THE CP/M USER'S JOURNAL $2.95

Microsystems

Volume 4 / Number 9 September 1983

"UNIX hierarchical directories— the greatest invention since file-clerks!" —D. Gewirtz

In the Fast Lane to UNIX

Bill Machrone reviews the CompuPro 8/16, running a variant of MP/M-86; the Altos 586, running XENIX, and the Ithaca Intersystems Encore, running MP/M-80. Leland Wilkinson reviews the Dual Systems 83/20, running a full UNIX. Introductory books on UNIX are reviewed in Ian Darwin's column; Chris Terry draws attention to three tutorials on the C language and some new books on CP/M.

UNIX & C with CP/M

David Gewirtz reviews Micronix, a UNIX subset with a CP/M emulator that runs on the Morrow Decision I. A.G.W. Cameron reviews a RATFOR preprocessor from Carousel MicroTools, and a set of UNIX-like Software Tools written in RATFOR. David Fiedler reviews the Supersoft C compiler. Scott Nowell presents "Cross Check," a C program to check double usage of blocks as an aid to CP/M file recovery.

North Star, and Other Topics

D.J. Yates presents step-by-step procedures for converting Microsoft Basic programs to run under North Star Basic. Allen Ashley shows how to run N* DOS as a .COM file under CP/M with a N*-to-CP/M file transfer facility. W. Lambert delves into the inner workings of WordStar with a Pascal program that reveals the markers and control codes hidden in a WordStar file. Andrew Bender presents the first article in a 4-part tutorial on relocating assemblers.

Hardware

Loren Amelang shows how to add a "rescue" key to your system that will get you out of infinite loops without destroying data. David Bray discusses the problems involved in upgrading an old 8080 computer such as the IMSAI to use the CompuPro 8085/8088 dual processor board.

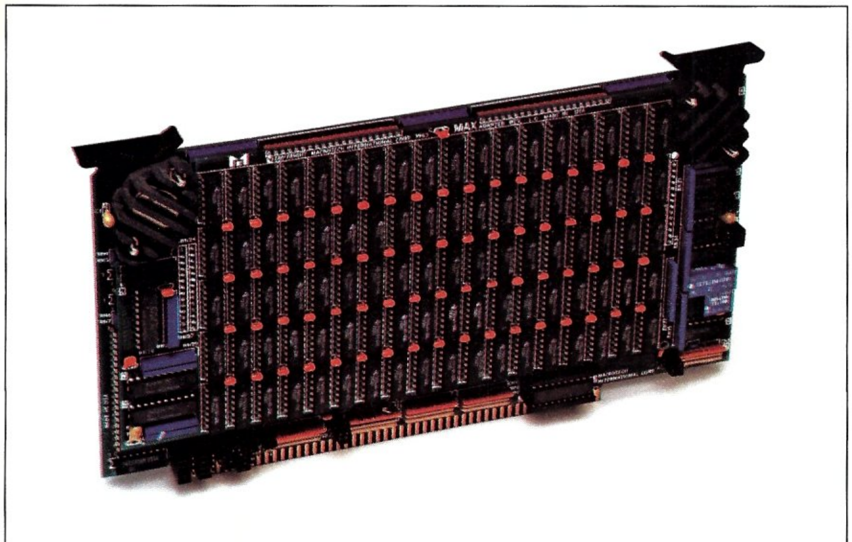
09
0 14024 14352

EXTRA EXTRA

S-100 World News

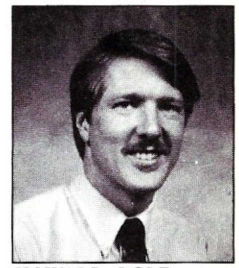
MACROTECH International Corporation 20630 Lassen Street, Chatsworth, California • 213-700-1501

NOW 1 MEGABYTE *MAX* FOR ALPHA MICRO



CHATSWORTH—June 30, 1983—Mike Pelkey, Macrotech International President, announced today that a special version of ***MAX*** is now running in Alpha Micro Systems.

This special version is available only through Soft Machines of Champaign, IL. (217) 351-7199. Howard Ogle of Soft Machines stated, "The new **AM-*MAX*1** runs full speed with all three Alpha S100 machines." Ogle also said, "The **AM-*MAX*1** is not only the most economical memory for Alpha, but the most versatile as well. The system is even faster with Soft Machines' 'GO FAST' disk cache utilities."



HOWARD OGLE

Bob Rubendunst of Soft Machines reports, "Every ***MAX*** is shipped with software that greatly simplifies implementation on bank switched systems. Also included are detailed installation instructions and diagnostic programs."

Dealer inquiries and orders should be directed to Bob at Soft Machines.

MAX SERIES GAINS WIDE ACCEPTANCE

CHATSWORTH—June 30, 1983—S100 systems manufacturers, integrators and users have been ordering and implementing ***MAX*** for a wide variety of environments and applications. These environments would include both 8- and 16-bit processors. Typical examples would include graphics and virtual disk implementations.

These environments include 16 bit systems such as those manufactured by Empirical Research Group, Dual Systems, Compupro, Cromemco, Lomas and Seattle. ***MAX*** has been used in non IEEE/696 systems such as Alpha Micro.

These ***MAX*** users have taken advantage of the density, high speed and low cost per bit to bring large system memories to S100 buyers.

Ralph Ring of Compatible Systems Engineering of Annandale, VA, (703) 941-0917 has used 4 ***MAX*** boards in a dual system UNIX* environment. Mr. Ring stated, "My application required a 4 megabyte system memory. Quarter meg boards were impractical, the ***MAX*** is ideal for this system."

Systems builders are using the M³ option to utilize large memories in 8-bit environments. Using the ***MAX*** board in this environment means using a single board for all memory needs. This includes functioning as system memory, virtual disk or cache memory. Some of these 8-bit environments include Compupro, CCS, Tarbell, IMS & Ithaca Intersystems.

M³ translates the 16 logic bits from an 8-bit processor into 24 physical address bits. This opens up the system to a 16 meg address space. Using software provided in the manual, virtual disk can be implemented using CP/M 2.2,* CP/M 3.0,* or MP/M 2* operating system.

If you think about it—quality, price, performance, and the reputation of Macrotech International—it's no wonder so many demanding systems builders have used ***MAX*** series dynamic memory. The S100 world's only full function one megabyte IEEE/696 memory.

Virtual Disk Gives *MAX* Split Personality

BURBANK—June 30, 1983—"Many current operating systems permit ***MAX*** to double as both virtual disk and system memory," stated Dan West of Westcom Systems. As an example, an MP/M 2.1* system using ***MAX-M*** could be configured as a 512K system memory and a 512K Vdisk. A typical CP/M 3.0* configuration could be 256K of system memory and up to 768K Vdisk. CP/M 2.2,* of course, only permits a 64K system memory, leaving the balance for a virtual disk. With ***MAX***, or the **128ST**, both functions can run simultaneously in a single memory board.

MACROTECH Moves

CHATSWORTH—June 30, 1983—Macrotech has moved to larger facilities located at 20630 Lassen St., Chatsworth, CA 91311. The new phone number is (213) 700-1501. "Due to a healthier marketplace and a phenomenal demand for the ***MAX*** series, larger facilities were necessary. This permits additional staffing, increased production, and customer support levels," said Mike Pelkey, President of Macrotech.

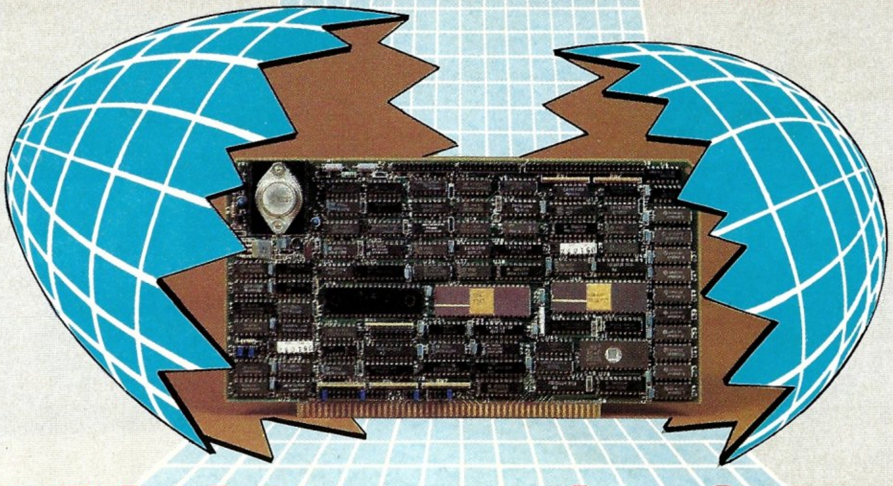
Virtual Disk for CP/M 86*

Dan West, Westcom Systems

BURBANK—June 30, 1983—Most of the CP/M 86* application programs available today fail to take advantage of the possible one megabyte address space. Virtual Disk for CP/M 86* will convert this unused space into RAM resident disk capacity for greatly improved disk access processing. The easily installed Virtual Disk 86 software module has been added to Macrotech's applications soft-

*CP/M 2.2, CP/M 3.0, CP/M 86 and MP/M 2.1 are registered

Incubation Complete



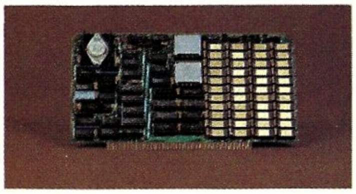
A Third Generation is Born

SBC 300

A Z80 based microcomputer board with memory and I/O functions

- ***Fully complies with IEEE 696 Standard***
- 4/6 MHz Z80 A/B
- Supports CP/M® Plus
- Operates as bus master/slave for multi-user, multi-processor architecture.
- 64K on board memory, dual ported, parity checked
- 2 serial ports, 1 SASI port
- All I/O drivers on board
- Memory management
- Full 24 bit address capability
- 3-16 bit CTC's

(Pictured above)



ExpandoRAM IV—Random access memory board utilizing 64K or 256K NMOS RAM chips

- ***Fully complies with IEEE 696 Standard***
- 256K capacity with 64K chips
- 1024K capacity with 256K chips
- Error checking and correction (2 bit detection, 1 bit correction)
- On board refresh
- Supports both 8 and 16 bit data transfers
- 24 bit addressing

SD300—A new series of compact yet expandable S-100 microcomputers.

- Compact size approximately 4" × 14" × 17"
- 6 Slot motherboard
- Rugged metal enclosure

OEM Version: Designed for ease of integration and maximum flexibility

- Z80 CPU
- 256K RAM
- Versafloppy II with free CP/M Plus™

Discless Version: An ideal high performance system for disk intensive applications. Eliminates disk wait states for spread sheets, spelling checkers, and network operation. Utilizes SDSystems RAMDisc and ROMDisc modules.

VFW-3: A single board controller for floppy and Winchester disk drives:

- ***Fully complies with IEEE 696 Standard***
- Up to 4 floppies and three Winchester drives may be controlled by VFW-3 • Data transfers to and from board under DMA or programmed I/O control • Supports 24 bit address space.

CP/M Plus™: high performance single user operating system.

- CP/M® 2.2. compatible—no modification! • When used with SDSystems 256K memory board speeds are up to 7 times faster than CP/M® 2.2. • High performance file system • MP/M® II file password protection • Time and date stamps on files • Support for 1 to 16 banks of RAM • Easy to use system utilities with HELP facility • Powerful batch facility • Sophisticated programmer utilities.

RAMDisc 256: A solid state disk emulator that greatly increases system performance by eliminating disk waits in disk intensive applications. Excellent for spreadsheets, spelling checkers and software development.

- 256K capacity • 1 mb total bus capacity • CP/M® 2.2, PLUS™ compatible • I/O port addresses user selectable • Storage locations addressed by on board 20 bit counter • On board refresh.

ROMDisc 128: An EPROM board that replaces a floppy disk drive for the purposes of booting CP/M® and loading application programs.

- Provides non volatile, permanent storage of programs and data • Utilizes 2732 or 2764 EPROMS, (16 max) • 128K capacity per board • 512K system capacity • Use with SDSystems RAMDisc to configure a stand alone or network discless system • Serial port provided.

SDSYSTEMS
A Syntech Company

10111 Miller Road • Dallas, Texas • (214) 340-0303 • TLX-682 9016

CIRCLE 3 ON READER SERVICE CARD

NOW IS OUR TIME.

Ithaca's Encore Computers, Designed For Those Who Desire The Best.

You've acquired a taste for computers. Now's the time to experience the benefits of America's finest.

Ithaca's ENCORE™ series offers matchless quality, performance, reliability, application versatility and expandability in 8 & 16 bit configurations.

The Standard of Quality

In both hardware and software, Ithaca is the standard of quality. Five years of experience in the production of S-100 systems means unsurpassed quality and reliability. As you grow, your ENCORE system expands with you. Unlimited options for growth are available through a variety of configurations which will meet all of your peripheral, memory and I/O requirements.

Proven Performance

ENCORE's unique design can provide up to a 500% increase in throughput on disk-intensive programs.

CP/M®, MP/M® and XENIX® operating systems as well as a selection of utility, development and application software, give the Ithaca ENCORE the increased programming capability you would expect from the best.

Ithaca's ENCORE™ . . . simply the finest price/performance, 8 & 16 bit computer systems available.

So advanced, that Now is Our Time.

For immediate reply call toll-free (800) 847-2088, New York call 607-273-2500. Ithaca is backed by Docutel Olivetti nationwide service.

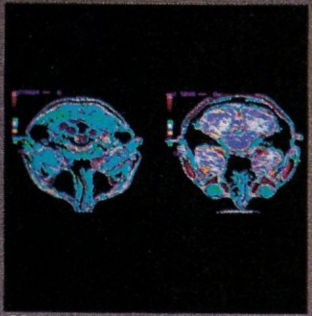
Ithaca InterSystems™
200 E. Buffalo St. Ithaca, N.Y. 14851

TM ENCORE is a trademark of Ithaca Intersystems, Inc. ® CP/M and MP/M are registered trademarks of Digital Research, Inc. ® XENIX is a registered trademark of Microsoft.

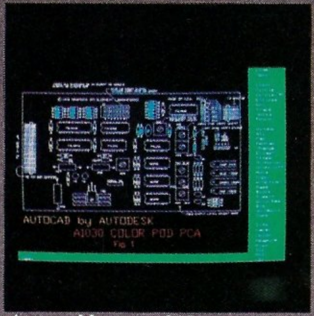
CIRCLE 231 ON READER SERVICE CARD

GDL's A-1000*™

Why Pay More For Less?



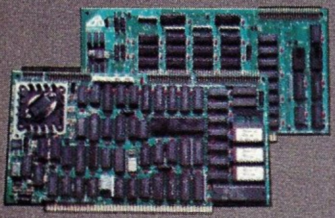
N.M.R. Brain Scan
J. Libove, Dual Systems



Autocad by Autodesk
R. H. Hymes, GDL

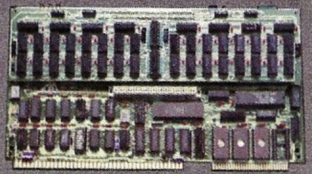
Supercalc Data Graph
Pacific Basin Graphics

Octehedron With Hidden
Surfaces Removed
Dr. C.L. Morgan,
Calif. S.U. Hayward



S100/696 A1000

- On-board 16 Bit Processor
- High-level Commands
- S100/696 or Multibus
- High Resolution
- Multiple Display Formats
- Simple Interface
- Extensive Software Support
- Download Capability
- Up to 16 out of 4096 Colors
- On-Board Self-Test



Multibus A1000

Graphics Development Laboratories

has finally made high performance color graphics affordable. These S100/696 and Multibus compatible boards are currently at work in such diverse areas as Medical, CAD, Education, Scientific, and Stock Market Analysis. And it's easy to see why, with their on-board 16-bit 8088 processor and extensive firmware, they act as intelligent graphics sub-systems, relieving the host of time intensive graphics processing, thus maximizing system throughput. Display memory is completely isolated from the host's bus and all communications occur through I/O ports. This simple interface and the high level commands allow for quick intergration into any S100 or Multibus system.

Software Support

The A-1000 command set not only includes pixel and vector draws but also **Polygon Area Fills, 2D rotation, scaling, clipping, dither fills, terminal emulate mode, stroke and raster character sets, circles, windowing and viewporting**. A Microsoft compatible subroutine library and C driver are included with every A-1000, at no extra cost. A **PLOT 10** driver and **GIOS** driver for **GSX** are available.

The A-1000 is supported by extensive third party software including:

AUTOCAD by AUTODESK	A 2D drafting package for use by Architects, Engineers or anyone requiring high quality diagrams.
PBG100	Business software that graphs data from **Supercalc** and **DBASE II** files and keyboard input. Also **Hershey** character sets are available.
PBG100 Library	A **Calcomp** compatible subroutine library.
The Analyst	Stock market analysis.
Graftalk	Business graphics.
Ugraf	Business graphics.

OEM's and Systems Integrators

GDL offers generous discounts to OEM's starting in small quantities. So whether you are a small systems integrator or large OEM, you will find our pricing attractive.

Dealers

Attractive Dealer discounts are available on software/hardware or hardware only packages.

Reliability

All A-1000's must past rigorous functional burn-in process. The self-test command allows the user to verify system integrity and isolate most problems.

To find out more about the A-1000, call or write. **Dealer inquiries welcome.**

A-1000 is the trademark of graphics Development Laboratories. Analyst is the trademark of Kates Komputers, PBG100 is the trademark of Pacific Basin Graphics, Graftalk is the trademark of The Redding Group. UGRAF is the trademark of Transparent Data Systems. Supercalc is the trademark of Sorcim Inc. DBASE II is the trademark of Ashton Tate Inc. GSX is a trademark of Digital Research Inc. Autocad is a trademark of Autodesk Inc.

* Formerly called the **Aurora 1000, AURORA** is the trademark of Aurora Systems.

