REPEAT without locating screen four (the first error message screen in fig-FORTH) at the FIRST screen buffer.

A fig-FORTH screen buffer consists of two bytes for the screen's number, 1024 bytes for the screen's data, and two guard bytes. +BUF points directly at the screen's data location, in the case of pages leaving useless the space for the screen number. The two bytes holding the screen's number are where Forth looks to determine whether a disk block should be written back to the disk.

Because their value cannot be guaranteed (unless EMPTY-BUFFERS is executed), the two bytes holding the screen number were recovered, thus ending the 2 - in GETBUF. ASCII text bytes in the recovered location will not appear as UPDATEd. (Random data may.)

When loading from text pages, the error screens are moved into the buffer area only once. Once loaded, the fig-FORTH disk routines "know" error screens four and five are located in memory. After the initial use of GETBUF, subsequent uses only take time performing arithmetic and branching.

If an error causes INTERPRET to fail, the terminal input buffer is left pointing where FILECOMPILE aimed it. After using FILECOMPILE a year or so ago, QUIT was altered to reset TIB from the user variable initialization table at fig-FORTH's origin. QUIT may be forced to read the start-up value for TIB by inserting 22 +ORIGIN @ TIB ! between RP! and CR. Since these alterations, jimmying around with TIB causes no odd behavior—even when an error is encountered.

GEOS on the '64 has facilities to ease the process of reading one page of a text file from its word processor. GETPAGE was written using these facilities, reading one page from the current disk drive into memory. GETPAGE is passed the address of a file's name, the page number, and the address of the buffer into which to put the page. After reading the disk, GETPAGE returns the GEOS "empty" flag (which is empty if zero, otherwise non-empty).

(GETPAGE handles GEOS disk errors, issues error messages on the terminal, and aborts if appropriate. The following word FILTERPAGE is implemented in 6502 code. Both words are highly specific to GEOS and GeoWrite. Neither are shown here, but the entire system—called *Brian*—may be downloaded from Q-Link. *Brian* is unique in that symbol files for the GEOS symbolic debugging are supplied. This allows users to execute Forth under the total control of a powerful debugging tool, and to examine its structure.)

After a text file's page is

Figure One. Initialization routine.

```
: GETBUF   ( -- buf_addr )
  BEGIN
    4 BLOCK FIRST 2 + = 0=
  WHILE EMPTY-BUFFERS REPEAT
  5 BLOCK +BUF DROP 2 -  ;
```

Figure Two. Loading a GeoWrite page to memory.

```
: PLOAD   ( page# -- )
  FN$ SWAP GETBUF GETPAGE   ( -- empty-flag )
    IF GETBUF FILTERPAGE GETBUF PCOMPILE
    ELSE ." Empty VLIR "  QUIT THEN   ;
```

Figure Three. Loading a range of pages, inclusive.

```
: TTHRU   ( begin# end# -- )
  FN$ 16 TYPE SPACE
    1+ SWAP DO
      I . I PLOAD
      ?TERMINAL IF LEAVE THEN
    LOOP   ;
```

Figure Four. Loading an entire file.

```
: TLOAD   ( -- )
  1 61 TTHRU   ;
```

moved into memory, execution is passed to FILTERPAGE. Its purpose is to remove the very structures producing good documentation, but which Forth doesn't understand. Written in machine language, FILTERPAGE is fast.

I had been told that the real way to load from text files is to make the text-formatting commands Forth no-ops. For example, a tab command's name would be an ASCII tab and would do nothing when encountered. This technique will not work easily without changing WORD. INTERPRET's parser—i.e., BL WORD—needs a space delimiter. (Unless you have HS-FORTH, which allows the user to define delimiters.) There are other difficulties as well.

With most word processors, you may write something like DUP. While a word processor may handle that, Forth cannot: Commands that change the fonts are *inside* the word DUP in the text file. Forth needs these structures stripped (or you'll end up rewriting the Fortran parser, if you add enough commands). FILTERPAGE simply replaces tabs, font commands, graphics, and so on with blanks. The user must remember not to use font changes inside words. (Though font and emphasis changes may be used immediately before or after a word.)

