Abstracts of the 1988 Rochester Forth Conference

The Pasha Hardware Diagnostic Package

Romain Agostini

Within the last year, a software package for testing and debugging the SNOOP module, a dedicated FASTBUS logic analyzer developed at SLAC and also programmed in FORTH, has been designed. It enables users to automatically test new SNOOP modules off-line and generates a series of test protocols on the printer which reflect the state of health of the module under test.

CASE: Extending Its Scope and Utility

Bapi Ahmad

Druma Incorporated

This paper describes a new case construct called **IFCASE** which preserves the generality of the if statement and the syntactical conciseness of the case statement. **IFCASE** extends the scope and utility of the case statement and removes weaknesses previously associated with the use of case.

The **IFCASE** construct is simple to implement, and it is as efficient as the case statement both during execution and compilation. Although **IFCASE** is presented in the context of FORTH, its concepts may be applied to other computer languages as well. Examples and compiler extensions in DRUMA FORTH-831 are also presented in the paper.

A Preferred Forth Meta-compiler

Allen Anway

The more technical the Forth, usually the less friendly it is. In contrast, I went out of my way to make my meta-compiler friendly to reduce debugging time of itself and the newly-compiled languages. Two goals of the Forth meta-compiler were to (a) catch errors and notify the operator and (b) compile six Forths — some very different — from one listing only. (a) is effected with predefined forward references, label tables, and other checks. (b) is effected with an elaborate comment system and by preventing the new Forth from operating at all during meta-compiling. One listing for all Forths makes it easier to update them. Despite all the meta-compiler does, it takes surprisingly few screens.

Evolution of Remote Technology Incorporated Remote Manipulator and Mobile Surveillance/Maintenance Robots

Duane Bozarth

Remote Technology Inc. developed the Remote Manipulator (RM-10A) man-in-the-loop servomanipulator in 1986. This manipulator consisted of two arms, each with six degrees of freedom plus torso rotate. The system was developed on M68000 processors using Laboratory Microsystems Incorporated FORTH running under the CP/M 68K operating system. The current RM-10A incorporates advanced features including camera auto-tracking and a graphical user interface utilizing a touch screen.

A mobile surveillance robot (SURBOT) vehicle incorporating a single RM-10A arm mounted on a three wheeled vehicle and including instrumentation for radiation monitoring, temperature and humidity as well as television cameras and microphones was developed for surveillance monitoring in nuclear power plants. This vehicle also has several unique features, including an automatic retrieval function and a data archival capability which allows instrument readings to be recorded on video tape for later playback. The RM-10A arm control software was utilized and expanded to include the vehicle control functions.

The SURBOT has been expanded to include two interchangeable options: either a set of RM-10A arms or the instrumentation package mounted on a tracked vehicle which has the capability to maneuver over obstacles and up or down stairs. This presentation will display many of the features of this series of surveillance/manipulation robots and describe the advantages of using FORTH as the development environment and some of the interesting features incorporated.

Nonvolatile Semiconductor Memory for Forth Blocks

D.B. Brumm

Small Forth systems used as controllers often cannot use conventional mass storage devices such as magnetic disks because of the adverse environmental conditions under which the system must operate. Various available and upcoming nonvolatile semiconductor memory technologies present the Forth system designer with some attractive choices for replacing conventional disk subsystems in such cases.

The characteristics and performance of electrically-erasable PROMs, battery-backed CMOS random access memories, and magnetic bubble memories as they apply to Forth systems are described. A method for achieving a high-speed transfer of a mass-storage block to a main-memory buffer is presented. The potential usefulness of recently-announced ferro-electric memories is also discussed.

HS/Forth's Rosetta Stone Dynamic Linker

Jim Callahan

The primary advantage of programming in Forth is the ability to program interactively at any level from machine code through very high level language, and to carefully adjust the level of programming for each aspect of an application. Modern versions of Forth provide convenient interfaces to all BIOS and DOS facilities as well. Harvard Softworks has taken the next logical step and made subroutine libraries written in or for another compiled languages available within the interactive HS/FORTH environment. Such libraries can be loaded and discarded dynamically at run time. Other techniques requiring use of the DOS linker with Forth object libraries unnecessarily sacrifice Forth's interactive character. Effective dynamic linking entails the establishment of call and data structure addressing, the observance of stack and other parameter passing protocols, and sometimes the interpretation of data structures of varying complexity. Rather than reinventing yet another data structure syntax, we extended HS/FORTH to directly accept data structures in the syntax found in "C" INCLUDE files. This makes interfacing to "C" subroutines particularly simple.

Autoopt: Interactive Optimization in HS/FORTH

Jim Callahan

AUTOOPT compiles Forth colon definitions of virtually any complexity or nesting depth into machine code without sacrificing the interactive character of Forth. It can be used to optimize subsections of colon definitions, add efficient code primitives to HS/FORTH or to metacompiled threaded systems, or to create non-threaded programs. AUTOOPT can use as source almost all HS/FORTH word types including colon definitions that contain literals, control structures, multiple code field data types, and words defined with `CREATE ... DOES>` . Word types can be at source level or buried in previously defined colon definitions. About the only word type that can not be optimized is one that contains `EXECUTE` since in this case the required word is not known until run time. Even this restriction is relaxed for user vectors, since it can often be assumed that the required rum time action is the one currently assigned. Increase in load time is generally unnoticeable, and the compile time remains faster than that for nonoptimized systems. Execution speed approaches that for well written assembly language programs and is at least as good as provided by dedicated non-interactive Forth compilers.

