6 Interactive Embedded Software Development
Garth Wilson
You can achieve complete interactivity for developing Forth software on a target computer. The method requires no hardware or software except the host computer, the target, the text editor, and the metacompiler—no communications software, no emulator. And the metacompiler will only be used to compile code for ROM after it is developed and working.

13 Quicksort and Swords Redux
Wil Baden
After more than 30 years, C.A.R. Hoare’s quicksort is still the fastest general algorithm for sorting in place on a single processor. Following up on earlier work, this FORML-award-winning author presents his implementation in ANS Forth (except for NOT), and shows how to implement in your Forth three words new in ANS Forth. And he narrowly avoids tempting fate…

21 Understanding F83 Vocabulary Usage
Byron Nilsen
Vocabularies are a unique feature of the Forth language. They provide a means to isolate groups of definitions so as to avoid naming conflicts, to preserve order in large applications programs, to reduce compilation time, and to achieve other worthy ends. We can ignore them at first, but eventually we will want to access words defined in other vocabularies, and when this need arises some confusion may follow. The discussion is for novices but, as another author points out in this issue, implementors have a different lesson to learn from it.

24 Generation and Application of Random Numbers
Dr. Everett F. Carter, Jr.
The world’s computers generate ten billion random numbers per second. Many subtle problems can occur, and various compromises have to be made in order to even pretend to generate random numbers with a computer. This article explores the generation of random numbers and some important applications that use such numbers.

37 Pygtools—A Library of Reusable Utilities
L. Greg Lisle
One often hears of the need for reusable libraries of Forth tools. The most common call is for the ability to access C libraries from Forth. An alternative is a library structure designed for use with Forth, where the programmer can do whatever he or she wishes. In addition to that, this package demonstrates the flexibility of a screen-based disk structure. Written in Pygmy Forth, the concepts should apply to any Forth that provides block access.
6 Zero-Overhead Forth Interrupts
Garth Wilson
The author provides high-level Forth interrupts in a simple way that works for most typical indirect-threaded systems. The zero-overhead interrupt support is simple, adds only about 100 bytes to the overall code, and can be nested as many interrupt levels deep as you wish. No additional stacks are required. It's another natural for Forth. (Some assembly required.)

12 Generation and Application of Random Numbers
Dr. Everett F. Carter, Jr.
(Closing from preceding issue.) The world's computers generate ten billion random numbers per second. Various compromises have to be made in order to even pretend to generate random numbers with a computer. This article explores the generation of random numbers and some important applications that use such numbers.

25 Top 10 List—Ways to Simplify Programming
Mike Elola
In the language of modern programming paradigms, why do you use Forth? Here's one way to explain your choice of language, and perhaps to persuade others, in ten easy steps...

34 Forth Nano-Compilers
K. D. Veil and P.J. Walker
This paper describes a highly efficient microcontroller programming system which could offer significant advantages in a wide variety of Forths. The authors' alternative approach to Forth object code generation makes each Forth keyword a small and highly specialized "nano-compiler" which generates the optimal target machine code for that particular keyword.

38 Some Vulgar Functions
Gordon Charlton
This article expands on the earlier article, "Rational Numbers, Vulgar Words." The author notes that much is to be gained from the graphical representation of data. A review of work in Logo showed that, in addition to the arithmetic primitives, frequent use is made of square roots and random numbers. These constitute an acceptable subset of vulgar math functions.

45 Convert Real Numbers to Fractions
Walter J. Rottenkolber
Most math formulas use real numbers, but Charles Moore prefers Forth to use scaled integers. So how do you find the fraction that best describes a real number, especially if the fraction needs to be small enough to use in signed numeric operations? This program generates a list of fractions equivalent to a real number by means of the concept called continued fractions.
7  **File Format Sleuth**  Bradley R. Olson
Ten years ago, file formats were no mystery. But now, data is hidden under objects, file formats are hidden under applications, and hardware is hidden under an API. When you really need to know how an application puts together the files it generates, this "format ferret" will show you the internals in a way you can understand.

15  **Engineering Notation with Integer Math**  Richard W. Fergus
Standard Forths are very satisfactory for accumulating and massaging data but may not provide an appropriate format for data display in certain applications. The format should account for data range and precision. The author's definitions in standard Forth provide engineering notation with selectable significant digits, decimal point positioning, and exponent offset.

18  **Interactive Remote Target Compilation**  Alan M. Robertson
In these days of fat Forths and heavyweight hardware, some companies still thrive in a minimalist landscape. The author shares his techniques for working in 2K of code space, with an eight-deep stack and 35 bytes of RAM, showing how a little RISC can pay off.

21  **Switch in Forth**  Walter J. Rottenkolber
Most Forthwrights are familiar with CASE to replace multiple branching statements. Its C equivalent, Switch, is less commonly seen. So, for frustrated C mavericks, the incurably curious, and the enlightened who are converting C to Forth, herewith: Switch in Forth.

23  **The Essence of Forth...**  Randy Leberknight & Dennis Ruffer
Programmers at the grande dame of Forth vendors are investigating Forth systems of the future. Here they remind us of the long-standing observation that Forth has an almost magical effect on productivity. But why? And how do Forth systems designers meet the requirements of the future without sacrificing the unique leverage that the language offers?

26  **Simple Mouse and Button Words**  Richard C. Wagner
DOS-based Forth systems too often lack—unnecessarily—the ubiquitous, point-and-click graphical user interface. But you can have such tools, even without a Windows-based Forth. This type of environment can be supplied with only ten blocks of code. The system presented here provides words to communicate with the mouse drivers, and to display the buttons, detect a button "press," and execute the code associated with a button.

---

**Departments**

4  **Editorial**  A gauntlet tossed—choose your weapons.

5  **Letters**  The Scientific Forth Library Project; Yes, Virginia; Producing correct code; Random erratum; Pictures worth a thousand comments.

22  **Advertisers Index**

38  **Fast Forthward**  Exposing Forth's modules; correction to ANS Forth Quick Reference Card.
7 **WLOAD for WRI Files**  
Hank Wilkinson  
Contrary to general "knowledge," Forth may be documented so well people compliment you! Windows' word processors can serve as editors, easing the documentation task. Understanding Write's file structure allows Forth to load its files without using the "Save As..." option.

11 **MuP21: Evolution of a Forth Chip**  
C.H. Ting  
Charles Moore, father of Forth, continues exploring the design and manufacture of Forth hardware. Moore has been demonstrating his newest Forth chip, enticing those who dream of executing Forth code on a Forth processor and those intrigued by do-it-yourself chip design.

13 **Jump & Execute Tables**  
Walter J. Rottenkolber  
Modifiable and economical, jump and execution tables are often overlooked as alternate "selector" words by new Forthwrights. Selecting options is common in programs; we can use tables as selectors because, in Forth, the distinction between data and functions is not sharp.

17 **A Bit of History**  
Jaanus Pöial  
Forth has been found in many places around the world. Here the author tells the story of Forth in his native Estonia, where it has occupied a distinct niche for over a dozen years.

18 **Algebraic Specification of Stack Effects**  
Jaanus Pöial  
How to validate a complex program's correctness is the subject of intense investigation and research. This article attacks the problem with a formalism which allows one to check the stack effects according to the program text.

