AUGUST 1980 Volume 5; Number 8 $2.50 in USA/$2.95 in Canada

BYTE®

the small systems journal

A McGRAW-HILL PUBLICATION

THE FORTH LANGUAGE

OFTEN FIRST - ALWAYS THE BEST

When we introduced the "S" system last year we knew that we were ahead of the industry. We didn't realize just how far.

WE KNEW THE NEEDS–

When we began designing the S/09 computer, we knew that the normal eight-bit microprocessor system was not adequate for any but the smallest, single user business applications. What was worse there was little that could be done to expand the capabilities of the system if the customer needed it. There is nothing much worse to a business customer than a "dead end" system.

MEMORY IS THE KEY–

Obviously a business system should be able to operate with multiple terminals if needed. It should also be able to do a variety of jobs; not just data processing, but also word processing and computer aided instruction. With a system limited to 64K bytes of memory addresses such a system is just not practical. The amount of user memory available to each terminal is too small for useful work.

HOW DO YOU GET IT–

The common solution to this problem is called bank switching. This process is similar to a selector switch that turns on the bank of memory that you want to work with. This, however, has a few problems. It is inefficient, therefore expensive, plus being slow. It is also extremely clumsy when data must be exchanged between two different programs. Besides with all this you still cannot use more than 64K of memory for any one program. So what is the alternative?

DO IT RIGHT–

The alternative is an address bus with more than the normal 16 bits found on eight-bit microprocessors. By using 20 address bits you can, for instance, address up to a million memory locations directly.

This way you have access to any part of memory at any time without any intermediate processes. Program interaction is now no problem at all.

SOFTWARE MUST MATCH–

So far we have a computer system with a large memory capacity and the ability to operate with many terminals, but this is not enough. You need an operating system just as sophisticated as the hardware to complete the job. It must be a multitasking (therefore multiuser) operating system and it must be fast if it is to be useful with multiterminal systems. UniFLEX® fills these requirements and more. It also has multiple directories, log-in and password features. UniFLEX® was patterned after UNIX™, which is one of the most highly regarded operating systems around.

PERIPHERALS TOO–

To complete the system we offer our smart terminals, and a variety of disk systems. We have everthing from a 390K byte floppy to a 40 Meg/byte Winchester drive. All peripherals are compatible and so you can start with a small single terminal system and upgrade if necessary to a fully expanded system—16 terminals, 768 bytes of RAM memory and 96 Meg/bytes of disk storage.

GET THE WHOLE STORY–

If you are planning to install, or sell business systems you should get our information package on the most versatile and cost effective system on the market, the S/09. You can get a 128K system (less printer) for a little over $5,000.00.

**UNIX is a Trademark of Bell Laboratories.*

SYSTEM SOFTWARE

Languages
- Assembler
- BASIC
- FORTRAN
- Pascal
- PILOT

Data Processing
- General Ledger
- Accounts Receivable
- Accounts Payable
- Payroll
- Jobcost
- Inventory
- Mail List

Operating Systems
- FLEX*
- UniFLEX

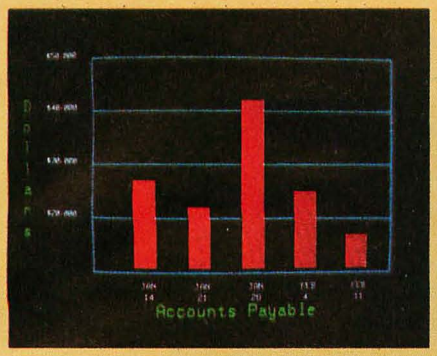
Word Processing
- Word Processing Editor
- Text Processor

Utilities
- Debug Package
- Sort-Merge
- Diagnostics

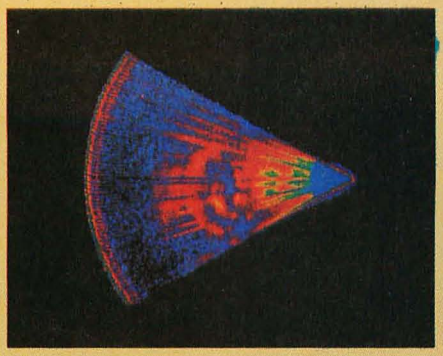
**Supplied with over 40 utilities*

SOUTHWEST TECHNICAL PRODUCTS CORPORATION
219 W. RHAPSODY
SAN ANTONIO, TEXAS 78216 (512) 344-0241

Circle 308 on inquiry card.



Management Information Display



Ultrasonic heart sector scan

High-resolution display with alphanumerics

Get the professional color display that has BASIC/FORTRAN simplicity

LOW-PRICED, TOO

Here's a color display that has everything: professional-level resolution, enormous color range, easy software, NTSC conformance, and low price.

Basically, this new Cromemco Model SDI* is a two-board interface that plugs into any Cromemco computer.

The SDI then maps computer display memory content onto a convenient color monitor to give high-quality, high-resolution displays (756 H x 482 V pixels).

When we say the SDI results in a high-quality professional display, we mean **you can't get higher resolution than this system offers in an** NTSC-conforming display.

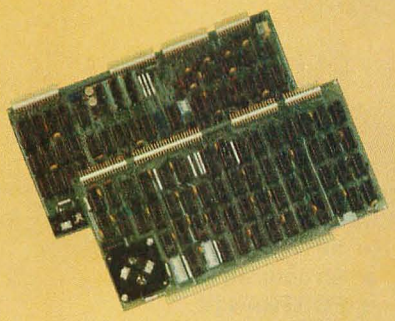
The resolution surpasses that of a color TV picture.

BASIC/FORTRAN programming

Besides its high resolution and low price, the new SDI lets you control with optional Cromemco software packages that use simple BASIC- and FORTRAN-like commands.

Pick any of 16 colors (from a 4096-color palette) with instructions like DEFCLR (c, R, G, B). Or obtain a circle of specified size, location, and color with XCIRC (x, y, r, c).

*U.S. Pat. No. 4121283



Model SDI High-Resolution Color Graphics Interface

HIGH RESOLUTION

The SDI's high resolution gives a professional-quality display that strictly meets NTSC requirements. You get 756 pixels on every visible line of the NTSC standard display of 482 image lines. Vertical line spacing is 1 pixel.

To achieve the high-quality display, a separate output signal is produced for each of the three component colors (red, green, blue). This yields a sharper image than is possible using an NTSC-composite video signal and color TV set. Full image quality is readily realized with our high-quality RGB Monitor or any conventional red/green/blue monitor common in TV work.

Model SDI plugs into Z-2H 11-megabyte hard disk computer or any Cromemco computer

DISPLAY MEMORY

Along with the SDI we also offer an optional fast and novel **two-port** memory that gives independent high-speed access to the computer memory. The two-port memory stores one full display, permitting fast computer operation even during display.

CONTACT YOUR REP NOW

The Model SDI has been used in scientific work, engineering, business, TV, color graphics, and other areas. It's a good example of how Cromemco keeps computers in the field up to date, since it turns any Cromemco computer into an up-to-date color display computer.

The SDI has still more features that you should be informed about. So contact your Cromemco representative now and see all that the SDI will do for you.

Cromemco
incorporated
280 BERNARDO AVE., MOUNTAIN VIEW, CA 94040 • (415) 964-7400
Tomorrow's computers today

Circle 1 on inquiry card.

Here's the state of the art in low-cost hard-disk computers

11 MEGABYTES OF FAST HARD-DISK STORAGE

Yes, the Cromemco Model Z-2H is in a class by itself in the computer field.

These Z-2H features tell you why:

- **11 megabytes of hard-disk storage**
- **64 kilobytes of fast RAM**
- **Two dual-sided floppy disk drives**
- **Z-80A type processor**
- **Fast 4 MHz operation—150 nanosecond access time**
- **Fast hard-disk transfer rate of 5.6 megabits/second**
- **Low cost**

And that's not all you get. Not nearly.

BROAD SOFTWARE SUPPORT

You also get Cromemco software support—the broadest software support in the microcomputer field. Software that Cromemco is known for. Like this:

- **Structured BASIC**
- **FORTRAN IV**
- **RATFOR (RATional FORtran)**
- **COBOL**
- **Z-80 Macro Assembler**
- **Word Processing System**
- **Data Base Management**

And more all the time.

FIELD PROVEN

The Z-2H is clearly in a class by itself. We introduced it last summer. It's field proven. It's reliable.

And it's rugged. Housed in a sturdy, all-metal cabinet.

EASILY EXPANDABLE

As always with Cromemco, you get expandability. The fast 64K RAM in this Model Z-2H can be expanded to 512 kilobytes. That amount of RAM combined with 11 megabytes of hard-disk storage gives you enormous computer power—the equal or even beyond what much larger computers sometimes offer.

What's more, this computer gives you a 12-slot card cage. That's to plug in your special circuits as well as additional RAM and interface cards.

This expandability is supported by still more Cromemco value — the Z-2H's heavy-duty power supply that gives you 30A at 8V and 15A at ±18V to support plug-ins.

LOW COST — SEE IT NOW

The Z-2H is real. It's been in the field for many months. It's proven itself.

You should see the Z-2H now. Contact a Cromemco representative and arrange for a demo. Learn that Cromemco is a survey-winner for reliability.

And learn that the Z-2H is under $10K.

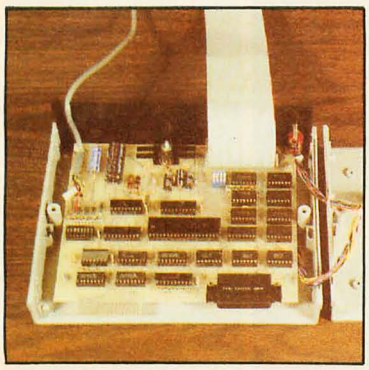
In the long run it always pays to get the best.

Cromemco
incorporated
280 BERNARDO AVE., MOUNTAIN VIEW, CA 94040 • (415) 964-7400
Tomorrow's computers today

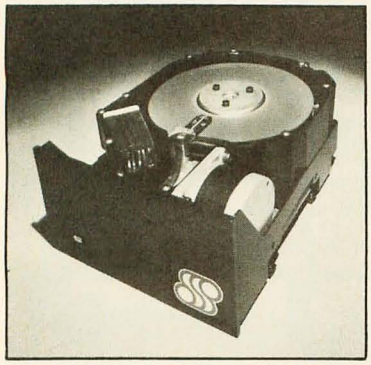
Circle 1 on inquiry card.

BYTE August 1980
Volume 5, Number 8

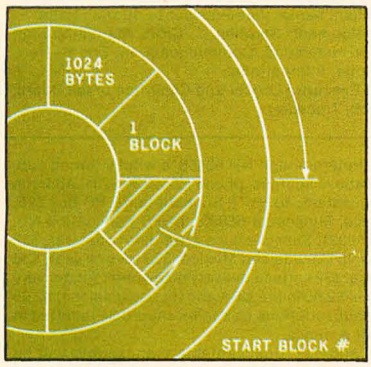
In The Queue



Page 22



Page 58



Page 164

Page 210

Foreground

Background

Nucleus

Publishers
Virginia Londoner,
Gordon R Williamson
Associate Publisher
John E Hayes
Assistant
Cheryl A Hurd

Editorial Director
Carl T Helmers Jr

Editor-in-Chief
Christopher P Morgan
Editors
Richard S Shuford, Gregg Williams,
Curtis P Feigel, Harold Nelson
Stan Miastkowski
Consulting Editor
Mark Dahmke
Book Editor
Bruce A Roberts
Chief Copy Editor
David William Hayward
Copy Editors
Faith Hanson, Warren Williamson,
Robin M Moss, Anthony J Lockwood
Assistant to the Editors
Faith Ferry
Assistants
Debe Wheeler, Karen A Cilley
New Products Editor
Clubs, Newsletters
Charles Freiberg
Drafting
Jon Swanson

Production Director
Nancy Estle
Assistant Production Director
Christine Dixon
Production/Advertising Coordinator
Wai Chiu Li
Production Art
Holly Carmen LaBossiere,
Deborah Porter
Typographers
Sherry McCarthy, Debi Fredericks,
Donna Sweeney

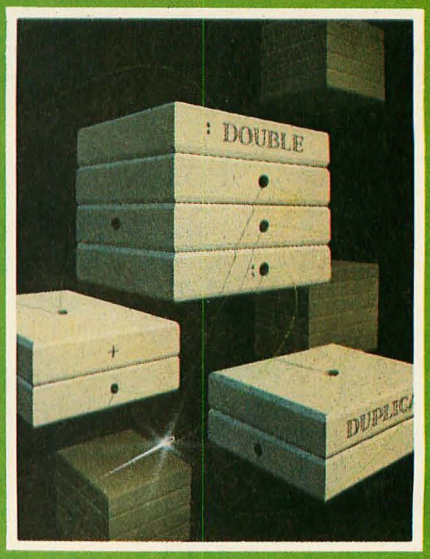
Advertising Director
Thomas Harvey
Assistants
Ruth M Walsh, Ms. Marion Gagnon
Barbara J Greene, Janet Ames

Special Projects Coordinator
Jill E Callihan
Marketing Coordinator
Laura A Hanson

Circulation Manager
Gregory Spitzfaden
Assistants
Pamela R H Spitzfaden, Agnes E Perry,
Melanie Bertoni, Barbara Varnum,
Louise Menegus, Andrew Jackson
Dealer Sales
Thomas Yanni

Controller
Daniel Rodrigues
Assistant
Mary E Fluhr
Accounts Receivable Specialist
Karen Burgess
Accounts Receivable Assistant
Jeanne Cilley

Receptionist
Jacqueline Earnshaw
Traffic Department
Mark Sandagata, Rob Hannings



ON THE COVER

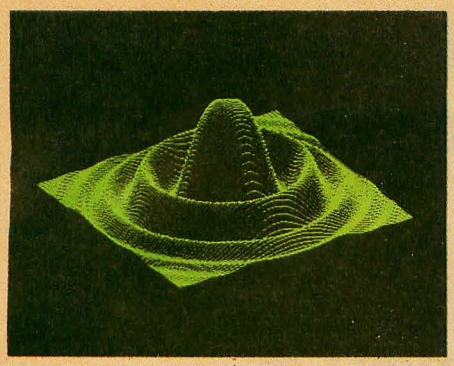
This month's cover by Robert Tinney shows a rocket-like needle threading its way through granite cubes labeled: DOUBLE , DUPLICATE , and + . The threaded path of the needle is a representation of the process used in FORTH and other threaded languages to create a new word (here, DOUBLE) with previously defined words (here, DUPLICATE and +).

