FORTH DIMENSIONS

JUNE/JULY 1978 VOLUME 1 NO. 1

EDITORIAL:
WHAT IS THE FORTH INTEREST GROUP?

The Forth Interest Group, which developed in the fertile ground of the computer clubs of the San Francisco Bay Area, grew in a few months from nothing to where we are now getting several letters a day from all over the country. With this increasing public interest we need to let people know what we are doing and why, what we would like to see happen, how others can be involved, and what we can and cannot do.

We are involved because we believe that this language can have a major effect on the usefulness of computers, especially small computers, and we want to see it put to the test. Increasingly software is becoming the critical, limiting factor in the computer industry. Large software projects are especially difficult to develop and modify. Few are happy with prevailing operating systems, which are huge, hard to understand, incompatible with each other, and without unity of design.

The Forth language is its own operating system and text editor. It is interactive, extensible (including user-defined data types), structured, and recursive. Code is so compact that the entire system (mostly written in Forth) usually fits in 6K bytes, running stand-alone with no other software required, or as a task in a conventional operating system. One person can understand the entire Forth system, change any part of it, or even write a new version from scratch. Run-time efficiencies are as little as 30% slower than straight machine code, and even less if the system's built-in assembler is used. When the assembler is not used, programs can be almost completely transportable between machines. Any large Forth program is really a special-purpose, application-oriented language, greatly facilitating maintenance and modification. We don't yet have conclusive data, but typical program development times and costs seem to be a fraction of those required by Fortran or assembly. Forth is especially useful for real-time, control-type applications, for large projects, and for small machines.

The problem is availability. Users have shown an ease of learning after they have a system available. The Forth characteristics of postfix notation, structured conditionals, and data stacks are best understood by use. To encourage Forth programmers, we need readily available systems even of modest performance. We hope that three levels will be available:

1. Demonstration - free (or under $20.) introductory version without file structure which compiles and executes from keyboard input.

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

2. Personal - low cost ($10. to $100.) with RAM or tape based files.

3. Professional - Commercial products for lab or industrial use and software development. ($1000. to $2500.)

Today the serious personal computer user holds the key to wider availability of the language. These users - generally engineers, businessmen, programmers - combine professional competence and commitment with the freedom to try new methods which may require a lot of time and tinkering with no definite guarantee of payoff. Practically everyone involved with the Forth Interest Group has both a personal and a professional interest in computers.

The Forth Interest Group is non-profit and non-commercial. We aren't associated with any vendor, no one is making money from it, and we are all busy with other work. We are an information clearinghouse and want to encourage distribution of all three of the previously mentioned levels of Forth. We do not have a Forth system for distribution at this time, and we don't want to get into the software or mail-order business because this is best left to companies or individuals committed to that goal. Naturally our critical issue is how to keep going over the long haul with volunteer energy. We need cost-effective means of information exchange.

At present we are writing for professional media, putting out this simple newsletter, and holding occasional meetings in the Bay Area. Also, we are developing a major technical and implementation manual, to be published in a journal form as four installments, available by subscription. While we cannot answer all of the mail individually, we certainly read it all, to answer it in the newsletter. While we cannot fill orders for software or literature, we will try and point you to where it is available. We welcome your input of information or suggestions, how you could help, what you would like to see happen, and where we should go from here.

Dave Bengel - Dave Boulton - Kim Harris - John James
Tom Olsen - Bill Ragsdale - Dave Wyland

EVOLUTION OF A F.I.G. FORTH FREAK

By Tom Olsen

I have been actively involved in the personal computing movement since early in 1974 when I shelled out $120. for an 8008 chip. Since that time my hardware and software have evolved into a very powerful and useful system, of which FORTH is a principal component. The system consists of an LSI-11, 28K of memory, 2 Diablo disks, an LA30 DECWRITER, a Diablo HYTYPE-I printer, a VDM-1 display, and dual floppy disk drives. Obtaining an operating system which would effectively utilize all of this hardware initially appeared to be much too expensive for an individual to buy, and far too complex to write from scratch. This attitude changed when early in 1976 I read a technical manual describing the internal organization of a relatively unknown "language" called FORTH. Here was a programming system which included not only an editor, assembler, and file management system, but the inherent capability to be rapidly expanded to perform any computer function I could define. The best part was the fact that the central core of this programming system was relatively small and would easily fit into 3K of memory. The large majority of the system programming could be done in terms of high level functions which I would have the freedom to define.

