Conference Abstracts

The following abstracts are from presentations made at the 1983 Rochester Forth Applications Conference, June 6–9, 1984. Many of these presentations appear as papers in the Proceedings of that Conference, published by the Institute for Applied Forth Research, Inc.

Computer Music Programming Environments

David P. Anderson

Computer Science Department University of Wisconsin Madison, WI 53706

Computer-based digital sound generation systems offer a wide range of new musical possibilities. Some applications for such systems are: (1) algorithmic composition and performance, possibly including random or human-specified elements; (2) programmed interpretation of human-composed works; (3) use of the computer as a human-playable instrument whose "notes" may include process invocations.

Technological advances have made the hardware capabilities for computer music affordable. Still lacking, however, is a widely available programming environment in which to do computer music. This paper explores the design and implementation issues involved in such an environment.

Among the needed features are:

(1) Concurrent programming with interprocess communication and synchronization.
(2) Accurate timing control of output events, independent of the CPU usage in computing the parameters of the events.
(3) A layered interface to the synthesis hardware providing a more powerful "virtual synthesizer."
(4) Programming abstractions for computer music. These abstractions are crucial for the convenient specification of musical structure and nuances.
(5) A fast, interactive user interface.

Forth provides the means for specifying and implementing these capabilities in a potentially portable way. We will outline a design for a Forth computer music programming environment, and will give some details of its implementation.

FORTH Processors in the Hopkins Ultraviolet Telescope

Ben Ballard, Bob Henshaw, and Tom Zaremba

The Johns Hopkins University
Applied Physics Laboratory
Laurel, Maryland

Engineers at the Johns Hopkins University Applied Physics Laboratory have designed and built two computers which will control and monitor the Hopkins Ultraviolet Telescope, a 1986 Space Shuttle experiment. These computers implement a microcoded FORTH nucleus in a word-addressed AMD 2900 series bit-slice architecture. All programs for them are written in FORTH, which takes the place of assembly language in this architecture.

The first part of this presentation briefly describes the scientific objectives of the Hopkins Ultraviolet Telescope and outlines the requirements which drove its processor designs. These requirements include speed, power, cooling, size and radiation tolerance. This introduction leads into a discussion of the computer hardware designs and special features incorporated to meet some of the more unusual requirements. The custom interface between the telescope's ultraviolet detector

and the computer system is described in some detail.

The second half of the presentation describes the way FORTH is implemented and used in the telescope. The microcoded inner interpreter, which forms the fetch execute loop of the processors, is presented in depth, as well as the dictionary structure which supports the outer interpreter. The microprogrammed concurrency routines are described briefly to demonstrate the multitasking features of the computers. The presentation concludes with some illustrative programming examples taken from the telescope's application software.

The VT52 — Terminal Emulation in Forth

James Basile
Department of Computer Science
Long Island University — C. W. Post Campus
Greenvale, NY 11548

This paper explores the general principles of terminal emulation and in particular focuses on an emulation of the DEC VT52 terminal. The use of state tables to process I/O is examined and general techniques for implementing an emulation in Forth are developed.

Four Vocabulary Tools

Robert Berkey
Dysan, Inc.
455 Bob Jones Ln.
Scotts Valley, CA 95066

The concept of VOCABULARY is decomposed into two elements: the search order and the ordered list. :SEARCH-ORDER is a defining word that defines search orders. GROUP is a defining word that defines ordered lists. An additional vocabulary access word, THE, is described for access to words outside the CONTEXT. GROWING is a name preferred to CURRENT.

Correction of Systematically Induced Instrumentation Errors on Streak Camera Data

Robert Boni
Laboratory for Laser Energetics
250 East River Road
Rochester, New York 14623

At the Laboratory for Laser Energetics (LLE), streak cameras are implemented to time resolve sub-nanosecond laser pulses. These streak cameras have three main components: the streak tube that time resolves the laser pulse, the intensifier which amplifies the resolved pulse, and the Optical Multichannel Analyzer (OMA) which digitizes the output for numerical analysis. The non-linear signal response of the streak tube and its signal saturation level cause intensity distortion of the data, after which the intensifier induces pincushion distortion (non-linear spatial stretching) of the data. Software algorithms are used to remove these distortions. These algorithms, which will be discussed, provide an efficient means for correcting these distortions.

Implementing FORTH on a New Processor FORTH for the 65816 and 65802

John Bowling
Starlight Forth Systems
15247 35th St
Phoenix, AZ 85032

This paper covers the creation of an assembler for a new processor in FORTH and generation of a FORTH meta-target compiler using that assembler. The techniques used can be implemented for any processor on any host system with FORTH. The host employed for this task was an OSI C8P-DF running fig-FORTH for the 6502 under OS65D DOS. The scheduled targets are OSI, Apple, Atari, and Commodore, each with a 65802 processor installed in place of the 6502 processor.

The reader will be shown how any present system using a 6502 or a 65002 can be upgraded by installing a 65802 processor in place of the 6502 and installing the newly created FORTH. With no other hardware changes, the system speed will more than double, and the space taken for CODE words will be reduced by nearly one-third.

Mealy Machines in FORTH

John Bowling
Starlight Forth Systems
15247 35th St
Phoenix, AZ 85032

This paper covers the implementation of a Mealy (State Active) Machine structure in FORTH. The Mealy Machine resides in a present state, and performs operations based on the present state and incoming parameters. The operation performed can change the state of the machine as it executes the desired function.

