Conference Abstracts

The following abstracts are from presentations made at the 1983 FORML Conference, November 23-25, 1983, published by FORML, P.O. Box 51351, Palo Alto, California 94303, U.S.A. Asterisk denotes presentations without papers.

Modern Control Logic

Wil Baden
339 Princeton Drive
Costa Mesa, CA 92626

Newcomers to Forth from more primitive languages such as Assembler and Basic (and Fortran) find the control structures of Forth restrictive and awkward. So do immigrants from more advanced languages as C and Ratfor (and Ada and Modula-2). The logical structures of Forth were designed at a time when the structured programming revolution was in its infancy, and we have learned much since 1968 about the composition of good programs. Implementation is given here for: (a) multiple 'else if' without further nesting; (b) 'LEAVE' from BEGIN-loops as well as DO-loops; (c) 'EXIT' from DO-loops; (d) Short-circuit AND-test.

COMPILER and INTERPRETER Coroutines

Robert Berkey
2334 Dumbarton Ave.
Palo Alto, CA 94303

Previous Forth systems required the system to handle compilation of colon definitions. This paper presents a new word, COMPILER, which can be compiled into other words. The interpret loop is named INTERPRETER. The constructs [and] are implemented as active processes which effect the transition between the COMPILER and INTERPRETER coroutines.

A Variable-Precision Floating-Point System For Forth

Sidney A. Bowhill
Electrical Engineering Department
University of Illinois
Urbana, IL 61801

Scientific computation frequently requires the use of floating-point calculations. The precision needed for such calculations may vary considerably from one case to another. The time of execution increases rapidly as the precision increases, so fast operation requires the use of the minimum precision that the calculation requires. A system that can be compiled to any desired precision will overcome this problem. In this paper, the design criteria for such a system are discussed, and a complete system for the 6502 processor is presented as an example, capable of a precision from 2 to 15 bytes in the mantissa. Execution times are compared with other floating-point systems.

Transcendental Functions in Variable-Precision Forth

Sidney A. Bowhill
Electrical Engineering Department
University of Illinois
Urbana, IL 61801

Transcendental functions in a floating-point system with fixed precision are optimized for rapid calculation using approximate formulas which apply only to that precision. For a variable-precision system, algorithms are needed which are robust over the entire range of precisions encompassed by the system. Several approaches are discussed in this paper for trigonometric and exponential functions, and a set of optimized algorithms is demonstrated.

CS-105 Controller Monitors Synchro Angular Positions and Generates Three Orthographic Views of Rotating Object

Glenn E. Bowie
Kelly Johnson R&D Center at Rye Canyon
Lockheed-California Company
Burbank, CA 91520

Lockheed-California Company acquired its first CS-105 from Controlex in November 1982 for antenna testing. The author uses a home computer to model antenna range objects, and describes motions in three orthographic views. In August 1983, Martin B. Petri, CS-105 creator, added the author's logic to Lockheed's CS-105. A synchro transmitter and graphics terminal are used now with the CS-105 to model antenna positions. This paper shows how to use these devices to compare two ways of programming rotations about a fixed axis. In the first, rotor angle is incremented uniformly, and in the second body angles are incremented.

A Portable File System Interface for Forth

W. M. Bradley
302 Easy Street #29
Mountain View, CA 94043

A file system interface for Forth has been designed and implemented. This interface may be used for Forth systems which run under almost any native operating system, or for stand-alone Forth systems. Native operating system files are used where available. The words which the program uses to access files are the same regardless of the native operating system. The interface is simple yet general. Performance is excellent; loading from files is roughly the same speed as loading from screens, and sometimes faster. This work is placed in the public domain.

Implementations of a Portable File System Interface

W. M. Bradley
302 Easy Street #29
Mountain View, CA 94043

This paper is a companion to the paper entitled "A Portable File System for Forth". Herein I present the system dependent interfaces to two different operating systems: 68000 UNIX and CP/M-80. This work is placed in the public domain.

A Quick & Dirty Programmable Controller In Forth

Michael J. Brockman
1342 S. 57th Street
Richmond, CA 94804

In this paper I will demonstrate a few Forth words and routines which I found useful in the design of a programmable controller. This system consists of a STD Z-80 CPU board with a Z-80 CTC (counter timer circuit) for a background timer, and OPTO-22 Solid State Relay module racks for binary I/O. Most of the words used in my system were adapted directly from 8080 FIG-Forth. The source for those words may be found elsewhere. Here I will show those words and routines which were used for initialization, the background timer, and the binary I/O. My intent with this system was to make the

programming of this controller as simple as practical for the end user. As such these words allow the end user to deal with the hardware as memory arrays with the actual handling of the hardware in the background.