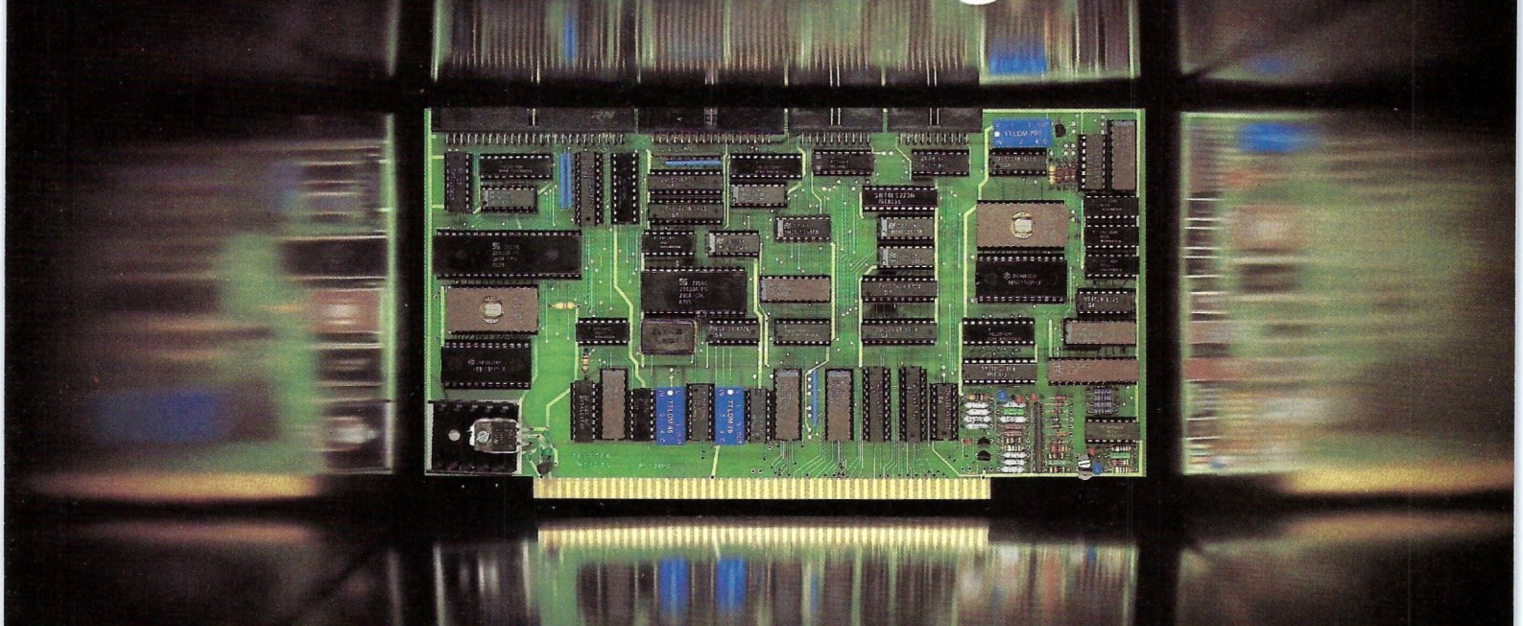
GRAPHICS DEVELOPMENT LABORATORIES
2832 Ninth Street
Berkeley, CA 94710
(415) 644-3551

in San Diego Area contact:
Corky Deeds
Martian Technologies
8348 Center Dr. Suite F
La Mesa, Ca. 92041
(619) 464-2924

NEW ADDRESS

CIRCLE 24 ON READER SERVICE CARD

Bored Waiting? Here's The Board You've Been Waiting For.



A hard disk and cartridge tape controller together on one board? Magic? Not really. It's Teletek's HD/CTC. The hard disk and cartridge tape drive controller provide the support necessary to interface both rigid-disk drives and a cartridge tape deck to the S-100 bus.

- A Z-80A CPU (optionally Z-80B) providing intelligent control of the rigid-disk and cartridge tape drives.
- Support of 5¼" rigid-disk drives with transfer rates of 5 megabits per second. Minor changes of the on-board components allow the support of other drive types/sizes and transfer rates up to 15 megabits per second. (Interface to disk drive is defined by software/firmware on-board.)
- Controller communications with the host processor via 2K FIFO at any speed desirable (limited only by RAM access time) for a data block transfer. Thus the controller does not constrain the host processor in any manner.
- Two 28-pin sockets allowing the use of up to 16K bytes of on-board EPROM and up to 8K bytes of on-board RAM.
- Individual software reset capability.
- Conforms to the proposed IEEE-696 S-100 standard.
- Controller can accommodate two rigid-disk drives and one cartridge tape drive. Expansion is made possible with an external card.

Teletek's HD/CTC Offers A Hard Disk Controller, Plus Cartridge Tape Controller, All On One Board.

TELETEK

9767F Business Park Drive Sacramento, CA 95827 (916) 361-1777 Telex #4991834 Answer back-Teletek

CIRCLE 220 ON READER SERVICE CARD

Contents

Microsystems

Volume 4/Number 9
September 1983

Microsystems

STAFF

Sol Libes	editor
Chris Terry	technical editor
Ian Darwin/Dave Fiedler/Dave Hardy/Bill Machrone/ Ernest E. Mau/Bruce Ratoff/Anthony Skjellum	contributing editors
Andrew Bender/David Gewirtz/Fred Gohlke/ Steve Leibson/Don Libes/Randy Reitz	assisting editors
Ann Ovodow	editorial coordinator
Tom Leander	editorial assistant
Mariano Nicieza	art editor
Jim Beloff	advertising manager

ADVERTISING SALES OFFICES

New England, Midatlantic
Jim Beloff, *Microsystems*
Ziff-Davis Publishing Company
One Park Avenue
New York, NY 10016
(212) 725-3452

Advertising Coordinator
Rosemarie Caruso, *Microsystems*
Ziff-Davis Publishing Company
One Park Avenue
New York, NY 10016
(212) 725-5386

Southeast
Mark Browning, *Browning Publications*
P.O. Box 81306
Atlanta, GA 30366
(404) 455-3430

Midwest
Jeff Edman, *The Pattis Group*
4761 W. Touhy Avenue
Lincolnwood, IL 60646
(312) 679-1100

Southern California, Southwest
Barbara Farkas,
Ziff-Davis Publishing
3460 Wilshire Blvd.
Los Angeles, CA 90010
(213) 387-2100

Northern California, Northwest
Jeff Cohen,
Ziff-Davis Publishing
3030 Bridgeway
Sausalito, CA 94965
(415) 331-7133

Canada
Frank Lederer, *The Pattis Group*
1623 Yonge St.
Toronto, Ontario M4T241
(416) 482-6288

Direct Retail Sales
Lynn Kujawa, *Ziff-Davis Publishing*
One Park Ave.
New York, NY 10016
(212) 725-7679

CONSUMER COMPUTER & ELECTRONICS DIVISION

President	Larry Sporn
Vice President Marketing	J. Scott Briggs
Vice President Circulation	Carole Mandel
Vice President General Manager	Eileen G. Markowitz
Creative Director	Peter J. Blank

ZIFF-DAVIS PUBLISHING

Richard P. Friese, *President* Albert S. Traina, *President, Consumer Magazine Division* Furman Hebb, *Executive Vice President* Paul H. Chook, *Executive Vice President, Marketing and Circulation* Phillip T. Heffernan, *Senior Vice President* Sidney Holtz, *Senior Vice President* Edward D. Muhlfeld, *Senior Vice President* Philip Sine, *Senior Vice President* Baird Davis, *Vice President* George Morrissey, *Vice President* Selwyn Taubman, *Treasurer* Bertram A. Abrams, *Secretary*

MICROSYSTEMS (ISSN #0199-7955) is published monthly by Ziff-Davis Publishing Company, One Park Avenue, New York, N.Y., 10016.

Second Class postage paid at New York, NY 10016 and at additional mailing offices. POSTMASTER: Send address changes to MICROSYSTEMS, PO Box 1987, Morristown, NJ 07960. Subscriptions are $26.97 for 12 issues. Canadian prices are $5.00 per year additional; other foreign $8.00 per year additional (U.S. currency only). For information or questions about subscriptions phone: (800) 631-8112.

Editorial correspondence is welcomed and should be sent to: MICROSYSTEMS, One Park Avenue, New York, NY 10016. Phone: (212) 725-6856.

For information on commercial advertising, write to: MICROSYSTEMS Advertising Dep't., One Park Avenue, New York, NY 10016, or call Jim Beloff at (212) 725-3452.

Introducing the powerful, multi-processing HORIZON® 8/16 from North Star.

The turbo-charged system with outstanding performance.

The new North Star HORIZON 8/16 microcomputer can handle up to eight individual users, supporting both 8-bit and 16-bit applications simultaneously.

Its advanced, multi-processor architecture makes this powerful performance possible. Unlike other multi-user systems, the HORIZON 8/16 doesn't load up its users on a single processor; instead, it provides a dedicated processor for each individual user—at a cost no greater than that of conventional multi-user systems.

The result? No degradation in processing performance, even when there are eight users on the system.

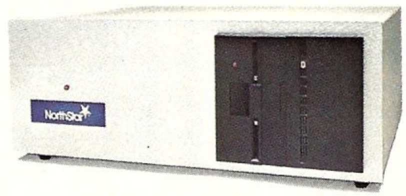
And North Star's industry standard S-100 bus gives you the flexibility to choose your options and tailor the system to meet your specific requirements.

What's more, the new North Star TurboDOS® is many times faster than standard, multi-user operating systems—and is compatible with CP/M-80,® CP/M-86® and MP/M.™

As for reliability, over 30,000 first generation HORIZONs are still in use. And each of these can be easily upgraded to the new 8/16 architecture.

The HORIZON 8/16 outperforms everything in its class. Costs no more. And is the only multi-user micro designed to meet your needs for today, and tomorrow—simply by plugging in the options you select.

You can discover North Star's HORIZON 8/16 at more than 1,000 computer stores and system houses nationwide. Call 800-722-STAR for the location nearest you. Or write North Star Computers, Inc., 14440 Catalina Street, San Leandro, CA 94577.



NorthStar™

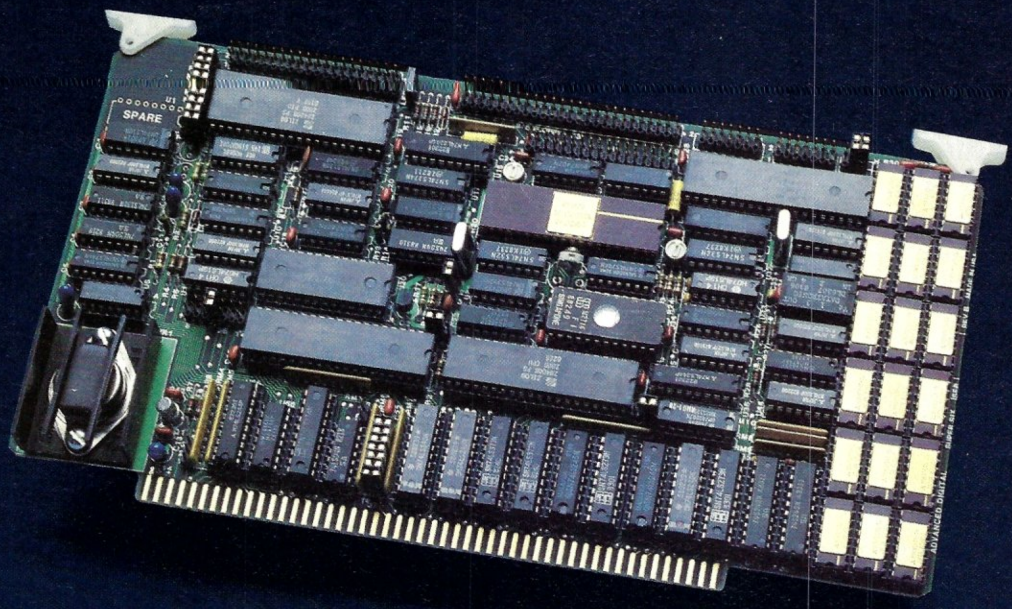
Simply powerful solutions.

Z-80 is a registered trademark of Zilog, Inc. TurboDOS is a registered trademark of Software 2000, Inc. CP/M-80, CP/M-86, MP/M and CP/M are either trademarks or registered trademarks of Digital Research Inc.

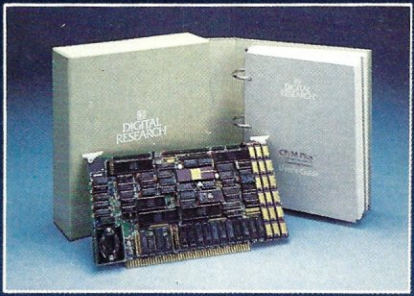
Systems serviced nationwide by MAI/Sorbus Service Division.

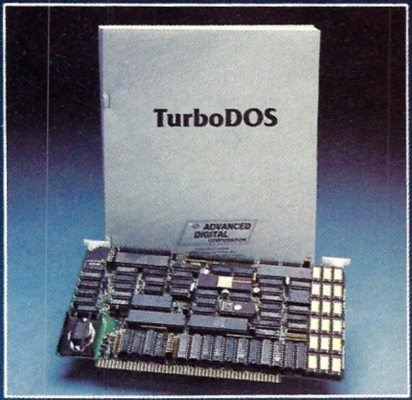
CIRCLE 11 ON READER SERVICE CARD

SUPER SIX™



SUPER SIX, THE FIRST 6MHz S-100 SINGLE BOARD COMPUTER TO SUPPORT BANKED CP/M™ 3.0





SUPER SIX FEATURES:

- 128 KB of Bank — selectable RAM
- 6 MHz, Z-80B CPU
- DMA Controller
- 6 MHz, Z-80B DART (2 Serial RS-232 Ports)
- 6 MHz, Z-80B PIO (2 Parallel Ports)
- 6 MHz, Z-80B CTC (Clock Timer)
- Double/Single Density Floppy Disk Controller — Supports 8″ and 5-1/4″ Drives Simultaneously
- 2/4 KB of Monitor EPROM
- S-100, IEEE 696 Compatible

*CP/M is a trademark of Digital Research Corp.
TurboDOS is a trademark of Software 2000 Inc.
SUPER SIX and SUPER SLAVE are trademarks of Advanced Digital Corp.

**SUPER SIX & CP/M™ 3.0
A PERFECT MATCH**

Advanced Digital has found the perfect match to its powerful, high-speed SUPER SIX single board computer. It's Digital Research's new CP/M 3.0. Because of SUPER SIX's 128 KB of RAM, it is the only S-100 board to support CP/M 3.0 in the banked mode; or run CP/M 2.2 with 64 KB of extra buffer.

**SUPER SIX & TurboDOS™
ANOTHER PERFECT MATCH**

When you combine the TurboDOS multi-user operating system with the 6 MHz SUPER SIX, you'll find your system running 1-1/2 times faster than before. Add the 4 MHz or 6 MHz SUPER SLAVE™ processor board(s) and you will have the fastest multi-user, multi-processor system available today.

See the SUPER SIX at your quality computer dealer or contact:

ADVANCED DIGITAL CORPORATION

5432 Production Dr.,
Huntington Beach, CA 92649
Phone: (714) 891-4004
Telex: 678401 TAB IRIN

CIRCLE 148 ON READER SERVICE CARD

Editor's Page

by Sol Libes

This is the second issue of *Microsystems* in which we are highlighting the UNIX operating system (the first issue was January 1983).

There is no doubt that CP/M is the de facto standard for single-user 8-bit microcomputer operating systems. The introduction of CP/M Plus and the forthcoming Z800 microprocessor for Zilog will ensure continued growth in the 8-bit micro market.

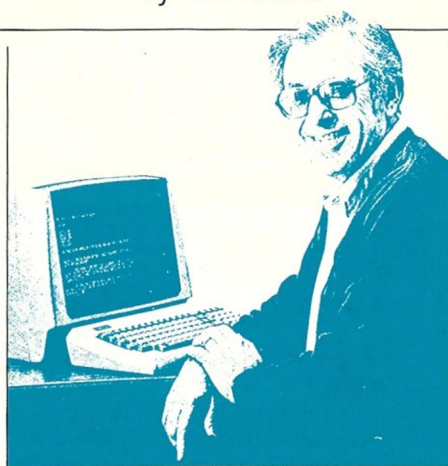
However, with the prices of single-user 16-bit micros dropping to only slightly more than those for the 8-bit machines, and with the added performance offered by these machines, it is only logical that the 16-bitters will overtake the 8-bit market at some point. The 16-bit single and multiuser micro markets are highly competitive worlds with no clear de facto DOS standards as yet.

The key to success for an operating system is the amount and quality of software available for it . . . and in the 16-bit single-user area, MS-DOS is the clear leader. Another factor is the number of OEM manufacturers operating the DOS on their hardware systems . . . and here both MS-DOS and CP/M are doing well.

The multiuser micro market is where 16-bit micros will dominate. Although DRI tried to enter the multiuser market with 8-bit versions of CP/M called MP/M and CP/NET, these lacked the necessary performance and features. MP/M, however, has worked out well as a multitasking operating system (*Microsystems* will be carrying several articles on MP/M). CP/NET has recently been greatly improved by DRI and should be successful (a review of the new release of CP NET will appear next month).

The TurboDos system (CP/M compatible) is achieving moderate success as a multiuser 8-bit operating system (we are also planning several articles on TurboDos). Phase One has also achieved some success with its OASIS operating system (see the Mar/Apr 1982 issue of *Microsystems*). But there is no doubt that the multiuser market will be dominated by the 16-bit micros.

DRI entered this market early with its 16-bit implementation of



MP/M and is achieving some moderate success with it. OASIS is also available for the 16-bit systems and is being offered by several OEMs. But the star of the 16-bit multiuser marketplace appears to be UNIX. Another comer is the PICK operating system. For these reasons, we have decided to devote another issue to UNIX, and at some point in the future expect to provide coverage of the PICK operating system.

Both UNIX and PICK have been in use on large multiuser minicomputer systems for about 10 years. They are mature products that provide reliable operation, a lot of features, and considerable application software support. They are being ported to many different hardware systems. At the recent National Computer Conference (the computer show-of-shows) more newly introduced computers were running UNIX or UNIX-like operating systems than any other operating system (although MS-DOS was a close second). And most were based on the Motorola 68000 microprocessor.