Once filtered, the address of the page is passed to PCOMPILE, which is Glen Haydon's FILECOMPILE. PLOAD (page load) contains the commands that load one page. Given a page number, it opens a file whose name is in a string variable FN$, puts the page where GETBUF told it to, closes the file, and passes execution to PCOMPILE. (Figure Two.)

When GETPAGE encounters an empty record, its flag causes PLOAD to QUIT to the command line. In GEOS terminology, an empty record does not exist. (A page with a lone page break is not empty.) With GeoWrite on the '64, the first empty page is the end of text.

PLOAD made it simple to build TTHRU (Figure Three). This word is useful for loading a range of pages. (You might have debugging words in pages seven through 11, for example.) TTHRU also reads the keyboard and stops loading pages after a key is pressed.

A GeoWrite text file on the '64 may have a maximum of 61 pages. TLOAD (text load) simply passes TTHRU the page numbers 1 and 61. Notice that TLOAD will try to load all 61 pages, but stops trying (via PLOADed QUIT) when the first empty page is encountered. (Figure Four.)

The dramatic impact of FILECOMPILE is that it turns inside-out the relationship between code and documentation. You are normally in a documenting mode (mood). With screens, I normally have a slight hesitation to document, just because of space considerations. (Note with screens, one comment line inserted may cause a word's definition to overflow a screen, aborting a LOAD.) FILECOMPILE allows documenting or coding with equal ease.

Another aspect FILECOMPILE makes clear is the idea that a word's definition must be completed on one screen. This may have evolved into "good programming practice," but reality is that Forth's screen-loading mechanism can't handle such circumstances. Using FILECOMPILE's switches, however, a word's definition may extend across multi-page boundaries. (You may be as verbose and graphic as desired.)

Hopefully, this discussion will encourage more programmers to try Glen Haydon's FILECOMPILE. (If I can do it on a '64 using fig-FORTH, maybe it's easy?) Implementation on a system of your choice makes it easier to teach Forth: Chances are, a beginner you know already uses your word processor.

If I can do it on a '64 using fig-FORTH, maybe it's easy?

CALL FOR PAPERS

for the thirteenth annual

FORML CONFERENCE

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

Following Thanksgiving
November 29— December 1, 1991

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

Theme: Simulation and Robotics

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

Presentations emphasizing virtual reality systems are being planned. Attendees are invited to enter a robot in a robotics contest where the robot solves a puzzle.

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

Completed papers are due November 1, 1991.

Registration and robotic contest information may be obtained by telephone request to the Forth Interest Group (408) 277-0668 or by writing to FORML, P.O. Box 8231, San Jose, CA 95155.

The Asilomar Conference Center combines excellent meeting and comfortable living accommodations with secluded forests on a Pacific Ocean beach. Registration includes use of conference facilities, deluxe rooms, all meals, and nightly wine and cheese parties.

This conference is sponsored by FORML, an activity of the Forth Interest Group. Information about membership in the Forth Interest Group may be obtained from the Forth Interest Group, P.O. Box 8231, San Jose, California 95155, telephone 408-277-0668.

(eForth, from page 17.)

user/implementor can examine or exercise any of the following features.

- 28 machine-dependent and 193 high-level words.
- Direct-threaded code.
- Separate name and code dictionaries.
- Deliberate use of user variables for ROM-ability.
- Vectored `?KEY`, `KEY`, and `EMIT`.
- File handling via the serial I/O interface (two computers are required).
- `CATCH-THROW` error handler.
- Singly indexed `FOR-NEXT`.
- Flexibility in memory mapping.

Implementing eForth on Other Processors

The eForth Model is designed for portability, and every effort is made to facilitate the porting process.