Other aspects of this fascinating language are described in the editorial, "Threads of a FORTH Tapestry," and in the theme articles for this issue.

BYTE is published monthly by BYTE Publications Inc, 70 Main St, Peterborough NH 03458, a wholly-owned subsidiary of McGraw-Hill, Inc. Address all mail except subscriptions to above address: phone (603) 924-9281. Address subscriptions, change of address, USPS Form 3579, and fulfillment questions to BYTE Subscriptions, PO Box 590, Martinsville NJ 08836. Controlled circulation postage paid at Waseca, Minnesota 56093 - USPS Publication No. 528890 (ISSN 0360-5280). Canadian second class registration number 9321. Subscriptions are $18 for one year, $32 for two years, and $46 for three years in the USA and its possessions. In Canada and Mexico, $20 for one year, $36 for two years, $52 for three years. $32 for one year air delivery to Europe. $32 surface delivery elsewhere. Air delivery to selected areas at additional rates upon request. Single copy price is $2.50 in the USA and its possessions, $2.95 in Canada and Mexico, $4.00 in Europe, and $4.50 elsewhere. Foreign subscriptions and sales should be remitted in United States funds drawn on a US bank. Printed in United States of America.
Address all editorial correspondence to the editor at the above address. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE.
BYTE® is available in microform from University Microfilms International, 300 N Zeeb Rd, Dept PR, Ann Arbor MI 48106 USA or 18 Bedford Row, Dept PR, London WC1R 4EJ ENGLAND.

AUDITED PAID CIRCULATION ABC

Subscription WATS Line: (800) 258-5485
Office hours: Mon-Thur 8:30 AM - 4:30 PM, Friday 8:30 AM - Noon, Eastern Time

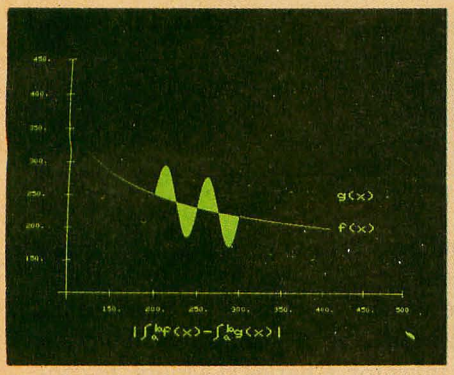


MICROANGELO

HIGH RESOLUTION GRAPHICS SINGLE BOARD COMPUTER

by

SCION
CORPORATION



RS-170 composite or direct drive output

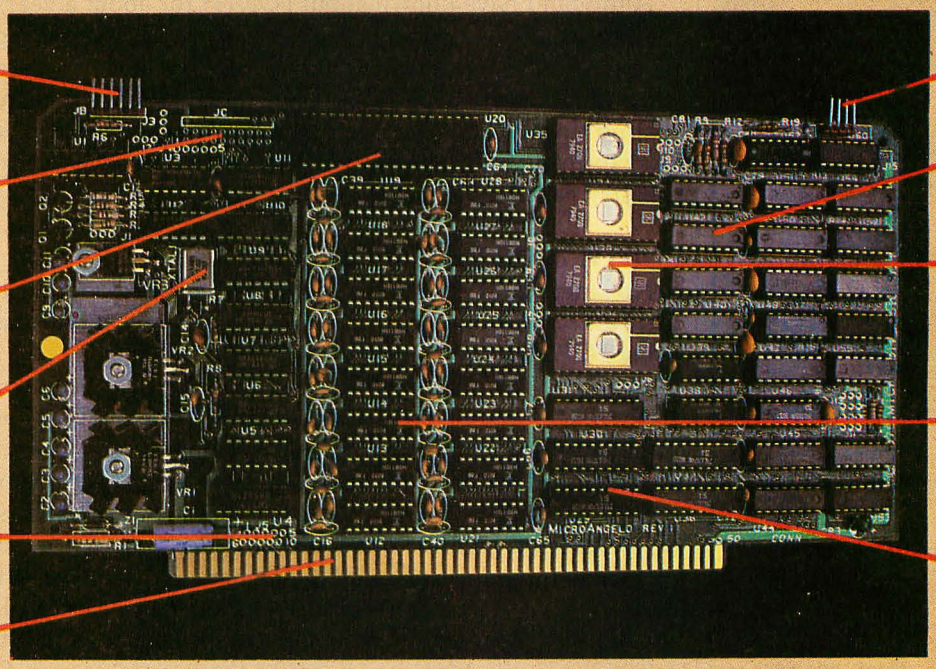
Local or external sync generation

4 or 5 Mhz Z80 microprocessor

60 hertz real-time clock

8 level interrupt tie-in

IEEE S100 bus compatible



Light pen interface

Time multiplexed refresh

4K resident Screenware™ Pak I operating system

32K RAM isolated from host address space

High speed communications over parallel bus ports

Screenware™ Pak I

A 4K byte operating system resident in PROM on MicroAngelo™ Pak I emulates an 85 character by 40 line graphics terminal and provides over 40 graphics commands. Provisions exist for user-defined character sets and directly callable user extensions to Screenware Pak I.

Host Resident Terminal Software

An interface software package that coordinates input/output from the MicroAngelo™ graphics board, the MicroAngelo™ keyboard, and your computer. The result is a flexible, yet sophisticated graphics terminal.

SCION Corporation
8455-D Tyco Road
Vienna, Va. 22180
(703) 827-0888

European Distributor:
Micro Diversions UK Ltd.
17/19 Mesnes Street
Wigan, England WN1 1QP
09-423 4311

Editorial

Threads of a FORTH Tapestry

Editor's Note: *This month's editorial is by BYTE Editor Gregg Williams. Gregg was responsible for the preparation of this month's special section devoted to the FORTH language. Carl Helmers returns next month with an editorial....***CM**

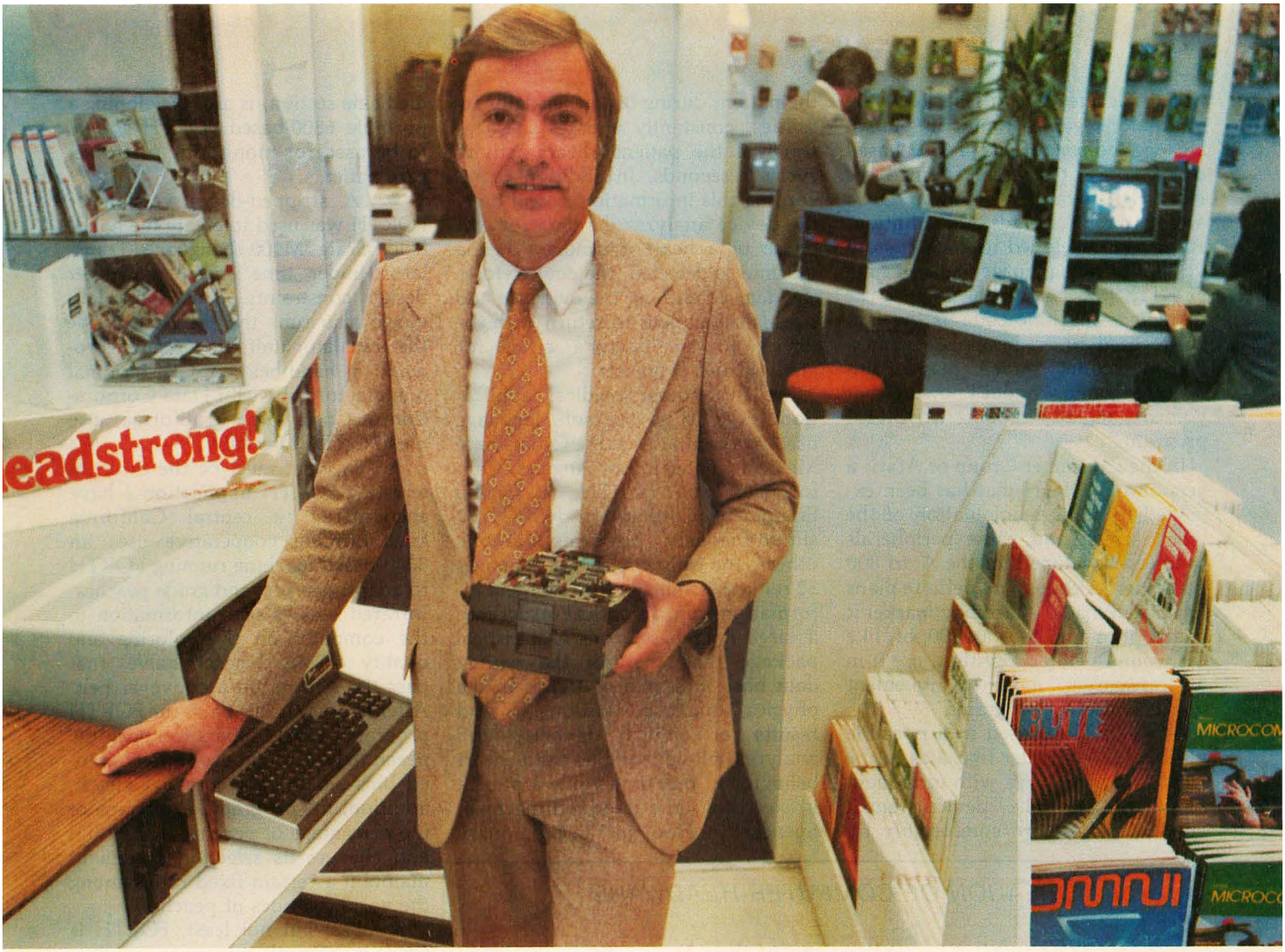
What do a portable heart monitor, the new Craig Language Translator, a peach-sorting machine, and a movie called *Battle Beyond the Stars* have in common? The answer is FORTH, a not-so-new language as comfortable in industrial machinery as it is in a personal computer. In fact, it was originally used by its inventor, Charles H Moore, to control the telescope and equipment at the Kitt Peak Observatory.

Although I have known about FORTH for about a year, it was only during the preparation of this issue that I began to actively keep my ears open for mention of this unusual language. I have uncovered a lot of information (and some experience) about FORTH and its variations. The language is so unusual that no single line of thought could give you a picture of what the language is like. Instead, the following sections represent several threads from the rich tapestry called FORTH.

FORTH in the Real World

No language I know of is as comfortable in real-world situations as FORTH. Here are some examples of the breadth of applications that have been created using FORTH:

- Elicon Inc of Brea, California, is using FORTH software to drive the same kind of computer-controlled cameras that were used to film the sophisticated space-battle scenes in *Star Wars*. New World Productions of Venice, California, is using this camera system to film the spaceship sequences in the motion picture *Battle Beyond the Stars*. In a related development, Magicam Inc (which devised a number of the special effects for the recent movie *Star Trek*) is in the process of converting control of its master-slave camera pair from an analog computer to a digital computer running FORTH software. In the Magicam process, the master camera follows actors on a special blue stage while the computer guides the slave camera across a detailed model. Later, the two images are optically combined, producing the effect of the actors actually being in the landscape depicted on the model.
- Allen Test Products of Kalamazoo, Michigan, has developed an ignition analyzer for use in service stations and automobile repair shops that analyzes the behavior of automobile ignition systems and displays both diagnostic and corrective information. Formerly, the voltage waveform from a spark plug was displayed on an oscilloscope, after which a mechanic would attempt repairs based on his interpretation of the waveforms.
- Atari Inc is using FORTH in two of its divisions and is rumored to be contemplating other uses for the language. In its Coin-Operated Division,



"For reliable data storage, you can't beat Shugart's Minifloppy."™

Raymond Schlitzer, Owner—
Computerland, San Francisco

"I sell systems my customers can depend on. That's why most of the personal and small business computer systems sold here feature Minifloppy disk drives. I know from experience I can rely on the Minifloppy."

Since 1976 Shugart's Minifloppy has been used by more small computer system manufacturers than any other drive. In fact, more than half-a-million Minifloppys have been installed. The Minifloppy looks small—but it stores a lot of data. 250 kilobytes on one side, or up to 500 kilobytes in the double-sided model. That's about 50 pages of printed information on a single-sided Minidiskette, and twice that on the double-sided version. You'll have plenty of storage capacity for your programs, letters, forms, or ledger entries. And you find your data fast, too, because the Minifloppy is a random access device that eliminates the need to search for your data serially as you must with a tape cassette unit.

No matter what problem you're solving with your computer system, you can rely on Shugart's Minifloppy for data storage. We're known as the Headstrong company for good reason. We're Headstrong about reliability, quality, and value. Ask your dealer. He knows us.

Rely on the Headstrong Company.

Shugart®

475 Oakmead Parkway, Sunnyvale, California 94086

TM—Minifloppy is a trademark of Shugart Associates.

which develops and markets the stand-alone games found in pinball arcades and restaurants, a 6502-based development system employs FORTH software to debug and test arcade circuit boards. In addition, Atari has developed its own custom version of the language, called game-FORTH, that is awaiting its first use to replace machine code as the language used to create arcade games. Someday soon, you may play a coin-operated game without knowing that you are actually running a FORTH program.

In the Consumer Group of Atari, a version of FORTH that has been extended to allow manipulation of the video screen and game peripherals has been developed for the Atari 800 computer. Although no definite plans have been made, Atari may market it as an option for the Atari 800, or, like the Coin-Operated Division, use it in a "transparent" mode to implement games and other programs.