UNIX Superstar

UNIX was developed at Bell Labs as a tool for software developers. It achieved popularity in-house and was greatly enhanced (one of its advantages is that it *can* be easily enhanced). BL made it available to the University community, where its power and features proved assets to software hackers. As these hackers moved from university to business environments, they carried the UNIX mystique with them. Thus UNIX has achieved quite a success in the engineering and scientific areas of business. However, it is in the nontechnical area of business where the big bucks lie; hence manufacturers are now in the process of adapting UNIX to this environment. Thus, companies such as Microsoft, with their XENIX package, are attempting to make UNIX easier to use and are adding features needed in a business environment (e.g., file security).

PICK—a challenger?

The PICK operating system is currently running on about the same number of minicomputers as UNIX (estimated at 10-15,000). Like UNIX, it is intended for multiuser environments serving a large number of users (e.g., up to 128 and even more). However, it is database oriented and much easier to use. Thus many sophisticated business application packages are available for it, and it has achieved a great deal of success in business system use.

The PICK operating system is already being offered by micro manufacturers such as Altos, General Automation, and Datamedia. And third-party vendors are selling versions for the IBM PC. A version for the XT is expected shortly, and Tandy and Apple are known to be checking it out for possible adoption on their 68000-based machines. It is interesting that the micro manufacturers offering the PICK system are also offering XENIX for the same systems. Thus, by offering both XENIX and PICK, these manufacturers can sell 16-bit multiuser systems better to both the business and scientific communities.

For those desiring more information on PICK, I recommend a special report entitled *The PICK Operating System: Markets & Strategies* issued by Yates Ventures (4962 El Camino Real, Suite 111, Los Altos CA 94022; 415/964-0130). To keep up to date on happenings on UNIX, I recommend a newsletter entitled *Unique,* published by InfoPro Systems (Box 33, East Hanover, NJ 07936; 201/625-2925; $54 for 12 issues). Yates Ventures also publishes a very good (but expensive: $400/yr) monthly newsletter entitled *The Yates Perspective.*

News & Views

by Sol Libes

CP/M '83 EAST in Boston, Sept. 29-Oct. 1

Following in the footsteps of the highly successful CP/M'83 show held in San Francisco last January, Digital Research is repeating the event in Boston September 29 through October 1. DRI reports that more than 50,000 attended the earlier event, which had 400 exhibitions. CP/M'83 EAST is expected to be equally large, if not larger.

CP/M'83 will be held in Boston's Hynes Auditorium, which is part of the Prudential Center Complex, adjacent to the Sheraton Boston Hotel. There is parking for 7100 cars.

The Industry Workshops for Independent Software Vendors (ISVs), distributors, dealers and manufacturers will be held each day from 9-11 A.M. The workshop topics are:

Thursday: Venture Capital and the Microcomputer Industry
Friday: Retail Marketing of Microcomputer Software
Saturday: Trends in Microcomputer Graphics

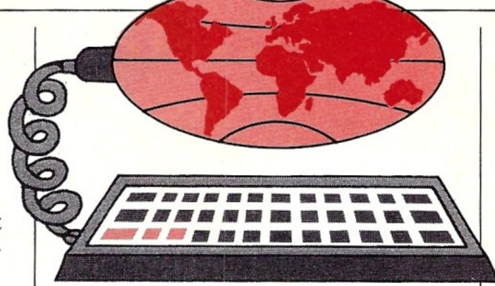
The Panel Discussion will be held daily from 2-4 P.M., and topics will be:

Thursday: Next Generation Software
Friday: Educational Software
Saturday: Microcomputer Industry Trends

The Seminars will be held daily from 11 A.M. to 6 P.M. The following topics will be covered:

Introduction to Logo
Public Domain CP/M Software
CP/M Cards
The Bridge from Unix to CP/M
Introduction to Accounting Applications
Introduction to Word Processing
Introduction to Communications Networks
Customer Training
Programmer Productivity Tools
Software Protection and Security
How To Get into the Software Business

The show and conference hours are 11 A.M.-6 P.M.; the industry workshop hours are 9-11 A.M. Admission is $10/day, or $25 for the three days. For more information on the show contact: Northeast Expositions, 824 Boylston Street, Chestnut Hill, MA 02167; (617) 739-2000.



CompuPro offers special deal

Bill Godbout, President of the CompuPro Division of Godbout Electronics, is making a special offer to *Microsystems* magazine readers that will allow them to step up their CompuPro 8-bit systems to dual-processor 8/16-bit systems. For a limited time, CompuPro will give our readers a half-price exchange of 8-bit CompuPro CPU-Z, Z80 CPU card (any vintage), for a CompuPro CPU 8085/88 card, which contains both 8085 and 8088 microprocessors. The list price is $425 (6 MHz 8085/8088) or $525 (6MHz 8085/8MHz 8088 with 200-hr burn-in).

To obtain a dual-processor CPU card at half price, you must mention that you read this in *Microsystems* magazine. Write: CompuPro Division, Godbout Electronics, Box 2355, Oakland Airport CA 94614

CP/M Emulator board for DEC PDP-11

Virtual Microsystems Inc., Berkeley CA, the supplier of the Bridge CP/M emulator, has introduced a plug-in board emulator for the DEC PDP-11/23 and Professional 300 computers. The board will be marketed through Digital Equipment Corp., while Bridge will continue to market its other software and hardware products for VAX and other PDP-systems. The new package is called the "z-chip" and is priced at $495 (Professional) and $695 (PDP-11/23). The board plugs into the floating-point processor slot of the computer.

The MS-DOS vs CP/M Battle

Digital Research is proving that it has not given up the fight for the 16-bit DOS market with a new strategy . . . if you can't beat em, then join em! DRI has announced that it plans to support the IBM PC-DOS and MS-DOS with the releases of its family of language products. This includes their CBasic compiler, Pascal/MT+, PL/I, C, Cobol, Access Manager, Display Manager, Programmer Utilities, and Symbolic Debugger. Thus, these languages may be run under PC-DOS, MS-DOS, CP/M-86 or Concurrent CP/M-86. This should prove to be a real asset for CP/M software vendors.

DRI has also announced that it will release new versions of CP/M-86 and Concurrent CP/M-86 that will run MS-DOS software in an emulator mode. They claim that MS-DOS software will run almost as fast as on a standard MS-DOS system.

Z800 Update

In the July *Microsystems* I reported on Zilog's introduction of the Z800 microprocessor, an enhanced version of the Z80 that will be very easy to retrofit into a Z80 CPU card. It now appears that first samples of this IC will not be available until the end of this year, and that significant production quantities will not become available until the end of '84 or the beginning of '85. Thus I do not expect to see the use of the Z800 in systems until 1985.

It is disappointing. Zilog "leaked" news of the Z800 at the 1981 National Computer conference. At the time, CP/M was unchallenged in the micro world, and the Z800 enhancements would have considerably slowed the current trend to 16-bit personal computers and the MS-DOS operating system. Many companies have enhanced their Z80 systems by adding a second 16-bit processor and gained most of the features offered by the Z800. And by the time the Z800 becomes available, micro designs will be in full swing with 32-bit devices.

By the time the Z800 becomes available, the software base for competing Intel chips will be larger and more attractive, and it is doubtful that many software developers will want to start to work on Z800 software. Thus there is a great likelihood that Zilog may miss the market.

In the Public Domain

by Chris Terry

Communications

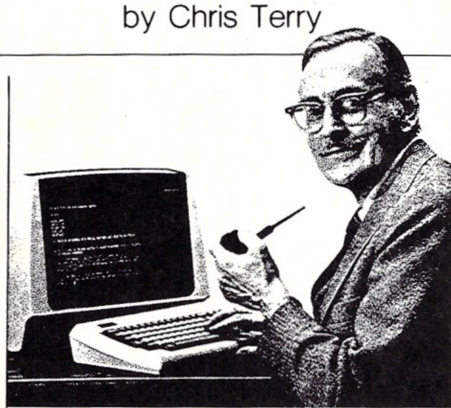
First, a correction to this column in the July issue: The volume number for MODEM798 was given as *CPMUG* Vol. 93A—this should have been *SIG/M* Vol. 93A. My apologies for any inconvenience caused by this error.

While we're on the subject of communications, **Omnitech,** *56 Baltusrol Way, Shore Hills, NJ 07078* wrote with an offer that may be helpful to users of Compustars and Superbrains. This company is offering a copy of MODEM7, in Superbrain format and with modest documentation, at their cost ($7.50). For further information, call them at (201) 376-6406.

CP/M enhancements

ZCPR2 and SYSLIB. The most notable (and also the most comprehensive) enhancement to CP/M that has so far appeared is ZCPR2, by Rich Conn. An introduction to this system by Rich Conn himself appeared in our June 1983 issue. An early version of the system (ZCPR) appeared in SIG/M Vol. 54, but ZCPR2 has many, many enhancements and, with SYSLIB, occupies volumes 98 through 107 of the SIG/M library. The documentation alone occupies three of these volumes.

The ZCPR2 system replaces the standard CCP (Console Command Processor) of CP/M. There are resident commands similar to those of the CCP, but more friendly and more powerful, and a number of useful resident commands have been added. In addition, utilities that work in conjunction with ZCPR2 greatly extend the convenience and power of CP/M; they include on-line documentation and the ability to search for a .COM file not only on the logged-in disk but also on all other drives in the system. A minimal ZCPR2 is still an improvement over the standard CCP and adds little in the way of overhead. A full ZCPR2 requires modification of the BIOS, with a consequent reduction of the TPA by 3K-4K, along with a considerable amount of disk storage—you should have at least 500K per disk to accommodate ZCPR and have room to do useful work.



SYSLIB is a library of relocatable assembly language modules that was also built by Rich Conn. Although not strictly a CP/M enhancement, I mention it here because it is a powerful tool for program development under CP/M. Standards for parameter passing and use of the modules are established in the documentation. Just as ZCPR2 increases programming productvity by making many operations far more convenient, so SYSLIB increases productivity by providing so many standard operations that the programmer can concentrate on the main logic of his program. The SYSLIB system is designed for use with Microsoft's M80 macroassembler and L80 linker.

An on-line SUBMIT. A program called /.COM by Ward Christensen (CPMUG Vol. 40) provides an on-line submit facility. This program builds a command file and runs it without having to edit a .SUB file. The commands are separated by semicolons—for example, typing:

```
A>WM FOO.ASM; ASM FOO.ABZ; LOAD B:FOO
```

will run WordMaster so you can change the source code of your program FOO. When you exit from WM, the source code is assembled with the hex file on drive B:, and a .COM file is generated without any further operator intervention.

SUPERSUB. SUPERSUB (CPMUG Vol. 81) is a replacement for the standard SUBMIT and XSUB utilities supplied with CP/M. It has been reviewed in detail in *Lifelines*; its principal advantage is that submit commands can be nested.

Directory utilities

SD-41. SD-41 (SIG/M Vol. 44) is a powerful directory utility that can search all drives and user areas for unambiguous or ambiguous filenames. Free-form options following a dollar sign on the command line allow the search to start at any specified drive or user area. Other options allow the resulting output to be sent to the LST: device or to a disk file.

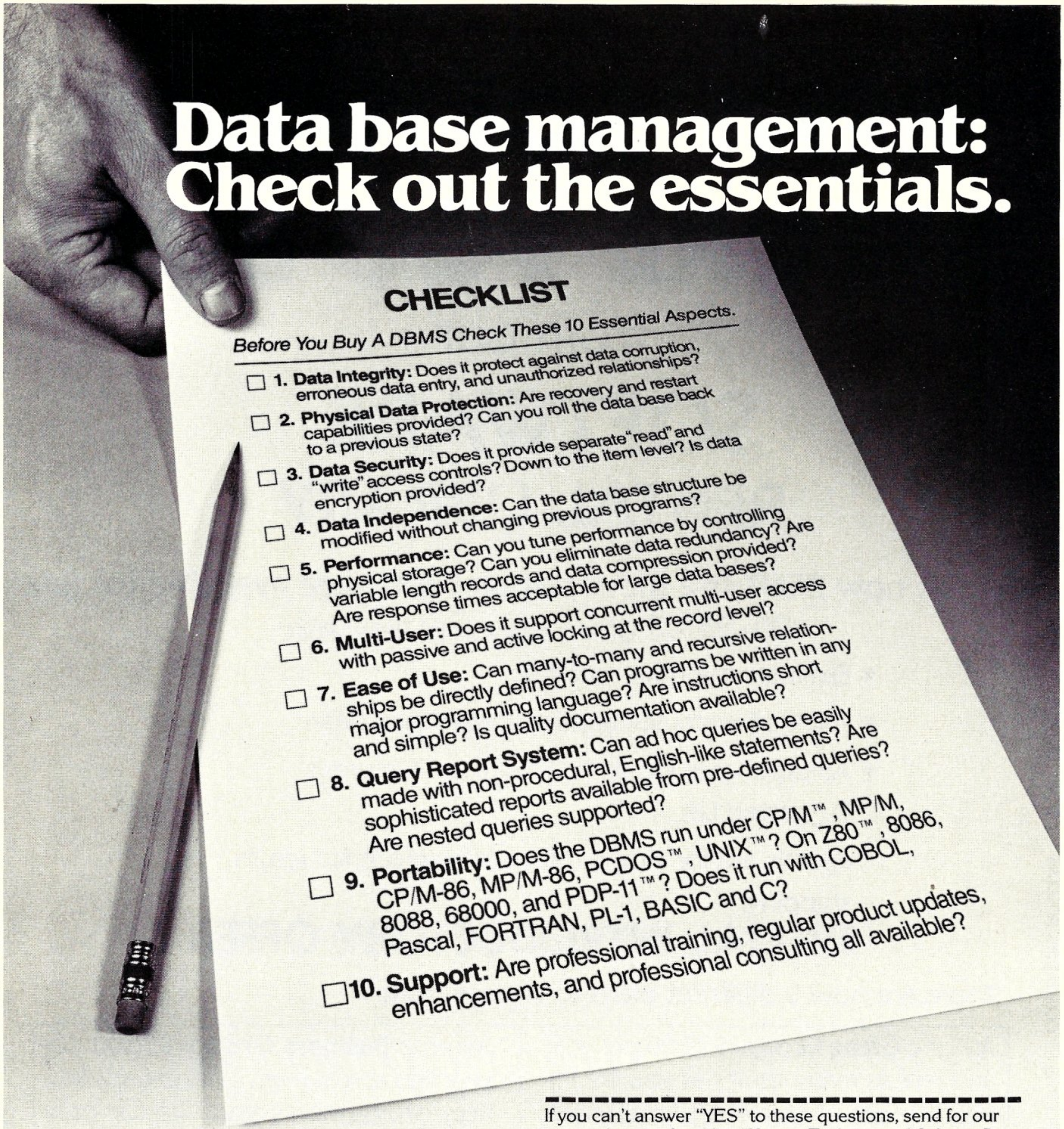
WASH. WASH (SIG/M Vol. 44) is a directory maintenance utilty that allows the operator to review the directory in detail. The filenames are displayed one by one, and as each filename appears the cursor pauses at the end of the line for an operator command. The file may be left unchanged, erased, copied, or renamed. When operations are complete, the operator may exit from the program without reviewing any further names.

XDIR. XDIR (extended Directory), available in SIG/M Vol. 17, prints an alphabetically sorted directory in 3-column format, with file sizes.

SAP. SAP (Sort And Purge) is a directory utility originally issued in CPMUG Vol. 19 for CP/M version 1.4. This utility discards zero-length files and entries which have been marked with E5 for deletion, and alphabetically sorts the valid directory entries, packing together the entries for multi-extent files. The packed and sorted directory is then rewritten to the disk and displayed on the screen.

An update for CP/M version 2.2 has been issued (SIG/M Vol. 7, CPMUG Vol. 61) but this utility should be used with extreme care. Since it discards zero-length files, it may destroy disk identifier entries used with Ward Christensen's CATALOG system, and it can also be upset by System and R/O attribute bits. It is handy if you want a permanently sorted directory, but try it out on a scratch disk containing System and R/O files as well as disk identifiers before using it on any important disks, particularly if you are using double-density formats.

Restart. I mentioned last month that I would discuss the null file as a means of restarting the last program that was run. This technique is particularly useful for restarting a program that has bombed for some reason, without destroying variables or



There is only one DBMS that satisfies all these criteria. That's MDBS III, the only authentic DBMS running on a wide variety of microcomputers and the PDP-11.

ISE™ PROFESSIONALS KNOW THE DIFFERENCE

In U.S. & Canada contact:
ISE-USA
85 W. Algonquin Rd.
Arlington Heights, IL 60005
(800) 323-3629/(312) 981-9200

Elsewhere contact:
ISE
PO Box 600
Lafayette, IN 47902
(317) 463-4561/2581

If you can't answer "YES" to these questions, send for our comprehensive booklet, "How to Evaluate and Select a Data Base Management System." Sold previously for $5.00, but it's yours *FREE* if you return this coupon now. But hurry, supplies are limited.

MAIL TO: Your local ISE affiliate.

Name__________ Title__________

Company__________

Address__________

City__________ State_____ Zip_____

Phone__________
(area code) (ext.)

Z80 is a registered trademark of Zilog; PDP-11 of Digital Equipment Corp; CP/M of Digital Research; UNIX of Western Electric; PCDOS of IBM.

CIRCLE 48 ON READER SERVICE CARD

Public Domain
continued . . .

output created by the program. Create a null file with the command:

```
A>SAVE 0 RESTART.COM
```

Then, next time a program hangs in a loop or terminates prematurely, just do:

```
A>RESTART
```

CP/M will attempt to load this program and run it at 100H; however, since RESTART has zero length, nothing is loaded, and when control is handed to 100H, the result is to restart whatever program you last ran, without any change.

Letters to the Editor

Dear Mr. Libes,

This letter provides a correction, a clarification, and IBM-PC information regarding my article "PIP Data Between Computers" (July 1983 *Microsystems,* page 48).