The eForth Model assumes that the CPU can address bytes in memory. All memory-accessing words use byte addresses. If your CPU cannot address bytes, you have to synthesize a byte address space from the cell addressing space and provide a mechanism to translate byte addresses into cell addresses the CPU can use. The eForth Model does enforce alignment to cell boundaries to facilitate byte-to-cell address translation.

The following procedure is suggested for porting it to a new CPU. After the generic eForth Model is ported to a target CPU, you might want to consider optimizing it to improve its performance.

- Determine the memory map in the target system. Set memory pointers in the EQU section properly to reflect the physical memory of the target; i.e., ROM, RAM, stacks, user area, code dictionary and name dictionary.
- Study the machine-dependent kernel. If you have access to an assembler for the target CPU, rewrite the machine-dependent words in the assembler language of the target; then insert the binary object produced by the assembler into the eForth source code using `DB` and `DW` statements. If you do not have an assembler, hand assemble the code words.
- If you have tools to exercise the assembly code words, try to debug them.
- Use MASM to assemble the eForth source files, producing binary object code.
- Move the object code into your target system via EPROMs or other means.
- Debug the target system.

The originators of eForth expect that a programmer familiar with the machine code of a target CPU should be able to port this model in one month. Porting eForth could even be a class project for college seniors or graduate students in an advanced assembly language class. The following list of processors are good candidates for eForth implementations (the Forth microprocessors will require a different porting strategy than that outlined above):

8080, Z80, 8051/31, 8096, 80960, 6502, 680x0, DSP16, DSP32, TMS320, 340x0, 56000, ADSP2100, 1750A, RTX2000, SC32, T425, T800, 88000, R3000, SPARC

News from the Forth RoundTable

Best of GEnie

Gary Smith (GARY-S on GEnie)
Little Rock, Arkansas

News from the GEnie Forth RoundTable—If you have been sleeping under a rock for the last couple of months, you may not be aware that the last BASIS document produced by the X3J14 ANS Technical Committee has, in fact, been decreed the review standard. (Only if you were sleeping under a boulder the last couple of years would you not have been aware that a ANS Standard Forth was being drafted. In that case, go back to sleep.) If you are interested in what the proposed standard contains, you can get your very own copy from the Forth Vendors' Group, c/o FORTH, Inc., 111 N. Sepulveda, Manhattan Beach, CA 90206 for the princely sum of $10.00.

If you conclude that this ends discussion of the proposed standard, you have been sleeping under the Rock of Gibraltar! There has been a torrent of discussion that seems to increase daily. The following exchanges concern primarily the ALSO ... ONLY debate, and the messages were gleaned from the week prior to the drafting of this article. This very brief look at but one area of discussion should make it abundantly clear there is a lot of pro and con still to be debated.

I must clarify something before proceeding to the exchanges. These messages were captured on GEnie, but obviously contain comments from the Usenet newsgroup comp.lang.forth as well as the RIME/pc-board Forth message base. That is because this discussion rages on all branches of Forthnet, of which the GEnie Forth RoundTable is but a part. It would be quite impossible to separate the contributing members and maintain any sense of coherency. You, too, can participate by responding on any branch. If you lack access to Usenet and your RIME BBS does not carry the xCFB Forth message base originating on The Grapevine, we cordially invite you to join the GEnie Forth RoundTable.

Now, on to the discussion regarding ALSO/ONLY and GET/SET-ORDER, with a side glance at strings and portability.

Could someone enlighten me as to the evil of ALSO ... ONLY?

Topic 2
ANS Forth Technical Committee

This topic is for discussion of the ANS FORTH Technical Committee and their recommendations/actions; and your reactions. Please note: points raised in this topic will be relayed to the committee.

Category 10, Topic 2
From: Brad Rodriguez
Doug Philips writes:

"...what I really, really want is a way to write Forth code that will run on more than one Forth and more than one processor. Portability is my main interest in this ANSI process."

Interesting. I've had relatively little trouble porting applications from processor to processor. It's moving from "standard" to "standard" that gives me headaches.