An example using the Mealy Machine as a multi-layered CASE: array will be given. This operation implements several functions, each with its own interpretation of the keyboard. It may reside as a turn-key overlay to FORTH, and allows Spread Sheet, Data Base management, Text Editing, FORTH editing, Help, etc. to be operated from what appears to be a single high level package.

From power up, a user selects an operation by a single keystroke. The user is then placed in an operating structure that allows temporary entrance to another operational package, return to previous or entrance to an additional, etc. all controlled by the Mealy Machine. Transfer may save the status of the existing function, and upon return enter it where it left off. Data may be transferred when going into another package, or data may be shipped off to another function to be manipulated, somewhat transparently, allowing spread sheet manipulation of numbers while editing in the data base.

Command Completion in Forth

Mitch Bradley
Sun Microsystems
2550 Garcia Ave.
Mt. View, CA 94043

Command completion is a technique for speeding-up user input to a computer. Instead of having to type entire names, the computer accepts any unambiguous initial substring of a command name as an abbreviation. The computer then completes the command name for the user. If a substring is ambiguous, the computer can tell the user all the valid choices.

Command completion has been used in a number of operating systems, notably TENEX and TOPS-20. It has proven to be a very well-liked feature of these systems.

Command completion fits in well with Forth, since most words that the user types are expected to be already present in the dictionary. The implementation of command completion is surprisingly simple.

This paper discusses the concept and use of command completion, and presents the author's implementation of command completion for Forth 83.

Structured Data with Bit Fields

Mitch Bradley
Sun Microsystems
2550 Garcia Ave.
Mt. View, CA 94043

A structured data item is a data item composed of several subfields of various types, analogous to the "struct" declaration of C or the "RECORD" types of Pascal and Modula-2.

Address Space Unification and Forth

James C. Brakefield
Technology, Inc.
5803 Cayuga
San Antonio, TX 78228

The following talk gives a philosophical background for the Forth software system. It places Forth within a larger context of the programming universe. I am interested in how this is viewed by the active, experienced Forth community.

The address interpreter can be generalized so that direct threaded code and indirect threaded code can coexist, even to the point that a given code string contains both. A second modification of the address interpreter permits compaction of the addresses so that a P-code style is possible. This coexists with the aforementioned direct and indirect threaded code.

The resulting interpretation of Forth code strings is that the Forth primitives are reserved addresses and that addresses are executable objects. Philosophically, the programmer is using addresses as op-codes. Subroutines are known by their addresses just like primitives. Subroutines have the same stack interface as primitives (op-codes) so that conceptually they are no different and thus may be abstracted in the same way.

The various concepts of advanced computer science and the artificial intelligence community can be mapped directly into the Forth context. Thus this extended Forth provides a foundation on which to build the rest of computer science.

Examples:
—Lisp is Forth with 1 plus 1 addressing,
—Address code strings are a possible internal parse tree representation for compilers,
—Forth has the most general extensibility of any language,
—Type checking and overloading are simple modifications of the text interpreter.

Status Threaded Code

Bob Buege
RTL Programming Aids
10844 Deerwood SE
Lowell, MI 49331

RTL is a Relocatable Threaded Language which I developed to overcome some of the weaknesses of FORTH. Because of the requirements which I demanded from RTL, I was forced to invent a new class of threaded languages which is more flexible than languages based on either direct threaded code or indirect threaded code. The purpose of this paper is to describe this new type of threaded code which I call status threaded code and show how it was used to advantage in the creation of RTL.

MA2301 FORTH: Modular Software to Complement Modular Hardware

Gordon Cook
National Semiconductor
2900 Semiconductor Dr.
Santa Clara, CA 95051

National Semiconductor's new MA2XXX Macrocomponent™, the ROM-based MA2301 FORTH module, reflects the modular theme of the MA2XXX family. A description of the features of the FORTH Macrocomponent™ is followed by a brief introduction of the compact, lower power, MA2XXX Macrocomponents™ utilized in the specific applications addressed by the paper.

A section on specific applications of the MA2301 FORTH module includes a program for a 300 BAUD MODEM, and an ANALOG INPUT unit, both companion Macrocomponent™ units. A program for use with an LCD display and keyboard interface further demonstrates the MA2301 FORTH module's flexibility.

A summary of the MA2301 FORTH module's capabilities concludes the paper, and a detailed appendix is attached.

The Use of FORTH in the Instruction of Introductory Robotics

Alan J. Cotterman, Daniel M. Willeford, and James E. Brandeberry
Dept. of Computer Science
Wright State University
Dayton, OH 45345

At the Digital Control and Robotics Laboratory at Wright State University, students use an implementation of fig-FORTH to design a teach mode control program for modified toy robot arms. This program allows the arm to be taught multiple trajectories of up to 100 points (unique arm positions) and then permits repeated execution of these trajectories. The program successfully realizes real-time position monitoring, motion-execution algorithms, management of the trajectory database and features a menu-driven, user-friendly operation.

The hardware environment consists of the host PDP-11/34 minicomputer operating under RSX-11, and four LSI-11 microprocessor workstations. TASK4TH, a standalone derivative of fig-FORTH, is down-line loaded from the host into the workstation and is capable of supporting peripheral equipment which includes serial and parallel I/O and also analog-to-digital and digital-to-analog conversion. Toy robot arms with five degrees of freedom are connected to the workstations and have been modified to operate under computer control. Position detection is accomplished through analog-to-digital conversions on the outputs of Hall-effect sensors (mounted on the axes of rotation) and arm motion is accomplished by actuating solenoids via the parallel port.