An Efficient Algorithm For Finding the Global Maximum Of an Arbitrary Univariate Function

Richard E. Haskell, Gabriel Castelino and Bahram Mirshab

Department of Electrical and Computer Engineering
GMI Engineering and Management Institute

A Forth implementation of an algorithm for finding the maximum of an arbitrary univariate function using a global search technique is presented. The algorithm is based on a method suggested by Kushner. The method will not get stuck on local maxima and does not depend on the function being well-behaved or easy to calculate or measure. For the case where the value of the global maximum is known a priori (a not uncommon case) the algorithm becomes very simple and efficient. Using a discrete test function that has two local maxima with values very close to the global maximum an experiment was performed in which the location of the global maximum was shifted to all possible values within a specified fixed range of the independent variable. The average number of search points needed to locate the global maximum was found to be 12% of the total number (compared to 50% for an exhaustive search) and the maximum number of search points needed (worst case) was found to be 25% (compared to 100% for the exhaustive case).

An Integrated Resource Management System

Gary Chanson

This Forth-based operating system attempts to unify all available system resources into a flexible, programmer friendly environment. The object is to free the programmer from the chores of implementing such things as pop down menus, mouse drivers, overlay handlers, etc. so that he may concentrate on the real program.

The system includes an archival file system interface with installable device drivers. Source code is normally loaded from text files but blocks are available as a loadable option. An extended addressing system allows easy access to any memory or I/O allocation. Overlays are automatically loaded when needed and stored in high memory for later re-use. An event manager handles all asynchronous events such as timers and the mouse. Defining words are provided for menus (including pop-down menus). To round out the user interface, is a window manager which automatically saves and restores designated are as of the display.

Early applications include a text editor, a file utility, an RPN calculator, a reconfigurable terminal emulator, and a block editor. Libraries of tools and useful routines are also provided.

The Separation and Classification of Semantic Knowledge Using Hierarchical and Linear Vocabularies

Andrew R. Diaz

Florida Institute of Technology

With the rapid expansion of semantic knowledge, separation and classification must be achieved in order to obtain optimal use of this knowledge. As the domain of a knowledge base increases, it becomes vital that the knowledge be linearly separable. Once this separation has been established, the inference engines of an Expert System, or any other system using the knowledge base, has one distinct place to search for the knowledge. However, linear separability is often quite impossible, especially when searching for answers which deal with complex situations, or deducing new knowledge from existing knowledge in various other domains.

One method for achieving linear separability in non-separable domains involves the identification and extraction of closely related knowledge into a more specific, disjoint domain, to which the authors apply the term *Class*. Within a class, the knowledge and rules may be further separated into *Subclasses*. The very structure of Forth facilitates both classes and subclasses in a manner familiar to all Forth programmers, via Forth Vocabularies. The linear approach to vocabularies provides for the domain, a method of classification. Once classified, the domain is further separated into differing types of knowledge such as: rules, facts, variables, etc. This is achieved through the traditional implementation of vocabularies in Forth.

This paper explores separation and classification of knowledge domains using both linear and hierarchical representations to achieve linear separation. Also explored is how the combination of these now linearly separate domains can be applied to reasoning and deduction.

Neural Network Simulation Using Forth

Frank N. DiMeo Ph.D.

Electrical Engineering Department, Villanova University

The renewed interest and recent activity in the field of neural networks has created a need to develop a user-friendly software tool to explore different approaches, algorithms, and applications.

This paper presents two different approaches to this simulation problem. One method uses FORTH words to represent the nodes of a neural network where the body of a word contains all the inter-node connection information together with present status, recent past history and threshold levels for a particular network node. Therefore, a set of words will exactly represent the physical neural network. The information contained by the entire set is updated cyclically by the FORTH system until convergence is detected.

The second method makes use of a rich matrix extension package to the FORTH environment. With this method the neural network is interpreted as a matrix manipulation problem whereby a fixed transformation matrix (which is the result of "training" the network) is used iteratively to update a vector which converges to become the output. This second method was applied specifically to the well known Hopfield neural net as a test. The results will be presented.

The Novix C Compiler

Brian Donovan

Novix Inc.

The C programming language has gained increasing popularity and over the years has become the preferred language for developing a variety of products. Novix has developed a C compiler to provide portability of existing software to the Novix microprocessor environment. Novix is currently Beta testing its Full-C compiler which adheres to Kernighan-Ritchie and proposed ANSI standards. The Beta version, running on the NB4100 8 MHz Novix system, runs the Byte Fibonacci benchmark 2.5 times faster than the 80386 at 16Mhz.

The Novix Express Forth Development Environment

Brian Donovan

Novix Inc.

The Novix Express Forth development environment provides all the tools necessary to rapidly develop applications software for the Novix NC4016 microprocessor. Using either a serial port on any NC4016 system or the high speed bus interface of the Novix NB4100 IBM PC plug in board, Express provides the same powerful user interface. Full file I/O to a host PC is provided, as well as a full screen editor. A wide and growing range of utilities are provided with each system. These free the application programmer from reinventing systems code so they can concentrate on the application code.

Express is the first 83 standard Forth from Novix for the NC4016 microprocessor. Out for about a year now, it has proven to be a powerful tool for the creation of embedded applications software. It is based on Henry Laxen and Mike Perry's F83 public domain Forth.

Interrupts and the Novix NC4016

Brian Donovan

Novix Inc.