From: Doug Philips

> "Interesting. I've had relatively little trouble porting applications from processor to processor. It's moving from "standard" to "standard" that gives me headaches."

Okay, perhaps I should be a bit more explicit. I was thinking of PC versus Unix (Sun, Apollo, etc.). PC-Forths versus C-Unix Forths. Since there is no widespread *de facto* standard, switching from processors implies (or so I tried to assume) switching between standards.

From: Mitch Bradley
Subject: Portability problems

The differences between standards don't cause me too much trouble; the problems that bother me the most are coping with things that I have to deal with but which neither fig-FORTH, Forth-79, nor Forth-83 bothered to address at all. Specifically: files, floating point, 32-bit machines, memory allocation, strings, and error handling.

In the areas where previous standards differ, there is a small finite number of possibilities, and it is fairly easy to make a few redefinitions to cope with the two or three possibilities. The exception is vocabularies, where previous standards differ so widely that the only portable thing to do is to not use them!

In the areas that previous standards totally ignored, the range of variation across implementations is much greater, from "no solution at all" to "a set of words that works on this particular vendor's system but no other."

That is why I am enthusiastic about ANS Forth, whose extension wordsets address the issues that concern me most. Even if the ANS Forth extension wordsets are not 100% perfect, they are very, very useful. In my opinion—with the exception of strings—the extension wordsets are all good enough that it's probably counterproductive to tweak them any further.

From: Mitch Bradley

Figure One. Primitive tools for constructing run-time search orders.

```
WORDLIST  ( -- wid )
```
Creates a new wordlist and returns its wid ("wordlist id"). You can give it a name with CONSTANT, or you can use WORDLIST inside the CREATE portion of a CREATE ... DOES> word.

```
FORTH-WORDLIST  ( -- wid )
```
Wordlist id for the Forth wordlist.

```
GET-ORDER  ( -- widn .. wid1 n )
```
Returns the number of wordlists in the current search order, and their "wordlist ids." wid1 is the wordlist that is searched Forth.

```
SET-ORDER  ( widn .. wid1 n -- )
```
Sets the search order so that the n wordlists widn ... wid1 will be searched (wid1 first). If n is -1, sets the search order to the implementation-defined minimum search order, containing at least these words.

```
GET-CURRENT  ( -- wid )
```
Returns the wordlist id of the compilation wordlist (where new definitions are created).

```
SET-CURRENT  ( wid -- )
```
Sets the compilation wordlist to the wordlist "wid."

Subject: Processors and Standards

"Ok, perhaps I should be a bit more explicit. I was thinking of PC versus Unix (Sun, Apollo, etc). PC-Forths versus C-Unix Forths. Since there is no widespread *de facto* standard, switching from processors implies (or so I tried to assume) switching between standards."

This is an excellent point, but I would come to a slightly different conclusion. You don't need to switch between standards, as Forth-83 derivatives exist in both environments.

Rather, when you move to any workstation-class machine, or indeed any machine based on a 680x0 (e.g., Macintosh, Atari ST, Amiga), you start getting into uncharted territory, as far as existing standards are concerned. In these kinds of environments:

- Source code in blocks makes little sense.
- 32-bit addressing is the name of the game.
- The existence of the operating system cannot be ignored (leading to problems with the traditional Forth memory model, and also providing market requirements for access to OS features).
- Hardware alignment restrictions sometimes apply.
- C is a viable, useable, and ubiquitous development environment, and Forth has to be competitive with it to succeed.

The Forth-83 Standard does not address these issues. Successful Forth implementations in these environments have addressed these issues, but without the guidance of a standard there has been great divergence.

In the last two or three years, even the PC world has begun to face the same issues (because of Turbo-C, '386 chips, cheap megabit RAM chips, and Windows 3).

From: Chip Salzenberg

Ray Duncan writes:

"We strongly support the proposals to delete ONLY/ALSO (I proposed this myself some time back, but my proposal was voted down)."

Could someone enlighten me as to the evil of ONLY/ALSO?