21 **HDTV Format Convertor**  
Philip S. Crosby  
Forth was used extensively to build a High-Density Television (HDTV) Format Convertor for the Advanced Television Test Center, generating and evaluating video in the proposed U.S. HDTV standards. Engineers relied on several Forth dialects, about 300 ICs, and an A/D and D/A conversion process suitable for producing high-quality video.

**Departments**

4 **Editorial**  
Maximum advantage; A call for authors; Bottom line.

5 **Letters**  
Fractional math; The essence of Forth; Spaghetti in any language; Switch suggestions; Making FIRE.

27 **Stuck on Stacks**  
A guide for stacrobats and stacrophobes.

30 **FIG Chapter report**  
Maryland hears from Bliss Carkhuff and Julian V. Noble.

34 **Guest Essay**  
Object code vs. metacode: a market for Forth expertise.

35 **Advertisers Index**

38 **Fast Forthward**  
A reconciliation with ANS Forth and an exercise in interface design; corrections to ANS Forth Quick Reference Card.
8 **Forth in 32-bit Protected Mode**
   Richard Astle
   The author demonstrates his straightforward technique for getting Forth into 32-bit protected mode—without an assembler, linker, or a protected-mode program loader. In the bargain, because this method results in a system that actually consists of both 16- and 32-bit Forths, its user can switch between modes and keep using his familiar tools without porting them to the new environment.

21 **A Forth-Oriented Compiler Compiler and its Applications**
   Mati Tombak, Viljo Soo, Jaanus Pöial
   Stack-oriented languages (Forth, PostScript, etc.) are often used as intermediate or target languages in software systems because of their portability, flexibility, compactness, and simplicity. In the field of compiler compilers, the concept of a "virtual stack machine" is often used to describe the source language semantics and program interpretation. Unfortunately, every author uses his/her own stack machine. The main idea of the approach given here is to use a real, widely known, and standard language in the role of intermediate code in the compilers. Forth is the system's implementation language and the target language of the compilers it generates.

23 **Forth2LaTeX—a Pretty-Printer**
   Ronald T. Kneusel
   Forth's beauty should shine—even through a printout—so the author wrote this program to bring out the beauty of Forth code by transforming it into LaTeX, a variation of Donald Knuth's famous TeX typesetting system. Forth2LaTeX is for anyone who would like to create eye-catching source-code listings. It permits straightforward text within Forth source code, and a programmer can write code that runs and generates its own formal report when finished.

28 **Using Zeller's Congruence**
   Walter J. Rottenkolber
   This classic demonstrates how to use Forth to calculate the day of the week when given a date. What day of the week were you born on? When is the next Friday the 13th?

30 **Reports from euroForth '94**
   Gordon Charlton and Tim Hendtlass
   An international audience attended last year's euroForth conference to hear presentations by a diverse group of Forth programmers, vendors, and academicians. As befits Europe's most distinguished Forth gathering, both the papers and the organizing effort scored high marks.

---

**Departments**

4 **Editorial**
   Challenge and opportunity.

5 **Letters**
   Beginner's perspective, Structured comment, Backhanded critique, Communication without BIOS, Branch without doubt.

7 **dot-quote**
   Elizabeth Rather rebuts the "death" of Forth.

34 **Nominations for FIG Board of Directors Commence**

37 **Advertisers Index**

38 **Fast Forthward**
   Fine-tuning Forth.

Forth Dimensions 3

January 1995 February
6 GUI Application Development  Gary Ellis, Roy Goddard
Forth can provide GUI-friendliness without losing the interactivity we take for granted. This paper demonstrates that careful design and algorithm choices are essential to support the tools required. If these choices are made well, the tools produced can interact transparently with each other, the system, and the user to allow cost-effective commercial application development.

11 Object-Oriented Forth in Assembly  András Zsótér
When choosing a system to use in laboratory automation and robotics, the author searched for a system which would give him freedom and interactivity while generating standalone applications. Naturally, the solution was Forth. He had used an object-oriented language and, as an obsessed assembly programmer, decided to implement an OO version of Forth in 32-bit protected mode.

20 Simulating NASA Shuttle Robot Arm  Edward K. Conklin
NASA’s space shuttle carries a robot arm for satellite operations, and for tasks such as the recent repair and upgrade of the Hubble Space Telescope. Its complex motions are directed by rotational and translational joysticks, and software does the complex calculations. There are two ground-based versions, and last year Forth, Inc. provided the control program for one of them.

22 Vehicular Rollover Reconstruction  J.V. Noble
A motor vehicle sliding sideways on a pavement can roll over as a result of collision with a barrier such as a curb. The behavior of a car under these conditions can be quite complex. This paper presents a numerical simulation of vehicular rollover accidents on both wet and dry pavements, with graphical display of “flying” cars.

36 PowerMacForth Optimizer  Xan Gregg
This article presents an optimizing direct-code Forth compiler for the PowerPC, an ANS version of Creative Solutions, Inc.’s system for the PowerMacintosh. The architecture makes some stack operations painful, but the PowerPC’s pre-increment addressing mode makes possible a one-instruction push, and the fixed-length instructions are a boon to optimizers and decompilers.

41 MuP21—a MISC Processor  C.H. Ting, Charles H. Moore
Whether you are hungry for Forth hardware or are just interested in the design and production of custom microprocessors, follow along as these authors provide details of the design constraints and philosophy behind their latest project. Once again, we see that the Forth paradigm can be as rewarding in hardware as it is in software.

Departments

4 Editorial ............... Leadership, wizardry, and building bridges.
5 Letters .............. Forth’s northern exposure;
On the learning curve.
18 Scientific Library ...... Progress report on a valuable project.
26 dot-quote ................. Forth strategems.

27 Stretching Forth ...... LZ77 Data Compression.
40 Advertisers Index
45 Product Watch
46 Fast Forthward ........ Can a Forth kernel use objects?; correction to ANS Forth Quick Ref.
6  Debugging ANS Forth  
Joerg Plewe

Something was missing from several Forth systems compliant, or at least nearly compliant, with ANS Forth: the debugger. It can be challenging to design a debugger working strictly within an ANSI Forth system. ANSI Forth systems may generate code in widely differing ways. So how to write a debugger? When considering some clear facts, there seems to be a clear path to the solution...

16  Distributed Shared Memory  
Jeff Fox

Distributed Shared Memory is a simple construct upon which to build inexpensive parallel processing systems. It is widely accepted that workstation farms are more economical than supercomputers. Since these networks of workstations are often already available and interconnected, DSM software permits these machines to be used as supercomputers. This paper will discuss the use of DSM in parallel programming in the Forth programming language.

20  Forth Link to C Subroutines  
Michael Christopher

This article describes a technique for using C subroutine libraries with Forth. It was born out of the need to use—within a Forth program—existing C routines that came with a nine-track tape system. This is done using software interrupts. To use this method, a C and a Forth program must be written, both provided here.

26  Yet Another Interpreter Organization  
Mitch Bradley

There has been a mild controversy in the Forth community about how to implement the text interpreter. The particular problem is how the distinction between compiling and interpreting should be coded. At least three distinct solutions have been advocated over the years. The author proposes a fourth one, and claim that it is the best solution yet.