Correction

I omitted a line from Figure 1 ("Installing PIPIO by hand"). It illustrates installing the PIPIO protocol when using an older version of DDT that does not allow overlaying a program with a HEX file. The missing line depicts telling DDT to begin memory substitution at location 0103H. It should be inserted as the sixth line of the example.

```
A>DDT PIP.COM
DDT VERS 2.2
NEXT  PC
1E00 0100
(insert PIPIO protocol)
S103
- 50 C3
```

Programmers are trained to strive for perfection; nearly accurate accounting systems and occasional system crashes are just not good enough. To those readers who were confused or inconvenienced by the omission, I offer my most abject apologies.

Clarification

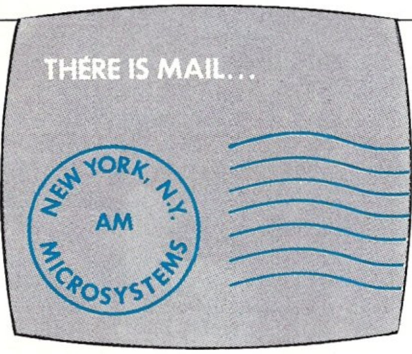
When using PIPIO to communicate between computers, you must invoke the receiving computer first. That way the first character sent is sure to be echoed. If the transmitting station is started before the receiving station, the first data byte may be sent before it can be detected. Then you would have a receiver waiting for input while the transmitter is waiting for the echo. While the article showed the proper sequence, this point was not streesed.

IBM-PC information

You can use PIPIO to send data to an IBM-PC running CP/M-86. Use the ASSIGN program to direct the PC console output to both the screen and a serial port. Also ASSIGN the auxiliary input to the same serial port. Then begin PC reception with the command:

```
A>PIP d:filename.typ=AXI:[E]
```

Data is then displayed on the screen and echoed to the serial port.



Unfortunately, you may lose three bytes within the first line or so of data. This is because the PIP command is sent to the serial port before reception begins. The last three characters of the command are still in the serial port circuitry when PIPIO handshaking begins. These bytes appear to the transmitting CP/M-80 system as data echoes. Each false acknowledgement loses one data byte.

You could write a simple Basic program to read and echo bytes from the communications port, saving the characters in a disk file. This program should close the file and stop upon receiving and echoing a Control-Z (1AH). You could use such a program to send to IBM-PC's running PC-DOS. In fact, this method was used to PIPIO to Alpha-Micro and DEC systems.

I am presently writing LOAD and UNLOAD programs in 8086/8088 assembler for use with CP/M-86; these may be the subject of a future article in *Microsystems.*

Steven Fisher CDP
Controlled Information
Environments
(a division of Steven Fisher
Enterprises)
P.O. Box 457
La Mesa, CA 92041
(619) 271-4327

Dear Mr. Libes,

Please ask readers not to write to me now about my letter on **single-density formatting** (June 1983), but await publication of my article. The number of enquiries has been large, and I cannot possibly reply to them individually.

Robert Lurie
8 Tingley Road
Morristown, NJ 07960

Editor's note:
We expect to publish Bob's article on single-density formatting in the October issue.

Errata

Here are corrections and additional information on the Computime SBC-880 (reviewed in Sept. 1983). Prices are: SBC-880: $275 A&T, $245 kit, $45 bare board; FDC UFDC-1: $275 A&T, $245 kit, $60 bare board. The East Coast distributor of these products is:

GSR Computers
60-10 69th St.
Maspeth, NY 11378
(212) 476-2091

A letter from MicroDynamics (to be published later) draws attention to the fact that their Microsprite Graphics board, listed in our S-100 Directory (May 1983), can display 16 colors on the screen simultaneously.

Dear Mr. Libes,

Mr. Leibson's article, "S.A.I.L.-ing Without a Lifeboat," in the May issue was very interesting. However, I have to take serious issue with one comment in it. The author stated that " . . . North Star does not publish information on how to run their disk controller . . . "

On the contrary, back in the days when their boards were available in kit form, N* published very detailed programming information on how their board works. I have had my manual for several years. I recently rewrote a disk driver for the current production board in polyFORTH II™, and found N*'s information to be complete and accurate.

I have cited North Star's programming documentation detail and quality to several other manufacturers as an example of how it should be done. The N* data makes some of the competitor's documentation look sick. For example, it is a real chore to get CP/M running on a Versafloppy II board in a non-SD system with the data SD gives you. No manufacturer should sell individual boards as stand-alone products without the depth of detail that North Star provides.

Joseph Reymann
ComType
P.O. Box 374
953 Avenida Ladera
San Dimas, CA 91773

HOW TO CONTROL YOUR CP/M™ MICRO—AT ONLY $3. PER COMMAND.

A Remarkable Program For CP/M Users.

Of course, CP/M is a wonderful operating system. That's why so much serious business software has been created for it.

BUT, CP/M is not easy to work with. That's why you need to take the **POWER!** trip.

POWER! is a super-power-packed, user-friendly program that lets you take immediate and complete control of CP/M. And at a cost of only $3. per command, it's the software buy of the year.

Over 55 Housekeeping Utilities.

POWER! is over 55 prompted, user-friendly CP/M utility programs all rolled into one 15k package. It takes care of all of these frustrations and more:

—BDOS errors? POWER! ends BDOS errors and gives you a way out.

—Accidentally erased a file? If you accidentally erase a program or disk file, **POWER!** restores the erased files.

—Can't remember file names? POWER! assigns a number to each file on your disk. So, to copy files from disk to disk, you don't have to fiddle with PIP anymore. You just pick the file from a numbered menu and **POWER!** copies it for you. No more typing errors! **POWER!** also marks original files and their copies for you; and you can compare files to find identical copies regardless of name.

—Lose data on a glitched disk? If a glitched disk makes it impossible to call up a long word processing text, **POWER!** can fix the glitch. This means you may have to retype only a couple of sentences instead of losing 20 pages of text.

—Trouble with "bargain" disks? POWER!'s disk testing function gathers any bad sectors of the disk into a special file so that CP/M thinks those parts of the disk are already used and never attempts to write to them. The rest of the disk is then safe to use.

—CP/M scrolls too fast through text files? POWER! spools through files for you, page by page, file by file, or line by line with instant halt by touching the space bar.

—Need to reorganize files? POWER! sorts and formats the directory in 4 different ways. And you can easily copy or move files from user area to user area. **POWER!** creates 32 user areas instead of CP/M's 16.

—Need to change memory? POWER! searches, displays and lets you change memory wherever you want. You can even automatically run software anywhere in memory. And you can inter-mix your search with as many wild card jokers as you need to find, for instance, all occurrences of "Sam Jones" and "Sid James" just by typing "S??J??". And **POWER!** also lets you read or write to any sector or track very simply.

—Changing disks? You can forget the ubiquitous Control C to change disks. **POWER!** can do it for you automatically. And **POWER!** doesn't require a system disk in any drive, so Drive A is open for use, when **POWER!** is in control of CP/M.

—Afraid of HEX numbers? POWER! automatically converts Hex to Decimal, Binary or ASCII.

Special Password Protection, Too.

POWER! now includes a special program that lets you lock sensitive files, so that only you can access them. Without the secret PASSWORD which you can create and change at will, no prying eyes will ever know your secret file even exists. A great way to protect financial or scientific data from unauthorized eyes. Just this single program alone would be worth the price of **POWER!**, but there are over 55 more just as valuable programs in this power-packed-package.

At $169., It's A Bargain.

Space doesn't permit describing all the wonderful ways **POWER!** can put you in complete control of your CP/M micro. But see for yourself. There's a Money Back Guarantee. At the low price of $169., each powerful command costs you less than $3. A true bargain!

InfoWorld
Software Report Card
Power Version 2.55

	Poor	Fair	Good	Excellent
Performance	☐	☐	☐	☑
Documentation	☐	☐	☑	☐
Ease of Use	☐	☐	☐	☑
Error Handling	☐	☐	☐	☑

© 1982 by Popular Computing, Inc. A subsidiary of CW Communications Inc., Framingham MA. Reprinted from InfoWorld.

POWER! Is Better Than Ever!

Eventhough "InforWorld", "Microsystems" and "Interface Age" call **POWER!** great, we have improved **POWER!**—including a completely rewritten 120-page easy-to-read documentation. (Previous purchasers of **POWER!** may exchange their original disk for an updated version with the new commands and a brand new manual—for only $35.)

Take The POWER! Trip Today!

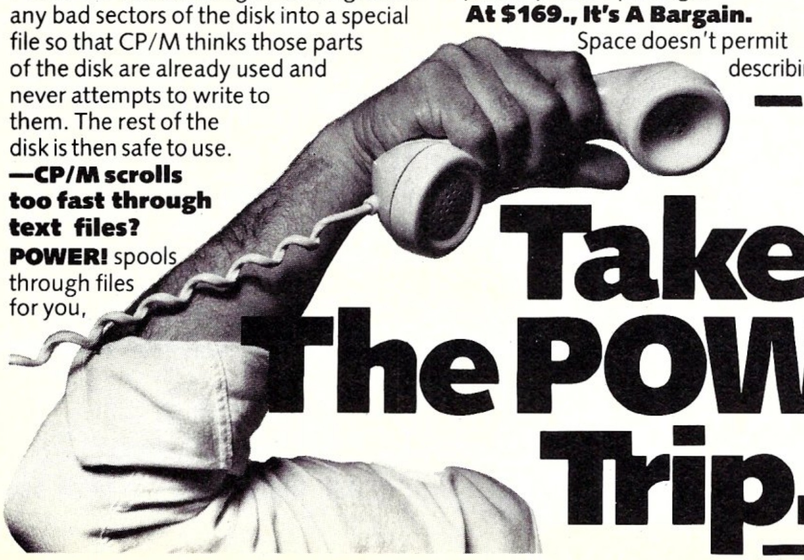
POWER! will operate in any standard CP/M or MP/M system, including CP/M-86, IBM PC, Apple (Z80 card), Osborne, Kaypro, HP, TeleVideo, TRS-80 conversions, S100's including NorthStar Vector, Morrow, CompuPro, etc. Up and running immediately, no configuration necessary—for hard disks and floppies.

At only $3. per command, you can afford to **Take the POWER! Trip.** Call or send in your order today.

NOW AVAILABLE FOR MS-DOS, TOO.

ONLY $169. Money Back Guarantee.
Charge & COD Orders Welcome.

TOLL FREE (800) 428-7825 Ext. 96AI
IN CA: (800) 428-7824 Ext. 96AI
DEALERS AND OEM'S (415) 567-1634 Ext. 96AI



Take The POWER!™ Trip.

COMPUTING!
2519 **AI** Greenwich San Francisco, CA 94123

TOLL FREE (800) 428-7825 Ext. 96AI
IN CA: (800) 428-7824 Ext. 96AI
DEALERS AND OEM'S (415) 567-1634 Ext. 96AI

ONLY $169. Calif. add 6½% sales tax
☐ **CP/M $169.** ☐ **CP/M-86 $169.** ☐ **MP/M $249.**

Card No. ____________
Exp. Date ____________
Computer ____________
Your Name ____________
Company Name ____________
Address ____________
City/State/Zip ____________

The S-100 Bus

by David Hardy

The IEEE-696 Temporary Master Interface

As we discussed last time, the IEEE-696 "special bus operations" make it possible for an S-100 system to have up to 16 temporary masters, along with its permanent master.

Now it's time to discuss some of the actual requirements of a temporary master interface. The basic definition of the temporary master interface, as listed in the IEEE-696 standard, section 2.4.1 is:

"*The temporary master interface provides the capability to transfer device dependent messages to and from a selected set of bus slaves. The temporary master thus differs from the permanent master in that it need not generate all possible bus cycles.*"

In other words, a temporary bus master can function almost exactly the same way that a permanent master functions. This includes performing the same kind of transfers to and from bus slaves, except that it need only work with a subset of the signals and functions used by the permanent bus master.

The output signals that must be furnished by the temporary master are:

- the Address Lines A0-A23,
- all status signals
- all control output signals
- the Data Output lines
- the DMA Arbitration lines, and
- the Hold Request line, HOLD*

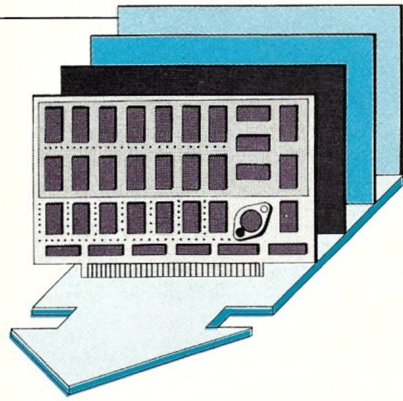
Although permanent bus masters are not required to use address lines 16-23, temporary bus masters are required to either generate these signals, or place logic 0's on them.

In addition, temporary masters should also provide a jumper on the pSTVAL* line, because some old 8080-based CPU boards do not properly transfer this line with the control output lines.

The input signals that are used by a temporary bus master are:

- the Ready lines, RDY and XRDY,
- the Hold Acknowledge line, pHLDA,
- the Data Input lines, and
- the system clock, ϕ

Thus, a typical basic TMA system



would look like the circuit shown in Figure 1.

The way this simple TMA system works is like this: First, the temporary master asserts the HOLD* line, saying that it desires temporary bus access. After the permanent master finishes executing its current instruction, it answers the request by making the hold acknowledge signal (pHLDA) true. When the temporary master sees pHLDA, it takes over the control, status, address, and data buses by turning off the master processor's tri-state bus buffers and turning on its own bus buffers. In this simple example, arbitration is not required, because there is only one temporary bus master.

After taking control of the bus, the temporary master provides the signals mentioned above, and appears to all bus slaves as their normal master controller. It is now free to perform whatever bus cycles it desires, for as long as it wants to use the bus.

When it has finished its task, the temporary master returns control to the permanent master, which was left idle during the TMA, by turning off the HOLD* signal, disabling its tri-state buffers, and re-enabling the master's buffers.

Note that, although the master must sit disabled and idle while a temporary master controls the bus, the same is not true for temporary masters while the master has bus control. In fact, it could be possible for temporary masters to be doing almost anything else while not using the bus, including talking to another S-100 system using a different S-100 bus. In this respect, the temporary bus master can be looked at as more of a "temporary bus I/O controller." If proper care were used during its design, a temporary bus master could even be powered down without affecting S-100 bus operations.

All in all, the idea of TMA promises to expand the usefulness of the S-100 bus quite a bit. But before you run out and buy a box of temporary masters for your system, you should recognize that the following problems occur with TMA:

- Most non-IEEE-696 boards use signal lines that would cause prob-

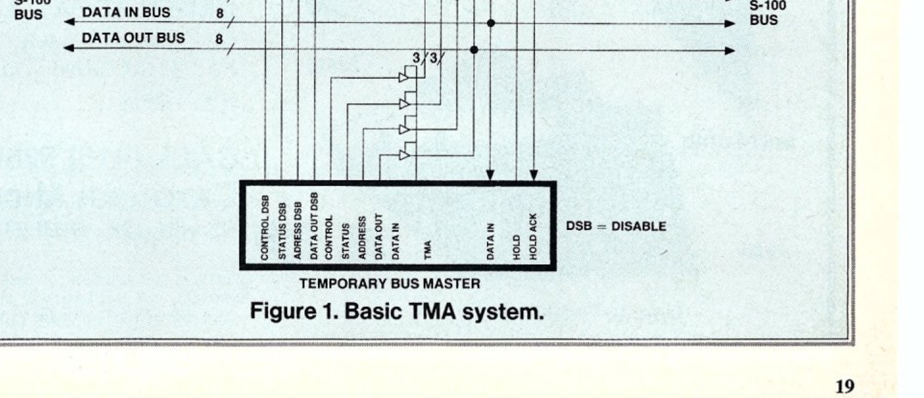


Figure 1. Basic TMA system.

S-100 Bus continued . . .

lems for TMA controllers, especially when using bus arbitration.

- As mentioned previously, the permanent master is disabled and must sit idle during temporary master access. If your system uses interrupts, for example, they would not work during TMA. TMA used on a system running under MP/M would also require some special programming to keep its TOD clock and timers correct.
- Finally, TMA devices actually take over the address and data buses of the system. Thus certain dynamic memory boards could be starved for refresh time, which could, in turn, cause them to lose data.

More reader questions

I've received several notes from S-100 hardware hackers who would like to know which lines of the IEEE-696 bus are available to them for their own special use. Basically, there are three undefined lines (called NDEF in the IEEE-696 standard) available in the S-100 bus that were intended to be used by manufacturers for their own special sig-

S-100 Bus continued . . .

nals. They are S-100 bus pins 21, 65, and 66. Effectively, these signals are also available to the hacker. The standard states that if a manufacturer uses these lines, their use must be fully documented, and a jumper must be provided to allow the line to be disconnected from the S-100 bus in the event of a conflict with other boards. It would probably be a good idea to include a jumper on any "hacker" installations that use these lines, too.

For the more adventurous, there are also four additional unused lines in the S-100 bus called RFU, which means "Reserved for Future Use." Although the IEEE standard states that these lines may not be used for *any* purpose, they are currently connected to nothing, and could probably be used by a careful hacker, at least until they are assigned by the IEEE. These lines are bus pins 27, 28, 69, and 71.

In any case, these lines should be checked before use. In the old S-100 bus layout, for example, pin 71, which is now marked Reserved for Future Use, was the front panel

LOOKING FOR A MULTI-USER, HARD DISK COMPUTER SYSTEM YOU CAN AFFORD?

It's the Morrow Decision I.™

Starting at $5,495 retail. That's the basic Morrow 10MB system with 256K of RAM. Including a 14-slot S-100 (IEEE-696) motherboard, a 400KB floppy, one parallel and three serial ports.

A bundle of software. Beginning with our multi-user Micronix™ operating system, with both UNIX™ and CP/M® 2.2 compatibility. The Decision I also includes WordStar® word processing, the Correct-It™ spelling checker, Microsoft® BASIC 80, the LogiCalc™ electronic spreadsheet and Personal PEARL,™ a relational data base manager.

Economical expandability. For individuals with growth on their minds, the Decision I offers very affordable possibilities. Another $2,000 buys 6 MB more memory, 256K more RAM and the ability to add three more users.