From: Ray Duncan

Chip Salzenberg writes:

"Could someone enlighten me as to the evil of ALSO/ONLY?"

First of all, it was never a part of the Forth-83 Standard. It was appended to the standard as an experimental word set, as a political ploy to keep Bill Ragsdale (one of the founders of FIG) happy.

Second of all, ALSO/ONLY was apparently originally conceived of as a "Forth-like" (i.e., stack-like) way to control search order, but the realization of the concept was brain damaged. To wit: there are ways to push things onto the search order stack but no way to pop them off; no way to interrogate what is currently in the search order stack; no way to rearrange the search order stack except by dumping the whole thing and starting over; and so on.

Third of all, ALSO/ONLY is completely incompatible with at least two of the commercial Forth systems with the largest installed bases (Forth, Inc. and LMI). It seems sort of silly to me to put something into the ANSI Forth draft standard—which is supposed to be founded on consensus and common practice—which will totally break the code of thousands of serious Forth programmers and *cannot* be mechanically edited/translated from old to new.

Topic 14
ANS TC Magnet for Vocabularies

ANS—Vocabularies and discussions about ANS Forth vocabularies. Magnet: John Stevenson

From: Ray Duncan
Subj: ALSO/ONLY

Craig Treleaven writes:

"What's wrong with ALSO/ONLY ... a Forth vocabulary with 1500+ words is not my idea of easy to use."

It's not my idea of easy to use, either. I have nothing against vocabularies, just the brain-damaged ALSO/ONLY scheme.

From: Craig Treleaven
Subj: ALSO/ONLY

Ray Duncan writes:

"I have nothing against vocabularies, just the brain-damaged ALSO/ONLY scheme."

Sorry, I still don't understand what is wrong with ALSO/ONLY? It seemed to me to be a nice way to control the search order. Words could be grouped into as many vocabularies as necessary, and the search

order set to only search the relevant ones.

From: Mitch Bradley
Subject: Brain damage

> "SET-ORDER can be done portably in ANS Forth only if everyone implements ***all*** the extension word sets. In our own case, I can definitely promise you that LMI will ***not implement the*** extension words it considers brain damaged, which definitely includes ALSO and ONLY."

GET/SET-ORDER was invented in order to resolve the conflict between the proponents of run-time search order specification and those who consider ALSO/ONLY to be brain damaged. (ALSO/ONLY indeed has some serious technical flaws; nevertheless, it is useful.)

I believe that the design of GET/SET-ORDER addresses the problems with ALSO/ONLY, while being simpler than ALSO/ONLY. Indeed, GET/SET-ORDER was originally proposed as a "fix" for the most fundamental flaw of ALSO/ONLY. Then I realized it is sufficiently powerful that ALSO/ONLY can be easily implemented in terms of GET/SET-ORDER. John Hayes and I have figured out how to implement a few other popular search order schemes in terms of GET/SET-ORDER. From what Martin Tracy has told me of LMI's search-order mechanism, I believe it can be expressed in terms of GET/SET-ORDER as well.

The Search Order "base" wordset now contains GET/SET-ORDER and a few related words (GET/SET-CURRENT, WORDLIST, and FORTH-WORDLIST). ALSO/ONLY has been banished to the Search Order Extension Wordset.

I would be interested to

learn of other wordsets that LMI considers to be brain-damaged, and why. (Don't bother mentioning the strings wordset; everybody in the world seems to have different and mutually-incompatible ideas about what should be in that particular wordset, so I expect that it will remain brain-damaged.)

From: Mitch Bradley
Subject: Problems with ALSO/ONLY

Disclaimer: I don't dislike ALSO/ONLY; I use them every day. But they do have some problems:

1. There is no standard way to save the existing search order, set the search order to a particular value, and later restore the search order.
2. In the common implementations, it is easy to get in a situation where the same vocabulary is searched twice. This is an implementation problem, not a specification problem, and it's only bad effect is slower compilation, but some people criticize it anyway.
3. A program has no standard way of knowing how many vocabularies it can add to the search order before the search order data structure overflows and the system crashes.
4. Many people think that the behavior of the search order as a "funny stack" (where the execution of a vocabulary *replaces* the top of the stack) is screwy.
5. There is no portable way of testing what is in the search order.
6. There is no portable way of removing a particular vocabulary from the search order.