30  Case Cookbook  
Walter J. Rottenkolber

Over a decade ago, there arose the great Case Controversy, an attempt to extend to Forth a familiar control structure. The great Case Contest elicited several versions, many published in volume two of Forth Dimensions. They demonstrated the means for extending Forth to generate your own control structures. So, for the Forth beginner, here is the Case Cookbook.
7 Compiling ANS Forth
Tom Almy
A fully compiled application is faster and smaller than an interpreted one, right? ForthCMP compiles Forth applications directly into executable machine code. These files are smaller than executables produced by other language compilers, even smaller than metacompiled Forth applications. Execution speed is comparable to current C compilers.

11 A Forth Story
Allen Cekorich
Why would someone downplay Forth's capabilities? Follow one professional career from college, to first job, to learning and mastering Forth, through corporate shakeups, to today. Learn from another's experience what you can about how to shape a Forth career and how to avoid certain pitfalls—or tell us what you might have done in his place.

15 Novel Approach to VLSI Design
Charles Moore, C.H. Ting
This simple but powerful software package for VLSI chip design runs on a '386. Its editor produces and modifies IC layout at the transistor level, and generates standard output for IC production—no schematics or net lists required. It displays the layers of an ASIC chip with panning, zooming, and layer-selecting, and can simulate the electrical behavior of the entire chip. Its functionality has been verified in the production of several high-performance microprocessors and signal-processing devices.

18 Forth in Control
Ken Merk
Finding the leap from software design to hardware control to be intimidating? Forth is a powerful tool to interface the computer to the outside world. Its speed and interactive qualities let you get immediate feedback from external hardware as easily as from new colon definitions. The author shows how—with your PC, Forth, and a few electronic components—you can build a simple interface to control devices in the real world.

25 Code Size, Abstraction, and Factoring
John J. Wavrik
From comp.lang.forth, we find a concise explanation of these keys to elegant Forth design. Aided by an astute querent, the author dispels any confusion between factoring as a way to make code smaller, and factoring as a device to make code more comprehensible.

Departments

4 Editorial — Guiding the lily.
5 Letters — No better course.
24 FIG Board News — Election results.
26 From FIG Chapters
27 Advertisers Index
28 ANS Forth Clarification Procedures
29 Stretching Forth — Pinhole optimization.
38 Fast Forthward — Organizing code.
39 Backspace — Doug Philips responds to the preceding Fast Forthward.
6 Sets, Stacks, and Queues  Marty McGowan

What more can be said about stacks? Rather than floating-point or compiler or exception stacks, this article discusses using stacks in software applications—meaning stacks in the more general realm of sets and queues. Sets, stacks, and queues differ only in their access methods: LIFO, FIFO, and "AIRO." Becoming conversant with Forth versions of each of these brings the freedom to use whichever is most appropriate to your application.

14 Bounds Checking for Stacks

On the Internet's comp.lang.forth, Russell Y. Webb started this discussion, which revolves around an interesting technical issue while also shedding light on Forth problem-solving in general. It all started with an innocent, on-line request for advice: "What is the most efficient approach to checking for stack underflow and overflow?...I'm interested in having a fairly secure, stack-based virtual machine, but it seems like a lot of overhead to check everything. Any ideas are welcome."

20 Nanocomputer Optimizing Target Compiler: the Processor-Independent Core  Tim Hendtlass

New nanocomputers—small single-chip processors with integrated RAM, ROM, and I/O—appear regularly, and a simple alternative to assembly language can speed the development of applications for them. This processor-independent core only needs to be matched with a processor-specific library to provide a compiler that accepts Forth input and generates absolute machine code. (In the next issue, a library for the PIC16C71 and PIC16C84 processors will be presented.) The compiler supports chips with different word lengths and different architectures; it only expects that the target processor executes a series of instructions taken from some type of ROM and has some RAM in which to keep variables and stacks.

Departments

4 Editorial ............... Will the real Forth please stand up?
5 Letters ................. Challenged by macros; Forth vs. not-Forth
32 Forth On-line .......... Forth on the net, in the web, and at other electronic locales.
35 Advertisers Index
36 Forth Vendors .......... Where to find Forth systems, services, and consultants.
38 Stretching Forth ...... Extending CASE by simplifying it.
42 Fast Forthward ........ Vocabularies are overworked.
Contents

6 Forth in the HP100LX  M. Edward Borasky
The HP100LX and HP200LX computers fit in a jacket pocket and weigh just 11 ounces. But add Forth, and these featherweights become contenders. As 80186/DOS 5.0 environments, a number of Forth packages are available. The author provides benchmarks and more that will put Forth in your pocket—you’ll never have to leave home without it!

13 Hashing Forth  Xan Gregg
It’s a topic discussed so nonchalantly that neophytes hesitate to ask how it works. Hashing provides fast ways to search unsorted data—at the cost of memory. Robert Sedgewick describes hashing as “directly referencing records in a table by doing arithmetic transformations on keys into table addresses.” That should make sense to you by the end of this article...

17 OOP, Forth, and the Future  Ronald T. Kneusel
Unless it changes, Forth might miss the opportunity of its lifetime. Whatever its drawbacks, C++ is deemed more powerful because of its object-oriented features, and users demand ever-more-powerful applications. The object-oriented paradigm is winning the day, and C++ is the forerunner in the fight. But need it be?

19 RETRY, EXIT, and Word-Level Factoring  Richard Astle
RETRY is not a mere Forth-hacker’s whim. Since he found it a decade ago, the author has come to rely on it more than its well-known brethren, REPEAT, UNTIL, and AGAIN. RETRY and EXIT, taken together, make possible flexible control structures with the word as the unit.

22 Making Forth Professional  Peter Knaggs
Forth vendors say that memories of early, poor public-domain implementations still deter potential customers. And software managers tend to see only code, they do not appreciate that Forth is not just another language, but a philosophy. Many contemporary practices—structured programming, reusability, libraries, etc.—have been available and used in Forth for many years. Here’s one way to win back some much-deserved credibility.

26 Nanocomputer Optimizing Target Compiler: the PIC Library  Tim Hendtlass
As promised in the preceding part of this article (FD XVII/3), here is a library for the PIC16C71 and PIC16C84 processors. The processor-independent core teamed with this library accepts Forth input and generates absolute machine code for the PIC. The compiler supports chips with different word lengths and architectures, just requiring a different library for each.

Departments

4 Editorial ............... The programming public
5 Letters ................. ‘Two cents’ worth
10 Advertisers Index
18 President’s Letter ... To the new board of directors and to members
24 Conference Report ... Emerging technology at the 1995 Rochester conference
33 Forth On-line ........ Expanded and updated list of on-line Forth connections
36 Stretching Forth ...... Associative lists
43 Fast Forthward........ Objects promote new programming style
5 Scattering a Colon Definition
M.L. Gassanenko
The author's prefix-notation Forth assembler performs initialization actions before processing any new instruction. The assembler's switches are set according to defaults and instruction operands, and they determine what happens; some switches select a Forth word to be executed. The problem was that initialization actions belong to two different modules at the same time: to the module they initialize and to the general initialization module. The author wanted to distribute these actions so they would be located in the modules they initialize, but used as a single definition.