Give us a call. We'll get you the name of your nearest Morrow dealer. We'd like to introduce you to Morrow's complete line of S-100, multi-user microcomputers, disks and peripherals. Systems you can afford.

MORROW

600 McCormick Street
San Leandro, CA 94577
(800) 521-3493
(415) 430-1970 In California

CIRCLE 183 ON READER SERVICE CARD

S-100 Bus continued . . .

RUN signal. Pin 21, which is now marked NDEF, was originally the front panel SS (single-step) signal. In addition, in the old S-100 bus layout, all of the RFU and NDEF lines were at one time used by some manufacturer for something, so use them with caution.

In the queue

The floppy disk drive information that I promised last time is not yet completed, as I am still awaiting a few manufacturers' replies.

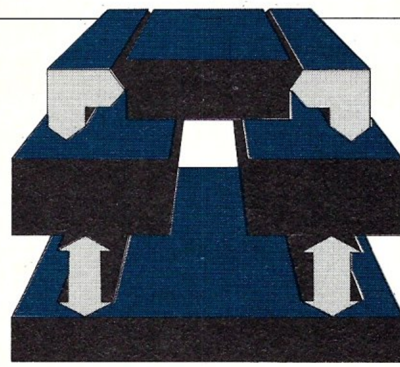
Next time, we'll cover more of the IEEE -696 standard, including how to interface simple I/O and control circuits to the S-100 bus, and we'll have some more reader feedback, including a simple repair for mysterious missing interrupts.

This column is intended as a forum on S-100 topics. Readers are encouraged to send in questions about the S-100 bus, which I will attempt to answer. Please write to: Dave Hardy, 736 Notre Dame, Grosse Pointe, MI 48230.

The UNIX File

by Ian F. Darwin

The UNIX File appears every other month and will spotlight important aspects of UNIX. If you have any questions, send them in and I will attempt to answer them. Write to Mr. Darwin at: Univ. of Toronto Computing Services, 10 King's College Rd., Rm. 4306, Toronto M5S 1A1 Canada. The opinions expressed in this column are those of the author, and not necessarily those of the University of Toronto or UTCS.



The UNIX explosion and some more introductory books

This issue of The Unix File starts with a cautionary note, which could save you a lot of money if you are buying UNIX-related expertise. I have found a few more introductory books about UNIX, and am offering my opinions on them. Finally, there's a minor elaboration on something I said earlier.

The UNIX information explosion

Now that UNIX has become a popular operating system, there is a cloudburst of activity as people rush to get to market with books, newsletters, courses and the like. Many of these are reputable, but beware! There are some quick-buck operators whose knowledge of UNIX may have been learned in the month or two before their publications appeared in print.

Because UNIX has been in use for so long at so many different places, however, there is no quick and easy way to evaluate an author's or instructor's credentials. If you are buying UNIX expertise of any sort, look around very carefully. Check the suppliers' reputation. See how long they've been in the UNIX business. Talk to some of their customers. And ultimately, *caveat emptor*.

Books

Last time, I promised to review *The UNIX Book* and *Introducing the UNIX System* in this issue. Well, half of that promise will be delayed. *The UNIX Book* still hasn't arrived, despite its announcement in *Online Books in Print* for March, 1983. Taking its place is *The UNIX Operating System* by Kaare Christian.

Writing a book about UNIX is a risky undertaking. You must either try to cover the entire gamut of topics, and run the risk of cutting some short, or concentrate on a selection of topics to the exclusion of others. *The Unix Operating System*, by Kaare Christian, seems to be the one book which covers all the topics without really snubbing any. By not trying to write a detailed guide to all aspects of the system, Christian manages to cover all the bases.

One of the critical parts of any introductory book on computers is the use of metaphor to explain new concepts. Metaphors are very helpful in relating new material to the familiar. And Christian uses what I find a very imaginative illusion, the idea of the "prototypewriter," to describe why computers need an operating system. He also used the familiar "algorithm = recipe" formulation to explain how computers follow instructions. And there are other places where the author relates the new to the known. If you already know computers very well, this book may be dull reading. You might prefer the Bell Labs manuals, which are complete but terse, or the Bourne book, reviewed here in July 1983, which assumes a programmer's knowledge of computers.

The Christian book, however slow it may be in places, covers a wide range of topics—some in just enough depth to whet your appetite. For example, the chapter on text formatting with **nroff/troff** does *not* try to make you an **nroff** expert in 25 pages or less. Instead, the author presents the key ideas of what **nroff** can do, gives examples, and moves on. You are shown how **nroff** works, advised to use formatter macro packages and told how to find out about them, and then given an introduction to the preprocessors used with the formatters (**tbl, eqn**). The other preprocessors (such as **pic/ideal** and **refer**) are not mentioned. If your only purpose in getting the book were to learn **nroff,** then you would be disappointed. But if you want to get a general feeling for all the power that UNIX offers, you might well be satisfied.

This book gives information on writing programs in the C language and on using the Shell as a programming language. The information is not as detailed as that found in the Bourne book. Unlike many books, however, the Christian book gives more than passing mention of the powerful program development tools, **yacc** and **lex**. These merit a chapter of their own—enough to show what they can do. There is also a section on system administration. The final chapter describes the internal organization and operation of UNIX. This information appears to be accurate. Again, some of the material is tutorial; it describes what a kernel is, why you need one, and how UNIX provides it. Diagrams illustrate the relationships among the data structures inside the operating system, the file system, the user process, and so on.

There's a quite detailed description of the process by which the system—and user sessions—are started up and terminated. The appendix is the author's version of the "manual" entries for 40 common commands. These are in the same format as the Bell Manuals, but are not reprints from any Bell Manual I've ever seen. Christian has rewritten them to be a little less terse. The most common command—the editor—has no description here, probably due to the variations among editors in use. The book ends with a glossary of computer terminology and an index.

The other half of my promise I am keeping. I have spent some time curled up with a copy of Henry

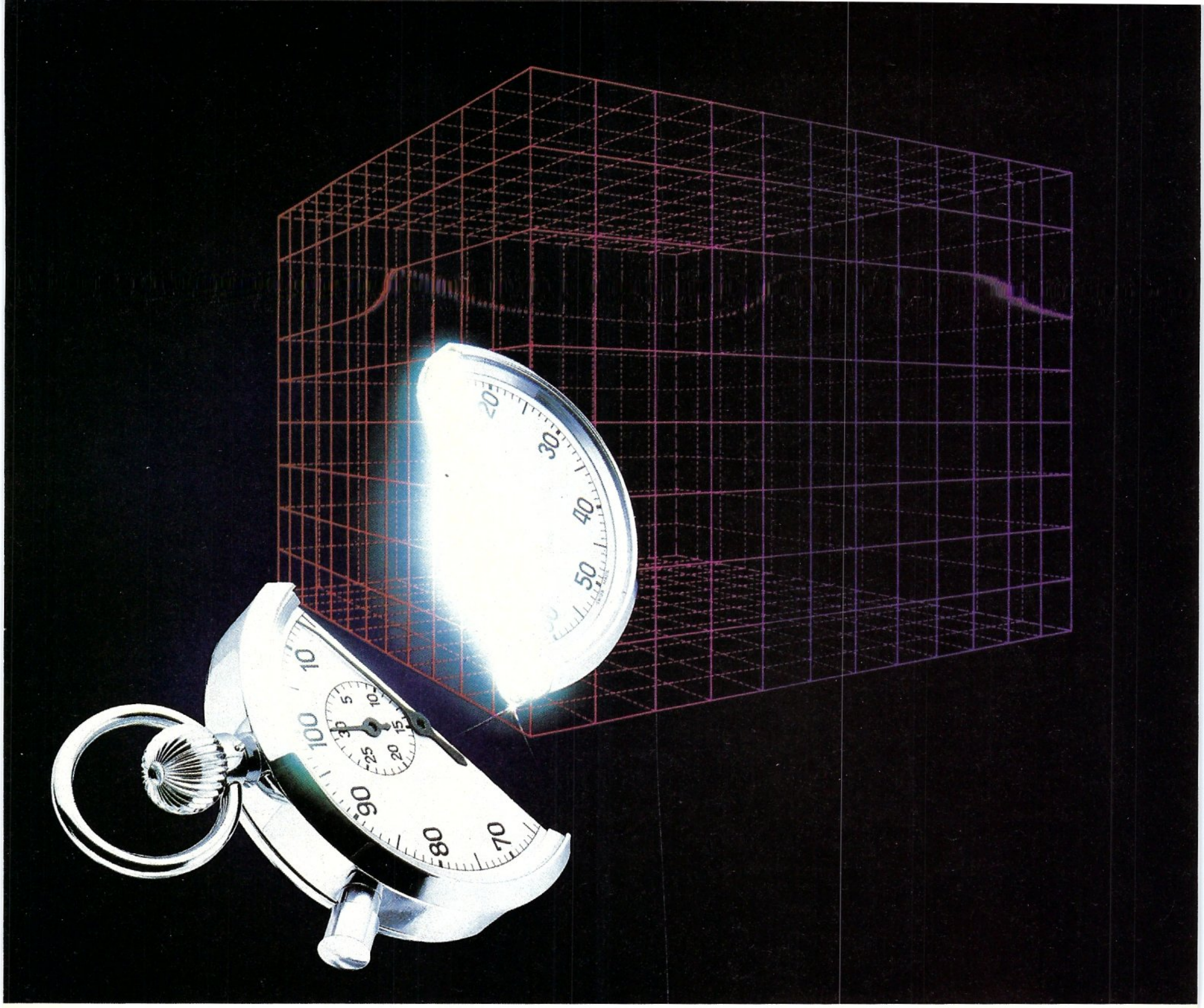
UNIX continued . . .

McGilton & Rachel Morgan's *Introducing the UNIX System.* It bills itself as "an introductory guide for users who are new to the UNIX system." This book, too, tries to cover practically all bases, from a comprehensive introduction to a chapter on system management. In between there are chapters on text manipulation, the **ed** and **sed** editors, the **ex** and **vi** editors, formatting documents with **nroff,** advanced formatting, advanced use of the Shell, tools for program development, and "The UNIX System at Berkeley." In fact, most of the basic information about using UNIX is here. There are many, many examples of using common programs that are part of UNIX. There are no specific exercises, but each chapter concludes with a quick review and suggestions on trying out the parts of the system introduced.

Why, then, do I have lingering doubts about the book? There are no glaring technical errors. There are some omissions, minor distortions in the name of simplicity, and a few mistakes. The authors refer to the growth of UNIX on machines "such as those using the Motorola MC68000 and the Zilog Z8000 microprocessors." This is reiterated a few pages later. In neither case is there any mention of the Intel 8086, which is the heart of many, many small UNIX systems—mainly XENIX implementations. The authors refer to the system file **/usr/pub/greek** as "the file which gives information to the text formatters about how Greek symbols can be printed on certain terminals." This shows a real confusion: this file is never read by text formatters, but by *people* to see how to enter requests *to* the text formatters to produce Greek symbols!

Like all these books, *Introducing the UNIX System* talks about text formatting. And there's a discussion of the differences between text formatters and word processors. The latter, the authors inform us, "cannot cope with automatic chapter and section numbering . . . neither can they generate the table of contents automatically. These tasks have to be done manually, and are a potential source of error." This may be true, but it's interesting to note that the book suffers from just this type of error in its contents listing. The appendix, "A Selected UNIX Bibliography," appears in the table of contents as "Responses from UNIX Commands." And in the acknowledgements the writers express their gratitude to a maker of CP/M software "packages which ease the job af [sic] catching spelling mistakes and grammatical errors." And that seems to summarize the book! No major, glaring, catastrophic errors. But a continuing parade of minor errors, stylistic gaffes, and other oversights. Extraneous commas crop up like weeds on the terrain of this book, and the authors violate many of the "Elementary Rules of Usage" in the well-known *The Elements of Style* by Strunk and White. Ironically, Strunk and White are highly recommended in a *Byte* editorial on "What's Wrong With Technical Writing Today" (December 1980, page 8). The McGilton-Morgan book was published in the "Byte Books" series by McGraw-Hill, owners of *Byte*.

Is all this to the point? I believe that it is, for if ever there was an operating system with a built-in emphasis on *style,* that system was (and is) UNIX. UNIX appeals to many sophisticated computer users because of its elegance, simplicity, and clear style. There is room—in UNIX and in the English language—for variations in style. There is not room for syntax errors in most programming languages and there is not, I submit, room for sloppy grammar in a textbook from a reputable publishing house. A shoddy programming style leads to bad programming. If unchecked, it leads to rockets that sometimes fail to make it home. And a shoddy use of English leads to poor communication. If the misuse of words, and the sprinkling of commas, where, they don't, belong, bothers you, then, you will, have trouble, reading this book. The authors have basic information about UNIX. They just don't introduce the UNIX system very elegantly.



Invest in Time

Your Customers Can't Wait

Getting your project completed on time is critical. Investing in the right software tools puts time on your side.

We write software for a living. We understand the importance of time to the professional programmer. To complete projects on time you need good software tools. So we create high quality, timesaving tools for users of DEC and MC68000-based computer systems.

Pascal-2 Compiler It generates fast, compact code. Because the compiler does the optimizing, programmers can spend time on other programming tasks. Because Pascal is a structured language, other programmers can easily read your programs. Indeed, it's the language most programmers are learning today.

Additionally, programs written in Pascal-2 are portable. Now you can change hardware without having to rewrite your software.

Additional TimeSavers The time you save with our Pascal-2 compiler is only the beginning. We also provide a full line of other TimeSavers. *Pascal-2 source-level Debugger* for true high-level debugging ease. *SourceTools* for control and management of changes to source programs. *Concurrent Programming Package* for real-time scheduling and device drivers. *Profiler* for identifying performance bottlenecks in programs.

The Pascal-2 Compiler is now available on UNIX for the MC68000 and PDP-11 computer systems.

TimeSavers from Oregon Software.
Your customers can't wait. And neither can you.

Oregon Software

2340 SW Canyon Road
Portland, Oregon 97201
(503) 226-7760
TWX: 910-464-4779

The Pioneer in Performance Pascal

DEC and PDP-11 are trademarks of the Digital Equipment Corporation. MC68000 is a trademark of Motorola Inc. UNIX is a trademark of Bell Labs.

CIRCLE 69 ON READER SERVICE CARD

Some other UNIX books

The original UNIX documentation is still available. In fact, Bell has reprinted the Version 7 Manual Sets in 8.5″ × 11″ paperback to take advantage of the mass market for UNIX information. Also, the detailed *Bell System Technical Journal* issue on UNIX is now in its fourth printing. Also planned for this fall is a new *BSTJ* issue on UNIX. And Bell has also brought out a range of other manuals for UNIX System V. I hope to describe these in more detail later.

Most vendors of UNIX systems include reprints of the appropriate manual sets (V7, Berkeley, System III) adapted for their system. A few reprint the manuals without adapting them for the vendor's particular hardware. A large computer manufacturer has announced UNIX for one of its minis, and the User's Manual for this product appears to be an exact reprint of the Bell Labs System III User's Manual. Vendors of systems based upon UniSoft "UniPlus+" UNIX seem to deliver Bell/Berkeley manuals adapted for the UniSoft system. UniSoft also reorganized the "documents" section into "Program Development Tools" and "Tutorials and Document Preparation." For System III, they revert to Bell Manual sets. I'm not familiar with every vendor's documentation, but most tend to be derived, like UNIX, from the Bell manuals.

There are also several recent books on the C Programming Language. Since these are not completely specific to the UNIX system, I am not going to review them here (see Book Review, page 120, for some brief notices).

A minor correction

It's been said several times that Dual has a strange policy of charging extra for the "reconfiguration files." These, like the BIOS on CP/M, are needed to reconfigure the operating system to conform with new hardware, other than just the addition of memory or devices identical to those on the system. In fact, DUAL has this policy, but it's not their doing. The decision was made by UniSoft, who supplies many manufacturers with the UniPlus+ port of Bell Labs' UNIX (with some Berkeley enhancements). The list of UniSoft OEMs reads like a sort of "Who's Who" to the 68000 industry—Callan, Codata, Sun, Pixel, Wicat, Dual, and many others all got their UNIX effort started from a UniSoft port. Many of them still sell UniSoftware, so you'd expect to pay extra for driver sources from any of these companies. Either that, or the price is bundled into the system.

Next time, I'll have a few more books (hopefully *The UNIX Book* and *The UNIX Guide*) as well as some details from the Summer of '83 USENIX Conference (Toronto, July), and a look at obtaining public domain software for UNIX. Until then, keep (those disks) spinning!

Errata

Mr. Darwin's article, "The 50-Line Text Formatter," (August 1983) contained an error. The line

FILE INFILE;

should have read

FILE *INFILE;

We apologize for the error.

Life in the Fast Lane: Three Multiuser Microcomputers

A comparison of the Altos 586, CompuPro 8/16C, and the Ithaca Intersystems Encore

by Bill Machrone

The number of fast, inexpensive multiuser computers is increasing daily. They occupy a position once reserved for the low end of the minicomputer market. Now, though, you can get a multiuser machine in one of two major flavors: MP/M-based machines that provide the operating familiarity of CP/M, and UNIX-based machines that virtually duplicate the capabilities of their minicomputer forebearers. [*Editor's note: There is a third option: a multi CP/M system running CP/NET on Turbodos. We will publish a review of CP/NET in the October 1983 issue, and a review of Turbodos later.*]

Here we look at a trio of machines, dissimilar except for their multiuser might at an affordable price. One runs Microsoft's XENIX, another runs MP/M-80, and the third uses a homegrown hybrid MP/M-86 that runs 8- and 16-bit programs simultaneously. The XENIX machine is Altos's 586, the compactness of which belies its computational power. The hybrid is CompuPro's 8/16C, an S-100 system built around their famous 8088/8085 dual processor card. [*Editor's note: This was the first S-100 dual processor board to appear on the market.*] Ithaca Intersystems' offering is the Encore, an extension of their longstanding experience with fast Z80 systems. The Altos is a single-board design, the 8/16 is a pure S-100 design, and the Encore uses a slightly modified S-100 bus. This article will be less a direct comparison than a description of their capabilities and limitations.