The Novix NC4016 Interrupt Controller (NIC) programmable logic devices (PLDs) described here permit the handling of interrupts that are less than one microsecond apart. Maximum interrupt latency is only four clock cycles. There is no need to insert special "safe" instructions in programs to make them interruptible: interrupts may remain enabled at all times. Thus, with a single 20-pin PLD such as a PAL16R4, the NC4016 can easily handle interrupts at rates over 1 MHz. No other "mainstream" microprocessor even comes close. Additionally, fully Vectored interrupts, with over 16,000 unique vectors and 7 priority levels, can be implemented with only four more ICs.

Novix makes the information in this paper, including the source code for these NIC PLDs, available for use by anyone with no licensing fees or restrictions.

TICOL: A Development Tool for Fifth Generation Programming Environments

J. Dowe and T. Arai

Excalibur Technologies Corporation, Albuquerque, NM
Nikkei Information Systems, Tokyo, Japan

TICOL (pronounced Tie Call) is a development tool for the creation and maintenance of Fifth Generation Program Development Environments (5PE's) on computers ranging from 32 bit workstations to large scale mainframes. After a brief discussion of the need for 5PE's and the context from which TICOL evolved, this paper will describe the fundamental concepts which govern the object system upon which TICOL is based, the four primitive objects provided by TICOL from which a 5PE can be built, and the major internal elements used to implement TICOL. The paper concludes with a look at performance results to date and future plans and possibilities for the use of this new tool.

A Development Board For Modeling Embedded Forth Microcontroller Applications

Timothy Dwyer and Ken Lyons

Harris Semiconductor

The Real-Time Express Development Board (RTXDB™) speeds prototyping and testing of designs based on the RTX 20008™ Microcontroller. Together with the Real-Time Express Development System (RTXDS™) software, the RTXDB offers its users a complete development environment with extensive breadboarding and debugging capabilities. This paper describes the features and architecture of the RTXDB. Advantages of using the RTXDB to prototype a RTX 2000 based control system are discussed. An example of utilizing the prototype area is presented.

Semiconductor Memory Cards for FORTH Systems

Paul Frenger M.D.

A Working Hypothesis, Inc.

Semiconductor memory devices (RAM/EPROM/EEPROM/ROM) are now being marketed in in a credit-card-sized form factor. These may be used in a wide variety of practical applications. For example, the CMOS static RAM cards may be utilized for microprocessor main memory, the replaceable nature of which will allow convenient upgrades in capacity. Another significant application of memory cards is as a compact, low power consumption substitute for floppy diskettes in portable medical/scientific instrumentation. As diskette substitutes, memory cards are faster than Winchester disks, require no "drive" mechanism (only a socket) and are very durable. The necessary interface to an operating system/language such as FORTH is easy to accomplish. This then provides a mass storage capability for any microprocessor, for FORTH source code, compiled object code and data.

This paper describes the use of a commercially available 128 kbyte CMOS static RAM card, battery backup, as a substitute for a floppy diskette/disk drive in a portable biomedical instrumentation device. A high integration 8-bit CMOS

microprocessor with on-chip serial/parallel ports, 8-bit 8 channel A/D converter and FORTH-83 in ROM was utilized as the controller unit. The hardware and software interface details of this system are described.

RTXDS: A Development Environment for Embedded Applications with the Harris RTX 2000 Microcontroller

Harvey Glass

University of South Florida

Designers of software for embedded applications using Forth have had limited development options. To profit from the interactive power of the language, it is generally necessary to implement a Forth system on each new hardware configuration. This system can then be used to develop and test the applications software — but not entirely. The final version of the target software will typically lack headers so that testing interactively is impractical.

RTXDS (Real Time Express Development System) designed for the new Harris RTX family of Forth-based microcontrollers provides an environment uniquely well suited to microcontroller applications development. Programs can be tested on a host PC and later cross-compiled for almost any target hardware configuration. The cross-compiler maintains all header data within RTXDS on the host so that headerless user code can be executed on the target. A serial link between the host and target systems allows a programmer interactive real-time debug capability while applications software executes at full speed.

The AGSYS Programming Productivity Tool System

Andreas Goppold

EDV-Beratung — Software-Design

AGSYS is a software engineering system designed to deal with complexity. It is based on the principle of the subroutine processor. While there may be superficial similarities with Forth, the essence of AGSYS is different from Forth: The words "no programming problem should require more than 32K-byte code" (which are, as I believe, attributed to the inventor of Forth, Charles Moore), express what I think is the essence of Forth: simplicity. And this is a very good principle, except that the world is not simple. We are entering the age where we have to deal with complexities approaching that of a bacterium (giga- or terabytes of information). (The Problem of AIDS is the problem of the survival of the human race. This problem has to be solved within the next 20 years or there will be nothing left to solve). When we want to keep programs simple we must unload the complexities on the human organization that maintains those programs and connects them to a meaningful system. As ten years of experience show, Forth is not used by the organizations, and maybe it is because that offloading principle doesn't work. (See also the Valdocs experience with the Forth approach to organizational complexity. I believe this is no coincidence.)

RTX 2000™ – A FORTH-based Real Time Microcontroller

Michael H. Graff, Vice-President

Harris Semiconductor

The RTX 2000, a 16 bit microcontroller based on the FORTH language is discussed. Developed by Harris Semiconductor, this highly integrated microcontroller is ideally suited to the requirements of real time embedded systems.