- FORTH is used in a portable 1802-based computer that aids in the treatment of patients with infrequent heart flutter. The device, small enough to be worn comfortably by the patient during his or her daily activities, constantly updates a "snapshot" of the patient's heart activity every 7 seconds. In addition to recording this information in real time, the device analyzes the data for evidence of a heart murmur. When a murmur is detected, the device stores the data containing the evidence and signals the patient to return with the device to the doctor's office for analysis and diagnosis.
- In another medical application, FORTH is the sole language used in a computer at the Cedar-Sinai Medical Center in Los Angeles, California. Using FORTH, a Digital Equipment Corporation PDP-11/60 simultaneously performs, among others, the following tasks: manages 32 remote terminals; stores patient information from an optical reader into a large data base; runs a statistical package that analyzes the patient data base in search of trends in the physical makeup, treatment, and results of similar patients; and analyzes blood samples and heart behavior in real time while a patient is exercising on a treadmill machine. Spencer SooHoo, in the pulmonary medicine section, is also developing a portable 6800-based FORTH system to be used for monitoring intensive-care patients.
- A stripped-down version of FORTH was used to create the hand-held Craig M100 Language Translator under time, size, and other design constraints. This same language also runs the software inside the translator unit. In a related product, a hand-held ASCII terminal manufactured by MSI Data Corporation of Costa Mesa, California, also uses FORTH internally.
- In what must be the most interesting FORTH application I have encountered, a central California fruit farming cooperative uses an 8080-based machine running FORTH to adaptively sort and grade peaches. Infrared sensors send information to the computer on the coloring and quality of pitted peach halves that pass the sensors on a conveyer belt. After analyzing this data, the FORTH program causes flippers to knock the peach halves into appropriately graded bins—extra fancy, fancy, etc. In addition, the program keeps track of the percentage of peaches in each bin and changes its selection criteria to maintain a certain fixed ratio among the various grades of peaches.
- Last but not least, FORTH is used in several aerospace applications. A FORTH-like language called IPS (running on an 1802-based system) is orbiting Earth in an amateur radio satellite called the OSCAR Phase III. Avco Inc is using another 1802-based system (again, for the small size and power consumption of the 1802 microprocessor) to monitor temperature and take care of ground-to-satellite and satellite-to-ground telemetry in a military satellite.

Who Should Try FORTH?

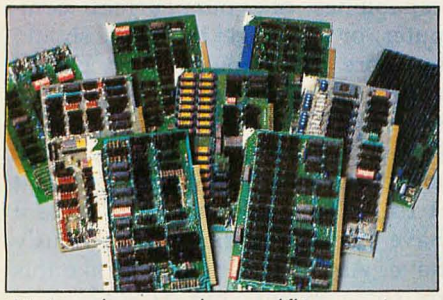
FORTH is an easy language: a high school student, Arnold Schaeffer, wrote an arcade-type game called BREAKFORTH. (See "Breakforth into FORTH," by A Richard Miller and Judy Miller, on page 150.)

FORTH is a difficult language: it easily beats APL as a "write-only language"; you can write a program in the language, but you can't easily read what you've written.

Given these two valid extremes, your initial reaction might be, "This doesn't make sense." True, learning

At Intersystems, "dump" is an instruction. Not a way of life.

(Or, when you're ready for IEEE S-100, will your computer be ready for you?)



We're about to be gadflies again.

While everyone's been busy trying to convince you that large buses housed in strong metal boxes will guarantee versatility and ward off obsolescence, we've been busy with something better. Solving the *real* problem with the first line of computer products *built from the ground up to conform to the new IEEE S-100 Bus Standard.* Offering you extra versatility in 8-bit applications today. And a full 16 bits tomorrow.

We call our new line Series II.™ And even if you don't need the full 24-bit address for up to 16 megabytes (!) of memory right now, they're something to think about. Because of all the performance, flexibility and economy they offer. Whether you're looking at a new mainframe, expanding your present one or upgrading your system with an eye to the future. (Series II boards are compatible with most existing S-100 systems and *all* IEEE S-100 Standard cards as other manufacturers get around to building them.)

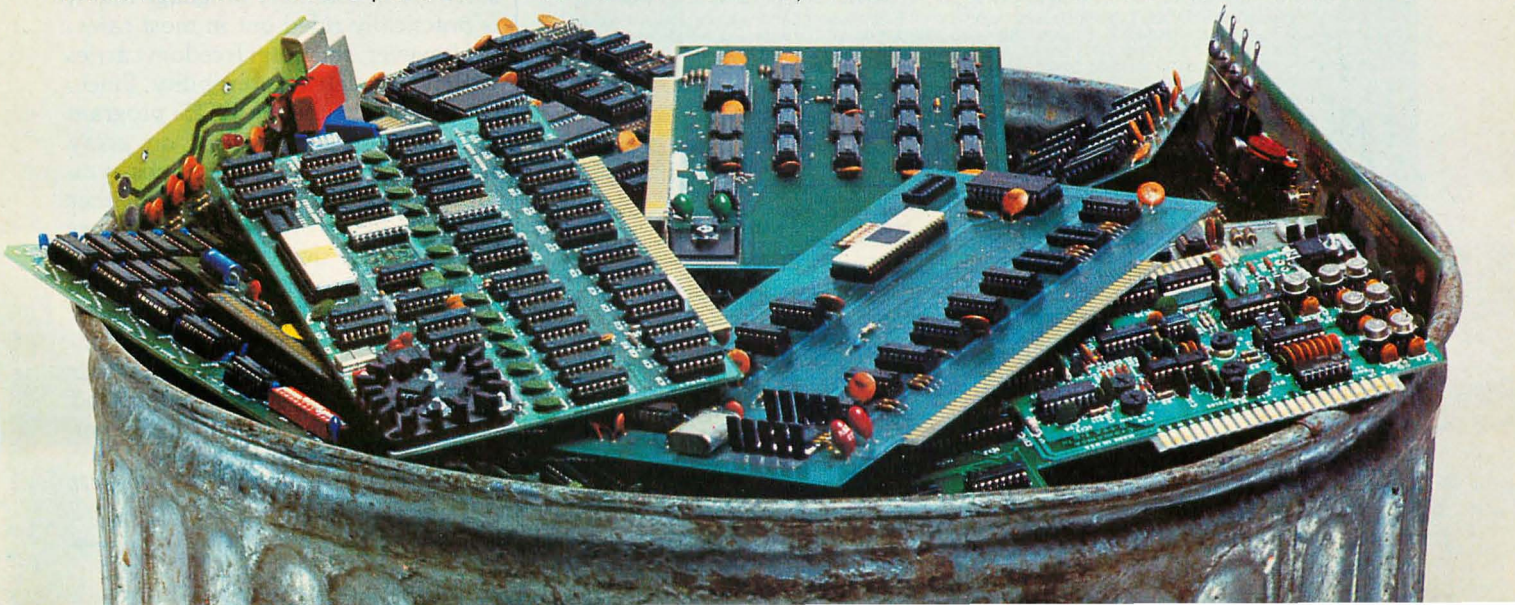
Consider some of the features: Reliable operation to 4MHz and beyond. Full compatibility with 8- and 16-bit CPUs, peripherals and other devices. *Eight* levels of prioritized interrupts. Up to 16 individually-addressable DMA devices, with IEEE Standard overlapped operation. User-selectable functions addressed by DIP-switch or jumpers, eliminating soldering. And that's just for openers.

The best part is that all this heady stuff is available *now*! In our advanced processor—a full IEEE Bus Master featuring Memory Map™ addressing to a full megabyte. Our fast, flexible 16K Static RAM and 64K Dynamic RAM boards. An incredibly versatile and economical 2-serial, 4-parallel Multiple I/O board. 8-bit A/D-D/A converter. Our Double-Density High-Speed Disk Controller. And what is undoubtedly the most flexible front panel in the business. Everything you need for a complete IEEE S-100 system. Available separately, or all together in our new DPS-1 Mainframe!

Whatever your needs, why dump your money into obsolete products labelled "IEEE timing compatible" or other words people use to make up for a lack of product. See the future now, at your Intersystems dealer or call/write for our new catalog. We'll tell you all about Series II and the new IEEE S-100 Bus we helped pioneer. Because it doesn't make sense to buy yesterday's products when tomorrow's are already here.

InterSystems™

Ithaca Intersystems Inc.,
1650 Hanshaw Road/P.O. Box 91,
Ithaca, NY 14850
607-257-0190/TWX: 510 255 4346



FORTH takes some time; it's somewhat like learning a foreign language. So far, my experiences with FORTH remind me of my attempts at learning a smattering of Russian; both languages are so different from any I've seen before—French or Spanish, BASIC or FORTRAN—that I have to mentally shift gears to work in the new language.

You should give FORTH a try if you are excited by what you see here. Especially important in this respect are the articles, "What is FORTH? A Tutorial Introduction," by John James, and " A FORTH Glossary," pages 100 and 186, respectively. Your best bet is to get to a computer that can run a version of FORTH; or, better yet, get someone who knows the language to demonstrate it to you.

My first experience with FORTH was at the Fourth West Coast Computer Faire in May 1979. A member of the FORTH Interest Group was demonstrating the language using an Apple II and an Advent television screen. First, he defined a word called COUNT, like this:

```
: COUNT   0  DO I . LOOP ;
```

Then he said { 6 COUNT } (note: the braces are not part of the expression; see the accompanying text box), the computer replied with { 0 1 2 3 4 5 OK }. I was instantly hooked on learning more about FORTH. What he had done closely paralleled the *iota* function in APL, and anything that even resembled APL was going to get my full attention.

If you are at all dissatisfied with the capabilities of your current computer, or if you feel that there should be more to computers than BASIC and assembly language, you should try FORTH. Once you get accustomed to its peculiar syntax, you can make it do nearly anything you want it to. In fact, you can even make it have features it did not previously have. Assembly language is like this to some extent, but FORTH is a higher-level language with the same abilities—only magnified. FORTH is what I call a "homebrew" language; its enthusiasts carry with themselves the same look-how-this-works enthusiasm as do most hardware hackers who build their own hardware. If we ever have a homebrew *software* issue, FORTH will certainly be included.

FORTH is the ultimate software hacker's language because, like a bag of components before a hardware hacker, you can do anything you want to with it. It can be argued that assembly language is the ultimate programming language; strictly speaking, this is true, but it takes so much more time to craft a piece of software in assembly language that it is practically ruled out in most cases.

However, this total freedom carries with it complete responsibility. Since, for example, the FORTH program you write is free to use an array subscript that is out of bounds, you must be responsible enough to either (a) put in error-checking routines (you can take them out later), or (b) build your program up from small tested modules to assure that your program will never execute an improper subscript. If you would rather have the language system do this kind of work for you, stick to BASIC or whatever you're running now.

Text continued on page 128

The Evolution of FORTH, an Unusual Language

Charles H Moore
FORTH Inc
2309 Pacific Coast Hwy
Hermosa Beach CA 90254

Introduction

When I invented FORTH about 10 years ago, my goal was simply to make myself a more productive programmer. When I first worked with computers at MIT and Stanford in the early 1960s, I figured that in 40 years a very good programmer could write forty programs. And I wanted to write *more* programs than that. There were things out in the world to be done, and I wanted a tool to help me do them. As I worked on programs that ranged from satellite orbits to chromatography to business systems, I developed FORTH in line with my overall goal. For several years now, I have been able to work at ten times my original rate.

As I began thinking of rather drastic improvements to programs, I think I was arrogant. I wanted to do things my way. I was not convinced that I should not be permitted to, and I was a bit hard to get along with. The arrogance was necessary because I felt insecure. I was promoting ideas that everyone said were wrong and that I thought were right. But, if I were right, that meant that all the other people would have been wrong, and there were many more of them than me. And it took a lot of arrogance to persist in the face of massive disinterest.

FORTH is a polarizing concept. There are people who love it and people who hate it. It's just like religion and politics. If you want to start an argument, say, "Boy, FORTH's really a great language."

This is partly because FORTH is an amplifier. A good programmer can do a fantastic job with FORTH; a bad programmer can do a disastrous job. I have seen very bad FORTH code and have been unable to explain to the author exactly why it was bad. There are some visible characteristics of good FORTH, such as very short definitions (many of them). Bad FORTH often takes the form of one definition per block—big, long, and dense. It is quite apparent, but difficult to explain, why or how a FORTH program is bad.

BASIC and FORTRAN are less sensitive to the quality of the programmer. I was a good FORTRAN programmer; I thought that I was doing the best job possible with FORTRAN, but it was not much better than what everybody else was doing. In this sense, FORTH is an elitist language.

On the other hand, I think that FORTH is a language that a grade school child can learn to use quite effectively, if it is presented in bite-size pieces with the proper motivation.

FORTH is the first language that has come up from the grass roots. It is the first language that has been honed against the rock of experience before being standardized. I hesitate to say it is perfect; I will say that if you take anything away from FORTH, it is not FORTH any longer—the basic components are all essential to the viability of the language.

History

What might be called the prehistory of the FORTH language goes back much further than 10 years. The first element of FORTH to exist was the text interpreter, shown in listing 1. This early version, programmed in ALGOL at the Stanford Linear Accelerator Center in the early 1960s, was part of a program called TRANSPORT, which designed electron-beam transport systems. Besides the text interpreter, this printout also shows an early version of the dictionary. The influence of LISP is evident in the indivisible entity (which in FORTH is called a *word*) named ATOM. As the interpreter reads a word from a punched card, it executes the associated routine, as for DRIFT in this example. The style resembles that of modern FORTH: there is no limit on the length of a word, as you can see by the length of the word SOLENOID, but only the

About the Author

Charles H Moore is Chairman of the Board of FORTH Inc, a firm created in 1973 to provide application programming services and packaged FORTH systems. This article is adapted from a speech delivered at the FORTH Convention held in San Francisco in October 1979.

The best in data base management for your micro-computer

Get the most out of your micro-computer. Use our advanced and progressive data management system.

HDBS is an extended hierarchal data base system offering

- fixed length records
- file-level read/write protection
- one-to-many set relationships

MDBS is a full network data base system offered as an upgrade from HDBS...or it may be ideal as your initial system. **Unique and versatile,** it adds these features:

- full network CODASYL-oriented data structures
- variable length records
- multiple levels of read/write protection
- one-to-one, many-to-one, and many-to-many sets
- non-redundancy of data, easy updating
- occurrences of a record type may own other occurrences of the same type
- a single set may have multiple owner and member record types

MDBS-DRS. As an add-on to MDBS, the DRS system offers extraordinary flexibility in data base restructuring to meet new needs.