A bit of history

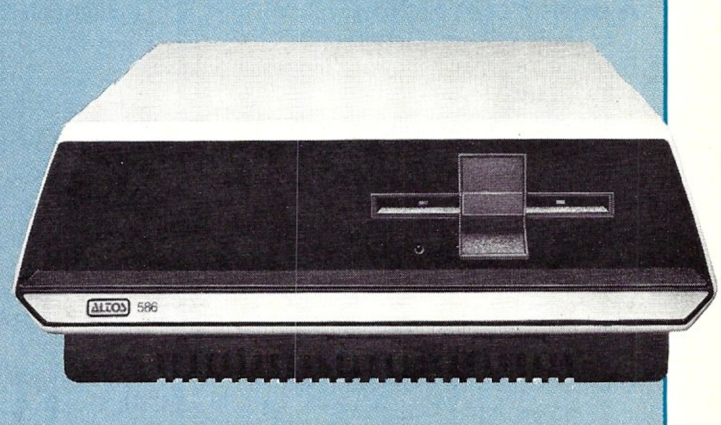
Before we begin, we must first express a debt of gratitude to the progenitor of all microprocessor-based multiuser systems, the Altos 8000 line. Back when many of us were getting started in the microcomputer field, it was great fun to sneer at Altos machines for their single-board design. We S-100 purists wanted nothing to do with them. Altos, unfazed by this rejection, sneered back at the hobbyists and continued on its course of producing inexpensive, reliable, small business computers. Even some of their best efforts left the hobbyists unimpressed.

A case in point is the 8000. In production for about three years, its specs still don't sound all that bad: 208K, 6 serial ports, a parallel port, a double-density 8″ floppy, and 10 to 40 MB of hard disk, with MP/M as the resident operating system. The 8000 sported a 4 MHz Z80 and DMA, could boot from hard disk or floppy, and permitted four simultaneous users. It used all the power available in the Z80's Mode 2 interrupt structure and Zilog's excellent peripheral family, including the SIO. The SIO is one of the main reasons the machine works so well. In addition to fast communication with the CPU through the interrupt structure, it has a 4-byte FIFO buffer for incoming characters. This buffer makes it virtually impossible to lose a keystroke.

On the down side, the fourth user area provided only about 30K of transient program area, too small to run significant user programs like WordStar and dBASE, so it was really only a three-user machine. Utilities could run in the short bank, but no serious application programs would fit. The bottom line was that this machine allowed three users to run WordStar or dBASE II or other applications with relatively little conflict. If multiple users went to the disk at the same time, you noticed the degradation of response time. Otherwise, the interrupt-driven console handler kept your terminal well serviced. Altos has sold thousands of these machines to businesses, and they continue to do so. They made small multiuser systems believable.

Altos 586

The Altos 586 bears no resemblance to its 8-bit predecessor. Its stylish case conceals 512K of parity-checked RAM, an 8086 running at 10 MHz, 10 MB of hard disk, and 6 serial ports. Ethernet communications can be handled by the addition of a board containing the new Intel chip set. There is a socket that holds an additional 512K memory board, and expansion is provided for tape backup and additional serial ports. The serial ports still use Z80-SIO chips and are controlled by a Z80 dedicated to input/output tasks. In this manner, Altos has off-loaded I/O from the 8086. This, coupled with a buffered hard disk interface with DMA access to memory, results in a very



The compactness of Altos 586 belies its computational power: this XENIX-driven system is a very fast machine.

Bill Machrone, 121 North Ave., Fanwood, NJ 07023

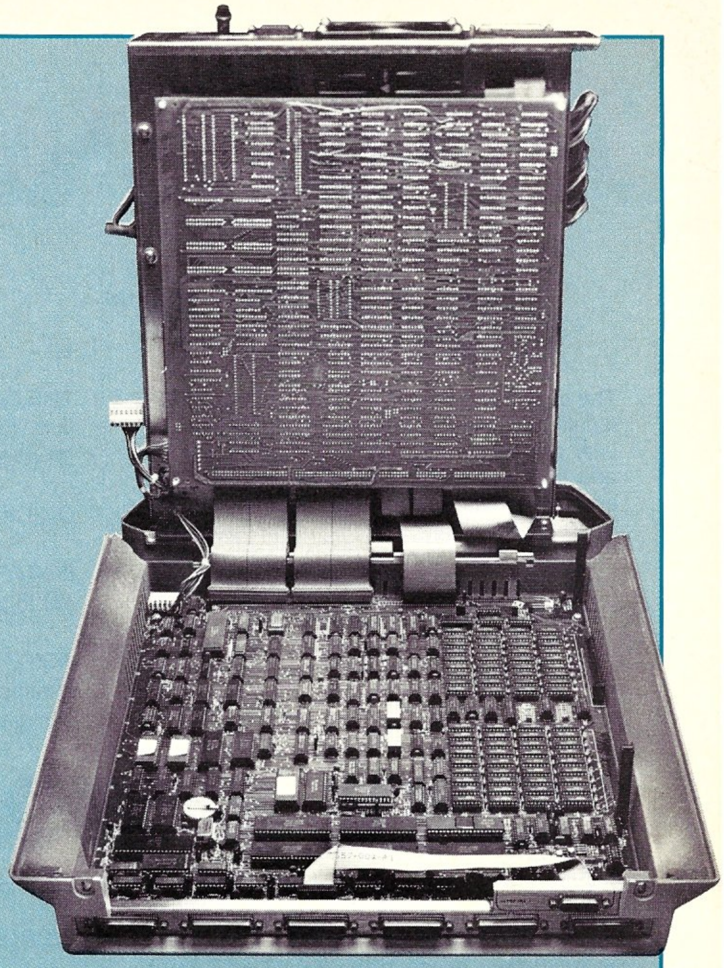
fast machine.

The hard disk controller has an 8089 I/O processor to make things simpler to program while providing a consistent interface to the 8086. Unlike Altos' 8600 series, the 8089 is not on the system bus with the 8086, but communicates through DMA. This allows the 8086 to crank along at 10 MHz, even though 8089s that can go that fast are still not available. There is a real-time clock backed up by a NiCad battery on the main board, but it is not interfaced to XENIX. You have to tell XENIX the time and date when you boot, and it keeps track from there. The minifloppy disk sports 700K of dual-sided quad-density storage. XENIX normally treats it as the tape device, but it can be configured as a random-access device as well. Architecturally, the 586 bears more resemblance to a minicomputer than a micro. Along these lines, Altos elected not to wait for the 80286 and designed their own memory management unit. It assigns workspace to each task dynamically from a memory pool, as though each task had as much contiguous memory as it wanted. In conjunction with XENIX's swapper, tasks share all the memory available on a demand basis.

The machine I tested had the optional XENIX development system in addition to the runtime package. This is essentially UNIX System III with Berkeley enhancements. You get a C compiler, Fortran 77, all the usual UNIX programmer productivity aids, and a selection of text editors. Also included is **uucp**, the UNIX-to-UNIX communication program, the "learn" library, and a host of utilities (7MB worth) that will take you months to go through.

Altos's Business Shell provides a convenient method of harnessing the system's power, even for neophytes. All the system utilities that you are likely to use are callable from one of several menus available to you. Further, every menu item has an associated help screen. There is even a "shell compiler," which permits you to create your own menu-driven applications. The user setup program, also menu driven, permits you to assign each user to a specific shell, based on login ID. A bug in the Business Shell program prevented me from substituting my own menu for the standard, but a UNIX-knowledgeable friend showed me a way around it. Getting the system up and running was a little time-consuming, but straightforward. A utility verifies the structure of the file system following installation, and whenever necessary afterward.

This system is productivity heaven. UNIX utility programs tend to be large compared with equivalent CP/M programs. The time required to load them from the hard disk is noticeable. After that, though, look out! Multiple tasks from one terminal, spooling, other users—bring 'em on! It was tough to find anything that would slow this system down. With two or three users on the system, it appeared to do most everything in memory, with few disk accesses. Only when five users had the machine fully busy did the operation of the swapper become apparent, and then only when one or two of the users were running disk-intensive relational database stuff or compiling. I suspect that the 512K add-in board would make much of that go away, too. I've used some largish minicomputers that were slower than the 586. Maybe they handled more users, but they also cost at least an order of magnitude more.



The Altos 586 has 512K of parity-checked RAM, an 8086 running at 10 MHz, 10MB of hard disk, and 6 serial ports.

CompuPro 8/16

CompuPro is committed to excellence in the S-100 field. As such, they offer the most comprehensive line of high-performance S-100 boards available. Recently, they have offered their products integrated into systems of varying capabilities. Most interesting of their offerings is the 8/16C, built around the dual-processor board and fast static RAM. The unit I tested has 384K of RAM (six RAM 17s) and 1.5 megabytes of M-Drive/H fast disk-emulating RAM. It also had nine serial ports and provisions for handling up to seven users. Eight of the ports are provided by an Interfacer 3-8, while the ninth is the serial port on the system support board. The system support board also contains a battery backed-up clock, ROM sockets, a socket for a math chip, and a bagful of interrupt control lines. The test unit contained the CSC (Certified System Component) boards, offering 12 MHz operation of the 8088 and a 6 MHz rate on the 8085. The Disk 1 floppy disk controller uses DMA to transfer data in and out of memory. The box, motherboard, and power supply are simply the toughest and best-built around.

Multiple tasks from one terminal, spooling, other users—bring 'em on! With 2 or 3 users, Altos' 586 appeared to do most everything in memory, with few disk accesses. It was tough to find anything that slowed it down.

Programming

dBASE II™ $398

The most widely sold DMBS for micros, dBASE II defines the state of the art. An inexperienced user can create a dBASE file, begin data entry, and print out a report in minutes; the experienced programmer can create complex program systems using dBASE's unique programming language. Both can use automatic program generators and other programming tools to create applications quickly and easily.

THE DATABASE ACCELERATOR: Automatically creates menus and "intelligent" data entry screens which check input for correctness. Eliminates 80—100% of dBASE programming time! **$195**

QUICKCODE: Fox & Geller's popular generator. **$185**

dGRAPH: Allows you to see your dBASE file in graphic form—a real aid to understanding. **$195**

ABSTAT: Allows you to analyze your data using a wide variety of statistical tests. Can be used with dBASE files or by itself. Also does bar graphs and plots. **$359**

EDIX & WORDIX . . $279

EDIX is a full-screen editor specifically designed to take advantage of the advanced features of the IBM PC; WORDIX is the best text formatter on the market. Together, they are an unbeatable combination, the finest word processor available on a micro—at an unbeatable price!

EDIX alone **$149**

WORDIX **$149**

MULTIPLAN™ $189

The world's hottest-selling spreadsheet, Multiplan™ is now setting the standard for ease of use and quality of documentation. Each of the Multiplan™ series comes complete with sophisticated training programs.

Multi-Tool Budget 'expert system': used with Multiplan, enables new users to get budgets out in minutes **$69**

Multi-Tool Financial Statement: Expert help in getting out financials. **$109**

SMARTMODEM 300 $209

1200 $509

1200B $479

These Hayes Stack Modems link your computer's RS232 serial port directly to a modular phone jack. They will dial the phone for you, and answer it as well. The Smartmodem 300 runs at 0—300 baud; for those needing higher speed, the Smartmodem 1200 can run at 1200 baud.

NEW! NEW! NEW! **Smartmodem 1200B:** for the IBM PC only............ **$469**

(800) 222-8811

International

CP/M-86 (8″)

Database

	LIST	SALE
Ashton-Tate		
dBASE II 700	429	
dBASE II w/user's guide	729	419

Language

	LIST	SALE
Digital Research		
Pascal MT+86 With SPP-86	800	600
XLT 86	150	132

Utility/Comm/Graphics

	LIST	SALE
Woolf		
Move It	150	99

CP/M-80

8″ standard SSSD and Apple 5¼″.
Please call to confirm price & availability for other formats, available as special orders.

Business

	LIST	SALE
ATI		
Power for CP/M	75	54
Artificial Intelligence		
Dental (PAS-3)	995	749
Medical (PAS-3)	995	749
Ashton-Tate		
Bottom Line Strategist	400	279
Aspen		
Grammatik	75	56
Proofreader	50	38
Balcones		
BOSS Fin. Acctg. Sys	2495	1496
Comshare		
Combo Pak	125	94
Masterplanner	325	253
Planner Calc	99	74
Target Application Guide	50	38
Designer Software		
Palantir	425	319
Dictronics		
Random House Thesaurus	150	319
Digital Research		
Display Manager	400	295
Epic		
Supervyz	150	94
Lexisoft		
Spellbinder	495	281
Metasoft		
Benchmark Mail List	250	184
Benchmark Word Processor	500	367
Micro Pro		
Calcstar	145	96
Mailmerge	250	165
Spellstar	250	165
Word/Mail	695	426
Word/Spell	695	426
Word/Spell/Mail	845	558
Wordmaster	150	99
Wordstar	495	327
Microsoft ®		
Edit 80™	120	94
Multiplan™	275	198
Text Editor™	120	94
Oasis		
The Word Plus	150	112
Peachtree		
Inventory †	400	275
PEACHPAK SERIES 9	500	345
Payroll †	400	275
SERIES 4- GL	600	398
SERIES 4 PAK (GL,AR,AP)	595	398
SERIES 40 PAK (GL,AR,AP) †	595	398
SERIES 40 GL †	400	275
SERIES 40 AR †	400	275
SERIES 40 AP †	400	275
SERIES 9 - PEACHTEXT	500	345
SERIES 9 - PROOFREADER	300	206
Series 4 - AR	600	398
Series 4 - AP	600	398
Series 4 - Inventory	600	398
Series 8 - GL	750	515
Series 8 - AR	750	515
Series 8 - Inventory	750	515
Series 8 - Payroll	750	515
Series 8 - Sales Inv.	750	515
Series 8 - PAK (AR,AP,GL)		
Series 80 PAK (GL,AR,AP) †	595	398
Series 80 GL Videoterm †	400	275
Series 80 AR Videoterm †	400	275
Series 80 AP Videoterm †	400	275
Series 80 Inv. Videoterm †	400	275
Series 9 - Mailing List	375	258
Series 9 - Peachcalc	375	258
Select		
Select Word Processor	595	356
Sensible		
Sensible Speller †	125	94
Software Dimensions		
Accounting + GL	645	398
Accounting + AR	645	398
Accounting + AP	645	398
Accounting + Inventory	645	398
Accounting + Payroll	645	398
Accounting + SOE	645	398
Accounting + POE	645	398
Accounting + POS	645	398
Sorcim		
Supercalc	195	129
Supercalc 2	285	179
Superspellguard	195	
Superwriter	295	185

Database

	LIST	SALE
Anderson-Bell		
Abstat	395	359
Ashton-Tate		
dBASE II	700	429
dBASE II w/user's guide	729	449
Caxton		
Cardbox	245	177
Computer Control		
Fabs-BTREE File Access	195	146
Ultrasort II	195	146
Condor		
Condor 1	295	189
Condor 3	650	398
DJR		
FMS 80	395	275
FMS 80-1	250	299
FMS 80-2	145	179
Digital Research		
Access Manager	300	225
FYI		
Superfile	195	129
Faircom		
Micro B+ (CBASIC)	260	211
Micro B+ (OTHER)	260	211
Fox & Geller		
Quickcode	295	195
Quickscreen/dBASE II	149	125
Quickscreen/CBASIC	149	125
Quickscreen/MBASIC	149	125
Quickscreen/FMS-80	149	125
dGRAPH	295	195
dUTIL	99	59
Human Soft		
DBPlus	125	95
Micro Pro		
Datastar	295	195
Infostar	495	327
Reportstar	350	231
Microsoft ®		
Sorting Facility (MSORT)™	195	151
Optimal		
Accelerator		195
Organic		
Datebook	295	229
Pear I Soft		
Personal Pear I	295	229
T/Maker Co.		
T/Maker III	275	215

Home/Education/Games

	LIST	SALE
Infocom		
Deadline	60	43
Starcross	50	38
Suspended	50	38
Zork I	50	38
Zork II	50	38
Zork III	50	38
ISM		
Mathemagic	100	72
Supersoft		
Dungeon Master	40	30
Nemesis	45	34

Language

	LIST	SALE
Digital Research		
CB80	500	395
CBASIC	150	94
CIS COBOL	800	595
MAC	90	75
PL/1 80	500	375
Pascal MT+	350	262
Pascal MT+ with SPP	500	395
SID	75	63
SPP	200	150
ZSID	100	88
Ithaca		
Pascal BZ	450	375
Pascal Z	450	375
Pascal Z & BZ	650	542
Microsoft ®		
ALDS™ †	125	94
Basic 80	350	252
Basic Compiler	395	296
Cobol 80	750	562
Fortran 80	500	360
Fortran 80 †	195	146
MACRO 80	200	150
Mumath/Musimp	250	194
muLisp/muStar	200	156
Sorcim		
ACT 65	175	126
ACT 68	175	126
ACT 69	175	126
ACT 80	175	126
ACT 86/88	175	126
Pascal/M-80	395	284
Pascal/M-86/88	495	356
Supersoft		
ADA Compiler	300	225
C Compiler	250	188

Utility/Comm/Graphics

	LIST	SALE
Byrom		
BASTAM	200	149
BSTMS	200	149
Digital Research		
Despool	50	38
Micro Pro		
Supersort	250	165
Microstuf		
Crosstalk/Smartmodem	195	135
Peachtree		
Series 9 - Telecom	375	258
Southwestern		
Z-Term †	100	75
Z-Term Professional †	150	109
Supersoft		
Disk Doctor	100	75
Woolf		
Move It	125	83

HARDWARE, NOT MACHINE SPECIFIC

	LIST	SALE
Amdek		
300 (12″ green)	200	157
Color 11 (R&G)	899	645
Corona Data Systems		
11MB Hard Disk	3295	2695
20MB Hard Disk	4295	3495
6MB Hard Disk	2295	1895
PC Interface,Cable,Man.	300	239
Corvus Systems		
11MB Hard Disk	3295	2695
20MB Hard Disk	4295	3495
6MB Hard Disk	2295	1895
Hayes Microcomputer Products		
Chronograph	249	199
Smartmodem 1200	699	509
Smartmodem 300	279	209
NEC		
Monitor 12″ Green Hi-Res	285	167
Novation		
J Cat	149	119
Smart Cat 103	259	207
Smart Cat 103/212	595	476
Quadrum Corp.		
Microfazer	189	160
Microfazer, in-line serial	330	270
Microfazer inline paralle	225	185

™ = Manufacturer's trademark
® = Microsoft is a registered trademark
† Apple CP/M only

Orders must be **PAID** by **October 31** to qualify for these special prices

TELEPHONE ORDERS

Outside Calif.: **(800) 222-8811**
Inside Calif.: **(415) 324-3730**

Order lines are manned 6:30—6 Monday thru Friday and 9—5 Saturday. Other lines are open 9—5 Monday thru Friday.