The RTX 2000 is capable of executing over 40 million FORTH operations per second, and its sustained performance can easily exceed 10 million Forth operations per second. The RTX 2000's on-chip features include a single cycle 16X16 bit multiplier, three 16-bit counter/timers, 256-word parameter and return stacks, and an interrupt controller. Combined with a hardware implementation of the FORTH "Virtual Machine," conceived by Charles Moore, this microcontroller provides the speed and predictable response to asynchronous external events, required in high performance real time system environments.

Software Metrics in Forth

Dr. Tom Hand

Department of Computer Science, Florida Institute of Technology

In 1977, Maurice Halstead published his now famous book *Elements of Software Science* that presented the first systematic summarization of a branch of experimental and theoretical science dealing with the human preparation of computer programs. He illustrated that computer programs are governed by natural laws, both in their preparation and in their ultimate form.

This paper defines several software metrics that are appropriate to Forth and discusses what these metrics measure. Because Forth is different form other languages, additional software metrics are needed that measure these differences. In addition, many of the traditional software metrics evaluate to significantly lower values in Forth. These metrics indicate that, in some ways, Forth is less complex than other languages.

Forth For Geometric Modelling

Henning Hansen and Niels J. Christensen
The Technical University of Denmark

The suitability of Forth as a programming language and interactive environment for geometric modelling has been studied.

Like Forth programming, geometric modelling is an interactive task, in which complicated objects are defined from simpler ones. The Forth dictionary structure and interactive compiler can therefore be expected to be well suited for the representation and definition of geometric model structures. In this project, this has been demonstrated for simple implementations of the two best known schemes for solid modelling: Constructive Solid Modelling and Boundary Representation. Simplified code examples are provided to illustrate this point. Further more, higher level modelling procedures have been developed for Boundary Representation, including a geometric sweeping operator. The main advantage of using Forth for geometric modelling is its interactive environment, in which the geometric modelling features can be embedded. The Forth environment is thus inherited by the modelling system. This makes Forth especially suited for fast development of small geometric modelling systems.

In a second project, more advanced schemes for Boundary Representation of geometric models are under study. Forth is used for experiments and prototype development in this project.

BoxNet: A Peer-to-peer Communication Datalink Layer In a Forth Multi-tasking Environment

Jesse W. Hartley

Establishing a reliable communication link between two computers requires, in addition to a compatible physical connection, a method of information exchange management for the flow of information. Most computers have some form of RS232 type asynchronous serial interface that can provide communication path. Using this as a minimum physical connection, BoxNet provides an adaptable point-to-point protocol for synchronization, error detection and correction, and flow control between peer computer systems.

The protocol used is a byte count character oriented type that uses the full-duplex communication channel. Error control is accomplished using a cyclic redundancy code (CRC-16) for error detection and retransmission for correction. Message acknowledgement is piggybacked within packets transmitted in the reverse direction. Flow control is by a sliding-window protocol. Two Forth tasks are used to implement the transmit and receive functions in an asynchronous manner. Multiple linked lists are maintained for communication to higher network functions and between the transmit and receive tasks. The scheme is adaptable to other types of physical connections including synchronous links, half-duplex links, and parallel links. Current implementations are on the Commodore Amiga and the DEC PDP-11 under the RSX-11 operating system.

Computer Learning Using Binary Tree Classifiers

Richard E. Haskell and Bahram Mirshab
School of Engineering and Computer Science, Oakland University

It is shown that a binary tree classifier provides an attractive alternative to neural networks for many types of pattern recognition problems. Unlike neural networks a binary tree classifier is well suited for efficient implementation on current sequential computers. At each node in the binary tree a decision is made based upon the value of one of many possible attributes or features. The leaves of the tree represent the various classes that can be recognized. A new algorithm is presented that allows the computer to automatically learn the feature and threshold to use at each node based upon a set of training data. The learning process uses an efficient global search technique to find the best feature and threshold at each node. Examples are given using a Forth implementation of the algorithm.

Calculating Shape Features in a Binary Image

Richard E. Haskell and Bahram Mirshab
School of Engineering and Computer Science, Oakland University

This paper presents an efficient algorithm for extracting object size and shape information from a binary image. To obtain the shape features rapidly, the binary image is scanned left to right and top to bottom. At each scan line the coded image data for that line and the previous scan line are processed by a boundary tracking procedure. This procedure detects continuation of the objects from the previous to the present scan line, as well as termination of objects on the previous scan line. Object merging and creation are also decided by the boundary tracking procedure. Once these conditions are determined, the calculated shape features for the objects are updated. At the completion of a single pass over the entire image view, shape features such as perimeter, area, circularity, height and width for all objects in the image are extracted. In addition, the algorithm is capable of extracting features such as the first and second moments, the number of holes in the object, and complex shape features based on the segmentation of the boundaries. This algorithm has been implemented

in FORTH on a Novix NB4100 single board computer. The results indicate that real-time shape feature extraction may be achievable.

WISC and the Forth Dilemma

Glen B. Haydon

WISC Technologies, Inc.

With Forth vendors tending more and more to providing all ofthe functions to which users have become accustomed, Forth loses its philosophical goal of "keeping it simple." For the WISC processors to be accepted in the marketplace, we must provide an acceptable development system even if that system appears quite different from conventional Forth. In this paper, I will discuss the dilemma of creating a development system that meets the user's perceived needs.