Basically, with ALSO/ONLY as described in Forth-83 (as an experimental wordset), you could set the search order to a particular value, but that is all. Anything else you wanted to do required knowledge of exactly how it was implemented.

I would be interested to hear of other problems that I may have missed.

From: Mitch Bradley
Subject: Search order user interfaces

I hereby propose a "search order user interface" contest.

Background:

Because there is a lot of Forth community support for ALSO/ONLY, it has remained in ANS Forth. Because there is a fair amount of opposition to it, it has been banished to the Search Order Extension wordset (i.e. the "extension" portion of the optional Search Order wordset), so it is sort of "doubly optional."

The base portion of the optional Search Order wordset contains a set of primitive tools for constructing run-time search orders. Those tools are given in Figure One.

From these primitives, various search order "user interfaces" may be readily constructed, including ALSO/ONLY, and the fig-FORTH, Forth-79, and polyFORTH vocabulary mechanisms.

Contest:

Design your own "user interface wordset" (as in ALSO/ONLY) for specifying search orders. Hopefully, it should be buildable on top of GET/SET-ORDER (if not, tell us the crucial deficiency of GET/SET-ORDER). Tell us what is good about it. Convince us that it is not "brain damaged."

The prize:

There is no prize, other than the chance that other people like it and will think you are a hotshot.

From: Ray Duncan

Thank you, Mitch, for exactly summarizing my objections to ALSO/ONLY.

In LMI Forth systems, FIND uses a three-level search order:

```
CONTEXT→
CURRENT→
FORTH
```

However, we also have a building block word (find) that is called by FIND and can search an arbitrary string of vocabulary threads—from one to many. So it is very easy to implement other schemes for search order control in LMI Forths and, in fact, the ALSO/ONLY scheme can be layered on top of our systems in one or two screens of code.

From: Mitch Bradley
Subject: Re-inventing the wheel

> "I'm bothered a little by the rationale for the use of WORDSET in lieu of VOCABULARY; what I heard of of the reasons given for the change there simply didn't seem to be enough for my liking..."

If the word VOCABULARY exists in the standard, then the upgrade path for several important commercial implementations is made considerably more difficult. In particular, polyFORTH, LMI Forth, MacForth, and MVP-FORTH all have the word VOCABULARY in different forms. Any definition of VOCABULARY that you choose will break at least three of those four systems.

This seems like a strong argument to me. Those systems represent many thousands of Forth programmers between them.

> "Not to mention the forms of signed division (We now need the code for both forms in the dictionary instead of one- or so I understand it to be)..."

Remember, the code *doesn't have to be in the dictionary.* It just has to be available. It could even be on a paper listing that the user could type in to define, for instance, FM/MOD in terms of SM/MOD or UM/MOD.

> "...they both give incorrect answers in the third quadrant (both numbers negative...) the MOD part of /MOD for either algorithm is ***negative when*** it should be positive (which is mathematically wrong...)."

Is it? Knuth (volume one, page 38, says that if $y < 0$, then $0 >= x \bmod y > y$.

Also see Robert Berkey's excellent discussion of division in the proceedings of the 1982 FORML conference.

Closing Comment

There is going to be a dpANS Forth. You are running out of opportunity and time to help create the model. I might add, I have deliberately scheduled some online conference guests with diverse views of what the final product should look like. The point is this: If you do not partake of the opportunity to impact the shape of "our" language it will *not* be because you were denied access to the process.