Solution Introduction to Continuous-Flow Analyzer by Rotary Valve and Robot Arm — a Forth Application

Don C. Cox, Frank W. Kerner, Leonard C. Jones, and William B. Furman
Center for Drug Analysis
Food and Drug Administration
St. Louis, Missouri 63101

Our laboratory monitors the quality of drugs produced in the U.S.A. We use commercially built, automatic continuous-flow analyzers to measure the amount of drug in individual tablets or capsules. Each tablet or capsule is dissolved in a known volume of liquid. Small portions of these sample solutions (and of standard solutions at intervals) are manually poured into disposable cups and placed on the carrousel of a commercially built, automatic sampler device. The automatic sampler mechanically turns the carrousel in steps and inserts a probe sequentially into each cup; each solution is pumped through tubing from the probe into the automatic analyzer. Between each sample or standard solution, the sampler lifts the probe briefly into the atmosphere and then inserts the probe into blank solvent. In this way, an air-segmented stream is pumped into the analyzer in the sequence: wash solution, air bubble, sample solution, air bubble, wash solution, air bubble, sample solution, air bubble, etc. A standard solution of known drug concentration is used to calibrate the automatic analyzer at regular intervals in the sequence. In a typical hour's work, an operator must pour 40 to 120 solutions into cups and place them on the carrousel; from five to 15 analyzers are in use on a given day.

We want to eliminate the manual transfer of prepared sample and standard solutions to the small cups required by the commercial automatic sampler. To do this, we replace the commercial sampler with a five-port rotary valve and a robot arm. The robot arm sequentially inserts the probe directly into the same containers of liquid that were used to dissolve the individual tablets or capsules, thus eliminating manual transfer of solutions. The rotary valve sequences the flowing streams properly for introduction of air, sample solution, or standard solution to the rest of the automatic analyzer. Forth programs control the valve and position the robot arm over a rectangular matrix of containers for sequential introduction of prepared solutions to the analyzer in real time.

Extensions of FORTH for Functional Programming

R. D. Dixon, W. M. Edmundson, R. D. Franklin, and J. L. Sloan
Dept. of Computer Science
Wright State University
Dayton, OH 45345

In order to provide facilities in a FORTH environment similar to those in Backus' FP, an interface was written between Ray Duncan's Z-80 FORTH and a functional language developed at Wright State University called BADJR. BADJR provides automatic storage allocation, garbage collection and compactification, unlimited precision numbers, strings and sequences. BADJR differs from FP in that functions may have several input and output parameters as well as local variables which may be assigned a value exactly once.

The result is an extended FORTH, called FORJR, which is pleasant and easy to use. Recursion is supported and stack reduction is done for tail recursion. This system is experimental and not meant for other than testing ideas but it has led us to some further proposals for a new FORTH model which incorporates the functional support features in its kernel. Such a FORTH would have much more flexible conditions on the addition and deletion of dictionary entries.

The Smallest Outer Interpreter

R. M. Dumse
New Micros, Inc.
808 Dalworth
Grand Prairie, Texas 75050

When an application program is target compiled, the ability to use the interactive nature of FORTH for debugging is lost with the removal of the dictionary and outer interpreter. Creating an "open" system by including the complete outer interpreter and dictionary headers in the final program would defeat the purpose of target compilation. In most instances, the added overhead to the compiled kernel would be unacceptable. A happy medium does exist, however, as evidenced by the "Micro Monitor" of the R65F11 Single Chip FORTH Based Microcomputer. This small program included in the kernel of the chip allows interactive use of all kernel and user defined words, even in target compiled systems. The details of this and other enhanced versions of "small outer interpreters" are the subject of this paper.

Operating System Services in Forth for VAX/VMS

David L. Forster
Telelogic, Inc.
One Kendall Square
Cambridge, MA 02139

The Forth programming language has been extended to facilitate access to system services provided by the VMS operating system on the DEC VAX super minicomputer. The complete interactive access to system services thus provided allows quick and efficient manipulation of system resources and information. Complex data structures, which are used by some system services, can be built interactively in Forth. Fields within these structures have been made directly accessible by name, through the use of some of the C programming language's "Struct" concepts. Second order defining words, which are defining words that build defining words, have been used to automatically build common system, service data structures, thus eliminating the task of building these structures repeatedly by hand. These features eliminate many of the difficulties in working with system services.

A Forth-Based Operating System for Embedded Real-Time Applications

Harvey Glass and Ted Neff
College of Engineering
University of South Florida
Tampa, FL 33620

Forth has been used extensively in the implementation of small specialized applications — particularly in real-time control and in interactive packages. This paper discusses the implementation of

a real-time operating system written principally in Forth.

The system was originally designed to operate in a high speed, high volume, time critical communications environment. It furnishes a set of sychronization primitives and real-time clock support. The development environment provided by the operating system was designed to allow interactive simulation and debugging as well as in-circuit emulation of the prototype application.

Although the system was used initially to perform a highly specialized function, it has proven to be both transportable and useful in a variety of applications. There is no doubt that Forth contributed greatly to the generality and transportability of the system.

Forth Coding Conventions

Kim Harris
Dysan Corporation
455 Bob Jones Lane
Scotts Valley, CA 95066

A set of conventions for Forth programming will be presented. The conventions include spacing within definitions, screen layout, interface and stack comments, and indentation.

Controlling a PDS Microdensitometer with FORTH

Arne A. Henden
Goddard Space Flight Center
and
Systems & Applied Sciences Corp.
5809 Annapolis Road
Hyattsville, MD 20784

A PDS Microdensitometer is an instrument with a motorized X-Y stage and a light source/detector mounted perpendicular to the stage. It is used to digitize photographic plates. We have written a FORTH program to control the stage motions, acquire the A/D data, and store the data on magnetic tape. Unique aspects of this control program will be discussed.