8 Mobile Computing in Brazil
Klaus Blass
Brazil boasts the world's eighth-largest economy and imports now enter freely, including computer hardware. Local companies are investing in sales force automation, setting the stage for mobile computing solutions such as those this author's company develops in Forth. While other firms are developing for the same market, Forth makes maximum use of limited resources, typically packing four times the functionality of a competitor's C program into an executable one-half to one-fourth the size.

18 FORML 1995
András Zsótér
Once again, intrepid Forth practitioners converged on Pacific Grove, California, to present their latest works, to collaborate informally on new ideas, and to seek a consenscus about technical, organizational, and political issues facing the Forth community. Our reporter from Hong Kong reports on his first experience of the annual FORML Conference.

19 Stepper Motors
Skip Carter
This is the first installment of the new "Forthware" column about using Forth to interact with the real world. It will explore how to control motors of various types; this issue discusses using the PC parallel port to control stepper motors—adopting the useful fantasy of working on a microprocessor-based control application and using the PC parallel port as a proxy for the digital I/O channels on the controller.

Departments

4 Editorial .................. Forth jobs, FIG works.
4 Dot-quote .................. Postpone, the inevitable.
7 Advertisers Index
10 FIG Board Meeting .... "New faces, fresh approaches."
11 President's Letter .... Skip Carter takes the helm.
12 Embedded Systems Conference & "A Prayer for Forth"
13 Stretching Forth ...... All the standard Forth you need.
42 Fast Forthward .......... The prospects for ++Forth.
7 Coordinating Pygmy and C
Frank Sergeant
A few changes allow Pygmy Forth to be wrapped inside a C program. The C program can pass strings to be INTERPRETED by Pygmy. Pygmy can call C library functions or functions defined in the C program. The essence of this method is this: The C wrapper program allocates RAM into which it loads the Forth image. The C wrapper contains a table holding the addresses of the functions or structures (variables) we want Forth to be able to access. C calls the loaded Forth image as a function, passing along the address of the table. Via the table, Forth can call the C functions, or read or modify the C structures (variables). Eventually, Forth terminates, returning control to the C wrapper.

21 Differential File Comparison
Wil Baden
"Stretching Forth"—Every programmer needs a utility to compare files—particularly source files—to find where and how they are different. In the course of making a series of small modifications to fix a misbehaving application, the programmer can easily lose track of just what has been done. Then the file comparison utility can be used to show the changes. To do the job right is not a trivial task. The obvious algorithm will sooner than later fail miserably. The trick is not to look for differences but to look for the longest common subsequence—the longest set of lines which are the same in both files and in the same order with what's different interspersed. What's left are the differences. The author has been refining his version of this tool since 1976 and shares here the benefits of his experience.

30 Getting to the Hardware from Linux
Skip Carter
"Forthware"—Those who move to Linux without previous experience with minicomputers and workstations are probably shocked to discover one fact about sophisticated operating systems: you no longer control the machine, the operating system does. The essentials are covered here: which Forth to use, how to access the parallel port, how to add device drivers; the Linux code for the preceding issue's topic (stepper motors) is included.
7 More Than A Simple State Machine
   John Rible
Familiar with some "traffic intersection" boards made for a programming course, the author
realized they seemed ideal to illustrate state machines, and he wrote a simple state machine
"mini-language" in Forth. Because of the hardware-oriented approach, the lexicon turns the
normal method on its head. The result models the logic equations, clock, and flip-flops as a set
of software equations that can be directly represented as a circuit diagram.

12 DOS-Compatible Disk Access for Targets
   Dwight Elvey
As a development environment for target devices the PC works fine; but when the application
changes, new code must be transferred to the target. Burning PROMs is a solution. A more
practical method is possible if both PC and target support serial connection. A better way may
be to add target disk I/O—the code becomes accessible to the target simply by moving the
disk. This also permits the target to log data in a format that can be stored or used on the PC.

22 Safety Critical Systems
   Paul E. Bennett
The proliferation of microprocessor-based systems throughout nearly every aspect of first-world
cultures means that more of us are engaged in work capable of great impact—for better or
worse. That brings inherent responsibilities and potential liabilities. Engineers, manufacturers,
and operators must not only deliver a system that works (and works safely under varying
conditions), but also protect themselves against claims for compensation and against criminal
charges and the resultant destruction of their careers.

27 Taming Variables and Pointers
   Chris Jakeman
Forth programmers enjoy unlimited access to code and computer, but there are times when
so much freedom is counter-productive. Fortunately, the tools are always at hand to try a new
idea and, perhaps with the assistance of on-line Forth colleagues, to refine it. When the author
had trouble debugging some code with lots of pointers, he added a new word to Forth—an
alternative to VARIABLE—to find his mistakes.

31 Does Late Binding Have to Be Slow?
   András Zsótér
Object-oriented programming entices many programmers and, perhaps, its performance pen-
alties discourage a like number. Many Forth dialects already have some sort of OOP support,
but some people in the Forth community argue that the overhead involved is unacceptable for
time-critical applications. This paper is directed towards them; its main goal is to make object-
oriented techniques efficient enough to become more attractive to those who like the "close
to silicon" approach.

Departments

4 Editorial
   Forth in Cyberspace

5 Letters
   A Forth History. On Target. Corrigendum to "Stretching Forth."

30 Advertisers Index

36 Stretching Forth
   LIFE that knows when to stop.

45 Forthware
   An introduction to power control.
Contents

Features

9 Safety Critical Systems, Part Two  Paul E. Bennett
The author argues that Forth is eminently suited to tasks which require formal certification as reliable and safe. He elucidates the requirements of such "safety critical systems," discusses various models and environmental influences, and shows how he uses Forth to minimize risk and liability.

13 More Than a Simple State Machine  Devin Wilson
Could a fifteen-year-old program a traffic-intersection controller? He could if he were a homeschooler learning Forth! John Rible's state machine engine and C.H. Ting's hardware provide an environment in which to learn more about Forth and real-world mechanisms and how subtle design decisions influence our lives.

18 A CGI Shell for the Apple Macintosh  Ronald T. Kneusel
The World Wide Web is more than passive, interlinked documents: it is interaction between server and client. CGI applications can amplify the degree of interactivity, providing increased value to the users and earning prestige for the Web site. This Mac shell might inspire you to stretch both your imagination and the Web... (CGI experts: how about an on-line ANS Forth tutorial with self-checking programming exercises?)

22 PC Floppies for non-DOS Hardware  Dwight Elvey
Custom embedded systems often can benefit from a floppy disk drive during development and for data logging during operation. You can easily avoid the labor and expense of a custom interface by using hardware generally available for PCs. The author steps you through the formats, port addresses, and other considerations that will arise with a project like this one.

30 hForth: a Small, Portable ANS Forth  Wonyong Koh
hForth is a minimalistic, public-domain ANS Forth based on eForth and intended for embedded systems. The basic ROM and RAM models are designed for portability, but can be optimized for specific CPUs, as demonstrated by the 8086 EXE model. The author incorporated ideas and techniques contributed by the Forth community, including an elegant new multitasker.