(Continued from page 11.)

```
( Memory release )
 : F-L    ( ad -->    :  free list cells )
  DUP NIL? IF DROP ELSE DUP LST?
    IF DUP CAR RECURSE DUP CDR RECURSE FWMC
    ELSE FWMC THEN THEN ;
 : OCC   ( ad1 ad2 --> f : ad1 -  occur in list ad2 ? )
  OVER OVER = IF DROP DROP TRUE ELSE DUP LST?
  IF OVER OVER CAR RECURSE
    IF DROP DROP TRUE ELSE CDR RECURSE THEN
  ELSE DROP DROP FALSE THEN THEN ;
 : SF-L   ( ad1 ad2 -->   :  free cells of ad1 not OCC in ad2 )
  OVER NIL? IF DROP DROP EXIT THEN
  OVER OVER OCC IF DROP DROP EXIT THEN
  OVER LST? IF OVER CAR OVER RECURSE
               OVER CDR SWAP RECURSE FWMC
            ELSE DROP FWMC THEN ;
```

(Continued from page 23.)

letes are unbearably tedious, even with a RAM disk. An invalid number, a zero, or a CR only, aborts the procedure.

When deleting, a confirmation request gives you a second chance to abort before anything is actually deleted.

Error Checking

When adding screens, MORE? checks the end of the source-screen set to ensure that enough empty screens are there. A confirmatory query comes up if more screens are needed. I did this primarily for floppy-disk systems, to prevent an 'out of room' error. The routine is a bit simple-minded. If four empty screens are present but five are needed, it will increase capacity by five, not one. For shadow screens, the number is doubled and the shadow screens are shifted up so they remain in sync with the source screens. When using shadow screens, I find it safer to start with the capacity as an even number and to keep it that way.

When deleting, a calculation by OK#DEL? checks that you don't try to delete more screens than exist in the file. After that, the screens are shifted and the duplicate screens are wiped clean. You will find that most of the operation is automatic and undemanding.

After tweaking the original code to my liking, the promise of PDE as an all-purpose development environment has been fulfilled. It's like a Forth-screen version of *Codeview* in just 34K bytes. I hope these screen-handling words only add to the pleasure.

Walter J. Rottenkolber says that Forth provides the same close-to-the-silicon feel as assembler, but without the pain. Early on, he experimented with fig-Forth and other languages, but gravitated toward assembler until he was dazzled by Bill Kibler's NC-4000-based computer.

reSource Listings

Please send updates, corrections, additional listings, and suggestions to the Editor.

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
Fax: 408-286-8988

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

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

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

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

George Shaw
Shaw Laboratories
P.O. Box 3471
Hayward, CA 94540-3471
415-276-5953

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

David C. Petty
Digitel
125 Cambridge Park Dr.
Cambridge, MA 02140-2311

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

Forth Instruction

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

On-Line Resources

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

GEnie
For information,
call 800-638-9636

- Forth RoundTable *(ForthNet*)*
  Call GEnie local node, then
  type M710 or FORTH
  SysOps:
  Dennis Ruffer (D.RUFFER),
  Scott Squires (S.W.SQUIRES),
  Leonard Morgenstern (NMORGENSTERN),
  Gary Smith (GARY-S)

- MACH2 RoundTable
  Type M450 or MACH2
  Palo Alto Shipping Company
  SysOp:
  Waymen Askey (D.MILEY)

BIX (ByteNet)
For information,
call 800-227-2983

- Forth Conference
  Access BIX via TymNet, then type j forth
  Type FORTH at the : prompt
  SysOp:
  Phil Wasson (PWASSON)

- LMI Conference
  Type LMI at the : prompt
  LMI products
  Host:
  Ray Duncan (RDUNCAN)

CompuServe
For information,
call 800-848-8990

- Creative Solutions Conf.
  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 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)

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

- British Columbia Forth Board
  604-434-5886
  SysOp: Jack Brown

- Grapevine
  501-753-8121 to register
  501-753-6859
  StarLink node 9858
  SysOp: Jim Wenzel

- 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

- Druma 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*)*
  United Kingdom
  0905 754157
  SysOp: Jon Brooks

- Sky Port *(ForthNet*)*
  United Kingdom
  44-1-294-1006
  SysOp: Andy Brimson

- SweFIG
  Per Alm Sweden
  46-8-71-35751

- NEXUS Servicios de Informacion, S. L.
  Travesera de 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

**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.*

This list was accurate as of March 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!glsrk!gars

FIG Chapters

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

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

U.S.A.