Expert Systems Using FORTH

John J. Cassady
339-15th Street
Oakland, CA 94612

An "expert system" is a type of artificial intelligence program that simulates the conduct of a human expert in such activities as planning, advice-giving, and medical diagnosis. Based on work from the 1950's and before, such programs have lately captured wide attention. Part of the reason is the decision of the architects of the Japanese "Fifth Generation". The software associated with the Japanese program focuses generally on artificial intelligence with special emphasis on expert systems. Notable successes have also created popular interest. Medical diagnosis, mineral and oil exploration, chemical analysis and the configuration of DEC's VAX computers have all been much publicized subjects of expert system building activity. A system for diagnosing the ills of diesel locomotives by Johnson and Bonissone at General Electric Co. is of particular interest because it is implemented in FORTH. Most other work is done in Lisp and similar functional languages.

Typically, an expert system has three, clearly separated components. First, the "database" contains the "knowledge" expressed as IF-THEN rules. Secondly, there is a collection of facts about the problem at hand. The third component is an interpreter that applies the rules to generate additional facts and, eventually, produce a result. Existing systems tend to be large. Databases of over 1000 rules are not unusual; systems of 10,000 and over are being discussed. In systems of this size, speed is naturally a problem. The currently successful systems are on the larger computers.

The challenge is to write expert systems in FORTH. The hope is that FORTH's inherent speed and flexible data structures will provide enough advantage to make the systems feasible on smaller computers. Our point of embarcation has been an article which appeared in *Byte Magazine*, September, 1981, entitled "Knowledge-Based Expert Systems Come of Age" by Richard O. Duda and John G. Gaschnig. The article, which is "required reading" for this paper, is not only an excellent introduction and survey, but it contains the complete, fully documented, Basic source code for an expert system. This is the well-known "ANIMALS" programs, a "toy" application that attempts to determine the kind of animal a user has in mind by use of deduction based on rules. A FORTH functional equivalent of this application is the subject of the rest of this paper.

User Specified Error Recovery In FORTH

Don Colburn
Creative Solutions, Inc.
4801 Randolph Road
Rockville, MD 20852

The technique presented allows the user to specify the action to be taken when an ABORT" is encountered either within the current definition or any word called by the current definition. If no handler is specified, the default action of ABORT" occurs.

Error handlers may be nested within the scope of other error handlers. When an error occurs, the most recently specified handler is executed. No action is required to remove an error handler at the end of the definition in which it is specified. It is automatically removed following the execution of EXIT. Any prior handler is automatically re-installed.

EGADS (Extensible Gate Array Design Spreadsheet)

Alan T. Furman
1634 Roll Street
Santa Clara, CA 95050

EGADS, a Forth function package, assists in the task of getting numerical results from a (pencil-and-paper) timing analysis of a gate-array logic network. It displays both independent variables and bottom-line results (timing safety margins) like a classical spreadsheet program. Once the topology-

dependent timing expressions have been compiled in, the user can rapidly perform "what if" trials to optimize a design and verify its reliability. The author's experience of writing and using EGADS leads to several observations on how Forth is best exploited in interactive design aids.

Forth and Functional Programming Systems

Harvey Glass
University of South Florida
Tampa, FL 33620

One hears often of the crisis in software facing our industry. While the computers with which we work get faster and more powerful their costs shrink. But as the hardware gets better, each new computer language and the latest software engineering techniques make only a dent in the software problem.

The architecture of the computer has in at least one respect changed little in the past 20 or 25 years. We still describe a system as a processing unit tied through a single channel to a set of mostly inactive stores. The language of the machine is a sequence of instructions that force all operations through this single channel. Backus [Back] describes the channel as the "Von Neumann bottleneck".

Most computer languages are little more than high level counterparts of the primitive machine instructions. Variables are used to identify storage cells and assignment statements store values into the cells. Conditionals and looping constructs are high level versions of conditional jumps. The syntax of existing conventional languages guides our thinking about programming.

The notation we use to solve a problem greatly influences how we approach the problem and can either enhance our ability or create serious obstacles [Iver]. In mathematics this principle has been demonstrated repeatedly. The classic case is that of long division with Roman numerals. The notation is sufficiently awkward that it creates obstacles for problems that would be otherwise easily solvable.