Departments

4 Editorial .................. Input and outreach.
4 dot-quote .................. Unix could use Forth...
5 Letters .................... Mousetrap myth; Java learns from Forth, Forth may benefit.
5 Product Watch
6 Stretching Forth ...... Circular string buffer.
21 Advertisers Index
34 Forthware ................ Controlling DC motors.
Features

6 Development Aids for New Micros  Richard W. Fergus
The author uses popular New Micros products, but wanted to enhance the facilities provided—to the point that his platform would have a significantly enhanced “feel.” He shares how he attacked downloading and terminal emulation, separated RAM variables, Motorola 519 record output, Brodie’s MAKE–DOER, multitasking, and a logging function in the terminal emulation mode to copy the 519 records.

11 The Elephant Who Refuses to Be Bagged  C.H. Ting
When attempting to use a compression algorithm in Forth that had been presented at a conference, the new data presented new challenges. The accompanying code documents the thorny trail of discovery, analysis, and “resolution.”

18 4tH, an Experiment in C  Hans Bezemer
4tH is a different Forth implementation. It contains much of Forth, but is translated to C. Yes, it has two interpreters, but they behave differently. It is token-threaded, but has no dictionary. It is a Forth using conventional compiler technology, but it isn’t a standalone compiler—it is a library. One result: 4tH produces bytecode, like Java, which can be used without any modification on every platform to which it has been ported.

31 FIG Board Increases Member Benefits  Elizabeth Rather
The Board of Directors of the Forth Interest Group met at the recent Rochester Forth Conference. It was the second official meeting of the currently constituted board. Major actions included instituting additional benefits for FIG members, clarification of benefits for corporate members, review of Forth Dimensions advertising rates, planning of promotional activities, and review of plans for managing the FIG office and activities.

33 Rochester Forth Conference  Nick Solntseff
Canada’s first Forth conference successfully presented papers on a wide-ranging variety of contemporary issues. “Rochester-in-Toronto” marked the first time the esteemed Rochester Forth Conference was held outside New York and, as this reporter comments, signs of Forth’s vitality were well represented.

Departments

4 Editorial
4 dot-quote
5 Letters .................. What ANS Forth needs now.
23 Stretching Forth ...... Breaking Code: Forth solves 150-year-old problem
32 Chapter News ........ Southern Ontario hosts Rochester Conference
34 Forthware ................ Digital input and synchronous I/O
5 Towards a Discipline of ANS Forth Programming  
M. Edward Borasky  
Recent on-line discussion about structured programming, multiple entries and exits, finite state machines, and other issues of Forth programming style—including readability and, especially, enhancing the ANS Forth control structures—inspired the author. In this article, he implements the Dijkstra guarded command control structures.

15 C-Style Arrays in Forth  
M.L. Gassanenko  
A feature of modern processors that is rarely utilized in Forth is based indexed addressing. This paper proposes a relevant notation for cell array indexing in Forth. The proposed syntax for the indexed access operations was inspired by C and Algol-68. It supports multi-dimensional arrays, and a similar syntax can be used for bit or double-cell arrays. The paper also shows how analysis of possible name conflicts should be performed.

22 Forth in Control — A Window Interface  
Ken Merk  
In his last article, the author showed how to build a parallel-printer-port interface with a series of LEDs representing the state of each bit on the port. In this article, he shows how to control that display using Microsoft Windows as the platform. The code builds a graphical user interface—arrays of buttons which can be activated by the mouse to control peripheral devices. This point-and-click environment makes it easy to manipulate the output port.

Departments

4 Editorial
4 dot-quote
27 Stretching Forth ...... Filters and sponges.
33 Forthware ............... Measuring frequency and sampling time-dependent signals.

Advertisers Index

The Computer Journal ............. 42
FORML Conference ............. 44
FORTH, Inc. .................. 17
Forth Interest Group ... centerfold
Miller Microcomputer Services .................. 14
Silicon Composers.............. 2
8   Garbage Collection in Forth
    Jim Schneider
    The ANS Forth memory allocation wordset is a good start toward a standard method of garbage
collection, but it's only a start—and this author ran into some of its shortcomings. He says,
"Although I don't admire LISP's irritating syntactic quirks, I do like the fact that it keeps track
of memory items and will discard memory that's no longer needed. I think a Forth garbage
collection utility would be invaluable. The code in this article is a first step in that direction."

11  Back to Forth:
    An Object-Oriented Forth
    Anatole L. Medyntsev
    This article describes an experimental object-oriented Forth system for MS Windows. Certain
ideas are based on the author's experience in FoxPro 2.5, Visual C++, MS Access, Visual BASIC,
and Delphi for database development in the financial field. In his opinion, a combination of
an interpreted object-oriented language and the usual C or Pascal is best for database
applications development. Unlike other options, Forth is not a monolithic language, and it
provides a simpler, more flexible, and more open technology.

15  Zen Floating Point
    C.H. Ting
    Less is more, especially with the '486's free FPU—its floating-point registers are arranged as
a stack. But its instruction set is unnecessarily complicated because it allows the registers to
be accessed as a regular register set. So even less is even more: if we eliminate the register-
addressing instructions, a floating-point package can be built very simply and elegantly.

19  A Stack-Based
    Dataflow Operating System
    Barry Kauler
    "Visual programming" can offer high productivity, and can be used by people who know little
about conventional text-based languages—but this comes at a very high price. The visual
dataflow paradigm intrigued the author, as did operating systems for embedded applications.
He conceived a dataflow OS targeting real-time embedded applications.

22  Can POSIX Threads Be Used as a Standard
    Forth Multi-tasker?
    Dr. Everett F. ("Skip") Carter, Jr.
    Threads are multiple processes running in the same memory image—like multiple Forth
virtual machines running simultaneously in the same dictionary and data space. The POSIX
threads API consists of a set of calls that maps very closely to that of a traditional Forth
cooperative multi-tasker. This similarity could be used to leverage an ANS Forth multi-tasker
via conformance to the international POSIX standard.

Departments

4 Editorial
5 OfficeNews
6 Writer's Guide
23 Advertisers Index
24 FORML Report
27 Stretching Forth
30 Forthware

Experimenting with ANS Forth.
Squareroot and Golden Ratio
Closing the Loop — PID Controllers
6 Fuzzy Forth — ANS-compliant Extension  
Rick VanNorman

Fuzzy logic remains popular fodder for experimentation and discussion. Its controversy does not detract from its advantages, and many products provide programmers with fuzzy logic interfaces—but rarely do these packages come with source code which can be examined and understood by the programmer. This paper presents an overview of fuzzy logic, a set of ANS-compliant source code extensions to Forth to implement a fuzzy logic inference engine, and a simple example of a Fuzzy Forth control program.

14 Object-Oriented Programming in ANS Forth  
Andrew McKewan

Programming on today's resource-rich platforms amounts, to a large extent, to managing complexity. Looking for a way to tame this complexity, the author's study led him to adopt Yerk's underlying (and quite general) class/object/message ideas. He first ported these to Win32Forth, a public-domain Forth system for Windows NT and Windows 95. But his goal was much more general—to provide any ANS Forth system with the ability to use the object-oriented syntax and programming style. Success will mean the ability to begin developing a library of ANS Forth objects.