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

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

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

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

  North Bay Chapter
  2nd Sat.
  12 noon tutorial, 1 p.m. Forth
  2055 Center St., Berkeley
  Leonard 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
  John Hall
  (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) 472-7782

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

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

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

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

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

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

- **NEW JERSEY**
  New Jersey Chapter
  Rutgers Univ., Piscataway
  Nicholas 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 Lab
  AGS dept.,
  bldg. 911, lab rm. A-202
  Irving Montanez
  (516) 282-2540

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

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

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

- **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 (HAL-PC)
1200 Post Oak Rd.
(Galleria area)
Russell Harris
(713) 461-1618

- **VERMONT**
  Vermont Chapter
  Vergennes
  3rd Mon., 7:30 p.m.
  Vergennes Union High School
  RM 210, Monkton Rd.
  Hal Clark (802) 453-4442

- **VIRGINIA**
  First Forth of Hampton Roads
  William Edmonds
  (804) 898-4099

  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-8044 Unterschleissheim
  (49) (89) 317 3784
  e-mail uucp:
  secretary@forthev.UUCP
  Internet:
  secretary@Admin.FORTH-eV.de

- **HOLLAND**
  Holland Chapter
  Vic Van de Zande
  Finmark 7
  3831 JE Leusden

- **ITALY**
  FIG Italia
  Marco Tausel
  Via Gerolamo Forni 48
  20161 Milano

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

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

- **SWEDEN**
  SweFIG
  Per Alm
  46/8-929631

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

SPECIAL GROUPS

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

CALL FOR PAPERS

1991 CHINA FORML CONFERENCE

ON INDUSTRIAL AUTOMATION

Shanghai Jiao Tong University, Shanghai, China
August 9-10, 1991

Papers are invited that address issues related to factory automation and industrial applications of Forth. Since the Forth experts in China are still few and far in between, tutorial topics on Forth, its applications and its implementations on microprocessors and microcontrollers are also welcome.

Please send abstracts of approximately 100 words by May 1, 1991 to FORML, P.O. Box 8231, San Jose, CA 95155. Completed papers are due July 5, 1991.

Chinese Ancient Capitals Tour for Conference participants and guests

This special tour offers a unique itinerary through many of the ancient Chinese capitals in the Yellow River basin, off the oft-trodden tourist paths. Visiting Beijing, Zhengzhou, Dengfing, Luoyang and Xian presents a comprehensive view in very high contrast of the Chinese people, landscape, architecture, culture and history. Other highlights include the Shaolin Temple, ruins of Sha Capital, Zhongyue Temple, Yuanguan Observatory, the Brick Songyue Tower, etc. Conference and tour cost is about $2,000 per person, from/to San Francisco.

Conference registration information may be obtained by telephone from the Forth Interest Group business office (408) 277-0668 or write to FORML, P.O. Box 8231, San Jose, CA 95155, or to Prof. Zhou Xu, Computer Center, Shanghai Jiao Tong University, Shanghai, China, 200030. Inquiries about the conference programme and the special Capitals Tour should be directed to the Programme Coordinators, Dr. C. H. Ting, (415) 570-6667 (w) or (415) 571-7639 (h).

Shanghai Jiao Tong University is the oldest technology university in China with special emphasis on communication, electronics, and computer technology. It hosted the previous China FORML Conferences in 1984 and 1986, and has been one of the focal points of Forth activity in China. Shanghai is the largest metropolitan area in the world at the mouth of gaint Yangtze River, surrounded by many scenic cities like Hangzhou, Zhenjiang, Suzchou, Yangzhou, Wuxi, and Nanjing.

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

Second Class
Postage Paid at
San Jose, CA