- Item, record, and set types can be added, deleted, or renamed in an existing data base as well as other data base characteristics. You can redesign the data base after it is already on-line!

NEW!

MDBS-RTL. As an add-on to MDBS, the RTL (Recovery Transaction Logging) logs all data base transactions, so that in the event of a system failure, the data base can be recovered with minimal loss of information.

- The recovery processor permits selective reloading of the data base from the transaction file. Users can log messages, indicate complex transaction sequences, and effect selective control over the recovery process.

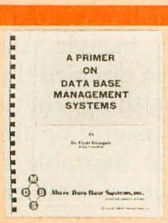
NEW!

MDBS-QRS. An interactive Report-Writer/Query-System for MDBS data bases. Features...

- easy data retrieval for non-technical users
- complex retrieval conditions may be specified
- detailed reports can be quickly generated
- wildcard and "match-one" string specifications included

HDBS and MDBS Packages Include:

- DDL data definition language analyzer/editor
- 260-page users manual
- DMS data management routines callable from host language
- Sample application program and DDL files
- Relocator to re-org all routines
- System specific manual for bringing up our software



Coming soon: Multi-User Versions of MDBS, and a Z8000 Version.

54-page "primer" on data base systems for micro-computers — $10.00 per copy.

NEW!

Both HDBS and MDBS Systems...

- Run under...
  - CP/M with Microsoft BASICs, FORTRAN or COBOL; InterSystem PASCAL/Z; Sorcim PASCAL/M; Micro Focus CIS COBOL; Digital Research PL/I
  - MVT/FAMOS with BASIC
  - OASIS with BASIC
  - TRSDOS and NEWDOS (Models I and II) with Disk BASIC
  - North Star DOS with North Star BASIC
  - Apple DOS and Applesoft BASIC
  - Machine Language Interface available on all above systems.
- Up to 254 record-types definable in the data base; each record-type may contain up to 255 item-types; each item-type may be up to 9,999 bytes in length.
- Names of data items, records, sets, and files are wholly user definable.
- Commands to add, delete, update, search, and traverse the data base.
- Straightforward use of ISAM-like structures.
- Records can be maintained in several sorted orders.
- Written in machine language for maximum execution efficiency and minimal memory usage.
- Independent of types and sizes of disk drives. Support data base spread over several disk drives (max.8); disks may be mini- or full-sized floppies or hard disks.
- Available inn versions Z80 (requires approx. 18K), 6502 (approx. 26K), 8080 (approx. 22K)
- 8086 version available. (Call or write for details and prices.)

NEW!

Ordering and pricing information:
(applicable to Z80, 8080 and 6502 versions):

HDBS	$ 300.00
MDBS	900.00
DRS	300.00
RTL	300.00
QRS	300.00
HDBS upgrade to MDBS	650.00
MDBS with DRS, RTL, and QRS	1500.00
HBDS/MDBS Manual	35.00
DRS Manual	5.00
RTL Manual	5.00
QRS Manual	5.00
System Specific Manuals (each)	5.00

Within a given operating system, add $125.00 for each additional language selected.

For prices outside the U.S. and Canada, please ask for price lists.

Add $2.50 handling fee for non-cash order ($5.00 outside U.S.).

When ordering, specify intended use with...

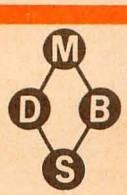
1. North Star DOS and BASIC
2. CP/M - Microsoft BASIC 4.XX
3. CP/M - Microsoft BASIC 5.XX
4. CP/M - Microsoft BASIC Compiler or FORTRAN-80
5. CP/M - Microsoft COBOL-80
6. CP/M - InterSystem PASCAL/Z
7. CP/M - Sorcim PASCAL/M
8. CP/M - Digital Research PL/I
9. CP/M - Micro Focus CIS COBOL
10. TRSDOS/NEWDOS and TRS Disk BASIC (Models I and II)
11. Apple DOS and Applesoft BASIC
12. MVT/FAMOS and BASIC
13. OASIS and OASIS BASIC
14. Machine Language Programs (Specify operating system.)

Indiana residents add 4%.

We accept Visa and Master Charge.

Finally, our software may cost a little more... but it's worth a lot more in quality and versatility.

Micro Data Base Systems, inc.



Box 248, Lafayette, Indiana 47902
317-742-7388 or 317-448-1616

Circle 52 on inquiry card.

first characters are significant and words are separated by spaces.

Other very early concepts have either changed in form or have evolved dramatically. In listing 2, the word that has become { : } (colon) in modern FORTH is called DEFINE , while END has become { ; } (semicolon). This listing also shows stack operators being defined. As an example of a concept that has evolved, consider the dictionary being sealed by the word SEAL and broken by the word BREAK . Such sealing and breaking has since been replaced by the idea of vocabularies.

Listing 1: *An early version of the FORTH text interpreter (written in ALGOL).*

```
IF ATOM="DRIFT" THEN DRIFT
ELSE IF ATOM="QUAD" THEN QUAD
ELSE IF ATOM="BEND" THEN BEND
ELSE IF ATOM="FACE" THEN FACE(-1)
ELSE IF ATOM="ROTATE" THEN ROTATE
ELSE IF ATOM="SOLENO" THEN SOLENOID
ELSE IF ATOM="SEX" THEN SEX
ELSE IF ATOM="ACC" THEN ACC

ELSE IF ATOM="MATRIX" THEN BEGIN IF NOT FITTING THEN BEGIN
   REAL A;
   WRITE1(3,0,0,CORE[S]); LINE(-(8+42×(ORDER-1)));
   FOR J←1 STEP 1 UNTIL 6 DO BEGIN
      FOR K←1 STEP 1 UNTIL 6 DO WRITE1(2,8,R1[J,K]×UNIT[K]/UNIT[J],2);
      LINE(0) END;
   IF ORDER=2 THEN FOR C←1 STEP 1 UNTIL 6 DO BEGIN
```

Listing 2: *An early version of the FORTH words { : } (called DEFINE here) and { ; } (called END here).*

```
"- "OPEN DEFINE   MINUS +   END
SEAL   "< "OPEN DEFINE   - <   END BREAK
"NOT "OPEN DEFINE   MINUS 1+   END
"> "OPEN DEFINE   •<   END
"AND "OPEN DEFINE   ×   END
"OR "OPEN DEFINE   NOT •NOT AND NOT   END
   "T 1 1 "REAL DECLARE
"= "OPEN DEFINE   T← ; DUP T< • T> OR NOT   END
"≠ "OPEN DEFINE   = NOT   END
"≤ "OPEN DEFINE   > NOT   END
"≥ "OPEN DEFINE   < NOT   END
"DUMP "OPEN DEFINE   NAME 10 "ALPHA WRITE; 3 10 "REAL WRITE 0 LINE END
```

Listing 3: *Another prototype of the FORTH text editor, again in ALGOL. In this listing, the word ATOM (the predecessor of the basic unit in FORTH, the word) has been replaced by the word W .*

```
120 CYCLE; FILL OUTPUT WITH BUFFER[1],BUFFER[2];
  1 WHILE WORD NEQ "END    " DO
  2 IF W=GM1 THEN REPLY("OK    ")
  3 ELSE IF NUMERIC THEN L:=MIN(W-1,$OF)
  4 ELSE IF W="+       " THEN L:=MIN(L+WORD,EOF)
  5 ELSE IF W="-       " THEN L:=MAX(L-WORD,0)
  6 ELSE IF W="T       " THEN BEGIN
  7    IF WORD=GM1 THEN W:=1; W:=MIN(L+W-1,EOF);
  8    FOR L:=L STEP 1 UNTIL W DO BEGIN
  9       POSITION; TYPE END; L:=L-1 END
130 ELSE IF W="R       " THEN BEGIN
  1    POSITION; REPLACE END
  2 ELSE IF W="A       " THEN BEGIN
  3    L:=EOF:=EOF+1; REPLACE END
  4 ELSE IF W="I       " OR W="D       " THEN BEGIN
  5    IF NOT RECOPY THEN BEGIN
  6       RECOPY:=TRUE; REWIND(CARD) END;
  7    POSITION; IF W="I       " THEN BEGIN
  8       PLACE; REPLACE END
  9    ELSE BEGIN EMPTY:=TRUE; IF WORD NEQ GM1 THEN BEGIN
140          L:=MIN(L+W-1,EOF); SPACE(CARD,L-L0+1); L0:=L+1
  1          END END END
```

Listing 3 shows another prototype in ALGOL, this time of a FORTH text editor. Here ATOM has become W and I am looking up plus, minus, and the commands T, R, A, and I, to edit a deck program.

Another method of implementing a dictionary is shown in listing 4. I am looking up the words in a conditional statement and setting NEXT, the key routine of modern FORTH's address interpreter, to the index.

Listing 5 shows an early implementation of a stack. Since it is written in BALGOL, which allows assignment statements inside other statements, I could replace STACK[J] with [J+1] in order to push items onto the stack. I did this so that I could manipulate parameters that were interpreted from the card deck as arguments to the routines. When I wanted, for instance, to convert angular measure from one unit to another, this added the ability to use arithmetic operators.

From Stanford I moved to the East Coast, where I programmed on a free-lance basis for several years. Some of you probably remember that, in the 1960s, a programmer at a typical computer center needed to learn about nineteen languages in order to function adequately: JCL (Job Control Language); languages to control utilities and facilities, such as the linking loader; assembly language and the assembler's control language; plus several high-level languages and the methods for controlling their compilers.

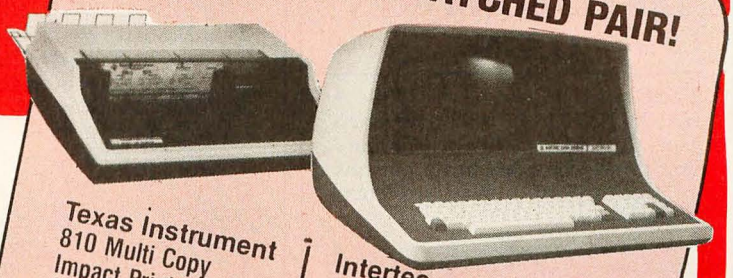
Listing 6 shows two of these languages, a PL/I program and the JCL necessary to run it. Note the obvious difference in syntaxes. FORTH developed in response to such conditions. In terms of modern FORTH, the importance of this example lies in the use of NEXT as a procedure that goes off to get the next word and do something with it.

Listing 7 shows a version of FORTH coded for the IBM System/360 with the routines PUSH and POP, which executed in about 15 μs. They include stack limit checking, which doubled the cost and was one of the things that led me to believe that execution-time stack checking is not desirable. This *was* coded in a macroassembler that did not have stack operations, which led to the deck full of statements like L19

SYNCHRO-SOUND

The ORIGINAL Computer People who KNOW Computers and offer EVERYTHING you need in Small Computer Systems

A PERFECT SUPER SPECIAL MATCHED PAIR!



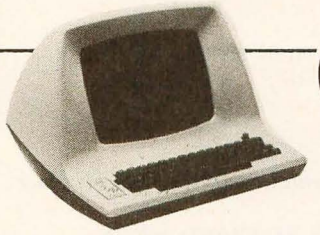
Texas Instrument 810 Multi Copy Impact Printer
150 characters per sec. bi-directional printing

Intertec Super Brain Computer Terminal
Dble. Density Dual Mini-Floppies, CPM based Development or Business System

ONLY $3995.

TERMINALS

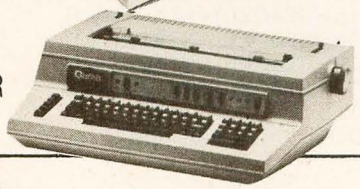
ADDS Regent 25



LEAR SIEGLER
ADM 3A
ADM 31
ADM 42

SOROC Technology
IQ 120
IQ 140

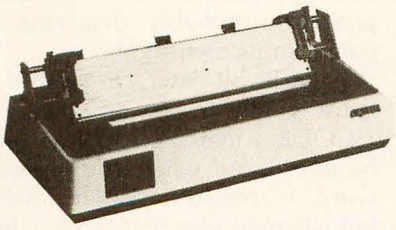
PRINTERS



QUME Sprint
5/45 KSR
5/55

DECwriter IV
LA 34

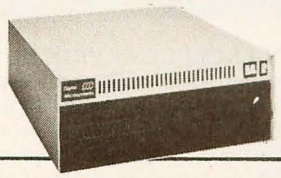
TELETYPE
43

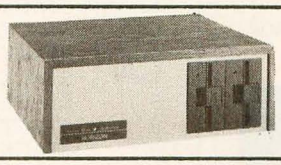


OKIDATA
Microline 80

COMPUTERS

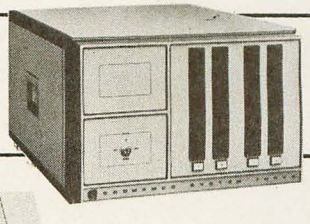
DIGITAL SYSTEMS
HDS 4004
14 Megabyte
Hard Disk

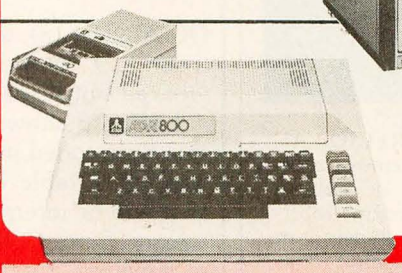




NORTHSTAR
HORIZON II
HORIZON II Quad

CROMEMCO
System 3





ATARI
400
800

MORE SPECIALS

Okidata SL125	$2595.00	Livermore Accoustic Coupler	$195.00
Okidata SL300	2995.00	Centronics Micro Printer	349.00
Persci 277	1395.00	5″ Scotch Diskette Box	34.95
Integral Data Systems Paper Tiger Printer	895.00	8″ Scotch Diskette Box	39.95
Televideo 912	825.00		
Televideo 920	895.00		

MANY OF OUR PRICES ARE TOO LOW TO ADVERTISE. PLEASE CALL OR WRITE

We carry a full line of Alpha-Micro Products.
We have a full staff of Programmers and Computer Consultants to design, configure and deliver a Turnkey Computer System to meet your specific requirements.