After about three months of late nights and pulling my hair out, I finally had a stand alone FORTH system which I could bootstrap and then use to load application vocabularies from disk. Once the basic implementation was fully debugged my general throughput of useful application software increased to a level I never would have thought possible. I can't over-emphasize the satisfaction associated with implementing the language from scratch. An added benefit of this approach is the flexibility derived by having a 100% understanding of ALL of the code your machine is executing.

Today I have application vocabularies which can do everything from playing a BACH minuet on a computer controlled synthesizer to generating, sorting, and printing the FORTH INTEREST GROUP mailing list. It is my hope that with the continued growth of the FORTH INTEREST GROUP and the establishment of some syntactical standards, widespread exchange of applications vocabularies will greatly enhance the computing power of all users of FORTH-like languages.

USING FORTH FOR TRADEOFFS BETWEEN HARDWARE/FIRMWARE/SOFTWARE

By Dave Wyland

FORTH provides a unique capability for changing the tradeoffs between software, firmware, and hardware. This capability derives from the method of nested definition of operators in FORTH.

Firmware (i.e. microprogramming) can add new instructions to the instruction set of an existing machine. New hardware designs can also add instructions to an existing set, with the Z80 upgrade of the 8080 serving as a good example. These new instructions could significantly improve the throughput of a system in many cases: however these new instructions cannot be used with existing software without rewriting the software. This is because current software methods such as assembler language, BASIC interpreters, FORTRAN compilers, etc., create software programs as one list of instructions. To add improved instructions to a program generated by a FORTRAN compiler involves rewriting the compiler to efficiently use the new instructions and then recompiling the program. This is not a trivial task since decisions must typically be made as to when to use a new instruction. This is why each new machine requires a new, rewritten FORTRAN compiler. The fact that the job has been done many times before is not very comforting.

With FORTH, however, operations are built-up of nested definitions with a common functional interface between operations. If a new instruction has been added to the computer's instruction set, it can be added to the FORTH system by changing a small number of definitions, in the typical case. The method can be quite straightforward. Each new instruction does an operation which would have required several instructions in the previous case. By identifying those FORTH definitions which could benefit from the new instructions and recoding them to include it, all software which uses the modified definitions is immediately improved. Also, very few definitions will have to be modified: only those elementry definitions with a high frequency of use need be modified to achieve throughput increase near the maximum possible.

Since the FORTH definitions are nested, the throughput gain of new instructions can be achieved by modifying a small amount of code, and the remaining structures remain unchanged. All existing application programs, translators, etc. are immediately improved without change!

Note that this process is reversable. Programs created for an existing system which has an extensive instruction set can be run on a simpler machine by converting the appropriate code based definitions to nested (colon) definitions. The process is also incremental: new instructions can be added one-at-a-time as desired.

FORTH LEARNS GERMAN

By John James

Forth now understands German, thanks to the following redefinitions of the operations in the Decus (Caltech) Forth manual.

Many of the operations were mathematical symbols, and of course they did not have to be translated. Of the rest, the control operations (IF, THEN, ELSE, BEGIN, END, DO, LOOP, +LOOP) are special, because they are "immediate operations"; that is, they are executed at compile time. Just redefining their names would not work, because they would try to execute right in the definitions. So their original definitions were copied, but with the German names.

Program development time for bilingual capability, two hours. Memory required, 600 bytes. Effect on run-time execution speed, zero.

```
BLOCK 30
 1 ( GERMAN.  JJ, 6/19/78)
 2 : ABWERFEN DROP ;     : UBER OVER ;
 3 : VERT SWAP ;    : SPAREN SAVE ;    : UNSPAREN UNSAVE ;
 4 : UNBEDINGT ABS ;     : UND AND ;
 5 : HOCHST MAX ;     : MINDEST MIN ;     : REST MOD ;
 6 : ODER OR ;    : /REST /MOD ;    : 0SETZEN 0SET ;
 7 : 1SETZEN 1SET ;    : HIER HERE ;    : VERGESSEN FORGET ;
 8 : SCHLUSSEL CODE ;    : BESTANDIG CONSTANT ;
 9 : GANZE INTEGER ; : ORDNUNG ARRAY ; : IORDNUNG IARRAY ;
10 : SETZEN SET ;
11 : AUFS-LAUFENDE UPDATE ; : AUSRADIEREN ERASE-CORE ;
12 : LADEN LOAD ;    : ZURUCK CR ;
13 : SCHREIBEN TYPE ;     : BASIS BASE ;
14
15
16 ( THE SAME: DUP, MINUS, =L, BLOCK, ;S, F)    ;S
```