We will examine a style of programming different than that of conventional languages like Fortran or Cobol. We are interested in discovering a means of improving our programming capability. We will show that Forth is a language with characteristics that are consistent with that of a functional style.

The Implementation of Extensions to Provide a More Writable Forth Syntax

Harvey Glass
University of South Florida

We describe the implementation of extensions to Forth that simplify the syntax of the language and offer a notation both easier to use and more easily read - sacrificing neither the interactive capability nor the power of Forth. The advantages are increased programmer productivity, more maintainable systems, and a syntax to which programmers of other languages can more easily relate. The implementation described is entirely high level Forth; demonstrating the versatility of the Forth language and its extensibility.

Forth Coding Conventions

Kim R. Harris
Dysan Software Development
Dysan Corporation
15495 Los Gatos Blvd., Suite 1
Los Gatos, CA 95030

Coding conventions are guidelines for the physical appearance of programs; they do not affect programming techniques or creativity in solving problems. Commonly used and accepted conventions allow one programmer to read another's program more quickly and accurately.

This paper allows proposals for such conventions. It is not intended to be the final specification; it is a working document from which agreements can be extracted. The conventions presented are based on years of experience in Forth programming and many diverse views. For controversial areas, choices are described and compared. The goal of this paper is to promote consensus for a commonly accepted set of Forth coding conventions.

Forth Coding Conventions: Indentation Choices

Kim R. Harris
Dysan Software Development
15495 Los Gatos Blvd., Suite 1
Los Gatos, CA 95030

There is a substantial controversy over whether control structure words should be specially formatted. There is a need for a single indentation style to be commonly used by the Forth community in programs written for distribution or publication. Several choices of indentation styles are presented and compared. No conclusion or recommendation is made. Instead, this paper should guide the choosing of a single, acceptable style.

Proposed Extensions to Standard Loop Structures

Kim Harris & Michael McNeil
Dysan Software Development
Dysan Corporation
15495 Los Gatos Blvd., Suite 1
Los Gatos, CA 95030

Some limitations and inconsistencies of standard Forth loop structures are discussed. Examples are presented which illustrate the need for loop structures which allow multiple exits from one loop and allow each exit to have its own termination body. Extensions are described which satisfy the new requirements. Comparisons are made between the standard structures and the extended structures.

DEAL

Glen B. Haydon
Box 429 Route 2
La Honda, CA 94020

When dealing from a deck of cards, each card when dealt is removed from the deck. This is another variation on the use of a random number generator.

A simple random number generator has been illustrated in *Starting Forth*. Its value is always in the range of a 16 bit value. Often that is not really what is desired.

The function CHOOSE, is also illustrated in *Starting Forth*. In this case a random number in a specified range from 0 to a limit can be selected. In each case another operation of CHOOSE has the possibility of selecting the same value again.

In the development of language instruction programs, it proved desirable to randomly select an item from a list of words and remove it from the list. The problem is the same as dealing from a deck of cards.

A simple routine has been implemented to accomplish this function. It removes a random value from a table, closes the values and places the number at the top of the table. By reducing the CHOOSE limit one on each access, the values are dealt from the deck.

Virtual Vocabularies

Glen B. Haydon
Box 429 Route 2
La Honda, CA 94020

Virtual vocabularies provide a way to utilize overlays.

Ideally, it would be nice to be able to compile many screens of source code within a second. However, many small micro-computers are unable to run this fast. As an alternative, binary images of desired FORTH vocabularies can be brought in much more rapidly.

Each binary overlay consists of a virtual vocabulary. The overlay is not on the top of the FORTH dictionary, but rather is located in a special virtual buffer whose pointers are linked in to the existing FORTH dictionary. In this way, new words added to the dictionary do not have the virtual vocabularies buried under them.

By modifying ideograms with a symbol indicating that the standard functions apply to the virtual vocabulary instead, only five new ideograms need to be defined: V-BUFFER, V-R/W,

V-VOCABULARY, V-DEFINITIONS, and R-DEFINITIONS. The last provides a return to Real-DEFINITIONS.

First Chinese FORTH - A Double Header Approach

Timothy Huang
Dai-E Systems, Inc.
29783 Town Center Loop West
P.O. Box 790
Wilsonville, OR 97070

Dai-E Chinese FORTH conforming with the 1983 standard was created by using (1) phonetic characters for name fields, and (2) double headers — i.e., original English bodies with one English header and one Chinese header.