SYNCHRO-SOUND ENTERPRISES, INC.
THE COMPUTER PEOPLE
193-25 Jamaica Ave., Jamaica, New York 11423 • TWX 710-582-5886

PHONE ORDERS, CALL:
New York 212/468-7067
Los Angeles-213/628-1808
Chicago-312/641-3010
Dallas-214/742-6090

Listing 4: *An early version of the FORTH dictionary.*

```
  8 PROCEDURE RELEVANCE; BEGIN REAL T,K0;
  9    J:=0; I:=-1; WHILE WORD NEQ "END    " DO
180    IF W="=       " THEN NEXT:=3
  1    ELSE IF W="GT      " THEN NEXT:=4
  2    ELSE IF W="LT      " THEN NEXT:=5
  3    ELSE IF W="NOT     " THEN NEXT:=6
  4    ELSE IF W="AND     " THEN NEXT:=7
  5    ELSE IF W="OR      " THEN NEXT:=8
  6    ELSE IF W="+       " THEN NEXT:=9
  7    ELSE IF W="-       " THEN NEXT:=10
  8    ELSE IF W="*       " THEN NEXT:=11
  9    ELSE IF W="/       " THEN NEXT:=12
190    ELSE IF K0:=SEARCH1(W) GEQ 0 THEN BEGIN
  1       NEXT:=1; NEXT:=K:=K0 END
  2    ELSE BEGIN
  3       NEXT:=2;
  4       IF BASE[K]=" " THEN NEXT:=WORDS[0]
  5       ELSE NEXT:=W END;
  6    NEXT:=0 END;
```

Listing 5: *An early implementation of the FORTH stack, written in BALGOL.*

```
  7 BOOLEAN PROCEDURE RELEVANT; BEGIN
  8    I:=J:=-1; STACK[0]:=1; DO CASE NEXT OF BEGIN
  9       J:=-1;
210       STACK[J:=J+1]:=CONTENT;
  1       STACK[J:=J+1]:=NEXT;
  2       STACK[J:=J-1]:=REAL(STACK[J]=STACK[J+1]);
  3       STACK[J:=J-1]:=REAL(STACK[J] GTR STACK[J+1]);
  4       STACK[J:=J-1]:=REAL(STACK[J] LSS STACK[J+1]);
  5       STACK[J]:=REAL(NOT BOOLEAN(STACK[J]));
  6       STACK[J:=J-1]:=REAL(BOOLEAN(STACK[J]) AND  BOOLEAN(STACK[J+1]));
  7       STACK[J:=J-1]:=REAL(BOOLEAN(STACK[J]) OR BOOLEAN(STACK[J+1]));
  8       STACK[J:=J-1]:=STACK[J]+STACK[J+1];
  9       STACK[J:=J-1]:=STACK[J]-STACK[J+1];
220       STACK[J:=J-1]:=STACK[J]×STACK[J+1];
  1       STACK[J:=J-1]:=STACK[J]/STACK[J+1];
  2       END UNTIL J LSS 0;
  3    RELEVANT:=BOOLEAN(STACK[0]) END;
```

DC AL2(*-L18), which gave me a link from L19 to the previous label. It worked but it was not pleasant.

Listing 8 shows a similar routine, this time coded in COBOL. I am setting up a table of identified words that will be interpreted from an input stream. Since COBOL does not allow parameters for subroutines, it is awkward to do anything meaningful.

New Concepts

About this time, I began to think of defining a word that would define other words; and at that time, this idea was staggering. For example, { ;CODE } was a very esoteric word. I explained it to people, but I could not express the potential I thought it had.

It took time to find out exactly what { ;CODE } should do (it specified the code to be executed for a previously defined word). I do not have the records, but I think the initial code for { ;CODE } was three or four lines long; to simplify that code was one of the driving forces behind the address interpreter—to make it possible to code { ;CODE } cleanly. This had implications as to what registers should be available.

The fact that W should be saved in a register for defining words led to *indirect*, rather than direct, threaded code. That was the most complicated concept I had coded in this evolving program—probably deserving of a patent in its own right.

A little bit later, it seemed that there ought to be an analog of {;CODE } that specified the code to be *interpreted* when you executed a word. It seemed the natural balance, but when the idea first arose, I did not have the foggiest notion of what to do or what the implementation should be. The first definition of this analog, called { ;: } (semicolon-colon), required three or four lines of code. It had to do what { ;CODE } did, and then more.

Out of that came the distinction between compile-time action and

Please send your free software catalog.

(Check which software is of particular interest)

☐ C COMPILER. Optimized native code for VAX 11/780, PDP-11, LSI-11, Z80, 8085, 8080. Full C language as defined in Kernighan and Ritchie, with comprehensive portable library. Cross compilers available. Runs under VMS, IAS, RSX-11D, RSX-11M, RSTS/E, RT-11, UNIX, Idris, CDOS, CP/M. From $600

☐ IDRIS OPERATING SYSTEM. System calls and file system identical to UNIX V6, including pipelines. Utilities include shell, editor, assembler, loader, archiver, compare, copy, grep, etc., plus system utilities for file system maintenance. Runs on LSI-11, PDP-11. From $1000.

☐ PASCAL COMPILER. Optimized native code for VAX 11/780, PDP-11, LSI-11, Z80, 8085, 8080. Full Pascal language as defined in Jensen and Wirth, with standard library. Includes C compiler and portable library, permitting intermixed C and Pascal. Cross compilers available. Runs *under VMS*, IAS, RSX-11D, RSX-11M, RSTS/E, RT-11, UNIX, Idris, CDOS, CP/M. From $750.

Name_____

Company_____

Street_____

City_____ State_____ Zip_____

Whitesmiths, Ltd.
Software for grownups.
(212) 799-1200
P.O.B. 1132 Ansonia Station, New York, N.Y. 10023

execute-time action. It was convenient for words to be coded to act this way, but it was expensive. It required not only the address of the code to be executed, but the address of the code to be interpreted, as well as the parameter to be supplied to the code being interpreted so you could do something useful.

Late in the 1960s I went to work for Mohasco Industries, where I put something strongly resembling FORTH on a Burroughs 5500, cross-compiled to the 5500 from an IBM 1130. (There is no assembler on the 5500; there is a dialect of ALGOL called SBOL that Burroughs used to compile operating systems, not available to users.) Listing 9 shows the code definitions of stack operations on the 5500, which was a stack-oriented processor at a time when stack machines were not popular. The names of some FORTH stack operators stem from that machine's operations; see, for example, DUP . The symbol ¢ stands for CODE and distinguishes the assembler's OR from the FORTH OR . (Vocabularies were not yet available.)

Listing 10 gives an example of FIND (a dictionary search routine) coded for the 5500. Notice the word SCRAMBLE , a colon definition making a hashed search. Apparently I had eight threads to the dictionary here, a concept we added back to FORTH when we developed polyFORTH last year.

Listing 6: *The NEXT procedure in PL/I and its associated JCL (Job Control Language) statements (lines 1 thru 8).*

```
 1 //UTILITY   JOB  SYSTEM,OVERHEAD
 2 //          EXEC PGM=IEBUPDTE,PARM=NEW
 3 //SYSPRINT DD   SYSOUT=A
 4 //SYSUT2   DD   DSNAME=OUTLIB,UNIT=2314,DISP=(NEW,KEEP),
 5 //                VOLUME=SER=MOORE,SPACE=(TRK,(100,,10)),
 6 //                DCB=(RECFM=F,LRECL=80,BLKSIZE=80)
 7 //SYSIN     DD   DATA
 8 ./         ADD   NAME=WORD,LEVEL=00,SOURCE=0,LIST=ALL
 9    DECLARE KEYBOARD STREAM INPUT,PRINTER STREAM OUTPUT PRINT;
10    NEXT: PROCEDURE CHARACTER(4);
 1    DECLARE (1 TEXT CHARACTER(81) INITIAL((81)" "),
 2       2 C(81) CHARACTER(1), I INITIAL(81),W CHARACTER(4),
 3       WORD CHARACTER(32) VARYING BASED(P),P,NUMERIC BIT(1)) EXTERNAL;
 4    DO WHILE C(I)=" "; I=I+1;
 5       IF I=82 THEN BEGIN; I=1;
 6          READ FILE(KEYBOARD) INTO(TEXT); END; END;
 7    P=ADDR(C(I));
 8    IF C(I)="-" OR C(I)="." OR "0" LE C(I) THEN BEGIN; NUMERIC="1"B;
 9       IF C(I) NOT="." THEN DO I=I+1 BY 1 WHILE "0" LE C(I); END;
20       IF C(I)="." THEN DO I=I+1 BY 1 WHILE "0" LE C(I); END; END;
 1 ELSE DO; NUMERIC="0"B;
 2      IF "A" LE C(I) THEN DO I=I+1 BY 1 WHILE "A" LE C(I) OR C(I)="-";
 3      END; ELSE I=I+1; END;
 4    W=WORD; RETURN(W);
```

Listing 7: *The FORTH words PUSH and POP written in IBM 360 assembly language.*

```
0056                              830 L18 DC AL2(*-L17)
                                  831     NAME 3,X'445550',0     DUP
03445550                          832+          DC     AL1(3),X'445550'
00                                833+          DC     X'0'
                                  834+          ORG    *-2-V0
                                  835+          DS     0H
                                  836+          ORG    *+V0+1
400004                            837+          DC     AL1(0*X'40'+X'40'),AL2(4)
5AC0 6014              00014      838 PUSH A SP,MFOUR    COSTS 15 US
5040 C000              00000      839     ST T,0(,SP)
19CB                              840     CR SP,DP
0729                              841     BCR 2,NEXT     BHR
47F0 667C              0067C      842         B ABORT
001A                              843 L19 DC AL2(*-L18)
0444D2CF50400008                  844     DC AL1(4),X'44D2CF50',X'40',AL2(8)     DROP
41C0 C004              00004      845     LA SP,4(,SP)
5840 C004              00004      846 POP L T,4(,SP)     COSTS 21 US
41C0 C004              00004      847     LA SP,4(,SP)
59C0 602C              0002C      848     C SP, SP00
07C9                              849     BCR 12,NEXT    BNHR
47F0 667C              0067C      850         B ABORT
```

FORTH and the IBM 1130

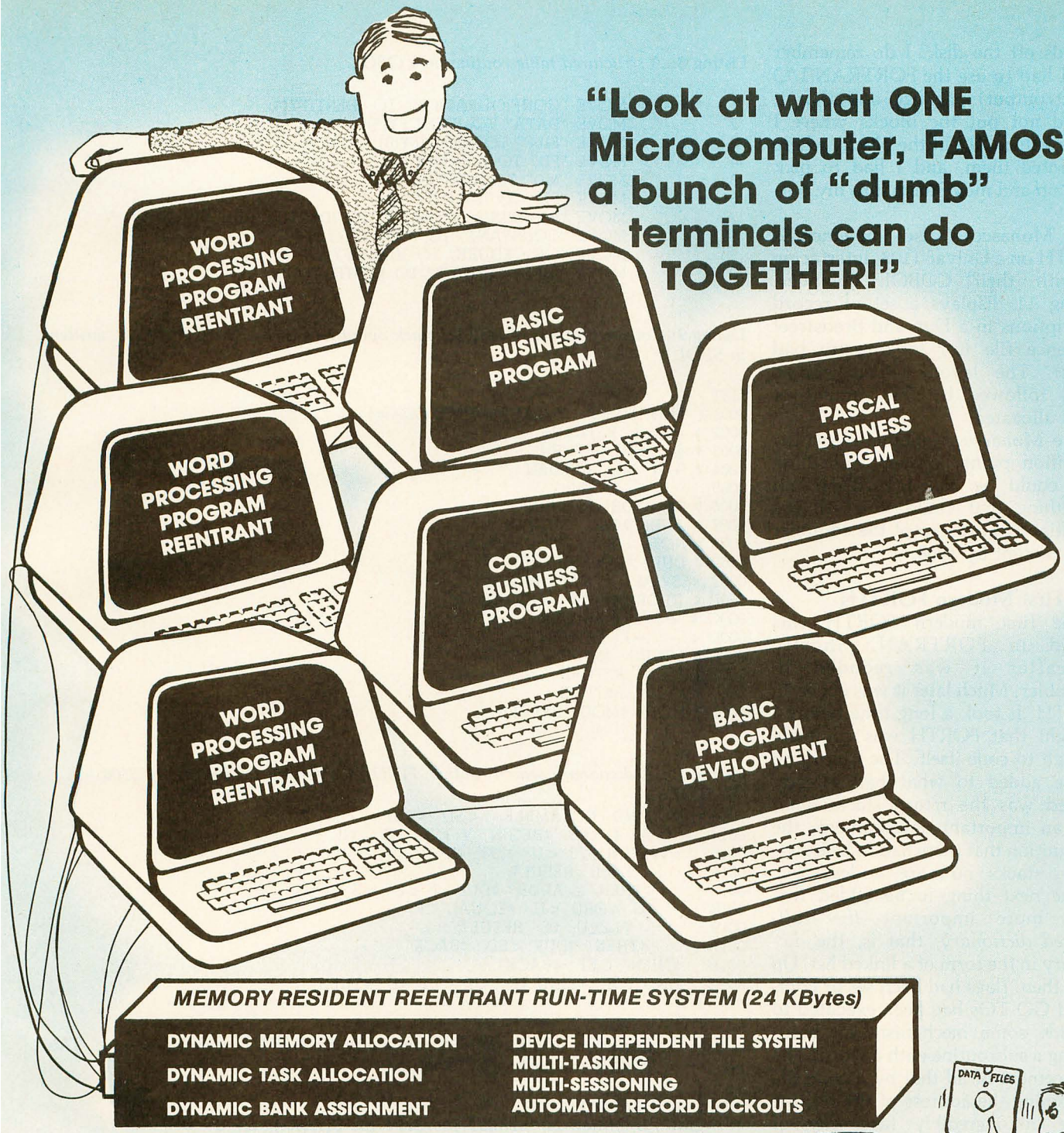
At Mohasco I also worked directly on an IBM 1130 interfaced with an IBM 2250 graphics display. The 1130 was a very important computer; it had the first cartridge disk, as well as a card reader, a card punch (as backup for the disk), and a console typewriter. The 1130 let the programmer, for the first time, totally control the computer interactively.