Computerized Process Control for Food Science Students

George Houghton
Cornell University
Food Science Dept.
Ithaca, NY 14853

Since students in Food Science can expect to work in such diverse lines of work as food processing, food engineering, quality control, product development, etc., they must be familiar with a wide range of scientific and technological fields. In the past few years this has come to include aspects of computer science. We are in the second year of developing a laboratory module to familiarize students with computerized process control. Our objective is to introduce this topic without requiring a background in computer technology. In this module, a small computer is programmed to control a simple, but realistic model process, namely a hot water heater. Since the software that controls this process is written in FORTH, students are given a short introduction to the language.

Forth Meets Smalltalk

Norman D. Iverson and Charles B. Duff
KRIYA
505 N. Lake Shore Dr.
Chicago, IL 60611

Several advantages are inherent to an "object-oriented" approach to programming languages: the source code resulting from use of such a system offers a clear insight into the underlying design; pathological couplings between the various parts of a program are minimized; and the close relationship between data structures and their associated operations encourages users to solidify their designs before any coding begins. Previous approaches to object-oriented systems have sacrificed efficiency for generality, often making them impractical for production use. We have implemented a set of extensions

to Forth that permit Class definition/instantiation and Subclass inheritance in the manner of Smalltalk and Simula. These extensions take advantage of Forth's compile time behavior to preserve runtime efficiency while providing many of the benefits of an object-oriented approach.

Our solution allows the following: definition of classes with hierarchical class/subclass inheritance; definition of methods for such classes triggered by selectors and messages; instantiation of classes as objects that associate private data areas with the owning class's methods; and the capacity to build a class's private data from previously defined objects, thus allowing the creation of nested data structures/defining words, previously impossible in Forth.

We will describe our formulation of the problem as well as design and implementation considerations. This work grew out of our creation of a Forth development environment on the Apple Macintosh computer.

Forth as a Development Tool for Micros

Man Chor Ko
Ascent Inc.
1983 Landings Dr.
Mountain View, CA 94030

Ascent Inc. is a software development company specializing in microcomputers. We are currently using Forth to develop a programming language for micros known for its graphics, and a productivity software package for lap-size computers. This talk will summarize our experience in using Forth as a professional development language for team projects.

Development of OMNITERM 2, a MS-DOS Communications Program in FORTH

David J. Lindbergh
Lindbergh Systems
49 Beechmont St.
Worcester, MA 01609

From May 1982 to March 1984 my principle project was OMNITERM 2, a highly capable general purpose communications, file transfer, and terminal emulation program for the IBM PC (and soon other MS-DOS machines). It is written under Miller Microcomputer Services MMSFORTH. I'll briefly describe the program itself (I don't intend to sell it, just give a quick overview) and my decision to use FORTH for its development.

I'll talk about the advantages and disadvantages I found with FORTH for developing such projects (the program runs about 80K of object code, and was about 300 blocks of FORTH and assembler code) including documentability and readability, ease of transfer to other machines, problems with 16 bit address spaces, etc.

Poster Session: Color and Sound for the IBM PC in FORTH

David J. Lindbergh
Lindbergh Systems
49 Beechmont St.
Worcester, MA 01609

The IBM PC has the capability to emit sounds through a software-controlled speaker. I have written some routines that allow me to specify the frequency and duration of any desired tone very easily in FORTH. The frequency is computed in Hertz to 7 digits of precision using integer math, and the duration is controlled by a heartbeat interrupt routine.

The IBM PC can also be equipped with a Color/Graphics card supporting the 16 foreground colors and 8 background colors, plus blinking. I have some very simple and elegant FORTH words that allow me to say things like "RED BLUEB" for red letters on a blue background, or "GREEN BRIGHT BLINK YELLOWB" for bright green blinking letters on a yellow background. Normally one must look up the desired colors from a table and specify the codes found, but FORTH allows me to simply type the desired color.

Number Crunching with 8087 FQUANs: The Mie Equations

Ferren MacIntyre
Center for Atmospheric-Chemistry Studies
Graduate School of Oceanography
University of Rhode Island
Narragansett, RI 02882-1197

By long-standing tradition, FORTH uses scaled-integer arithmetic in preference to floating-point operations, principally because of the inefficiencies of the latter. However, there are operations in which half of the possible precision is lost for any scaling. In addition, the appearance of the 8087 numerical co-processor has removed the stigma of inefficiency, and we have reached the point anticipated by Charles Moore, at which it becomes preferable to use floating-point operations.

Taking the IBM PC BIOS single-precision routines as baseline for the heavily numerical Mie equations, the 8087 is 115 times faster. Precision increased from one significant figure (round-off error dominates in the recursive calculations) to complete agreement with published 6-figure values.

Because arithmetic and stack operations are fast on the 8087, while 10-byte stores and fetches are relatively slow, it pays to design algorithms which keep operands on the 80-bit-wide, 8-word-deep 8087 stack. It is, for instance, possible to replace two complex numbers with their sum and difference, without leaving the stack.

The net result is an algorithm which produces a result in 15 minutes, compared to all day in the queue for a 1-second run on an available CRAY-1 supercomputer.

Implementing Forth on the NCR/32 Chip Set

Michael L. McBride
NCR Microelectronics
Colorado Springs, Co.
Greg Bailey
Athena Programming
Santa Barbara, Ca.
Mary Anne Ryan
NCR Microelectronics
Colorado Springs, Co.