Sidebar: methods of achieving object orientation in Forth. Many have tried, and many have succeeded—to some degree. A Forth programmer can use object-oriented programming to supplement normal Forth programming, treating it as just another tool in the toolbox; or can use the technology or everything.

30 Yet Another Modest Proposal  
Richard Astle

Thus far, there has been no consensus about object-oriented programming in Forth—other languages also evidence disparate ways of serving up object technology. But instead of waiting for a standard or new wordset, we can steal some of the thunder from object technology. A related topic, not talked often mentioned and far from standardized, is modules of source code containing implementation and interface parts, private and public definitions. It promotes modularity, results in fewer headers cluttering things, and permits reuse of names. In fact, the author has stopped using the vocabulary/wordset concept!

32 MPE's Forth Coding Style

This Forth layout standard covers the layout of Forth source code in text files. It covers the layout of code and comments, along with the use of the file, and the reasons for a standard, and the reasons for certain decisions and recommendations in the standard. This document reflects current programming practice at MPE

36 A Gentle Introduction to Digital Filters  
Skip Carter

Digital filters provide a means to condition a digital signal in order to achieve a variety of purposes. Depending on the problem, a filter may reduce unwanted noise, isolate or reject a piece of a signal, or enhance certain components of a signal. What makes digital filtering hard to master is that creating a proper filter takes a bit of calculus—usually in the complex plan. What makes this intimidating is the degree to which various authors handle the calculus. Here you will find the math—but starting at the shallow end of the pool.
A Forth Memoir by John Nangreaves
There are at least as many Forth stories as there are Forth users, and each sheds light on Forth and on how it fits into the toolbox of the working professional. In this case, after coding assembler (in hex, even) for a year and a half, the author concluded there had to be a better way. So this story begins...

A Simple Implementation of the Kermit Protocol in Pygmy Forth by Frank Sergeant
In the preceding article, John Nangreaves gives more than a passing nod to the public spirit of the Forth community, especially to individuals who distribute useful tools and generously continue to support them. A case in point is Frank Sergeant, whose Pygmy Forth has a following among those who appreciate Forth in its lean-and-mean aspect. Here, he implements Kermit in Pygmy Forth.

Transportable Control Structures by Randy Leberknight
ANS Forth formalizes an aspect of Forth's extensibility: the creation of new types of control structure words without writing any new words in assembler. For example, both IF and WHILE perform a conditional forward branch at run time; that common behavior can be factored out and shared. In a production environment, this ability can shave time off the development cycle.

Working Comments (long)? by Julian V. Noble
Code fragments can be tested prior to compilation to determine their effect on the data and return stacks, without risking system crashes or hidden bugs. Code for a preliminary version is given, together with discussion of possible improvements, should that prove desirable. A fine example of creative work spawned by a casual remark on comp.lang.forth.

Standardizing OOF Extensions by Anton Ertl
Andrew McKewan argued in the last issue that we need to agree on a model to start building an object-oriented library. This author's is the reverse: write a good object-oriented library that everyone wants to use, and the object model on which that library is based will become the standard.
A Platform-Independent Token System for Payment Terminals
by Peter Johannes, Stephen Pelc, and Elizabeth Rather
The financial industry is scrambling to implement new instruments intended to revolutionize personal finance. Smart cards, debit cards, electronic purses, and more are buzzwords that, in certain audiences, electrify with their implications. Hardware constraints are a significant part of the puzzle—how to pack transaction-supporting applications into the amount of RAM that can fit onto a plastic card of the usual proportions—which also happens to contain a microprocessor and I/O. Forth is a natural in constrained environments and, as it turns out, is playing a leading role in the development of this new technology.

A Simple Implementation of the Kermit Protocol in Pygmy Forth
by Frank Sergeant
Frank Sergeant, whose Pygmy Forth has a following among those who appreciate Forth in its lean-and-mean aspect, presented a description of his Kermit implementation in the preceding issue. Herewith: the code.

Yet Another Forth Objects Package
by Anton Ertl
Programmers often must treat several data structures similarly in some respects, but differently in others. A big CASE structure would not be very elegant, and would require maintenance. In a nutshell, this is the problem object-oriented systems solve. After criticizing the Neon model in the last issue, the author presents a model he finds better, and its implementation.

DEPARTMENTS

<table>
<thead>
<tr>
<th></th>
<th>EDITORIAL</th>
<th>20</th>
<th>STRETCHING STANDARD FORTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Errata; FORML; Corporate Members; Thanks...</td>
<td></td>
<td>Linked Lists</td>
</tr>
<tr>
<td>5</td>
<td>OFFICE NEWS</td>
<td>22</td>
<td>THE VIEW FROM GOAT HILL</td>
</tr>
<tr>
<td></td>
<td>Rochester Conference news, and an invitation.</td>
<td></td>
<td>The Search Paradigm</td>
</tr>
<tr>
<td>6</td>
<td>ISO/IEC FORTH</td>
<td>24</td>
<td>TOOL BELT</td>
</tr>
<tr>
<td></td>
<td>International standard released.</td>
<td></td>
<td>Simple Macros</td>
</tr>
<tr>
<td>19</td>
<td>FREeware &amp; SHAREware</td>
<td>31</td>
<td>FORTHWARE</td>
</tr>
<tr>
<td></td>
<td>Updates to Win32Forth and Pygmy</td>
<td></td>
<td>Adaptive Digital Filters</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td>MPE's coding style standard continues...</td>
<td></td>
</tr>
</tbody>
</table>
Writing a Macintosh Application with Pocket Forth by Ronald T. Kneusel
Forth is breathing some sanity into the world of GUI development. The author doesn't claim to be a Mac expert, but demonstrates event-driven programming with this pared-down system. Turns out, a familiar tool takes most of the curse off what many Forth programmers were avoiding for so long.

Yet Another Forth Structures Package by Anton Ertl
In the previous issue, the author proposed a model for object-oriented Forth, and referenced this paper. The package presented here offers support for features like C's struct or Pascal's RECORD, and includes automatic handling of alignment and optimization of fields with offset 0.

Approaching CREATE DOES> by Dave Taliaferro
Defining new defining words does not have to be the bête noir of new Forth programmers. Instead, it can be the switch that illuminates a deeper appreciation for, and greater proficiency with, the language. The author presents this article in the spirit of helping others while he is himself still near enough to the learning curve to remember the things that puzzled him.

Lookup Tables by Hans Bezemer and Benjamin Hoyt
Okay, they aren't glamorous, and they would rarely be called elegant. But the authors argue persuasively that lookup tables are utilitarian, flexible, maintainable, extensible in various ways, and indeed are the right solution to many problems. And with these tools, their implementation becomes easier.

Pygmy Embellishments by Richard W. Fergus
Years of using a Forth system brings more than proficiency, it likely brings a package of ancillary tools one has designed for certain application domains, to correct perceived deficiencies, and to provide "personal-favorite" facilities. Take this opportunity to explore the personal toolkit of a Pygmy pro.