Technical Support ***(415) 324-0311***
Order Status ***(415) 324-0306***
Sales Manager ***(415) 324-0305***

More CP/M®, APPLE®, IBM PC™, UCSD p-System™ software, hardware, etc.: call for quote.

PROGRAMMING INTERNATIONAL
505 Hamilton Avenue Suite 301
Palo Alto, California 94301

TERMS: All prices subject to change without notice and avialability. Cashier's check/MO/bank transfer. Allow time for company or personal checks to clear. Prices reflect cash prepaid discount. VISA/MASTERCARD/COD/PO's + 3%. CA residents add sales tax. All sales final for games & special orders.

SHIPPING: $3 per item for UPS surface ($6 for Blue Label) within Continental USA, except where shipping cost is specified in square brackets. UPS does not go to Canada, Alaska, APO's, FPO's; call for ship charge or add 15%—we will refund/credit difference.

RETURNS: Must have authorization number, obtained at 415-324-0305. Unauthorized returns will be refused; damaged goods will be refused. All returns subject to 15% restocking fee. No return after 30 days.

Also available at

Software to go

Call (800) 222-8811, (415) 324-1099, or local information for Software to go location nearest you. Software to go prices may differ slightly. Local sales tax applies.

OPENING SOON

California
- Los Angeles
- Orange County
- San Diego
- Palo Alto

Illinois
- Chicago

New Jersey
- Camden
- North Jersey

New York
- Manhattan

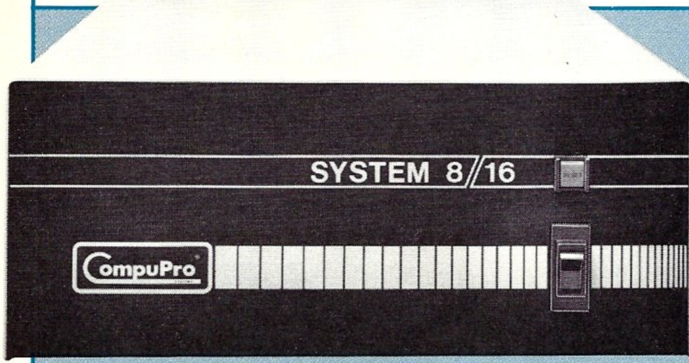
Texas
- Dallas-Ft. Worth
- Houston

to serve you better!

(800) 222-8811

CIRCLE 73 ON READER SERVICE CARD

Life in the Fast Lane continued . . .



The CompuPro 8/16 is built around the dual processor and 384K static RAM; 9 serial ports can handle up to 7 users.

Interior view: the 8/16's box, motherboard, and power supply are simply the toughest and best-built around.

The software that ties all these pieces together is MP/M 816. It is a modified version of MP/M-86 that detects when you want to run 8-bit programs and hands those tasks off to the 8085. It does this by requesting a 64K workspace from the 8088, then setting up a small BIOS at the top of the workspace. It intercepts CP/M calls, translates them to the MP/M-86 equivalent, then hands them back to the 8088 for execution. Thus, the 8088 is the I/O processor for the 8085. There is also a "shell" program instead of the usual terminal message processor (TMP). Through login IDs, it controls the user areas to which each user may have access and can be set up to force a given user to a specific disk and program. Terminal and printer setup is done through configuration files. This permits rapid and simple system reconfiguration by editing these files. Baud rates, number of stop bits, and the like, can all be changed at will.

The 8/16 came with a pair of dual-sided Qume 8″ floppies in a matching cabinet. No one would want one of these without a hard disk and, fortunately, they are available so configured from CompuPro's growing number of System Centers. These System Centers, or "super dealers" are a vital link in the chain from CompuPro to the customer. They provide technical support, service, custom systems, and support of smaller dealers with fewer technical resources. The first System Center was Gifford Computer, owned by Dale Gifford, also of G&G Engineering, the firm that first made hard disk subsystems a reality for CompuPro hardware. Most of the System Centers choose the Fujitsu or Memorex 8″ drive. All use the G&G/CompuPro system software. Having had one of each on other CompuPro systems, I can attest to their speed and reliability.

I had a few problems with the 8/16 in multiuser mode. One of them was traced to a faulty 50-pin connector going to the disk drives. Another one, which remained unsolved, was that one user would occasionally cause another to crash. For example, if two users were running WordStar and one exited, the other would sometimes find himself out of WordStar and back at the system level. This only happened when both users were running 8-bit programs and did not seem to follow any set pattern. The folks at CompuPro were stumped, as this is evidently not a problem on other 8/16 systems.

Although the 8/16 is fast, I liken its performance to that of a dragster compared to a sports car. It's super in a straight line, but it isn't so adaptable when things get twisty. Slower machines can appear to outperform it through good hardware design. Specifically, the lack of interrupt-driven I/O is a limiting factor. This is somewhat ironic, since CompuPro has a very sophisticated I/O processor (the MPX board) that could be integrated into this system. The real measure of how "fast" a system is does not lie in clock speed, but in how quickly it can process characters from the keyboard. A "fast" system is one that can convince the user that he is not being ignored.

RAM disk is one way to accelerate the performance of a system, but is not without some risk. There is no disputing the speed of the M-Drive/H on this system. However, I would never depend on it unless the system were connected to an uninterruptible power supply. Systems like XENIX do their work with RAM copies of disk files, but the operating system automatically writes changed buffers back to disk several times a minute. MP/M has no equivalent capability. Furthermore, you wouldn't want it in a floppy disk environment.

CompuPro's 8/16C, built around the dual-processor board and fast static RAM, runs 8- and 16-bit programs simultaneously. The software that ties it together is the MP/M 816, a modified version of MP/M-86.

On the other hand, I know of a fellow in California who teaches dBASE II programming with the aid of an 8/16C. He has half a dozen terminals hooked up to it, and his students use them simultaneously. There is an electronics design shop on the East Coast that uses an 8/16 to support four engineers and a secretary. They run everything from word processing to a C compiler to spreadsheets. The machine comes with dBASE II and Sorcim's Supercalc-86, by the way. As you might expect, Supercalc really flies at 12 MHz on the 8088.

Ithaca Intersystems Encore

The Encore is a departure from Ithaca Intersystems' standard line of S-100 machines. Although it uses an S-100 bus, the power lines are all regulated, and the on-board regulators have been removed. Some contend that the S-100 bus should have been designed this way in the first place, but it wasn't, so you can't use regular S-100 cards in this machine without modifying them. The reason they modified the boards was so they could use a compact, cool-running switching power supply instead of the usual behemoth. Another interesting difference in Intersystems boards is that they do a parity check on all board-to-board data transfers. One of the non-defined S-100 lines is used as a ninth bit, and another is used as a parity request line. True to the S-100 standard (even though the power arrangement makes them nonstandard), the feature is automatically disabled when boards incapable of providing parity are mixed into the system. Selection of state-of-the-art components permitted Intersystems to pack this machine with a lot of features. There's an 8″ floppy for compatibility with the CP/M world, a 5¼″ Winchester and room for a "removable Winchester" (which is surely a contradiction in terms) and a 5¼″ floppy. The Encore that I tested had a 10 MB hard disk and no SyQuest. The floppy is a Shugart half-height unit. It also had a half megabyte of dynamic RAM, which, in addition to user areas under MP/M, also holds a track buffer for disk access. It is an adaptation of the Intersystems Cache BIOS, reworked for MP/M. It buffers disk directory information, in addition to the most recently used sectors. This really accelerates the system's performance. Large chunks of WordStar overlays, files you are accessing, and system utilities wind up in the buffers when you use them repeatedly. Thus, even the hard disk is accessed less frequently, and everything gets done faster.

In addition to MP/M, the Encore also comes with a proprietary Intersystems word processing package and Pascal/Z. Pascal/Z was one of the first true Pascal compilers for CP/M which, despite a buggy beginning, has gained wide acceptance and has a very active user group. I didn't get a chance to try the word processor, but it looked like good quality stuff.

Booting the Encore for the first time produced a horrendous grating sound from the floppy disk drive. At first I feared that something was wrong, but it was nothing more than the sound the stepper motor makes at slow speeds. After the system is all set up, it boots from the hard disk. No more horrible noises. My test machine supported five users and a serial printer. If you're used to a 4 MHz unbuffered system like the Altos 8000-10, you'll notice the difference on this machine immediately. Cold loads of big programs are no faster, since it always takes a finite amount of time to get programs off the hard disk, but actual operation is significantly better. User setup is done via a convenient menu-driven program that permits each terminal to run at different speeds and have different device characteristics. Thus I was able to set up one port to answer a Hayes Smartmodem and used MP/M's version of Submit to route the caller into a password/login program. I've done this on several MP/M systems, and it works well. The configuration program, ICONFIG, is powerful, but potentially confusing for a new user. Some of the available options require "tuning" decisions, and there is precious little discussion in the documentation to guide you. For instance, into how many logical drives do you want to divide your hard disk? How many directory entries do you want?

The Ithaca Intersystems Encore uses an S-100 bus, with regulated power lines and on-board regulators removed.

Encore boards do a parity check on all board-to-board data transfers. A nondefined S-100 line is used as a ninth bit.

Do you want to use Cache BIOS's write-back feature? How often? There should be a simplified version of this program that takes some standard default values. But I'm carping. Experienced users will really appreciate the tailoring power of this program.

There is another system-tuning aid called Command Mode. Pressing a certain (user-definable) control code during normal operations puts you into a BIOS-resident system monitor. It allows you to display the contents of various queues, toggle various analytical and running modes, and get a general "window" into the operating system while it's running. I was impressed. The average business user will never know or care that this feature is there, but if you are at all curious about the internal operation of Cache BIOS, it is fascinating.

Documentation

I've reserved a separate subsection for documentation because it is becoming more and more critical to the success of a product. It is also an area in which such dissimilar machines can be successfully compared.

Intersystems was the hands-down winner. The documentation came in two large, well-prepared three-ring binders, one for hardware, the other for software. In addition to the usual Digital Research manuals, there are workmanlike manuals for the hardware and the software unique to the Encore. The hardware manual explains the system on a board-by-board basis, and an overview chapter ties it all together. There is complete documentation for the word processing package and the Intersystems-supplied utilities.

From the standpoint of sheer volume, the Altos 586 should get some kind of prize. The documentation for the XENIX development system, coupled with a book on XENIX and all of the hardware-specific manuals, nearly outweighed the machine. This is not an exaggeration. There are literally thousands of pages, most of which, fortunately, you'll never have to read. The sheer volume makes for some basic disorganization. Some of the facts you need to know are in the book, some are in a hardware manual, and others are in an "Intro to XENIX" prepared by Altos. I can't really comment on the quality of the documentation, since it comes from all over and was written by many different people at many different times. In a way, it would be like knocking the literary quality of the Dead Sea Scrolls, but the bottom line is that it's pretty unreadable. My UNIX-savvy friend (he works in the UNIX development department at Bell Labs) could only say that "It reeks of Berkeley."

If brevity is the soul of wit, then CompuPro's manuals are hilarious. You get little more than a hardware manual for each of the boards and a brief writeup on MP/M 8/16. You do get an impressive number of Digital Research manuals, since the machine is shipped with three operating systems; CP/M-80, CP/M-86, and MP/M-86. Yep, you get 'em all, regardless of the inherent duplication. CompuPro is still selling to the hardware sophisticate. No neophyte could make heads or tails of all this. They should have included a copy of their excellent primer, *Bits, Bytes and Buzzwords.* Maybe I'm wrong, but I think the days when you can sell hardware on its technical specifications alone are over. Since dBASE and Supercalc-86 are provided, you get documentation for them, too. The dBASE manual is rebound into book form instead of the three-ring binder—a convenience.

Summing it up

I'll make no bones about it. I fell in love with the Altos 586. The amount that I had to learn about UNIX and the amount of time I had to do it in were horrendous. Shells are friendly to the beginner; UNIX is not. But I saw the inherent beauty and simplicity of this operating system, and saw it executed quickly and flawlessly on an amazingly inexpensive microcomputer. So what if I had to throw away everything I ever learned about CP/M? The 586 is a time machine—one that will transport us into the future.

Ithaca Intersystems' Encore is another outstanding example of how to be a successful multiuser micro. It doesn't write off 8-bit processors as incapable of handling multiple users, but draws on the inherent strengths of the hardware in a design that maximizes performance. The result is an environment familiar to CP/M users that provides true multiuser capabilities.

The 8/16 is CompuPro's first cut at an integrated system. While the design and performance of the hardware, taken on a board-by-board basis, is exemplary, it needs better integration with the operating system. As I mentioned above, it also needs to be interrupt driven if it is to live up to its potential. I tend to view the 8/16C as a transitional system, one that sets us up for the potentially phenomenal performance of the upcoming 16-bit processors like the 80286, while maintaining ties to CP/M. CompuPro is very much on top of the upcoming generation of superchips, but by no means do they have an exclusive.

Dennis Thovson's review of CompuPro's MPX board in the May 1983 *Microsystems* makes the point that interrupts are beyond the capabilities of the casual hacker. So are the innards of the multiuser operating systems. Manufacturers must not be afraid to incorporate all the high-tech components necesssary to make a system perform properly. Furthermore, they must pay for the software engineering to make it work. Lastly, they should charge whatever they have to for such systems. If the quality is there, they can't miss. Sales volumes will pay for the development costs.

Finally, be prepared to leave CP/M behind (O Heresy!) if it doesn't do the job for you anymore. Don't get me wrong; I'm writing this article on a CP/M system and will have one at home for years to come. At work, however, wherever that may be, it's a whole different ball game.

Prices: Altos 586-10, $7,990; CompuPro's 8/16C, $8,995; Encore (configuration reviewed here), $8,995.

FOR MORE ON THE 586, CIRCLE #315 ON READER SERVICE CARD
FOR MORE ON THE 8/16C, CIRCLE #316 ON READER SERVICE CARD
FOR MORE ON THE ENCORE, CIRCLE #317 ON READER SERVICE CARD

A Review of the Dual Systems 83/20 68000 UNIX System

by Leland Wilkinson

Several years ago, I made a bold (rash?) proposal to a research group at the University of Chicago. At a time when 16-bit processors were coming on the OEM market, I said that in a few years I could put all their statistical computing on a micro. The economics of this proposal were simple. The group spent about a thousand dollars a month on basic statistical packages on an Amdahl and an IBM 370. On that budget, the cost of a micro using one of the new processors could be recovered in less than two years.

What happened since that proposal taught me a lot about systems development and advertisers' claims. I first investigated the Z8000. We already had an S-100 machine that worked beautifully on word processing and data analysis, so we ordered a Z8000 card and 256K of memory to begin upgrading the system. Just before delivery, the factory called us to say that there was no software for the machine; the employee who wrote the advertisement that offered an assembler and other higher-level languages for the Z8000 system had been fired. Today, over two years later, that manufacturer offers a fine machine with software for multiprocessing, but the Z8000 still does not have the power for the kind of computing we do.

Next, I considered the 8086. I knew that Digital Research was working on CP/M-86 and that Microsoft would offer Fortran for the 8086 shortly after CP/M-86 was ready. After examining several S-100 systems, however, I learned that the Fortran would be restricted to 64K data and program segments. Although overlays were possible, this would be a severe limitation. Some of our programs had large arrays and over 4000 lines of Fortran.

Then I looked into the 68000. Some S-100 CPU cards were just becoming available, but there was little sign of software. This processor, however, seemed custom-made for a mainframe type of higher-level language compiler. There would be no practical limit on addressability and program size. I contacted Dual Systems, the company that had one of the first S-100 cards on the market. After finding that Dual had a working S-100 prototype with UNIX and Fortran, I placed an order for the basic machine with a half megabyte of memory and two floppy disks.

The machine arrived in March 1982. I thought at first that the company had used a box from a minicomputer. The cabinet was unlike any other S-100 I had seen. Two fans forced out filtered air through the front of the machine, so that several cabinets could be stacked without impairing the cooling. Sliding the insides out like a drawer, I could see a Godbout motherboard, a constant-voltage power supply, and various Dual (CPU, memory, clock) and Godbout (disk controller, I/O) cards. The disk drives were NEC slimlines. My initial impression of this gorgeous hardware was to sustain me through the trials of the next few months.

Blessed with a patient client (a psychiatrist), I worked throughout the summer trying to do something with this machine. It had UNIX, of course, but UNIX didn't want to do anything. The simple catch was, how do you copy a file from a floppy disk when the bare minimum of UNIX occupies almost all of two double-sided, double-density drives? The factory had warned me that the floppy-only system would be limited, but I thought we could get around it because most of our processing was to be "batch." We would read a program and a data file into memory and let it go.

Not so. Nothing of any size was going into memory without a third drive or some utility like Morrow's old "SINGLE" program, which emulated an extra disk drive for CP/M. To add to the insult, the NEC drives began to fail intermittently.

The company's response to this situation made me their enthusiastic supporter. For less than factory cost, they upgraded the system to a hard disk (the dual-floppy system is no longer available). They returned the system in September with a 20MB hard disk and a Sanyo floppy drive in one cabinet, and the memory management chip and 68000 in the other. Elated, I hooked up every peripheral I could find to its four ports. Six megabytes of the hard disk contained *all* of Berkeley UNIX (as ported by UniSoft), including dictionaries, editors, games, manuals, and witty aphorisms for every user who logged onto a different port. The whole staff began printing Romanesque banners on a Decwriter while I worked on software.