The NCR/32 chip set is a high speed, full 32 bit, externally microprogrammable microprocessor family that is an excellent vehicle for implementing a powerful FORTH computer. The chip set will be discussed in detail. The particular architectural features that enhance a FORTH implementation will be examined. Finally, the performance of a FORTH implementation will be discussed.

Kitt Peak Multi-Tasking FORTH-11

Thomas E. McGuire
Health Systems International
100 Broadway
New Haven, CT 06517

An overview of the Kitt Peak National Observatory (KPNO) real-time environment is presented. The eleven optical telescopes at Kitt Peak employ a variety of experimental instruments, including Fourier Transfor Spectrometers and large array digital cameras, that impose a diverse set of real-time requirements. Presented next will be a historical perspective of FORTH at KPNO over the last decade outlining the evolution of the embryonic version of FORTH on an 8K byte minicomputer to the current KPNO implementations on mainframe (CDC), mini (Digital VAX-11 and PDP-11), and micro (Digital LSI-11) machines. The real-time attributes of FORTH in general and KPNO multi-tasking FORTH-11 in particular are discussed. Some details of the implementation of FORTH-11 are presented to reveal the power of this real-time programming environment. Finally, the KPNO CCD camera system will be used to illustrate a complex real-time application of multi-tasking FORTH-11.

A FORTH Profile Management System

John Michaloski
National Bureau of Standards
Industrial Systems Division
Washington, DC 20234

A new approach to program management is presented called Profile Management (PM). Differing from conventional file systems, PROFILE not only handles those problems of off-line source code management, but extends this concept to deal with the on-line status of the machine. The basis of this new approach is the "profile", i.e., a partition of source code on the disk. Each profile names a section of code that can be used throughout the loading process, plus allowing various forms of status information to be accessed via commands within the Profile Management system. Further, each profile can be subdivided into smaller partitions that themselves act as an individual profile or as a part of the parent profile. These subdivisions offer a flexible mechanism for loading entire programs or just individual components. In addition, the profile management system is only a stepwise upgrade from current Forth source block management systems. All profiles are embedded as a commenting structure so that PM can operate either on-line or off-line to extract loading information. On-line PM produces named profiles in the dictionary with status pertaining to disk block location, load status, and a pointer to the name on disk. Off-line PM does selective code generation through a query system to create load blocks for the user. The system requirements of this package are approximately 2K memory and 30 source blocks. Finally, source code and examples will be detailed to illustrate how to fully exploit the power of a PROFILE Management system.

Multiple Mirror Telescope Coalignment and Cophasing Software Control System

J. W. Montgomery
Multiple Mirror Telescope Obervatory
Tucson, AZ 85721

The Multiple Mirror Telescope (MMT) located at Mount Hopkins, 64 kilometers south of Tucson, Arizona, is a demonstrated revolutionary concept in telescope design. Instead of having a single large monolithic primary mirror, it employs a cluster of six 1.8-meter Cassegrain telescopes and computer controlled transfer optics to focus the incoming light at a common plane. The telescopes are supported by an altitude-azimuth mount and housed within a four story co-rotating building. Without the current advancements in computer technology the MMT would not be feasible. This paper discusses the successful use of the FORTH language for orchestrating open and closed loop coalignment as well as open loop cophasing of the six telescopes.

Forth as a Design Tool

Gary Nemeth
Hampton Corporation
20800 Ctr. Ridge Rd. #229
Rocky River, OH 44116

In this application, Forth was used to design an 8085 assembler program. Half of the Forth program simulated the operating environment, and half was analogous to the desired 8085 program.

The desired 8085 program was a communication protocol for a controller computer. Already coded in assembler, the controller sequenced a small group of factory tools. The communication protocol was to be added for strategic coordination. The author had never written such a protocol, was not normally on location, and was new to the project.

All aspects of the controller system were simulated in Forth: the existing timer modules, interrupt modules, the nature of the 8085 instructions and assembler, the USART chip, the ROM/RAM aspect of the final product, the new calling programs which would be required, and the symmetric protocol modules of the other network computers. Thus Forth was used to avoid a duplicate development system or being on location.

The Forth program evolved expeditiously. The design process was interactive, and thus more efficient than pencil, paper, and pseudo-code. Quickly rewritten in the 8085 assembler language, the program was squirted into the controller computer.

A Brief Note on the Kuiper Airborne Observatory C141 Submillimeter Spectroscopy

Hans Nieuwenhuyzen
University of Utrecht
Grunoplantsoen 10
Bunnik 3981 GT
The Netherlands

In the 1970's a number of different molecules were detected in the interstellar gas clouds in our Galaxy. They revealed unexpected gas components in the interstellar clouds, comprising some of the largest known galactic objects.

One of the regions where interesting molecular information can be found is in the difficult to access sub-mm to mm region (due to the very high radio-frequencies, e.g. 260 Ghz for CO(2-1) transition, and to the low transmittance of the earth's atmosphere).

In 1974, the European Science and Technology Center (ESTEC) and the Utrecht Observatory in Holland started a collaboration to develop a suitable spectrographic sub-mm and mm instrument package and to do observations from high ground-based observatories and from the Kuiper Airborne Observatory. The following is a note on how the language FORTH was chosen for the control and data-reduction program. It is presented as a Case Study on the use of FORTH in an interactive, highly demanding, environment.