FORTH first appeared as an entity on that 1130. It was called F-O-R-T-H, a five-letter abbreviation of FOURTH, standing for fourth-generation computer language. That was the day, you may remember, of third-generation computers and I was going to leapfrog. But because FORTH ran on the 1130 (which permitted only five-character identifiers), the name was shortened.

What came out of the 1130 was a cross-assembler that assembled the instructions, which were then to be executed by the 2250. I think the 2250 had its own memory, and these things had to be programmed carefully. What I accomplished was that the 1130 in FORTRAN in 32 K bytes could draw pictures on the 2250, fairly slowly; and FORTH, in 8 K bytes, could draw three-dimensional moving pictures on the 2250—but it could do that only if every cycle was accounted for and if the utmost was squeezed out. That is why FORTRAN had to go—I required an assembler and could not do an impressive enough job with FORTRAN.

But high-level or colon definitions were not yet compiled—the compiler came much later. The text was stored in the body of the definition, and the text interpreter reinterpreted the text in order to discover what it was to do. This contradicts the efficiency of the language, but I had big words that put up pictures and I did not have to interpret too much. The cleverness was limited to squeezing out extraneous blanks as a compression medium. I am told that this is the way that BASIC acts today in many instances.

This machine had a disk drive, *and* I am almost certain that the word BLOCK existed in order to access



"Look at what ONE Microcomputer, FAMOS,™ & a bunch of "dumb" terminals can do TOGETHER!"

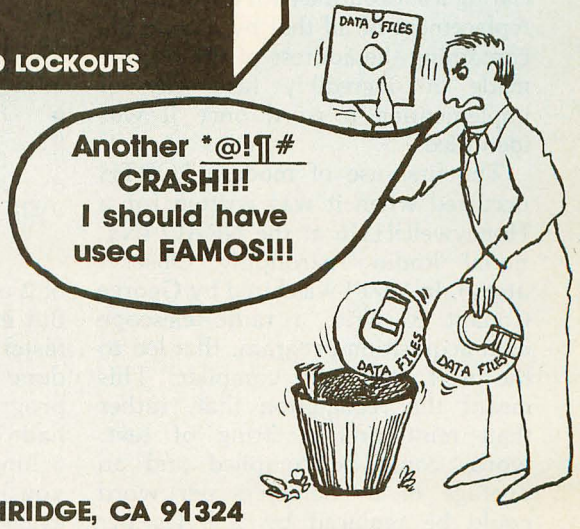
MEMORY RESIDENT REENTRANT RUN-TIME SYSTEM (24 KBytes)

- DYNAMIC MEMORY ALLOCATION
- DYNAMIC TASK ALLOCATION
- DYNAMIC BANK ASSIGNMENT
- DEVICE INDEPENDENT FILE SYSTEM
- MULTI-TASKING
- MULTI-SESSIONING
- AUTOMATIC RECORD LOCKOUTS

NEW BASIC COMPILER

- OPTIMIZED CODE
- AUTO INTEGER REPRESENTATION
- VARIABLE PRECISION (4-18)
- MULTIPLE KEY ISAM
- REENTRANT OBJECT & RUN-TIME
- SCREEN HANDLING UTILITY
- Z80 VERSION AVAILABLE
- BATCH OPERATIONS UNDER BASIC

Over 500 users.... APPLICATIONS AVAILABLE



MVT MICROCOMPUTER SYSTEMS INC.

9241 RESEDA BLVD., SUITE 203, NORTHRIDGE, CA 91324
Phone: (213) 349-9076 TWX 910 493-2291 MVT NTGE

System Software with unique file integrity.... for the OEM & Manufacturer.

records off the disk. I do remember that I had to use the FORTRAN I/O (input/output) package and that it would not put the blocks where I wanted them; it put the blocks where *it* wanted them, and I had to pick them up and move them into my buffers.

At Mohasco I also implemented FORTH on a Univac 1108, interfacing it with their COBOL compiler. Listing 11 displays a set of record descriptions in a Dun and Bradstreet reference file (for looking up bad debts). The layout shows named fields followed by the number of bytes allocated.

The Mohasco programs mark the transition point between something that could be called FORTH and something that could not. All the essential features except the compiler were present by 1968.

The First Modern FORTHs

The first modern FORTH was coded in FORTRAN. Shortly thereafter it was recoded in assembler. Much later it was coded in FORTH. It took a long time before I thought that FORTH was complete enough to code itself. The first thing to be added to what had already existed was the return stack. That was an important development; the recognition that there had to be exactly two stacks, no more, no less.

The next thing to be added was even more important—*the full-fledged dictionary*, that is, the dictionary in the form of a linked list. Up until then, flags had been set or computed GO TOs had been executed to provide some mechanism for associating a subroutine with a word. The replacement of all that by a code file containing the address of the routine made an incredibly fast way of implementing a word once it was identified.

The first use of modern FORTH occurred when it was written for a Honeywell H316 at the NRAO (National Radio Astronomy Observatory). In 1971 I was hired by George Conant to write a radio-telescope data-acquisition program: that led to the next step, the compiler. This meant the recognition that, rather than reinterpret a string of text, words could be compiled and an average of 5 characters per word could be replaced by 2 bytes per word. This gave a compression factor of 2 or 3, not drastic but appreciable. But execution speed would be much faster. Again I asked myself, as I had done when I first began modifying programs: if it was that easy, why hadn't anyone else done it? It took me a long time to convince myself that you could compile anything and everything.

Interrupts came around this time. It was important to utilize the interrupt capability of the computer, but it had not been done by me before that—I did not know anything about interrupts. I/O, however, was not yet interrupt-driven. Interrupts were available for the application if it wanted them—FORTH did not bother.

The multiprogrammer came along

Listing 8: *A structured table routine, in COBOL.*

```
 1            MOVE "CONFIGURATION" TO IDENTIFY(4);
 2            MOVE "DATA" TO IDENTIFY(5);
 3            MOVE "FILE" TO IDENTIFY(6);
 4            MOVE "FD" TO IDENTIFY(7);
 5            MOVE "MD" TO IDENTIFY(8);
 6            MOVE "SD" TO IDENTIFY(9);
 7            MOVE "WORKING-STORAGE" TO IDENTIFY(10);
 8            MOVE "CONSTANT" TO IDENTIFY(11);
 9            MOVE "PROCEDURE" TO IDENTIFY(12);
70            MOVE "INPUT-OUTPUT" TO IDENTIFY(13);
```

Listing 9: *Code definitions of FORTH stack operations on the Burroughs 5500, written in SBOL.*

```
LIST
0001 ( 'PRIMITIVES'  26 LAST   =   30 SIZE=)
0002 ¢ = _S  RETURN
0003 ¢ @ <SD  RETURN
0004 ¢ + V 241,  RETURN
0005
0006 ¢ OR  ¢OR  RETURN
0007 ¢ AND  ¢AND  RETURN
0008 ¢ NOT  115,  RETURN
0009 ¢ DUP  ¢DUP  RETURN
000A ¢ SWAP  ¢SWAP  RETURN
000B ¢ DROP  ¢DROP  RETURN
000C ¢ +   +1  RETURN
000D ¢ -   -1  RETURN
000E ¢ MINUS  ¢MINUS  RETURN
000F ¢ *   *1  RETURN
0010 ¢ /   /1  RETURN
0011 ¢ MOD  ¢MOD  RETURN
```

Listing 10: *A dictionary search routine, FIND , written for the Burroughs 5500.*

```
0013 ¢SM ¢FIND  SCRAMBLE   <SD  ¢DUP
0014  41 >A  41 >B  ¢BEGIN  V <U  1771, ¢IF
0015     ¢BEGIN  V0 <U  1771, ¢IF
0016         1 <L  RESULT
0017       ¢THEN  _ADDR  ¢DUP  1 <L  <S
0018       0S WORD <U  ¢EQUAL ¢IF
0019         V1 _U  0S  RESULT
001A       ¢THEN  ¢DUP <SD  ¢BACK
001B   ¢THEN  GET  ¢BACK
001C : FIND TOP  ¢FIND ¢IF  UR <UD ¢B  ¢THEN;
```

Listing 11: *Prototype of a file layout, running under FORTH on a Univac 1108. This version of FORTH was written in COBOL.*

```
3 DBI   DBI/MOORE 33 33
4    DUNS 8  NAME 24  STREET 19  CITY 15  STATE 4  ZIP 5
5    PHONE 10  BORN 3  PRODUCT 19  OFFICER 24  SIC 4  SIC1 4  SIC2 4
6    SIC3 4  SIC4 4  SIC5 4  TOTAL 5.0 EMPL 5.0 WORTH 9.0 SALES 9.0 MFG 1
7    SUBS 1  HDQ 1  HEAD 8  PARENT 8  MAIL 19  CITY1 15  STATE1 4
8    NAME1 19
9 END
```

For those who want to test the water before jumping in.

picoFORTH

If you're thinking of getting into polyFORTH and you'd like an introduction through hands-on experience, then picoFORTH is for you. picoFORTH has been designed by FORTH, Inc. to serve as your entry into a complete polyFORTH programming environment.

picoFORTH™ is a disk-based operating system and interactive high-level language, complete with compiler, editor, and assembler. It's upgradable to full polyFORTH.™ And it's priced at only $495.

So step forth and get your feet wet. The water's fine.

For information, call:

213/372-8493

FORTH, Inc.

2309 Pacific Coast Highway
Hermosa Beach, California 90254
(213) 372-8493
TWX 910-344-6408 (FORTH INC HMBH)

Circle 55 on inquiry card.

a couple of years later when we developed an improved version of the system for NRAO's telescope at Kitt Peak. This computer was a PDP-11; the multiprogrammer had four tasks. Input was still not interrupt-driven, which was unfortunate.

The Second FORTH Programmer

Ten years ago there was one FORTH programmer, me. The second FORTH programmer, Elizabeth Rather, came along in 1971. That is quite a quantum jump, from one to two; the next step was four (the next two came out of Kitt Peak National Observatory); the growth can be traced from there to the several thousand today.

The first FORTH user was Ned Conklin, head of the NRAO station at Kitt Peak, Arizona. NRAO runs a millimeter-wave radio telescope that is in great demand by observers, in part because it is responsible over the last 10 years for discovering half of the interstellar molecules that are known to exist. FORTH is still running on that telescope at Kitt Peak and on a lot of other telescopes.

Given interest from other astronomers, a few believers split off from NRAO in 1973 and formed FORTH Inc. We were deluged by requests for FORTH systems from astronomers and went into business to try to exploit that market. It would still be our principal line of business today except that there are so few new telescopes in the world that you cannot support a company on that market.

We developed miniFORTH™ (FORTH on minicomputers) with the idea of having a programming tool. An important implementation of the tool came when we put an LSI-11 and FORTH into a suitcase. I think I became the first computer-aided programmer—computer-aided in that I had my computer and took it around with me. I talked to my computer, my computer talked to your computer, and we could communicate much more efficiently than I could communicate directly with your computer before it could run FORTH. Using this tool, we have put FORTH on many computers.

We added the feature of interrupt-driven I/O when FORTH Inc produced its first multiterminal system. It did not speed things up particularly from the user's point of view, but it *did* prevent any loss of characters when several people were typing at the same time. You did not have to look quickly to get the character before the next one came along. They were all buffered and waiting for you, which is an important distinction for multiprogrammed systems.

Data-base management came along at this time. It has been extensively changed, just as FORTH has. But fundamentally, nothing has changed. The concept of files, records, fields, and relational pointers that polyFORTH™ offers dates back from 1974 or so—years and years ago. Listing 12 shows a recent application of the FORTH Inc data-base management system.

With microFORTH™ in 1976 came the first version of our current target compilers. They are very complex things, much more so than I expected them to be. At about the same time, we worked out the current implementation of DOES> .

This new form of { ;: } does not require the address of the code to be interpreted. Since that is supplied by a different mechanism, the parameter can occupy the parameter field as it is supposed to. You can "tick" it and change its value, which is nice. [*The FORTH word { ' } (called "tick" above) places the address of the word that follows it onto the stack....***GW**] But we save 2 bytes for every DOES> word, 2 bytes for very common words—and for 3 years, we did

Listing 12: *Field and record layouts for a recent FORTH Inc data-base management system.*

```
64 LIST

 0   ( GLOSSARY FILE)
 1 2  ( LINK)   12 BYTES WORD    12 BYTES VOC
 2   NUMBER SOURCE    NUMBER STACKS    70 BYTES PHRASE
 3   210 FILLER ( 4 LINES)    32 FILLER ( 340 B/R, 3/BLOCK) DROP
 4 2   24 BYTES WORD+VOC    DROP
```

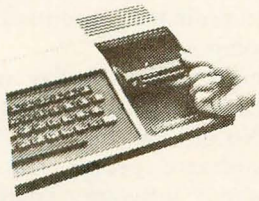
master charge THE INTERBANK CARD

VISA

free freight!

MICROWORLD®

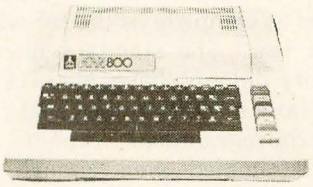
New Products



Texas Instruments 99/4 Home Computer

FREE Solid State Speech Synthesizer! The 99/4 talks! Superior sound, 16-color graphics; price includes 13" color monitor and TI BASIC.

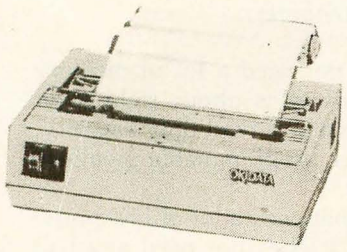
List Price: $1549.95
Your Price: $949.00 complete!



Atari 800

FREE joysticks and FREE Star Raiders game! System features expandable memory, advanced components and comprehensive software library; a "timeless" home computer!