The Architecture of FRISC 3: A Summary

John R. Hayes and Susan C. Lee

Johns Hopkins University / Applied Physics Laboratory

We have designed a 32 bit microprocessor that directly executing the Forth programming language. This is the third in a series of Forth Reduced Instruction Set Computers (FRISCs). FRISC 3 improves on our previous designs in its more efficient load, store, literal, and branching instructions; better support of multiplication and division; and a better approach to stack caching. The processor's instruction set consists of eight instruction types in three formats. The 32 bit wide data path contains two stack caches. The top portions of the parameter and return stacks are cached in the microprocessor to improve performance while retaining a single data path between memory and the CPU. The processor spends less than 2% of its time managing the data caches on typical Forth programs.

SKWK: An Experiment in Programming For Investigations and Identification

Alexander Luoma, Systems Programmer

Florida Department of Law Enforcement, Criminal Justice Information Systems

SKWK ("some kind of working knowledge") is an experiment based on the premise that investigation — criminal or otherwise — strongly resembles object-oriented programming. Although it is commonly accepted that all investigators and analysts will be provided workstations in the near future, perhaps the more important task is to define the software, user interface, and high level of interconnectivity between systems required by such professionals. Built on the writings and programming tools of Dick Pountain, as well as Charles Duff, Jack Park and others of previous Rochester Conferences, this project focuses on two areas: the development of automatically generated script languages to perform on-line database searches, and primary issues concerning the emerging technology of DNA "fingerprinting."

Investigations and identification procedures are meant to develop a tightly focused working knowledge, a kind of knowledge which can be greatly assisted by the certain kinds of information systems.

Designing Architecturally Extensible Stack-Oriented Processors Using the "RTX Toolbox" Approach

Chris W. Malinowski, Senior Scientist

Harris Semiconductor

The paper presents a new approach to the design and development of stack-oriented processors, modelled on the FORTH' concept of a "Virtual Machine." The design methodology described here is based on a set of software tools and building blocks developed for the Harris Standard Cell library implementation of the RTX Forth engine (the "RTX Toolbox"). This approach allows the end user to architect custom, application-specific processors capable of direct execution of the Forth language. The Toolbox is built around a dual-stack processor engine with a parallel datapath architecture. The RTX engine (described elswhere in these proceedings) is capable of executing each of its instructions in either one or two clock cycles. In addition to the core processor, single- and multi-tasking stack controller cells are included in the Toolbox to support high speed stack RAM operations; an interrupt controller macrocell offers high speed 16-input interrupt arbitration, and a set of 16x16 bit array multiplier cells allows the development of high speed DSP and Image Processing accelerators. A host of other support cells — such as a leading zero detector, counters and UARTs — are available to further enhance the functionality of a custom realtime processor.

The concept of implementing FORTH' extensibility in hardware is discussed, as related to the extensibility of the hardware implementation of the "Virtual Machine." Specifically, examples of extending the core's architecture using user-defined peripheral devices are given. Tradeoffs between firmware and hardware implemented algorithms are also reviewed.

Bryte's Forth-Based Microcontroller Development Environment

Clifton L. McLellan

Bryte Computers, Inc.

Bryte Computers' work with microcontrollers has led to the development of a Forth-based microcontroller development environment. The intent is to provide the programmer with a fast means of developing code in an interactive environment, with the added ability to cross-compile this same code into an efficient final product for distribution. Forth is particularly applicable to the world of microcontrollers, where fast, limited-duty applications running in very little memory is the norm.

TForth: A Forth Cross-Compiler For Embedded Applications with the Harris RTX 2000

Mike Mellen

Harris Semiconductor

The TForth cross-compiler is part of the RTXDS (Real Time Express Development System), an integrated environment for designing and debugging code for the Harris RTX family of microcontrollers. TForth translates Forth-83 source code into optimized, headerless machine code while preserving header data in a table for use by the RTXDS system.

This paper describes the evolution of the TForth compiler from a stand-alone "target compiler" to a part of a fully integated development environment. It discusses design issues and describes interactive debugging tools to support the development of target applications with limited resources.

Flying by Sound: The Acoustic Orientation Instrument

Brian C. Mikiten

BCM Designs

Spatial disorientation (SD), especially in single seat aircraft, is a continuing problem in flight operations. It accounts for over one third of USAF aircraft accidents involving pilot error and costs the USAF over $100 million a year. Several different approaches are being pursued to minimize this problem. One approach, additional sensory input via a Forth-based Acoustic Orientation Instrument (AOI), has been developed. The 6x6x4″ prototype described here uses a 68HC11/F board system and three custom PC boards to provide six channels of digitally controlled acoustic stimuli.

Knowledge Representation Techniques To Reduce Inference Engine Overloading

Rashesh Mody

Computer Science Department, Florida Institute of Technology

Knowledge representation, in conjunction with an inferencing mechanism, is directly related to the intelligence of a knowledge-based system. It is therefore vital that the knowledge representation chosen for a knowledge-based system, compliment the inferencing mechanism. Traditionally, Lisp frames are used to symbolically represent all forms of knowledge and facilitate inheritance. However, this often overloads the inferencing mechanism with exhaustive searching and the evaluation separation of knowledge, thus complicating the inferencing mechanism and greatly reducing its speed.

The elements which make up the knowledge in a knowledge-based system consist of: Rules, Facts, Objects, Attributes, and Variables. These elements are all related by way of the inferencing mechanism which contextually uses these elements. Each element has its own characteristics, thus suggesting different representation and inference techniques. If treated in such a manner, the knowledge elements may be separated. In addition, the use of inheritance will enhance knowledge reusability.

The authors have found Forth to be the environment best suited for such an implementation. By using unique Forth defining words to dynamically creat knowledge elements, each instantiation of an element is given a generic run-time action and stored in a vocabulary containing other instantiations of that element.