Complex Integer Arithmetic in Forth

Vic Norton
Department of Mathematics and Statistics
Bowling Green State University
Bowling Green, Ohio, USA 43403

Every Gaussian integer, $z = m + ni$, can be uniquely represented as a base $2 + i$ numeral,

$$z = a_k \ldots a_1 a_0 = a_k (2 + i)^k + \ldots + a_1 (2 + i) + a_0$$

with digits a_j from the set of "quints" 0, 1, i, −1, −i. We explore the arithmetic of this quinary positional system in FORTH by setting up a Q-stack (for quinary numerals) fed by the word

Q (-- quinary numeral ,) : put a quinary numeral
from the input stream onto the Q-stack.

The words Q+, Q−, Q*, QNEGATE are defined to do arithmetic on quinary numerals and the FORTH system for integers is further duplicated by the words QDUP, QDROP, QSWAP, Q., Q@, Q!, QVARIABLE, etc. The main problem is this: how does one define Q/MOD and is this even possible?

Using Native Machine Code Analogs of Interpreted FORTH's Elements for High Performance

Walt Pawley
Sun Studs, Inc.
2635 Old Highway 99 South
Roseburg, OR 97470

Sun Studs is a small timber products company which develops much of its own technology for process control — most notably nonanthropomorphic computer controlled 'robots'. The requirement for high performance has dictated the use of assembly language for the overwhelming majority of programs used in these systems. To investigate its feasibility in real time work, it was decided to develop a FORTH for the Perkin/Elmer computers Sun Studs uses. The kernel was adapted from a microcomputer version of FORTH and written using a macro processor to provide highly readable, but still interpreted, code. Before this was completed, however, it became clear that native machine code structures could be substituted for the interpreted structures allowing FORTH code to execute

directly,. It was only necessary to change the code-generating macros and the definition of a few of the kernel's words.

Several new words have been added to support the gamut of environments used at Sun Studs. The result is a fast, compact FORTH that supports both 16 & 32-bit systems operating in either user or executive modes. It also works with standard files from these systems, rather than defining its own mass storage structure. Unfortunately the jury is still out on the applicability of such a FORTH in our real time systems.

Nicolet DXFTIR: Real-Time, Multi-Tasking FORTH and Other Tricks

Joel V. Petersen
Nicolet Instrument Corporation
5225-1 Verona Road
Madison, WI 53711

The software package developed for the Nicolet DX series FTIR spectrometer represents the largest single FORTH program ever developed by the author. With the object code extending over 64K of 20-bit words, the program spans an enormous range of concepts. The heart of the DXFTIR program is a simple real-time multi-tasking FORTH that was developed for the Nicolet 5-MX, a ROM-based FTIR spectrometer. Under control of this multi-tasking supervisor, the DXFTIR program can collect and process a 8192 word data array once a second while displaying one spectrum with full ROLL and ZOOM function and plotting a second spectrum.

The operator interface starts with a menu-driven mode with single keystroke operation and numerous help messages and culminates with a programmable command language complete with a screen editor. With a configuration program, the operator can customize the DXFTIR program according to which spectrometer bench, which plotter, and how much memory he has. The operator can also write their own programs in FORTRAN, PASCAL, FORTH, or assembly language to be run as overlays to the DXFTIR package. Finally, the operator can access the FORTH vocabulary of the DXFTIR package and add his own commands to the command language.

Language Trade-Offs for Real-Time Programming Applications

Richard Poulo
Control Automation
P.O. Box 2304
Princeton, NJ 08540

We examine the advantages and disadvantages of FORTH vis a vis conventional programming languages, with Pascal chosen as a representative. Some properties of these languages can be compared by identifying the part of the software life cycle to which they are relevant, while some properties can only be measured by other criteria.

We also examine two similar industrial vision systems that are programmed in Pascal and FORTH and discuss the requirements and constraints of each system. These requirements originate from both technical and non-technical considerations but all have an impact on the choice of language. In particular the requirements determine the relative priorities of the properties of the candidate languages.

Historically the vision system programmed in Pascal was developed first and the FORTH system two years later. The change in product emphasis that led to a change in language is used as an example of how to apply product requirements to make a language choice.

High Speed Image Capture and Image Generation

Tom Sargent
Io Incorporated
1806 W. Grant Rd. #105
Tucson, AZ 85745

A modular image capture and graphic display system has been built using bit slice technology. Used as a graphics device, a dual, 32-bit, microcoded graphics engine copies, scales and rotates pre-defined patterns into a 1024 x 1024 color graphic display memory at 6 million pixels/sec. Used in the

frame capture mode, the system can accept a data stream at up to 96 million pixels per second.

The image capture/generation hardware is controlled by a VME bus computer running multiple MC68000 microprocessors programmed exclusively in FORTH.

The first graphics application of this system has been a Computer Aided Design machine for printed circuit board design. The frame capture capability is being used in cartographic and medical imaging applications.

User's View of FORTH for the Study of Optical Thin Film Coatings

Ansgar Schmid and Mark Guardalben
Laboratory for Laser Energetics
250 East River Road
Rochester, NY 14623

At the University of Rochester's Laboratory for Laser Energetics, FORTH is the language of choice for the automated control of experiments designed to investigate the interaction of intense radiation with thin film materials. A DEC PDP 11/23 is used in conjunction with a Chromatics Color Graphics computer in the acquisition, reduction, and subsequent color display of data. All processes involved in the acquisition of data are automated via a CAMAC serial highway.

A FORTH Application to Infrared Astronomy

Justin Schoenwald
Dept. of Physics and Astronomy
University of Rochester
Rochester, NY 14627

A PDP-11 is programmed for data collection from a 32 x 32 InSb array with charged coupled device readout. The same computer serves to process and display images at the telescope using a Peritek VCH-Q graphics board. Extensive image processing is performed in the lab, where coding of FORTH above 64K is accomplished with the LSI 11/23 extended memory feature.