The control operation definitions (OB, DANN, SONST, BEGINNEN, ENDEN, TUN, SCHLINGE, +SCHLINGE) are not shown here; they are all short (one line), and exact copies of the English operations. (Incidently this particular vocabulary is a rough draft; we have not seen the results of the International Forth Standards Team, which is currently at work.) The word GERMAN is defined on the load screen, so that the Forth user can call in the German vocabulary when desired. The following session shows the entry and execution of a fibonacci

sequence program in German (until 16-bit overflow).

```
FORTH LOAD
OK
GERMAN
OK
: PRUFUNG 0 1 30 0 TUN VERT DUP . UBER + SCHLINGE ABWERFEN ABWERFEN ;
OK
PRUFUNG
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 1771
 28657 -19168 9489 -9679 -190 -9869 -10059
```

On a (slightly) more speculative note, why not extend this scheme to a computer assisted international language for computer conferences, electronic mail, and international data-base utilities? Clearly natural language is too free, and computer languages like BNF are too restrictive, to be feasible. But a hybrid, a vocabulary of several hundred unambiguous words (each used in one sense only), and perhaps some computer-oriented syntactical markers, should be enough for useful dialog within a particular interest area. If it works for two languages it should work as well for any number. The final test - whether international teams could collaborate, after minimal training - would take a few weeks programming at most, after the vocabularies and terminal interfaces had been determined.

FORTH MAILBAG

By Dave Bengel

The Forth Interest Group developed from several people in the San Francisco- San Jose area who have been working on Forth for the last year and a half. Until about six months ago most of these users were unaware of each other. Until the publication of the article by John James in Dr. Dobb's Journal (May 1978), about 80 percent of the group was from the Homebrew Computer Club - whose publication should also be watched for news concerning the F.I.G.

We now have nearly 200 names on the mailing list, and are receiving about six letters a day. The writers' main question is how to get a version for their machine.

We don't yet have a detailed description of the versions of Forth now available, nor is there a standard form of the language available for various CPUs. Intense work on implementations is now underway; e.g. the Forth Interest Group implementation workshop. We will keep you informed as more documentation and systems become available.

Your answers to the questionaire in this newsletter will help us keep a mailing list for interest-specific applications, documentation, CPU versions, etc. We appreciate any information you can send us, particularly about Forth versions or variants which are running or being developed, or any software we can publish. We need your contributions to this newsletter (which we hope will grow into a journal).

FIG LEAVES

IMPLEMENTATION WORKSHOP

1771

The Forth Interest Group will hold an implementation workshop, starting sometime in July. The purpose is to create a uniform set of implementations for common micros. The assembly listings which result will be available through F.I.G. to those who wish to distribute specific versions.

This will be a small group, no more than ten, with only one person for each machine. There will be approximately four all-day sessions, over six weeks. Implementers must have access to their target machine, with an assembler and editor; floppy or tape is not required. The meetings will be to share notes and specific guidance on implementation details. At the end of the workshop we should have uniform Forth versions for all the machines.

We now have implementers scheduled for 8080, 6502, PACE, and LSI-11. We need implementers for 6800, Z80, 1802, F8, System 3 (32), 5100, etc. We also need a project librarian with Forth experience.

If you are interested write to FORTH IMPLEMENTATION PROJECT, 20956 Corsair Blvd., Hayward, Ca. 94545.

CONTRIBUTED MATERIAL

Forth Interest Group needs the following material:

(1) Manuals available for distribution. We can purchase copies and distribute, or print from your authorized original.

(2) Name and address of Forth implementations for inclusion in our publications. Include computer requirements, documentation, and cost.

(3) Technical material for the forthcoming journal. Both expositions on internal features of Forth and application programs are needed.

(4) Users who may be referenced for local demonstration to newcomers, on a regional basis. Indicate interest area (i.e. personal computing, educational, scientific, industrial, etc.).

(5) Letters for publication in this newsletter.

(PAGE 8 AND 9 BLANK)