List Price: $1159.90
Call For Special Price!



Okidata Microline 80

Compact, low-cost 80 cps printer; 9 x7 matrix . . . friction or pin feed! 80 col. w/132 column compressed print graphics, and more! Tractor fed optional.

Call for New Price

Prices subject to change without notice; products subject to availability.

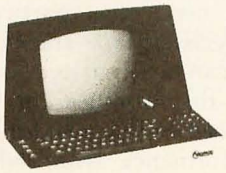
Complete Systems . . . Solution Packages from MicroWorld®

MicroWorld recognizes the need to provide customers with more than just hardware and peripherals. We provide total, integrated systems as part of a complete solution package . . . including all hardware, interfacing, software and operating systems required.

Our systems are set up especially for your own individual needs, based on the most reliable computers and peripherals, selected specifically for your environment, application and budget. Our staff is here to assist you in planning your data management needs.

Systems and solutions . . . the expertise of industry-trained professionals . . . backed by commitment and integrity. From MicroWorld; the source you can trust.

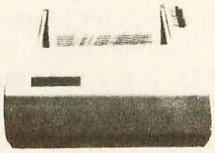
Soroc IQ 120



High-quality, text-editing terminal, 73-key board, built-in 2K RAM, RS232 interface.

List Price: $995.00
Special! $729.00

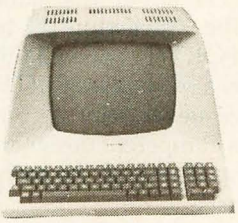
Comprint GP



Low-priced electrostatic matrix printer, 225 cps; ideal for personal applications requiring second printer.

Call for Price!

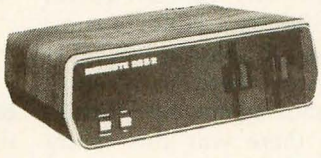
Televideo 920C



Low-cost terminal loaded with features; full-function keyboard, 24x80 display, blink, reverse, self-test!

List price: $1030.
Your price: $820.

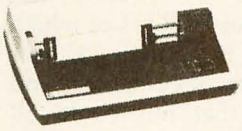
Dynabyte System DB8



Compact system with two Z80 processors, two serial and one parallel port, dual 8" floppies . . . double or quad density! Hard disk storage available.

Call for Price!

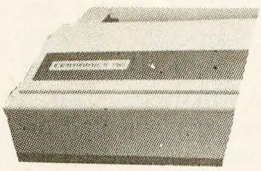
TI 810 Fully Loaded



RO printer; low price includes full ASCII, vertical forms control, compressed print, 150 cps, RS232, tractors, 3" to 15" form width, bidirectional printing!

Special! $1799.00

Centronics 737



Low-cost 50-90 cps, proportional spacing RO printer. Cable interface, generates full ASCIi, pin or friction feed!

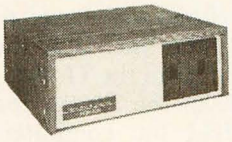
Call for Price!

Zenith-Health Data Systems Z-89 All-in-One Computer (16K)

FREE HDOS! Two-80 processors, minifloppy drive, 25 x80 display, memory expandable to 48K!

Call for Price!

NorthStar Horizon



Quad or double-density! Plus, hard disk drives for expansive storage requirements.

Call for Price!

TOLL-FREE 1-800-528-1418

1425 W. 12 Pl. • Tempe, AZ 85281 • 602-894-1193

not realize that we had missed the optimum by so much.

I know no way of speeding this process from initial thought to development, except to let a certain amount of time pass. We could sit, we did sit and debate this thing endlessly, and we missed the obvious.

I think that completes the capabilities that I think of as FORTH today. You see how they dribbled in—at no point did I sit down to design a programming language. I *solved* the problems as they arose. When demands for improved performance came along, I would sit and worry and come up with a way of providing improved performance.

polyFORTH is a condensation of everything that we at FORTH Inc have learned in the last 10 years of developing FORTH. I think it is a very good package. I foresee no fundamental changes in the design of the language except for accommodation to FORTH standards, which are becoming increasingly important.

Implementations of FORTH

I would like to review the implementations of FORTH of which I am aware. It is actually a tour through the history of computers and it is fascinating that this could all have happened in 10 years.

FORTH has been programmed in FORTRAN, ALGOL, PL/I, COBOL, assembler, and FORTH; and I am sure some of you can come up with other languages with the same history. My list is strictly personal.

FORTH has been implemented on the Burroughs 5500; the IBM 1130; the Univac 1108; the Honeywell 316; the IBM 360; the Data General Nova; the HP 2100 (not by me but by Paul Scott at Kitt Peak); the PDP-10 and PDP-11 (by Marty Ewing at the California Institute of Technology); the PDP-11 (by FORTH Inc); the Varian 620; the Mod-Comp II; the GA SPC-16; the CDC-6400 (by Kitt Peak); the PDP-8; the IV-Phase; the Computer Automation LSI-4; the RCA 1802; the Honeywell Level 6; the IBM Series 1; the Interdata; the 6800; the 8080; the 8086; the TI-9900; and soon the 68000, the Z8000, the 6809, and a Child Inc computer. Some independent groups have 6502s, ILLIAC, and others running FORTH. I raise the question—is it the case that FORTH has been put on every computer that exists?

Some people think FORTH ought to be machine independent, but that premise is wrong. The equivalence is FORTH—each machine requires meticulous attention to its individual characteristics. You must use all the hardware capabilities of each machine and must then work to force it into the mold specified by FORTH's virtual machine.

For example, we put a subset of FORTH on an SMS-300 microcomputer. It had only eight instructions. The internal characteristics of every machine can and must be exploited. You do not need any particular number of registers or stacks or anything. All can be simulated, but if you neglect the abilities of the machine, you can end up a factor of 2 down in performance from where you might otherwise be.

> At no point did I sit down to design a programming language. I solved the problems as they arose.

FORTH-in-Hardware Computers

The first FORTH computer I know of was built at Jodrell Bank in England around 1973. It is a redesign of an English Ferranti computer that went out of production. The observatory at Jodrell Bank was going to build their own bit-slice version; they discovered FORTH about the same time, modified the instruction set to accommodate FORTH, and built what I am told is a very fast FORTH computer. I have never seen it, but have talked to its competent designer, John Davies, who is one of the early FORTH enthusiasts.

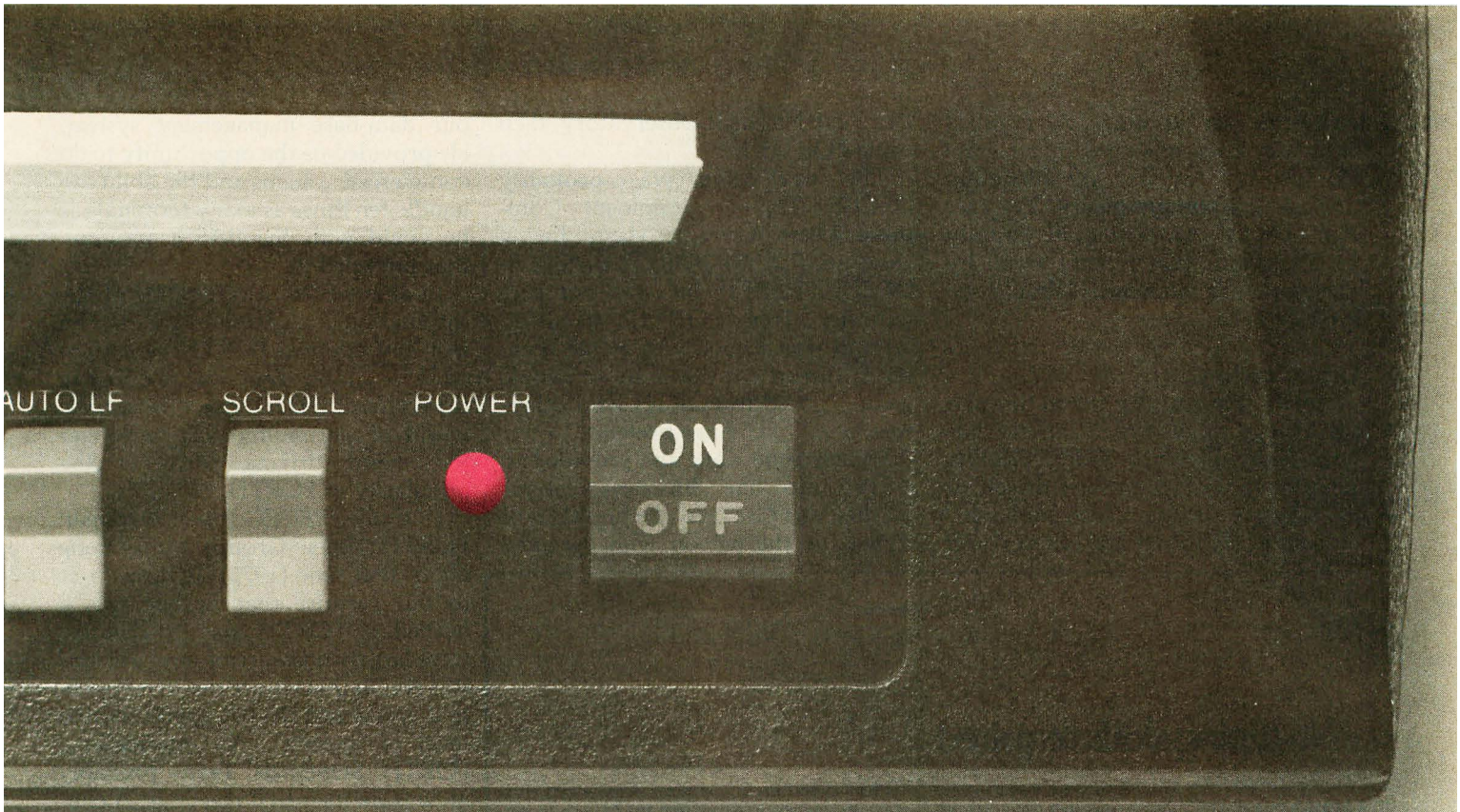
In 1973, before Dean Sanderson came to FORTH Inc to develop microFORTH, he had a FORTH computer at a company called General Logic. It qualifies as a FORTH computer because it has a FORTH instruction. And there is a story there. Dean showed me his instruction set, and there was this funny instruction that I could not see any reason for—I figured it was some kind of no-op or catchall or something; it had the weirdest properties, and it could not possibly be useful. It was NEXT. It was a one-instruction NEXT which was beautiful. And it was a very *simple* modification (this was a bit-slice computer) to the instruction set—a few wires here and there—and that is the first time I saw a FORTH computer, if you will. I call it a FORTH computer because it had the ability to change itself from an ordinary computer into a FORTH computer.

I think that hardware today is in the same shape as software was 20 years ago. No offense, but it is time that the hardware people learned something about software. There is an order or two of magnitude improvement in performance possible with existing technology. We do not need picosecond computers to make really substantial improvements in execution speed. Faced with that realization, there is no point in trying to optimize the software any further until we have taken the first crack at the hardware. The hardware redesign has to be as complete as the software redesign was. The standard microprocessors did not have FORTH in mind. Those minicomputers that can be microprogrammed cannot be microprogrammed well enough to even be worth doing. The improvements available are much greater than you can achieve by these half measures.

I have built a small FORTH computer. The design changes as fast as the chips can be plugged into the board. But it is not difficult to do. Here are the characteristics of a FORTH computer:

- It does not need a lot of memory (16 K bytes is about right—half programmable read-only memory, half user programmable memory, maybe).
- It does not need a lot of I/O ports; in fact, it does not need any I/O ports except for the application requirements.
- A serial line and interface to a disk drive are useful but not required.

We have put FORTH on an 8080-based machine with a virtual disk in memory, enough memory to hold eight blocks. The system is quite viable and has no particular problem with system crashes. Bubble



Diablo printers spend more time in this position.

A printer isn't much good if it can't do the job when it's needed.

That's why, at Diablo, we don't just design printers that work. We design printers that keep on working. In fact, we make them so reliable, you can just open the carton, plug in and play.

Diablo offers the widest range of reliable printers and options to give the flexibility you need. Which stands to reason. After all, we pioneered the daisy wheel technology and we're still the leader in it.

So if your printers spend too much time in the "off" position, you know what to do.

Switch.

Diablo Systems

XEROX

memories are coming. A FORTH computer does not need much mass storage; 100 K bytes are adequate, and 250 K bytes are plenty. The fact that FORTH can exist quite happily on a machine that is very small by contemporary standards should be exploited.

Organizations

Finally, I would like to run through the history of the organizations that have been involved with FORTH. They have formed another thread of the tapestry. It began with Mohasco, of course, followed by NRAO and Kitt Peak National Observatory: then came FORTH Inc.

The next step was probably DECUS (Digital Equipment Computer Users' Group). Marty Ewing gave his PDP-11 FORTH system to DECUS. FORTH Inc was not sure whether free FORTHs floating around was a good idea at the time. But it turned out that a lot of people were exposed to FORTH who otherwise would not have been.

Cybek came along and provided an entry into the business-systems market. Art Gravina, the president of Cybek, is the person who designed our data-base management system. He provided us the opportunity to do commercial systems and the ability to handle ten times as many terminals as he could with the BASIC program that preceded it.

In about 1976, a committee of the International Astronomical Union met and adopted FORTH as a standard language. That was a boost in the world of astronomy, although the world of astronomy was no longer the major driving force in the popularity of FORTH.

I think EFUG (the European FORTH Users' Group) came along about that time (1976). It turned out to our surprise that Europe was a hotbed of FORTH activity that we were largely unaware of (and perhaps still are, in that we are not involved in that world and do not appreciate the level of interest). An international FORTH Standards Team probably grew from their first meetings. A couple of years later, the FORTH Interest Group started. Now we have FORML—FORTH Modification Laboratory, an idea-generating organization.

Conclusion

The tendency seems to be for people to organize themselves in groups. Some of these groups are companies, others are associations. It looks like FORTH is going to be a communal activity in that sense—that it will grow from the work of unstructured clusterings of like-minded people. The suggestion is that this whole world of FORTH is going to be quite disorganized, uncentralized, and uncontrollable. It's not bad, perhaps it's good.