Forth as a First Language

John S. James
Dysan Corporation

The author has taught Forth to several groups of students who had no previous computer background or experience. Some approaches to organizing and communicating the material proved especially helpful. These concepts are outlined, and class handouts which were used are attached.

Logarithm and Exponential Functions, a Bit at a Time

Uwe Lange & Klaus Schleisiek
POB 202 264
2000 Hamburg 20
W-Germany

A high level implementation for deriving the exponential - and logarithm function a bit at a time is presented.

The underlying math is presented in an intuitive way without a formal proof - it works anyway. A side benefit is an implementation of the square root function which is simpler than was previously published in FD.

An Adaptation of F83 For Extensible Access to Files

Peter Midnight
P.O. Box 20264
Oakland, CA 94620

F83 uses CP/M files exclusively for all of its disk storage. In addition, F83 can access several CP/M files simultaneously, with each appearing as a separate, named set of numbered blocks starting at block zero. The structures in F83 which support this behavior are vectored read/write words for use by BLOCK, four byte labelling of the block buffers, a user variable to indicate which file is active, and file control blocks in the dictionary. This paper describes an adaptation of these structures to form an extensible system which supports not only external file systems and storage devices, but also internal file systems written in Forth.

Time-out RETURN Key*

Charles H. Moore
Drawer CP-66
Manhattan Beach, California 90266

The conventional use of RETURN to terminate keyboard input can be replaced by a 1-second time-out. When you stop typing, the computer takes action. This is friendly behavior, similar to spoken input. Advantages include single-keystroke entry (by setting time-out very short). A consequence is the requirement for forethought to prevent hesitation.

The FORTH Instruction Set*

Charles H. Moore
Drawer CP-66
Manhattan Beach, California 90266

The FORTH computer is designed for maximum parallelism. This is accessible with 16-bit instructions split into 7 fields. The compiler micro-codes these fields to execute as many as 7 words in one cycle. Most FORTH primitives (those words usually coded in Assembler) have 16-bit representations, though successive words are often combined.

ECL Signal Propagation*

Charles H. Moore
Drawer CP-66
Manhattan Beach, California 90266

ECL logic speeds are so high that signal propagation (at the speed of light) must be considered. Multi-drop transmission lines can be modeled by a simple pair of difference equations. Displaying their results, provides insight into propagation problems.

The integration of these equations requires a variety of elements and boundary conditions. These can be elegantly represented by compiling a model which, when executed, performs an integration time-step.

Data Security and File Management System

Pierre Moreton
Mountain View Press
P.O. Box 4656
Mountain View, CA 94040

This presentation describes a data security and file management system (FMS). The implementation has been written in high level FORTH, on the basis of a standard MVP-FORTH. Consequently this package runs on all the computers on which MVP-FORTH has been implemented.

F83
A Public Domain Model Implementation of the Forth-83 Standard

Michael A. Perry
1125 Bancroft Way
Berkeley, CA 94702

The F83 Model is a second generation public domain Forth system. It contains a reasonably complete set of utilities for program development. These include an editor, assembler, debugger, decompiler, source screen locator (VIEW), compile error locator (WHERE), support for shadow (comment) screens, and a screen printing utility. Also included are a simple multi-tasker, a host file system interface, and a meta-compiler. Unlike earlier systems, F83 does not rely upon conventional assemblers. Instead, it follows the Forth approach of regenerating itself through meta-compilation, and so is self-contained.

A Simple Multi-tasker for Forth

Michael A. Perry
1125 Bancroft Way
Berkeley, CA 94702

I will describe the use and implementation of a fairly simple multi-tasker. This multi-tasker is a part of the F83 model implementation of the Forth-83 standard. It allows a single user to create print spoolers, background counters, and other simple tasks.

Leadership For Forth Products

William F. Ragsdale
1842 Sabre Street
Hayward, CA 94545

Leadership in products springs from offering value, unavailable elsewhere, which responds to customer needs. This presentation advocates this form of leadership in Forth enterprises and products. Leadership examples are presented with suggestions for cultivating an attitude for leadership by focusing on customer service.

Key-Capture Macros*

John Rible and Thomas Dowling
Miller Microcomputer Services
61 Lake Shore Road
Natick, MA 01760

During system development or in an application, there are times when a sequence of key-strokes must be used in a repetitive manner. At such times it would be useful to be able to "teach" the computer that series of key-strokes for later "recall".