Forth Words in "C"

Wayne J. Naimoli

Gazza Lab

There are many situations when a C compiler must be used or is desirable for a system. This paper defines FORTH's data and return stack, stack manipulations, and arithmetic operations on a small C compiler. This allows quick conversion of FORTH programs onto a C system while maintaining FORTH's stack dataflow and word extension, yet having all the control structures and parser from C.

Little Universe: An Object Oriented State Table

Karl-Dietrich Neubert

Physikalisch Technische Bundeanstalt, Berlin, Germany

Little Universe is an object oriented state table. In essence, it consists of matrices, the rows of which represent methods and the columns states. Each matrix field represents an object, the properties of which are inherited from the defining method. A matrix is a world of objects, the set of all worlds is the universe. Within a given vocabulary, each name is unique and an object not only is identified by its row-, state- and world name, but vice versa is aware of these coordinates. The structure of Little Universe is explained and examples of its application are presented.

A Bit of Musing Over the NMI F68HC11

Michael Pandolfo

One of the more recent offerings of silicon based Forth is the F68HCC11 by New Micros, Inc. This chip uses the traditional method of implementing Forth on top of an existing architecture. In this case, Forth is mask programmed in ROM on a Motorola 68HCC11A8. Additionally, the chip includes RAM, EEPROM, A/D functions, and timers. This paper will describe the author's experiences in implementing a stand-alone computer system, from both the hardware and software development views.

A Consistent Interface for Firmware and Software

Michael Pandolfo

Stratus Computer

Stratus Computer manufactures a family of 68000 based fault tolerant minicomputers. The application processor is either a 68010 or a 68020, and with the newly announced I/O subsystem, each communication or peripheral adapter card employs a 68000 or 60008. This paper describes a single Forth environment that executes identically on any of these processors. The result of this project is a consistent programming interface and the ability to develop protocols within the application environment that are targeted for turnkey use in an adapter. In addition, since the interactive characteristics of Forth have not been sacrificed in the adapters, the adapter firmware can be used by the programmer for monitor and debugging purposes.

What Should a Programming Environment For Threaded Interpretive Languages Provide?

Mikael R.K. Patel

Department of Computer and Information Science, Linköping University

Given the low cost compiler and interpreter of Threaded Interpretive Languages, such as FORTH, an incremental and interactive programming style is more or less directly achieved. Traditionally program development may be regarded as a successive definition and testing of procedures and data structures. To reduce the complexity of the programming task and enhance reusage of code modules additional tools are required. This paper discusses some tools that should be available in a programming environment for Threaded Interpretive Languages. Some of the basic ideas from the advanced programming environments of INTERLISP and Smalltalk-80 are related and discussed.

Non-Invasive Tools Using Software Modules

Stephen Pelc

Microprocessor Engineering, Southampton, England

The conventional concepts of software modules or packages can be applied to Forth. With some additions tuned to Forth, classes of tools and utilities can be built that require little or no modification of the rest of the system. This paper discusses the word set and mechanisms that make such methods possible, illustrated by two practical examples. An assembler is described that resides outside the normal memory space of the module into which code is assembled, and the second example describes a debugger module designed from from the outset to take advantage of modules. Such modules have advantages for software coders, project managers, and third party vendors.

Generating Fibonacci Numbers on the Novix NC4016 Forth Engine

Douglas Ross

NASA Goddard Space Flight Center

The New Generation columns in the July and August, 1987 BYTE magazines listed some comparisons of various personal computers running six benchmarks, written in C, given in the July column. I took the first benchmark — an algorithm that computes the 24th Fibonacci number, recursively, 100 times — and translated it to cmFORTH to run on systems using the Novix NC4016 Forth Engine.

My initial intent was to compare the Fibonacci benchmark run on the systems given in BYTE against systems I had using the Novix NC4016. After getting impressive results (see BYTE, December 1987, Letters to Editor) implementing what is an inefficient algorithm — for the purpose of generating Fibonacci numbers — I decided to find efficient algorithms to generate Fibonacci numbers with the Novix chip.

I originally worked with two systems from Silicon Composers. One was their PC4000 PC-compatible board, running at 5 MHz, the other was their stand-alone Delta board system, running at 4 MHz. I eventually built a Delta board clone that can run at variable rates up to 6.67 MHz. All the routines were done in cmFORTH for each system.

The Novix NC4016 is a 16 bit processor that is designed to implement the high level language Forth in silicon. It has separate data stack and return stack data/address busses, separate main memory busses, and a separate I/O bus. There is no microcoding of the Forth primitive instructions, and all primitive Forth instructions execute in one or two clock cycles.

This paper can also be looked upon as a mini-tutorial on programming the NC4016, which in turn would make it a tutorial on programming in Forth. I hope it shows how much more efficient it is to program in Forth on a chip like the Novix.

Flexible Exception Handling in Forth-83

John Roye

This paper presents a Forth-83 implementation of flexible exception handling. Flexibility is achieved by allowing an exception to pass a value on the data stack so that the exception handling code can determine what to do. This makes selective handling and upward propagation of exceptions easy, without reference to global variables. It also allows many exceptions to be handled by one exception handler. This Forth-83 implementation is designed to be transportable across systems since it relies on only one simple system dependent word (`RP@`).

GEnie on 25 Cents a Day

Dennis Ruffer

Lead Sysop, Forth Interest Group RoundTable

Last September, we opened the Forth Interest Group RoundTable on the GE Network for Information Exchange, better known as GEnie. Through this paper I will introduce you to the system, and show you the fastest way to stay in communication with the rest of the Forth industry.