My view of the future is more unsettled today than it has been for years: promising, confusing, perplexing. The implications are perhaps as staggering now as they were 20 years ago. The promise of realization is much higher. My original goal was to write more than forty programs in my life. I think I have increased my throughput by a factor of 10. I do not think that that throughput is program-language limited any longer. So I have accomplished what I set out to do: I have a tool that is very effective in my hands. It seems it is very effective in others' hands as well. I am happy and proud that this is true.

WordPro™

"WordPro is the most sophisticated Word Processing Software package available for the Commodore Computer line."

Solve Your Paperwork Problem . . . Let WordPro Software Do The Work

Using standard typing methods, hundreds of valuable hours are spent erasing, revising, and retyping letters and documents as you work towards a final draft copy. The second, third, or fourth drafts take just as long to type as the first!

With WordPro word processing software you can transform your Commodore computer into a "state of the art" word processing machine with sophisticated word processing features at an affordable price.

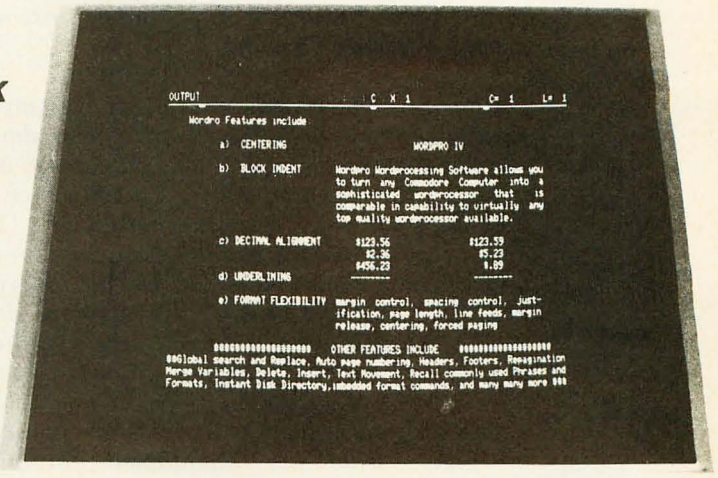
There are four versions of WordPro, ranging from the simple to the sophisticated. WordPro 1 on cassette will give computer enthusiasts a full range of text editing capabilities with cassette file storage. WordPro 2 is disk based and allows fast and easy file handling and manipulation. WordPro 3 was designed for professionals and contains the many features required in a business environment such as global search and replace, headers, footers, decimal tabulation, repagination, merging capabilities, and much, much more. WordPro 4 is our best. WordPro 4 runs on the new Commodore 8032, 80-column display computer. WordPro 4 has all the features of WordPro 3, plus additional features usually found only on the most sophisticated and expensive word processing equipment.

WordPro is a new breed of word processing software. Powerful, sophisticated, and easy to use, WordPro was field-tested by dozens of attorneys and commercial customers during 1979. WordPro is now installed and is saving its owners valuable time and money in hundreds of offices nationwide.

WordPro was designed with the user in mind. WordPro's unique "STATUS LINE" constantly interacts with the user by displaying the status of the system. Editing, storing documents, recalling letters, even the most sophisticated comands, are accomplished by a few, easy to remember, keystrokes.

You may find that WordPro alone is reason enough to own a computer.

WordPro can be found at most Commodore dealers worldwide. Call us for the number of the dealer nearest you. If you cannot locate a stocking WordPro dealer you may place an order with Professional Software via check or VISA/MasterCharge.



Actual Photograph of WordPro on CBM Model 8032

The many features of WordPro 1 - 4:

WordPro 1 - Cassette based • Status line • Test Editing • Insert/Delete • Screen Scroll Auto Repeat • String Search • Erase Functions • Link Files • Margin Controls • Tab Functions • Justification • Page Length

WordPro 2 - Most WordPro 1 Functions Plus + Disk Based • Paragraph Indent • Centering • Text Transfer • Hyphenation • Appending • Margin Release • Variable Blocks (Form Letters) • Multiple Copies • Automatic Disk Commands • Complete Disk File Handling

WordPro 3 - Commercial Disk Version for 40 Columns • WordPro 2 Functions Plus + Global Functions (Search/Replace/Copy) • Merging Disk File Linkage • 10 or 12 Pitch • Repagination • Duplicate Lines • Auto Delete Word/Sentence/Range • Numeric Mode • Underlining • Continuous Print • Headers/Footers • Auto Page Numbering • Proportional Justification • Forced Paging • Non-Print Comments • BASIC Language File Compatibility

WordPro 4 - Commercial Disk Version for 80 Columns • WordPro 3 Functions Plus + Displays and Formats Text to Screen for Review

WordPro 1 — For all 8K RAM units. Requires C2N Peripheral/integrated cassette drive - **$29.95**

WordPro 2 — For all 16K RAM units with 40 column screen. Requires 2040 disk drive - **$99.95**

WordPro 3 — For all 32K RAM units with 40 column screen. Requires 2040 disk drive - **$199.95**

WordPro 4 — For Model 8032 with 80 column screen. Requires 2040 or 8050 disk drive - **$299.95**

All four versions of WordPro are written in 6502 machine code.

Professional Software Inc.

166 Crescent Rd., Needham, MA 02194

(617) 444-5224

WordPro Dealer Inquires Invited

WordPro was developed by Steve Punter of Pro-Micro Software Ltd., and is marketed exclusively by Professional Software Inc.

Components of FORTH

FORTH is characterized by five major elements: dictionary, stack, interpreters, assembler, and virtual memory. Although not one of these is unique to FORTH, their interaction in FORTH produces a synergistic effect that creates a programming system of unexpected power and flexibility.

- Dictionary: *The resident FORTH system is organized into a dictionary that occupies almost all of program memory. The dictionary is a threaded list of variable-length items, each of which defines a word of the vocabulary. The actual content of each definition depends on the type of word: noun, verb, etc. The dictionary is extensible, growing toward high memory. In a multiterminal system, terminal tasks may have private dictionaries that are connected in a hierarchical tree structure.*
- Stack: *Two push-down stacks (last-in, first-out, or LIFO, lists) are maintained for each multiprogrammed task in the system. These provide the primary communication between routines as well as an efficient mechanism for controlling logical flow. A stack normally contains items one computer word long, which may be addresses, numbers, or other objects. Stacks, which are of indefinite size, grow toward low memory.*
- Interpreters: *FORTH is fundamentally an interpretive system, meaning that program execution is controlled by data items rather than by machine code. It is a common assumption that interpreters are severely wasteful of processor time; this is avoided in FORTH by maintaining two levels of interpretation.*

The first of these is the text interpreter, also known as the outer interpreter. *It works in a conventional manner, parsing text strings that come from terminals or mass storage and looking up each word in the dictionary. When a word is found in the dictionary, it is executed (unless the task is in compile mode) by invoking the address interpreter.*

The address interpreter (also known as the inner interpreter*) interprets strings of absolute memory addresses by executing the definition pointed to by each. Most dictionary definitions contain addresses of previously defined words that are to be executed by this interpreter. This level of interpretation requires no dictionary search since these words have already been compiled by the text interpreter, which generated the absolute addresses.*

The address interpreter has several important properties. First, it is fast. *Indeed, on some computers it executes only one instruction for each word, in addition to the code implied by the word itself. Second, it interprets compact* definitions. *Each word referenced in a definition compiles a single memory location. Finally, the definitions are* machine independent *because the definition of one word in terms of others does not depend upon the computer that interprets the definitions.*

- Assembler: *FORTH includes a resident assembler, which allows the programmer to define words that will cause specified machine instructions to be executed. This type of definition is necessary to perform device-dependent input and output operations, to implement elementary operations, and to do highly time-critical processing.*
- Virtual memory: *The final key element of FORTH is its blocks: fixed-length segments of disk space that may contain program text or data. A number of buffers are provided in memory; blocks are read into them automatically when referenced. If a block is modified in memory, it is automatically replaced on disk. Explicit read and write operations, therefore, are not required; programmers may presume that program text or data is in memory whenever it is referenced.*

[The above paragraphs present a concise overview of FORTH as a language; the following paragraphs describe features of a FORTH Inc product, polyFORTH...GW]

The standard polyFORTH system utilities include the following:

Text editor:	*Facilitates editing program source text, both by line and by character.*
Source listings:	*Prints program source listings and indexes.*
Disk copy:	*Provides for disk-to-disk copying of data file and program source files for backup purposes.*
Disk diagnostic:	*Produces a simple, read-only disk diagnostic that may be run at any time without disturbing other users. (More extensive hardware diagnostics are optional.)*

Each polyFORTH system also contains a Target Compiler™ *capability; this allows the user to develop, for run-time applications only, a computer system that does not require the entire operating system. Since FORTH is an interpretive language, an interpreter must always be present; but the target compilation process creates the minimum dictionary necessary, thus allowing a program to be run with a minimum of memory overhead. Typically, this overhead is less than 1000 bytes.*

Full data-base management support is available in an optional Extended File Management package. Included within its structure are the essential features of the CODASYL standard along with the characteristic speed, compactness, and flexibility of the FORTH language. Facilities include commands for file definition and formatting and for field and record descriptions, as well as several file-accessing techniques, operators for accessing individual fields by name and fields within specified files, and such utility functions as a report generator and an optional key-sort routine. ■

This selected list of FORTH vendors is meant to be an overview only. For complete details contact the vendors. Many of the products, listed as fig-FORTH versions, are implementations of the FORTH Interest Group software customized for a given machine and available in machine-readable (as opposed to printed) form.

When purchasing a version of FORTH, check to see what source the version is based upon. All good versions of FORTH are based on either the FORTH Inc or the FORTH Interest Group versions. Some existing implementations use nonstandard shortcuts that limit the usability of the product; these should be avoided.

Literature on FORTH is scarce, so be prepared to puzzle through cryptic documentation. Miller Microcomputer Services offers a wide selection of books on FORTH (the only selection we know of). Particularly suitable are microFORTH Primer *(supplied with the purchase of MMSFORTH) and* Using FORTH, *both written by FORTH Inc.*

*STOIC is a FORTH-like language available from the CP/M Users' Group and is listed because of its low price. N/A refers to information unavailable at the time this table was compiled....***GW** *and* **CHF**

Selected FORTH Vendors

Manufacturer	Product Name(s)	Machine Requirements	Format	Cost	Notes
Acropolis Software 17453 Via Valencia San Lorenzo CA 94580 (415) 276-6050	A-FORTH (based on fig-FORTH)	Any machine running Micropolis disks and MDOS operating system	Floppy disk	$150	Includes 8085 assembler, double-precision fixed-point math, enhanced disk access, other features.
Cap'n Software POB 575 San Franciso CA 94101 (415) 540-0202	fig-FORTH	Apple II with disk	5-inch floppy disk	$140	FORTH hot line available for questions. Extra packages (Apple high-resolution graphics, floating-point) available at extra cost.
CP/M Users Group 1651 Third Ave New York NY 10028	STOIC (*not* FORTH, but a FORTH variant)	Any CP/M machine.	8-inch floppy disk	$20	See editorial for further details.
FORTH Inc 2309 Pacific Coast Hwy Hermosa Beach CA 90254 (213) 372-8493	FORTH, polyFORTH, microFORTH, picoFORTH	Versions for various machines: 8080, 8086, 6800, 1802, LSI-11; also handles versions for minicomputers and mainframes	Varies with machine.	$2500 up ($495 for picoFORTH)	These are the inventors of the language; they supply custom packages and extensive support; picoFORTH (for 8080 or 1802) can be directly upgraded to polyFORTH.
FORTH Interest Group POB 1105 San Carlos CA 94070	fig-FORTH	Various machines with 16 K bytes or more: 8080, 6502, 6800, LSI-11, 9900, PACE; disk preferable	Printed listings; must be customized by user.	$20	$20 includes installation manual and assembly language source for one processor (8080, etc); requires some work by user to install; quality product at a low price.
FORTH Power 17390 Hawkins Ln Morgan Hill CA 95037 (415) 471-1762	fig-FORTH	Heath WH-89 or 6800 EXORciser	N/A	N/A	
Forthright Enterprises POB 50911 Palo Alto CA 94303 (415) 856-0450	fig-FORTH	CP/M machine, 16 K bytes	8-inch CP/M floppy disk	$30	Includes all source code.
John James POB 348 Berkeley CA 94701 (415) 526-8815	fig-FORTH	PDP-11, all models; stand-alone or running under RT-11 or RSX-11M; 24 K bytes or more	8-inch floppy disk	$140	Package includes all documentation and source code; also offers a book of FORTH reprints.
M&B Design 820 Sweetbay Dr Sunnyvale CA 94086 (408) 243-0834	polyFORTH-CP/M	8080 CP/M system	8-inch CP/M floppy disk	$4000	Multitasking version of FORTH running on CP/M system with 32 K bytes or more; includes utility programs and interface to CP/M; system uses CP/M I/O drivers only.
Miller Microcomputer Services 61 Lake Shore Rd Natick MA 01760	MMSFORTH (based on FORTH Inc micro-FORTH)	TRS-80 Model I, with Level II BASIC, 16 K bytes or more	Cassette or 5-inch floppy disk	$59.95, cassette $79.95, disk	Offers support of product, consultation, newsletter, additional FORTH products, and a wide selection of FORTH books.
The Stackworks 321 E Kirkwood Ave Bloomington IN 47401 (812) 336-1600	SL5 (FORTH under a different name)	Any CP/M machine, 8080 or Z80	8-inch CP/M floppy disk	$150 (noncommercial use), $1500 (commercial use)	This language is essentially an implementation of the 1977 FORTH Standard; SL5 includes a debug package and packages that allow the generation of condensed, stand-alone programs as either CP/M, .COM files, or as programs to be placed in read-only memory.
Talbot Microsystems 7209 Stella Link Suite 112 Houston TX 77025 (713) 666-7588	fig-FORTH	Minimum 12 K bytes (20 K better for FLEX 9.0) 6809 SwTPC FLEX 9.0	5-inch floppy disk soft-sectored	$39.95	Offers telephone support of product.