Miller Microcomputer Services has implemented an effective method of capturing key-strokes for later execution as a defined macro.

A function-initiator key starts and ends the capture, or "learning" function. During capture, key strokes are simultaneously captured and executed in the normal manner. Tones are generated as each key is pressed to remind the user that key-capture is taking place. Macros defined in this manner are also able to call other previously defined macros.

Key-capture macros are supported under MMSFORTH V2.2. They are application-transparent and can be used wherever a series of key-strokes would be awkward to include in a FORTH definition.

Floating Point Exceptions and Traps

Jonathan R. Sand
Dysan Corporation
15495 Los Gatos Blvd.
Los Gatos, CA 95030

Computer arithmetic, either integer or floating point, operates within the physical limitations of range and precision. When the result of a calculation exceeds these limitations, it is called an exception. An answer must be found that will fit, but several choices usually exist. An IEEE standard floating point implementation provides a default answer, and an alternative, called a trap. This paper is a presentation of 1) a very brief summary of the provisions in the IEEE proposed standard, and 2) a glossary of Dysan's implementation in FORTH.

Error Trapping
A Mechanism for Resuming Execution at a Higher Level

Klaus Schleisiek
POB 202 264
2000 Hamburg 20
W-Germany

This article presents a couple of words which allow the definition of software traps. When executed these trap routines allow FORTH to fall back several levels, execute a piece of specific exception code and resume execution at a higher level, adjusting the return stack. This trapping technique simplifies the handling of exceptional conditions at the man/machine interface and makes feasible sophisticated communication protocols.

Stacks for FORTH Machines*

R. L. Smith and J. Vaughan
ESL Corporation
Sunnyvale, CA

There are a number of approaches on selecting an architecture for a FORTH machine. If we assume that speed and efficiency are important parameters, it is likely that some form of hardware stacks will be

used. We will discuss some forms of hardware stacks, pointing out the simplicity of hardware stacks and some obvious advantages to having stack memory chips, such as substantially reducing the pin count. Slight modification and extension of one or two stacks leads to some interesting computer architectures on which one can base a FORTH or FORTH-like machine.

A Brief Look at Rational Arithmetic

Charles T. Springer
Mountain View Press
P.O. Box 4656
Mountain View, CA 94040

Accuracy and precision are common topics of conversation among programmers. The problem of dealing with numbers in computers and methods for improving results fill entire shelves of books on numerical analysis. Like most of life, there seems to be no "best way" for performing mathematical manipulations of data that is represented in a discrete form. We come to accept these limitations as part of the price we pay in order to get our computers to perform a desired task.

Tail Recursion Convoluted or Return Stack Optimization

Glenn S. Tenney
Fantasia Systems Inc.
1059 Alameda de las Pulgas
Belmont, CA 94002

A simple and straightforward method for return stack and performance optimization is presented. This method allows manual optimization of performance with no compiled overhead.

An example definition is described to demonstrate the concept and does no error checking. There are, of course, other ways of accomplishing the same objective. There are also other word names that could have been used.

FORTH Execution On Non-von Neumann Machines

John R. Wilson
Technology Development of Californa, Inc.
2431 Mission College Boulevard
Santa Clara, CA 95054

New generations of computer architecture, including the Japanese Fifth Generation project, have a distinctly non-von Neumann orientation, and FORTH can be expected to execute in this environment in the future. This paper briefly summarizes modern computer architecture styles, and then examines in some detail requirements for FORTH implementation and execution on some candidate machines. Specifically discussed are FORTH on dataflow and reduction machines, with some discussion of FORTH's relationship to other functional programming styles.

Improved Block Maintenance

Tom Zimmer
292 Falcato Drive
Milpitas, CA 95035

The Forth block structure can provide a good method of control for modularity, but blocks also create some of their own problems. Here are a few examples and a suggested solution.

1. Blocks are usually very wasteful of disk space.
2. It is very difficult to insert a new empty block in the middle of a sequence of blocks.
3. No automatic backup facility is provided when editing, you edit the only copy of your source text.

The solution I propose has the following characteristics:

1. It does not waste disk space, and is more space efficient than a sequential file system.
2. Inserting a new blank screen in the middle of a large group of blocks becomes simple, and does not require movement of large quantities of data.
3. Backup copies of earlier edited versions are automatically maintained, and are not discarded until you specifically request it.