DOE Current Energy Awareness Bulletins: An Application in STOIC

Norman E. Smith, CDP

Science Applications International Corporation

Stoic is being used as an integral element of major publishing software systems implementation at the DOE Office of Scientific and Technical Information (OSTI).

OSTI publishes a number of energy related publications. Most are in the form of bibliographies arranged by subject category. The bibliography is a collection of citations about books, journals, reports, etc. published about an energy subject. The data for the bibliographies resides in the DOE Energy Data Base (EDB) at OSTI.

The Current Energy Awareness Bulletins are a group of 25 or so publications that are run from once every two weeks to once every two months. Each bulletin is created automatically; the operator submits a batch job and a short time later a laser printer starts outputting pages ready to go to the printer. This whole process has three major pieces: the data base procedures, the publications software, and the typesetting software.

This paper examines the publications software which was implemented in Stoic in detail. The publication software prepares the data extracted from the EDB for typesetting by manipulating the data and inserting the proper typesetting commands automatically.

STOIC on the DEC VAX

Norman E. Smith, CDP

Science Applications International Corporation

Stoic is a very interesting Forth-like language that provides a wealth of ideas for the Forth community to draw upon. It is currently being used as an integral part of major publishing software systems at the DOE Offfce of Scientific and Technical Information for over a year.

Stoic was first implemented at MIT around 1977 on an Intel 8080 based microcomputer system. Roger Hauck implemented the version that runs on the DEC VAX in 1980 at the Smithsonian Astrophysical Observatory. Both versions of Stoic are in the public domain and still available.

This paper examines the VAX/VMS version of Stoic and how it compares to Forth. Stoic has many features that merit inclusion into the general Forth commmunity. Specific examples of syntax differences in the two languages are

examined. Finally, the reasons Stoic was chosen over Forth for developing publishing applications at the DOE Office of Scientific and Technical Information.

An Extended Case Grammar for Natural Language Processing

Joseph J. Sternlicht

Computer Science Department, Florida Institute of Technology

The Case Grammar approach to Natural Language Processing, presented in 1967 by Charles Fillmore, opened the door to semantic, as opposed to syntactic processing of natural language. However, in an attempt to implement Fillmore's case grammar, the authors encountered several shortcomings, the solutions of which are discussed in the following paper. In particular, the authors have found the cases themselves to be too restrictive, and thus have defined an alternate set of cases. In addition, the semantic representation of a natural language sentence has been formalized into a case frame structure which may be by other knowledge-based systems.

The authors also demonstrate the natural ability for Forth to parse natural Language. By dynamically building Forth words for each natural language word, as it defined at run time, and supplying it with a speech dependent run-time action, the natural language sentence practically parses itself! Furthermore, with a little semantic information from the lexicon, which may be incomplete, the grammarless parser can resolve conflicts during case assignment in the case frame.

Given this extended case grammar and in an artificial language as naturally suited for Natural Language Processing as is Forth, simple natural language may be parsed quite easily and without a grammar to guide the parse!

Symbolic Stack Addressing

Adin Tevet

Israel

Data on the stack can be given symbolic names while not increasing execution times significantly. Symbolic names are compiled into addresses relative to the top of the stack, thereby avoiding the run-time cost of setting up and dismantling stack frames. Data is fetched from the stack by `PICK` and stored into the stack by a new word `POST`. The compiler can know the run-time stack height from the change in stack height induced by each word in the dictionary. As an example, the following definition of the Greatest Common Denominator requires just 9% more time to execute than a standard version not using symbolic stack addressing. (The word `=>(` assigns symbolic names to items on the stack, the word `TO>` performs the "`TO` concept" symbolic names, the word `TOP>` resets the stacktop.)

```
: GCD  IN( A B )  A B < IF B A ELSE A B THEN  =>( LARGE SMALL )
    BEGIN  LARGE SMALL /MOD =>( REM QUOTIENT )
  REM WHILE  SMALL TO> LARGE  REM TO> SMALL  TOP> SMALL
    REPEAT  OUT( SMALL ) ;
```

An Imprecision Vocabulary

J.E. Thomas

Biomathematics Dept., University of Alabama at Birmingham

A vocabulary is presented which allows one to keep track of the supposed imprecision of data and its effect on calculations. It also detects the most grievous computational imprecisions, notably subtraction roundoff and some nonconverging averaging procedures. Although a theoretically correct treatment would be very difficult, the present simple method yields surprisingly good results.

Marine Corps Range Scheduling System

Martin Tracy

FORTH, Inc.

Forth, Inc. has implemented a program which assists the Marine Corps Air Ground Combat Center (MCAGCC at 29 Palms, California) in scheduling 28 training areas, 22 ranges, and 4 air spaces. The program tracks the allocation and actual usage of each area and warns of any potentially dangerous conflicts.

RePTIL, ReEntrant Quans, and Recursion

Israel Urieli

Ohio University

RePTIL (a **R**ecursive **P**ostfix **T**hreaded **I**nterpretive **L**anguage) is a token-threaded Forth-like language which is being developed in an attempt to bridge the communication barrier between computers and human(oid)s. Currently the development of RePTIL is continuing on the Apple Macintosh computer in terms of a 32 bit parameter stack. In this paper we discuss the development and usage of local quantities (verb parameters or local variables) which we call Quans. In order to retain the recursive nature of the language, the quans are reentrant. The concept of of a fundamental structured

verb set defining the bounds of existence of the local quantities is discussed. Two examples of usage are presented, being the Towers of Hanoi and Heighway's Paper (Folding) Dragon.