DEPARTMENTS

2 OFFICE NEWS 6 SPONSORS & BENEFACCTORS
4 EDITORIAL 28 STRETCHING STANDARD FORTH
5 ROCHESTER '97 REPORT Arcipher — alleged RC4
6 EUROFORTH '97 REPORT 32 FORTHWARE
30 MPE's coding style standard concludes...
Forth in Control: Analog/Digital Converters by Ken Merk
When controlling devices in the outside world using your parallel port, sometimes a need arises to monitor the device to see if it is responding properly. This feedback can be a digital signal that is fed directly into the computer, or an analog signal that needs to be converted into digital form so it can be understood by the computer. This article provides a tutorial approach to putting Forth in control, and provides information about relevant hardware resources.

MicroFont: 4x5 by Rob Chapman
A fair number of embedded devices and other small systems incorporate a small liquid-crystal or other display, such as the 128×64 one this author was using. Such displays may include fonts with a limited character set or which are otherwise not optimal for the application. This article presents a 4×5 font that contains all printable ASCII characters, and is quite readable; and the Forth code can be modified easily to extend or edit the provided characters.

Misty Beach Forth: An Implementation in Java by Mark Roulo
This article provides an overview of Misty Beach Forth, a Forth implementation running under Java. While, at first glance, the two technologies would seem an easy fit, the Java Virtual Machine (JVM) has peculiarities that make the fit awkward. Further, in the spirit of Java’s well-defined semantics, the author added some design constraints that made implementing Misty Beach Forth harder than it needed to be.

Factors Influencing the Use of Forth for Large Projects by Dr. Everett F. Carter, Jr.
The author looks at the factors involved in large projects and how these impact the choice of using Forth for them. Frequently the choice of languages is dictated by managerial, customer, or agency requirements that are really political and not technical decisions. For the most part, there is very little one can do to deal with these kinds of directives on a project-by-project basis. In the longer view, these decisions can be influenced by education and by ensuring that there are no technical issues regarding a particular language choice. Here we are primarily concerned with the technical issues that become important when the system being created is large.
Adventures in Debugging a Mix of New Hardware and Software by Randy Leberknight
Debugging new software or new hardware can be a challenge at any time. However, it is particularly challenging when the software is the firmware for the new hardware. In many cases, the presence of new hardware necessitates new firmware, and we must test them both at once. Some of the features of Open Firmware which help us deal with these challenges are discussed here.

Easy Target Compilation by Dave Taliaferro
Custom macro languages are easy to create in Forth. Building on techniques from his last article (FD XIX.3, “Approaching CREATE DOES”), the author demonstrates how simple it is to write custom compilers and assemblers using Forth. Best of all, the new languages retain the unique interpretive and compiling characteristics of Forth.

Forth Profiling Utility by Marcel Hendrix
LPROFILER counts the number of times a source code line is executed. Although not measuring the exact run time of a program line, LPROFILER provides a good start when hunting for performance bottlenecks. Once the most promising candidates for optimization are known, the word ‘TIME’ can be used to time the execution performance of individual Forth phrases. A fringe benefit of LPROFILER is that it shows lines of code that are not visited at all: it points out incompletely tested applications.

Manipulating Input Source Contexts in ANS Forth by M.L. Gassanenko
This paper presents a method of manipulating contexts, a technique which may be useful for programmers who have to switch contexts, e.g., when binding together two languages. The particular problem solved in this paper is to change the current input source parameters, having no special construct to do this or to establish a new input source context with the desired parameters. Doing it in ANS Forth is compared to approaches in non-standard Forth.
### Stack Gymnastics Made Easy
by Ronald T. Kneusel
Forth's stack nature is one of its strong points, but manipulating a large number of stack items can be somewhat difficult with traditional stack words. What is described here is based on a similar construct found in a Forth-ish programming language the author saw a number of years ago, and makes complex stack manipulations very easy.

### Building a Remote Target Compiler
by Dave Taliaferro
This represents the culmination of this three-part series about remote target compilation. A remote Forth target compiler runs on an embedded development host machine and allows interactive production of executable Forth and assembler routines for a target microprocessor. As keyboard definitions or source files are interpreted by the host Forth, the resulting code and data is transparently uploaded to the target for immediate testing and further programming.

### Mushroom Identification
by Charles Samuels
The plan was to allow users to point and click to describe an unknown mushroom to the program, and to get an identification. But the author had not done any serious programming for the past 12 years, and Windows had passed him by. So he had to create a serious data set and catch up on Windows programming. With LMI's WinForth and a third-party app to help with the interface, he developed what he hopes will become a commercial success.

### An Extensible User Interface
by John J. Wavrik
Interface design, whether graphical or not, deserves far more attention that it usually receives. It is, after all, how most others will experience—and, hopefully, use—our code. The author shows his research work to non-programmer mathematicians and students, and an obscure interface would result in even greater obscurity for his subject material. Here he provides his command-line interface tool.

### Runstk – Stack Utility
by Warren Heath
This text editor allows the display of the data stack symbolically anywhere within a word shown on the display. It allows the safe virtual running of source code, and can also automatically check cumulative stack effects against the given stack picture for the word. It can put the cumulative stack picture into the source for documentation, and can also show the stack picture of any word in the system.

### DEPARTMENTS

<table>
<thead>
<tr>
<th>Page</th>
<th>Department</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>OFFICE NEWS</td>
</tr>
<tr>
<td>4</td>
<td>EDITORIAL</td>
</tr>
<tr>
<td></td>
<td>Around the world</td>
</tr>
<tr>
<td>5</td>
<td>LETTERS</td>
</tr>
<tr>
<td></td>
<td>Why is Forth not more popular?</td>
</tr>
<tr>
<td>8</td>
<td>FORTH ON THE WEB</td>
</tr>
<tr>
<td>32</td>
<td>NEW PRODUCT ANNOUNCEMENT</td>
</tr>
<tr>
<td>32</td>
<td>STANDARD FORTH TOOL BELT</td>
</tr>
<tr>
<td></td>
<td>Character Literals</td>
</tr>
<tr>
<td>33</td>
<td>STRETCHING STANDARD FORTH</td>
</tr>
<tr>
<td></td>
<td>Double-Number Arithmetic</td>
</tr>
<tr>
<td>35</td>
<td>SPONSORS &amp; BENEFACTORS</td>
</tr>
</tbody>
</table>
eForth for Java
by Michael Losh
Extending Forth's potential to reach into the on-line world, this high-level implementation for the Java Virtual Machine (JVM) runs as a console-style applet which can be opened in a Java-enabled web browser such as Netscape Navigator or Microsoft Internet Explorer. eForth can open new opportunities for promoting and teaching Forth to a wide audience over the Internet.

Forth in Control: Temperature Monitoring
by Ken Merk
Temperature is one parameter of our environment which has an affect on all living things. Even machines perform differently through a range of temperatures. Many opportunities arise with the need to measure temperature accurately and then perform certain tasks accordingly. This article covers how to interface a digital thermometer sensor chip to your computer's parallel port. The device used is Dallas Semiconductor's DS1620, which contains the sensor itself and a three-wire serial interface inside an eight-pin DIP package.