A VAX Implementation of the FORTH 79 Standards

William Sebok
Dept. of Astrophysics
Princeton University
Princeton, NJ 08544

A public domain implementation of Forth has been written for the VAX819 super-mini that runs under 4.1 and 4.2 BSDUNIX829. It has been running now for about two years at the Astrophysics Dept. at Princeton and is used for image processing. It follows the 79-Standard except: 1) entries on the parameter stack are 32-bits wide, 2) addresses are 32-bits (rather than the demanded 16-bits) wide, 3) certain escape sequences beginning with a backslash are recognized in the printing word ."....."

Some extensions to the 79-Standard are: 1) A character string stack, with a full set of string operators. This also makes manipulating Unix file names infinitely easier. 2) A floating point stack. 3) A set of Unix interface words.

Colon definitions are compiled as a series of bsb, bsbw, or jsb instructions (the shortest one that will reach) rather than as a list of pointers. When a word defined by the ICODE operator is compiled its code is stuffed in-line rather than referenced. Number references are compiled as the shortest of the many possible instructions to push that number onto the stack.

TASK4TH — A Multi-Tasking FORTH Workstation

John L. Sloan
Department of Computer Science
Wright State University
Dayton, OH 45435

TASK4TH is a standalone multi-tasking FORTH system derived from the PDP11 version of the public domain fig-FORTH. TASK4TH provides facilities for interrupt handling, process synchronization and message passing.

TASK4TH is down-line loaded from a multi-user PDP-11 minicomputer to a single user LSI-11/2 microcomputer. The LSI-11 target system requires no native software other than the standard micro-coded ODT monitor, and no native peripherals other than a console terminal and a serial communications line to the host PDP-11.

Once TASK4TH is loaded, the target system may be used in either of two modes. In terminal mode, the target system can be used as a standard terminal to access the host system. In FORTH mode, the target system enters the FORTH interpreter, and may request FORTH screens from the host. A file server utility on the host reads screens from a screen file on a large shared hard disk and passes the data to the target over the serial line.

Each TASK4TH user's logical FORTH screen file is split into two physical files. The first file is common to all users on the host system, while the second file is unique to each TASK4TH user. The physical split is transparent to TASK4TH.

TASK4TH has been in use by under-graduate and graduate students for nearly a year. Projects include inter-processor communication, real-time grahics, and control of various robot arms.

SPHERE: An In-Circuit Development System With a Forth Heritage

Evan L. Solley
Infosphere, Inc.
4730 SW Macadam Ave.
Portland, OR 97201

The development of embedded realtime applications for industrial and scientific control and measurement is a complex task demanding an effective toolkit and development methodology. Interactive development within the domain of the execution vehicle has been shown to lower costs and development time.

SPHERE defines the minimum required functionality of an idealized, abstract engine for solving realtime problems. The current implementation, available on the most popular microprocessors, utilizes concepts from Forth to efficiently realize that functionality.

This paper describes the features of SPHERE that make it an ideal choice for system integrators solving industrial control problems. The Forth community will be especially interested in:

1. An ergonomic user interface,
2. A pre-emptive, priority-driven multitasking executive,
3. Advanced data types for realtime control solutions,
4. Conditional evaluation capabilities that enhance code portability,
5. An extensible mass storage interface, and
6. A ROMmed, "silicon software" package.

An Approach to a Machine-Independent Forth Model

*N. Solntseff and J. W. Russell**
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S 4K1

This paper is a progress report on the research work described in [1] and [2]. The earlier papers introduced an Abstract Forth Machine (AFM) and described an instruction-set architecture specifically designed for the Forth system. The aim of the project is to provide a description of the Forth Nucleus in terms of the AFM code, henceforth called F-code, instead of the machine language of the computer that hosts the Forth system. Once this is done, the F-code routines can be translated by a macro processor into the assembly language of any host computer for further processing by a standard assembler. The use of F-code for nucleus words and high-level Forth for the remainder of the Forth-system kernel leads to a Forth model which is machine independent to a high degree. (Only terminal and disc I/O routines remain to be coded in machine language.) The effort needed to implement a Forth system on any host computer is significantly reduced because the F-code machine has fewer than 32 instructions for which macro definitions have to be written in host-computer machine language.

The paper describes the design and implementation of a macro processor suitable for converting Forth words expressed in F-code into assembly language for Z80 and MCS6502 micro-processors.

Preliminary results and experience gained with this approach will be discussed. As an example, it has been found that the use of the macro processor results in a Forth-system kernel which is 15–20% longer than that produced by the original Fig-Forth78 model.

* On leave from Niagara College of Applied Arts and Technology, Welland, Ontario L3B 5S2.

[1] N. Solntseff, "An abstract machine for the Forth system," Proceedings of the 1982 Rochester Conference (1982), pp. 149–155.

[2] N. Solntseff, "An instruction-set architecture for abstract Forth machines," Proceedings of the 1983 Rochester Conference (1983), pp 175–183.