The complete paper has been submitted to the Journal of Forth Application and Research for publication.

Electrical Distribution Network Simulator

Michel Villeneuve

Cégep André-Laurendeau, Québec, Canada

The production of hydroelectricity in Northern Québec introduces many transportation problems to engineers. The customer service reliability depends partly on an efficient coordination of electrical protections. In the case of a permanent deficiency, its goal is to isolate the site of the power failure while maintaining services to as many customers as possible. In the case of a temporary deficiency, its goal is to try to remove the cause of the failure without any perceptive interruption of service.

The electricity distribution network may be compared to a huge cobweb running not only over the Province of Québec but also over the northeastern USA. Thus, distribution companies must train lots of specialized technicians in order to ensure services over their territories. We, from André-Laurendeau College, have been mandated to design an electricity network simulator. It is presented as a 4'x2' model. This $50,000 project was financed by the *Ministère de l'enseignement supérieur et de la science* as a technological transfer between industry and educational institutions. More than 100 inputs/outputs connected to components like switches, solenoids, buzzers, and 300 LEDs arranged in 18 sections had to be controlled. At this end, a micro-controller with a resident FORTH was used (DSC-12 from DIANAX inc.). The software scans the condition of switches, detects short-circuits and decides of the actions to be taken in order to protect the network. Depending on the origins of the short-circuits, more or less complex procedures are initiated since protection of the network may only relate to fuses or proceed through different types of power breakers, reclosers, etc.

This product will be sold to electricity distribution companies and educational institutions in many countries by a private enterprise. The project reaches its end and is a complete success.

Forth and WISC: The Marriage with Extensibility

Anil Visariya

Computer Science Department, Florida Institute of Technology

The Writable Instruction Set Computer (WISC) allows a systems programmer to write micro-instructions customized for the application. The WISC CPU/16 from WISC Technologies, Inc. is based on a bit-sliced architecture that uses the 74181 chip as its basic 4-bit ALU. It is basically a RISC using the low level functions provided by the 74181. By using this reduced instruction set, one can tailor application specific micro-instructions that execute the application very fast.

This provides a new technique for building special purpose processors from the basic WISC architecture. Thus costly special purpose processors or language oriented processors required in real-time applications or stand-alone applications can very effectively be substituted by a WISC architecture processor. WISC thus presents a new dimension of extensible hardware. Forth, an extensible software environment, in combination with WISC provides an ideal solution to reduce both cost and development time and increase execution speed.

This paper explores the basics of the WISC CPU/16 architecture and describes how it will be utilized in a prototype for a rule based expert system.

RTX: A High-Speed Forth Processor Engine For Development of High Performance Custom Microprocessors

David G. Williams, Market Development Manager

Harris Semiconductor

The paper presents the architecture and basic performance characteristics of a novel processor engine implemented in Harris Standard Cell CAD environment called the "RTX Toolbox." The engine's innovative architecture is based on the concept of Forth' "Virtual Machine." Unique characteristics of the engine, such as two-stack operation, high speed "Quad Bus" data flow organization, highly efficient instruction execution, and fast subroutine calls and returns are discussed. The concept of hardware extensibility of the engine via its high speed "ASIC Bus" is introduced.

Implementation of an OPS5 Derivative on The Novix RISC Processor

Laird C. Williams

Instrumentation and Controls Division, Oak Ridge National Laboratory

The recent popularity of rule-based programming and reduced instruction set computer (RISC) architectures has made the marriage of these two technologies an almost inevitable alternative for applying artificial intelligence (AI) techniques to real-time computing tasks.

This paper presents an overview of the software development environment for a rule-based expert-system language based on OPS5 that was written specifically for the Novix NC4016 CPU.

A Mesh-Structured Processor Array Architecture Based on a Stack-Oriented Instruction Set

Kel D. Winters

Department of Electrical Engineering, Montana State University

Large single instruction/multiple datapath (SIMD) computer arrays have been proposed since the late 1950s for high performance manipulation of large regular data structures. Such machines have so far failed to gain wide acceptance over vector-based super-computers in scientific applications. However, new techniques of custom VLSI design, surface-mount packaging, and reduced instruction set selection may enable new architectural approaches not practical in earlier technologies.

A new parallel computer architecture is proposed that applies a very concise stack-based instruction set to high speed VLSI processing elements in a mesh array parallel organization. This approach differs from prior SIMD arrays in that a stack-based instruction set is used to reduce the complexity and improve the performance of the processing elements while enhancing programmability. The processing element instruction set is a subset of the Forth language and provides a simple path to a Forth-based programming environment and user interface. Other unique features of this architecture include a dual high-speed LIFO (last-in-first-out) structure realized in a single register stack, hardware support of nested conditional forms, and improved processor control circuits.

Behavioral models of this system and accompanying user environments have been developed in the Forth language, and a custom CMOS VLSI processing element prototype is currently under development at Montana State University. This architecture is intended to support arrays of 10,000 or more processing elements at improved price/performance over conventional super-computers.

Zilog Super8: The "People's Forth Chip"

Jack J. Woehr

VESTA Technology Inc.

An important entry in the field of microprocessors aimed specifically at the Forth programming environment is the Zilog Super 8. The Super8 combines a rich instruction set, an amply-sized register file, low cost, high speed and several on-chip features making it ideal for controller applications, with three instructions specifically linking it to the ever-expanding breed we call Forth processors.