LOAD" Module"
by Dave Edwards
This article provides a method to organize programs into sections by using the first line of each screen—the line usually left for comments—and thereby attain a far greater degree of flexibility and control. The idea was developed to organize the loading of programs, but can even be used to implement, for instance, a simple help engine. The ability to use the data on the "header" line for structural information provides a surprising amount of functionality from such a simple mechanism.
Finite State Machines in Forth
by Julian V. Noble
Certain programming problems are simpler to solve using abstract finite state machines. This paper provides methods for constructing deterministic and non-deterministic finite state automata in Forth. The “best” method produces a one-to-one relation between the definition and the state table of the automaton. An important feature of the technique is the absence of (slow) nested IF clauses.

Safer Numeric Input
by Jerry Avins
"Safety" is, perhaps, something many programmers don’t have to worry about. But when your embedded system is controlling a medical device or a piece of heavy machinery, such concerns become paramount. Specialists in the field know there are many aspects to the subject; this article addresses one concern—that the operator’s display accurately represent the data the program is using.

EXPRESS Factory Control
by Allen Anway
EXPRESS is a Forth-based system for running machinery or a factory. Its specialty is real-time performance. This application runs four lime kilns. In this case, EXPRESS works in concert with Programmable Logic Controllers (PLCs) distributed at various use sites throughout the plant.

Forth to the IRS
by Len Zettel
The annual U.S. tax-filing ritual brings great stress as the April 15 deadline mars an otherwise lovely time of year. We can file extensions and go through the emotional trauma again in a few months, or do as this author did, and dilute the dread with the pleasure of Forth programming...

Report: EuroForth 1998 Conference
by Paul E. Bennett
Forth might have nominal roots in the United States, but it migrated to Europe almost immediately. It has strong supporters and developers there, and many significant applications. The annual Forth conference is the primary venue of intellectual exchange and camaraderie among Forth users in Europe.

Two Problems in ANS Forth
by Thomas Worthington
The author addresses his concerns about potential problems in ANS Forth: colon-sys on the control-flow stack, and the inability of the programmer to assign the input stream to an arbitrary block of memory. The problems are described along with solutions—which present a bonus benefit.
**Porting hForth to the StrongARM SA-110 RISC Processor**
*by Neil Crook*

The author was working for DEC's semiconductor division as an applications engineer and settled upon the idea of doing a port to the 64-bit Alpha RISC processor. But his group won the task of supporting StrongARM chip sales, and he started work on the design of a board that would be used as a hardware verification and evaluation platform for the first StrongARM chip, the SA-110.

**The Stuttering Context Switch**
*by Martin Schaaf*

How to build the context-switching part of a Forth engine? The author had been focussed on optimizing the time-wasting stack-shuffling operations, devising a method of buffering the top three items on the stack and performing stack shuffling in parallel with other operations. Task switching, however, he had to learn about from his plumbing.

**Linearizing a Thermocouple with Two-Step Interpolation**
*by Jerry Avins*

When building a profiling temperature controller for a small oven, one of the necessary details is a way to read a thermocouple that is to indicate temperature in degrees F and be suitable for use in a control loop. Thermocouples are only slightly non-linear. Nevertheless, a simple way to linearize them also works well with functions that have much greater non-linearity, and it is presented here.

**ANS Appendix to “Finite State Machines in Forth”**
*by Julian V. Noble*

ANS-compatible code to accompany the author's article (which appeared in our preceding issue), and an erratum to the code that appeared previously in these pages.

**A Forth Switchblade**
*by Rick VanNorman*

An example of a switch in Forth is the CASE statement. The execution-time behavior of CASE and OF can be optimized until your system implementor is exhausted, and performance will be similar to that of a C version. So why would anyone want to implement a new switch construct in Forth? For SwiftForth, the reason was the need for extensibility—to be able to define the base structure and to extend it at will. The traditional CASE statement does not lend itself to being extended after it is defined.

**Point and Do**
*by Richard W. “Dick” Fergus*

A pointing device can be very useful to interface the user with the intricacies of a program. Herewith, the author supplies relevant support code for Pygmy Forth although, with minor modifications, they should be applicable to other Forth dialects.

---

**DEPARTMENTS**

<table>
<thead>
<tr>
<th>2</th>
<th>OFFICE NEWS</th>
<th>26</th>
<th>STANDARD FORTH TOOL BELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>EDITORIAL</td>
<td>29</td>
<td>STRETCHING STANDARD FORTH</td>
</tr>
<tr>
<td>15</td>
<td>CROSSWORD — “Stacks”</td>
<td>34</td>
<td>Only Standard Definitions</td>
</tr>
<tr>
<td>16</td>
<td>PRESIDENT’S LETTER</td>
<td>35</td>
<td>URLs — a selection of on-line Forth resources</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>26</th>
<th>STANDARD FORTH TOOL BELT</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>STRETCHING STANDARD FORTH</td>
</tr>
<tr>
<td>34</td>
<td>Only Standard Definitions</td>
</tr>
<tr>
<td>35</td>
<td>URLs — a selection of on-line Forth resources</td>
</tr>
</tbody>
</table>

Forth Dimensions XX.3
DynOOF-style Objects for the i21 microprocessor by András Zsótér
The i21 is a stack machine designed to meet the minimum needs of the programmer; at first it does not seem the best candidate for implementing OOP into the Forth virtual machine. This implementation demonstrates that it can be done, and that such an implementation is suitable for commercial use.

Toward a Standard for Cross-compilers and Embedded Systems
by Elizabeth D. Rather
An agenda item for ANS Forth involves the issues raised by embedded systems and cross-compilers. Such systems represent a large body of Forth use. In 1996, FORTH, Inc. and MPE developed a joint set of standards for such systems. These now have been used in commercial settings and, as a result, a good body of experience is available from which to form the basis for a proposal for standardization.

FORML Conference #20 by Richard Astle
Perhaps the longest-standing tradition in the Forth community, the FORML Conference just celebrated its twentieth anniversary. It's never been said that "as FORML goes, so goes Forth," but this year's increase in both attendance and number of presentations was encouraging. Certain of the presentations may have set in motion technical developments which will bear fruit in the coming few months...

How and Why to Use Multitasking by Frank Sergeant
Most FortHS provide multitasking, which allows independent threads of control to run cooperatively. The author has used multitasking in his 16-bit Pyrgmy Forth and in its variants. This paper discusses some benefits of multitasking. The examples are for Pyrgmy, but the principles apply to other FortHS. If you don't already use multitasking, this article will break the ice and get you started.

Forth and Functional MRI by Ronald Kneusel
Functional Magnetic Resonance Imaging (fMRI) is a new branch of biophysics which studies brain function. In this article we will look a little at what MRI is and what the word "functional" means in regards to MRI. Then we will take a closer look at a Forth program developed for the analysis of functional MRI data.

The Problem with Buffers by Hugh Aguilar
It is commonplace that a program accept data and do something with that data. But data often comes in bursts, faster than the program can process it, so we need to buffer the data in memory. But buffers can suffer from a variety of constraints, not least of which is the amount of memory. And what if the data must be in contiguous addresses?

Reed-Solomon Error Correction by Glenn Dixon
Reed-Solomon is a type of forward error correction used in disk drives, CDs, satellites, and other communication channels. Redundancy is added to data before sending. At the destination, this data reveals if an error has occurred and may allow correction, reducing the necessity of retransmitting data.