A Break Point Utility for Forth

*N. Solntseff and J. W. Russell**
Unit for Computer Science
McMaster University
Hamilton, Ontario L8S 4K1

The goal of the work described in this paper is to provide the Forth programmer with a tool to examine the operation of high-level Forth words. The idea of such a tool was provided by Leo Brodie in [1] which describes a word BREAK to transfer control to a special interpreter. BREAK has to be compiled into a word being examined so that several recompilations are needed to examine any Forth word in a number of places. This approach precludes an easy way of stepping through a definition one execution address at a time. In the break point utility described here, the execution address of any word within a definition is saved and replaced by the execution address of BREAK. Whenever BREAK is executed, a special interpreter is available to examine the state of the Forth machine. Before execution is resumed, the break point is automatically moved to the next word in sequence, so that single stepping through an entire definition is very easy. The utility can be used to single step through any high-level Forth word, including those in the Forth-system kernel (provided that they are RAM resident), without the need for repeated re-compilations.

* On leave from Niagara College of Applied Arts and Technology, Welland, Ontario L3B 5S2.

[1] Leo Brodie, "Add a break point tool," Forth Dimensions, Volume V, No. 1 (May/June 1983), p 19.

Of Widgets and Clock Ticks

Dr. Michael Starling
148 Union Carbide Technical Ctr.
Building 770, P.O. Box 8361
South Charleston, W.VA 25303

Traditional programming is centered around a logical construct, the language. The program so generated must interact with the outside world in a time which is relevant to a human interface.

The programming of devices in the real world must often be based around the stringent timing concerns of the device, system, or experiment which the program knows as the real world. Real-time programming is much more concerned about time than is most other programming.

The device, or widget, which forms the visible heart of the real world interface is too often the central theme of discussion about the real time environment. Though a specific device may set the limits of a problem, several concerns form the nucleus of a real-time programmer's view of the world. This paper discusses the following elements of that view:

1. A real world interface
2. Time
3. Tasking as a means of simplification
4. Distribution as a means of simplification

HELLO, A REPTIL I AM

Israel Urieli
Sunpower, Inc.
225 Highland Ave.
Athens, OH 45701

REPTIL (a Reverse Polish Threaded Interpretive Language) is a Forth-like language which has been designed with the specific motive of being a viable alternative educational language to BASIC or LOGO. It is not limited to education, however, and can "grow" with the student over the entire range from computer literacy through interactive applications programming.

There are some significant departures of REPTIL from Forth in order to make it more palatable, consistent and pedagogically sound. Thus, the various structured constructs can be used either interactively, or within a verb definition. Furthermore, the defining verb is not limited to one level, and nested definitions are allowed. The prompt symbol includes a comprehensive line by line status report, including the number of elements on the stack, the radix base, and the various unclosed structures used. Syntax checking of the correct nesting of structured constructs is done throughout.

The basic set of fundamental verbs and structures have been designed for ease of readability and writeability, and the entire language is specified in terms of structured REPTIL algorithms.

REPTIL has been implemented on a 6502 based microcomputer using less than 8K of memory. The various utilities such as dissassembler, file storage, editor and high resolution 'turtle' graphics are currently being implemented.

Forth Machine Design Considerations

Christopher Vickery
Computer Science Dept.
Queens College of CUNY
Flushing, NY 11367

Two common approaches to the design of a Forth language processor are translation of a software model such as (Ragsdale, 1980) into hardware structures, and implementation of Forth language primitives as the microcode of a conventional processor augmented with a hardware stack. This paper discusses the strengths of these two approaches, and then explores the possibility of developing a "Forth Architecture" which transcends historical implementations and hardware details. Such an architecture must deal not only with compiler facilities and stacks, but also with such other issues as the structure of memory for code and data as well as the processors control flow mechanism. Furthermore, the paper also discusses multi-tasking structures for the architecture, justified by the argument that Forth is often used in process control applications and that such applications are often best developed as multitasking systems.

The paper concludes with a brief description of the architecture which has evolved from these considerations at Queens College and reports on our plans to build a machine which realizes this architecture.

Asynchronous Words for FORTH

R. G. Winterle — Institute for Materials Research
W. F. S. Poehlman — Department of Engineering
MacMaster University
Hamilton, Ontario L8S 4M1

The asynchronous word (A-word) has been developed to provide an extremely convenient way to program concurrent real-time interrupt service routines. By adding only three new words ("::", ";>" and "ACTION") to a programmer's vocabulary, A-words that support the full process model can be defined, quickly and easily. This includes the implicit operation (interrupt response code) plus up to (currently) eight process housekeeping command operations. All operations are user-definable in high level FORTH statements. The syntax model is:

```
:: <A-word name>
        [ interrupt service definition ]
;>      ( interrupt return )

INIT>   [ init housekeeping definition ]
START>  [ start housekeeping definition ]
   .
   .        [ up to eight process housekeeping definitions ]
   .

;       ( end of dictionary entry )
```

where INIT> and START> are mneumonic words that call ACTION to set up the housekeeping dispatcher. With this approach, these complex word definitions can be supported with all of the standard FORTH tools. Indeed, the A-word is a pure superset of the standard FORTH vocabulary. This paper deals with the implementation and evaluation of the above syntax model.

CCD Detectors and the Palomar 200 Inch Telescope

Barbara A. Zimmerman
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

The application of Forth to support real time data acquisition from instruments at Palomar using array detectors will be discussed. A brief history of Forth's support of such instrumentation will be given as a background for the use of Charge Coupled Device (CCD) detectors on the 200 in. telescope. Four instruments currently use CCD's at Palomar. Forth, running on a Digital Equipment Corp. PDP11/44 reads the detectors, stores the data on tape and disk, and displays the data on a Grinnell Image Display system. The Palomar Forth system will be described with emphasis on the latest instrument on the telescope, a ground based version of Space Telescope's Wide Field Planetary Camera, which uses four Texas Instrument 800x800 CCDs.