My troubles were not over. After a month or so, I had developed a statistical software base on the machine and was proudly demonstrating it one day when the screen went blank. The hard disk had been erased. After eliminating several hypotheses, I called the factory to find that we had the dreaded "Westinghouse disease." An identical system at Westinghouse had erased itself the week before. Dual had just figured out the problem (a subtle interaction between UNIX and the Godbout hard disk controller) and was sending to us by express a free updated UNIX to fix the problem. It took me 45 minutes to load the new UNIX from 5 floppies, plus my program floppy backups. No problems since.

The Machine

It is difficult to describe the experience of an S-100 user sitting at this machine. Other 8- and 16-bit systems can do multitasking and time-sharing, and some have versions of UNIX. Some systems that claim to have many of the features of this one can be bought for less than its $16,660 price. Few of these "state of the art" S-100 systems can be compared, however, to minicomputers and mainframes. This one can.

For example, this machine passes my mainframe version

Leland Wilkinson, Dept. of Psychology, University of Illinois, P.O. Box 4348, Chicago, Illinois 60680

of the Turing test: First, hide the box or put several users in a separate room with terminals. Now let them try to discover that this system is not a mainframe. After sending mail to each other, playing games, exploring the system libraries, and running large (> 256K) programs, show them the box. It's fun to do with veteran UNIX users.

This machine is rapid. The general response of the operating system is excellent. Several visitors who have tried other 68000 UNIX systems have reported that this system is faster. The delays while UNIX swaps (128K on the disk) and searches its libraries are scarcely noticeable. The longest I have found was three seconds, and the average response when a single user is logged on is comparable to a 4MHz CP/M system. The speed is most apparent when using an editor to search a 100K document, all of it in memory. Finding a string is essentially instantaneous.

The Software

The performance of the Dual Fortran 77 is another matter. On our large numerical programs, the Dual Fortran 77 is between 80 and 120 times as slow as Fortran H on the university Amdahl V7 mainframe. Using similar numerical programs, V. SethuRamen found Fortran 77 on a University of Chicago VAX 11/780 (with the DEC Floating Point Accelerator) to be 6 to 8 times as slow as Fortran H on the same Amdahl. Thus, we might expect the Dual Fortran 77 system to be about 13 to 15 times as slow as the VAX 11/780 Fortran 77. To explore this further, I tested the *Byte* magazine benchmark (January 1983, p. 286) on this compiler. In Fortran 77, the Eratosthenes sieve program took 29 seconds to execute on the Dual system. In the *Byte* test, a VAX 11/780 did the same program in 2.34 seconds with its Fortran 77.

This disappointing performance is due, I think, to the Fortran 77 on the machine. This compiler was subcontracted to Silicon Valley Software. It does not generate 68000 code directly. Instead, it formats code for the C compiler, which generates 68000 code. Coding the same algorithm in C, I reduced the CPU time to 9 seconds. A more efficient Fortran compiler is promised. In the meantime, frequently used routines might better be coded in C and linked to Fortran.

Having the full UNIX operating system available is both a strength and weakness of this system. For computer professionals and users who want to take the trouble to learn one of the most powerful and flexible operating systems currently available, this system is ideal. The hierarchical (tree) file structure, pipes, and shell make it preferable to other operating systems such as CP/M for this type of multiuser machine. New users, however, will require considerable supervision (I have trained three on both UNIX and CP/M). Confronted with commands like **cat** (list), **grep** (find), **pwd** (where-am-i), **nroff** (format), they can go to pieces. The editors and utilities provided in Berkeley UNIX are for professionals, not novices. Fortunately, UNIX has password protection, so that **rm*** and other commands, which execute before warning the user, cannot damage the system and other accounts.

Criticizing the human interface of a 10-year-old operating system in an era of Smalltalk, LISA, and iconic user interfaces is a bit gratuitous. UNIX was created before the term "user-friendly" was a commonplace, and the developers of UNIX evidently had less concern for the user than for system implementation. Interestingly, several recent human factors studies have shown that "meaningful" command names and even iconic interfaces do not necessarily improve performance for experienced users. But purchasers of a UNIX-based microcomputer should allow more training time than for CP/M.As more super-microcomputer manufacturers turn to UNIX, however, we can expect user-friendly interfaces to become available.

In addition to C, Fortran 77, and the enormous amount of applications software available to a full UNIX system like this one, Dual offers Pascal, Cobol, Basic, Forth, a 68000 macro assembler, a spreadsheet program, and a relational database system (INGRES). Also, CPM/BRIDGE is available for running CP/M application software under UNIX. Although we have not yet bought this program, we have used a Dual utility for handling CP/M disks. This has been invaluable for transfering files from our Intersystems S-100 machine. Downloading from other machines can be accomplished via a Dual UNIX handshaking utility easily piped to one of the serial ports.

I know of no other 68000 system even *advertising* this much software and, given our experience, I don't believe software announcements unless they are accompanied by a price and guarantee of immediate (as long as it takes to copy a floppy) delivery. Furthermore, I don't trust software companies that ask customers to pay for fixes. One supplier of a "scientific" Fortran for our CP/M system cheerfully announced that we could get an updated version which fixed serious bugs in the intrinsic functions for a "modest" handling fee. Dual provides updates for the cost of the media (floppies) and shipping.

The documentation with this machine is impressive. Three large looseleaf binders incorporate the original Bell Labs and Berkeley UNIX reference manuals and guides. The Fortran manual has extensive material on Fortran 77 file handling and I/O features. Every board is documented with detailed theory of operation and circuit diagrams.

Summary

This is hardly a hobbyist's machine. An enterprising hacker could construct the basic machine from Godbout and Dual cards (available separately) for less than its list price. As a single-user machine, however, it is overkill. Nor is it a small-business machine. Apple, IBM, Radio Shack, and others will probably run away with that 68000 market once they offer machines with some software. This Dual system is clearly aimed at sophisticated end-users in engineering, and industry, and academics who want high performance in computing-bound applications.

I have covered all the problems encountered initially (over a year ago) with this machine, mostly to indicate the type of company supporting it. There are tremendous discrepancies among advertising claims, computer show prototypes, and on-site performance of sophisticated microcoputer systems. This system is no longer a prototype; its software no longer a development program. This company doesn't disappear after delivering a shiny box.

Other 8- and 16-bit systems can do multitasking and time sharing, and some have versions of UNIX, but few of these "state-of-the-art" S-100 systems can be compared to minicomputers and mainframes. This one can.

Dual Systems 83/20 continued . . .

Now that this machine has been running flawlessly for several months, I have begun to appreciate what the IEEE S-100 standard makes possible. The 68000 is now available on several other buses, but to see room in this cabinet for 80MB of hard disk and 3MB of Dual 256K parity memory cards makes one think of a reservation system for a small airline, a relational database for a small company, or a solid modeling CAD/CAM system. Dual already has a 12MHz system running and is working on a faster hard disk controller. When enough potential Digital Equipment VAX 11/730 clients get to see this machine, DEC will be lowering its price.

For more information, contact **Dual Systems Control Corp.**, *2530 San Pablo Ave., Berkeley, CA 94702; (415) 549-3854.*

Note

Dr. Joseph Marcus is principal investigator for the project, and financial support for the machine was provided by Irving Harris.

CIRCLE 330 ON READER SERVICE CARD

Leland Wilkinson is Associate Professor of psychology at the University of Illinois, Chicago. He received his A.B. from Harvard and Ph.D. from Yale University, where he studied the psychometrics of human preferences. His current research area is the perception of statistical graphics—how the eye lies.

The Supersoft C Compiler

by David Fiedler

While just about two years ago there were only two or three CP/M-based C compilers on the market, now there are easily a dozen to choose from. Since most of these are priced in the $100-$250 range, it is necessary to choose carefully (it's *your* money, after all). Of course, compile time and execution speed are the easiest things to measure when considering a compiler, but there is also the general level of ease you feel: you're going to be spending a lot of time with it, so make sure you're comfortable with it, like an old shoe.

Having tested a fair number of C compilers, I tend to look for the features that make a compiler stand out from the rest. When I began this review, I didn't even go near the disk until I read through the manual, looking for interesting points. Here's what I found:

The Supersoft C compiler is actually a family of compilers that shares syntax, design, and even command line options to a degree. These compilers run on CP/M-80, CP/M-86, PC-DOS, UNIX, XENIX and ZMOS (for the Z8000 computer). The first pass of the compiler produces what Supersoft calls U-code (for universal) as an ASCII text file. The second pass performs peephole optimization and translation to the assembly language of the target machine, be it 8080, Z80, 8086, or Z8000. The full UNIX version 7 C language is said to be supported, except for **statics, longs, floats, doubles, bit fields, #if, #ifdef, #ifndef, #undef, #else, #endif,** and **#line.** Also unimplemented are parameterized **#define, #typedefs,** and initialization of data. Compared to other popular CP/M compilers, this list is quite long in the macro (#) area.

Also included on the disk with the two passes are runtime header and trailer files, a standard I/O library, memory allocation routines, and a good many library functions, all supplied in C source code. A careful look through this code found it reasonably commented and well-written. Apparently, a good deal of effort had been put into standardizing style, return values, and the like. While these observations don't prove anything about the performance of the compiler itself, they do give an insight into the priorities of Supersoft in attempting to produce a standardized, transportable product.

Various command line options for the compiler allow the user to include the C source code as comments in the generated assembly language file, turn off optimization (more about this later on), optimize for size rather than speed, and perform various assembly-level functions. An **#asm, #endasm** feature is supplied in Supersoft C to allow assembly code to extend directly in a C program; of course, such programs will be non-portable to other CPUs.

The manual is fairly comprehensive, and lists detailed information about each library function. These are listed in alphabetical order, which can be more useful than grouping them by function, as some other manuals do.

Performance testing

So now it's time to run the compiler, using the sample programs supplied by Supersoft and their recommended command syntax. Generally, I try to run all compilers in the same environment; a comparison between BDS C on my hard disk and Supersoft on its single-density distribution disk would hardly be fair. Therefore, the first thing to do is to copy the distribution disk onto one of my double-sided, double-density disks, and run it from there. No problem. The first pass is invoked for the first supplied program, SAMPL.C:

```
cc sampl.c
```

and it bombs. One of the **#included** function files is giving hundreds of syntax errors, the kind you have to reboot to stop. Second try: same result. The problem is "obviously" a bad copy.

This time I use PIP with the Verify option. No change. In desperation, I run the CP/M User Group's **CRC** program on both disks, only to find that every program matches the original disk.

Now I begin thinking of all the possible problems that could be causing this symptom.To eliminate memory problems and settle things down, I attempted running the first pass directly from the distribution disk (not recommended as general practice). It works fine. I am now faced with the possibility that this software only runs from the original disk!

When linking programs, the order in which they are linked might be the difference between success and failure. Maybe something here is sensitive to the order of the files on the disk? I make a careful copy of the original disk, preserving the order exactly, but that doesn't do it either. Just the kind of problem I need to help me write the review by the deadline . . . Finally, I hit on the idea of making another single-density disk containing the programs. It works!

My new hypothesis is that this compiler only runs from single-density disks. Obviously, this makes no sense. My system has been running for years with all sorts of densities, and I have never had a problem like this. But at least now it's running, so I can see how it works when it *does* run.

SAMPl.C, the first sample program, was 1575 bytes of source code. It ran through the first pass of the Supersoft compiler in just over 2 minutes, producing a U-code file (SAMPl.COD) of some 18K. This seemed a bit slow to me, but I'm not used to using single-density disks, and I figured that was the problem. Then I ran pass 2. It took 3 minutes and 40 seconds! Clearly something was wrong here, although everything was acting normal. But now I had an assembly language file. A 42K assembly language file, to be exact, and I was wondering, just how extensive was this function library of theirs, anyway? SAMPl.ASM took just under 2 more minutes for the standard CP/M assembler to turn into SAMPl.HEX, and 23 more seconds to load. At this point I was a bit apprehensive in running the .COM file, mainly because one of the functions the program called was found to be undefined by the assembler (*in a sample program?*). But it did run, although I noticed a line feed seemed to be missing at one point, garbling the output a bit.

Checking further

I sat back a bit to study the situation. Here was a distribution disk I had read without errors, with software that ran only on another single-density disk, and that contained er-

David Fiedler, InfoPro Systems, P.O. Box 33, East Hanover, NJ 07936.

Move up to Microsystems

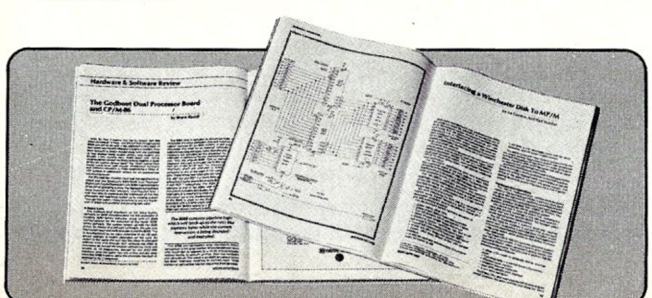
THE JOURNAL FOR SOPHISTICATED MICROCOMPUTER USERS.

Microsystems is not, in any sense, for beginners. Unlike "personal computer" magazines, *Microsystems* is written *exclusively* for expert users and designers in the software and support hardware field.

If you need a reliable, high-level, up-to-date journal devoted to CP/M,® MP/M,® MS-DOS,™ UNIX,® XENIX,™ OASIS and other operating systems, then *Microsystems* is for you. We feature tutorials on the modification and special use of systems, offer short-cut routines, show you debugging procedures, techniques for interfacing, and much more. Our one purpose is to keep the advanced microcomputer professional at the leading edge of DOS development and use.

In a few recent issues of Microsystems you would have seen:

- Reviews of four C compilers: BDS C, Small C, tiny c TWO™ and Whitesmiths C.
- A programming style comparison: Digital Research PL/I-80™ vs. Microsoft BASIC.
- Twenty-seven 16-Bit DOS compared.
- Virtual segment procedures under UCSD Pascal.
- An in-depth look at MP/M-80 II.™
- A simple 6-byte hexadecimal ASCII conversion routine.
- Using CP/M's undocumented "Autoload" feature.
- An explanation of double density disk controllers.



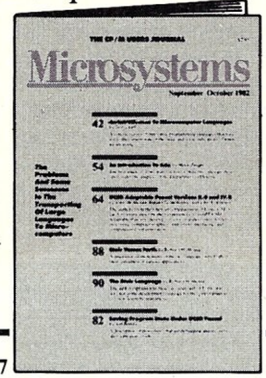
- A comparison of five popular S-100 disk controller cards.
- OS-1—a UNIX-like DOS with a CP/M adapter.

Every issue of *Microsystems* keeps you at the forefront of the industry in customizing software and adapting hardware to meet new requirements and standards. You'll also find industry news, book and new product reviews, plus our regular *Software Directory* listing newly available programs.

To get the high-level systems information you want, join the club of serious microcomputer professionals who subscribe to *Microsystems.* Use the coupon below to order, and save up to 31%!

PROFESSIONALS PLEASE NOTE:

Your subscription to Microsystems may be tax deductible. Check with your accountant.



Trademarks:
XENIX: Microsoft; tiny c TWO: tiny c assoc.; PL/I-80, MS-DOS, MP/M-80 II: Digital Research.
Registered Trademarks:
CP/M, MP/M: Digital Research; UNIX: Bell Laboratories.

Microsystems CN 1987
Morristown, New Jersey 07960

YES, enter my subscription to *Microsystems* for:

☐ 12 issues at $21.97—I save 19%.

☐ 24 issues at $40.97—I save 24%.

☐ 36 issues at $55.97—I save 31%.

Savings based on full 12-issue subscription price of $26.97

Mr.
Ms. 4D24

(please print full name)

Company

Address

City

State Zip

CHECK ONE: ☐ Payment enclosed. ☐ Bill me later.

Offer valid in U.S. and possessions only. Please allow 30 to 60 days for delivery of first issue. Satisfaction guaranteed or a prompt refund for unmailed issues.

NEW SUBSCRIBERS ONLY

rors in the sample program. Either this is mighty poor software, or I got a disk that was somehow very subtly defective. Then I realized I had been running all of this under MicroShell (see *Microsystems,* January 1983, which used 8K of TPA space. Maybe this was all just symptomatic of being low on memory!

But that wasn't it either, although Pass 1 got a bit further along before it started spitting out errors. This still wouldn't explain why it works from one disk and not the other, and anyone familiar with CP/M can tell you that none of this should mean a thing as long as the disks are working normally (which they were). But there's more to this story.

It didn't make sense that Pass 2 should take as long as it did, so I looked into the command line options, and found I could turn optimization off. When I did this, I virtually *halved* runtime of Pass 2. However, the output .ASM file was now 50K!! Without boring you with more details, Table 1 summarizes the findings.

While this wasn't intended to be another C compiler comparison article (see *Microsystems* Sep/Oct 1981), I couldn't resist seeing how BDS C stacked up against Supersoft. I copied the BDS compiler to the same disk I had been running Supersoft from. The difference is quite striking, especially considering that all three active portions of BDS C (pass 1, pass 2, and the linker) together take up less disk space than *either* pass of the Supersoft compiler. Of course, the Supersoft compiler is written in *itself* rather than assembly language,the way BDS C is.

It might be interesting to note that the only changes I had to make in order to compile the sample program under BDS C was to change the **#include** lines to pull in the BDS standard library and add one of Supersoft's non-standard functions (**putdec**) to the end of the source program. Remember the bug in the sample program? It *didn't* show up when I compiled and ran it under BDS C, suggesting that the bug is not in the program itself but in one of Supersoft's function libraries. Considering only the speed of compilation and the fact that BDS C actually supports a *larger subset* of standard C than does Supersoft (as well as costing about half as much), BDS begins looking better all the time.

Another problem is that all programs you write using this compiler will **sign on** with a message from Supersoft, giving compiler and runtime package version numbers. Not only is this a bit sticky for people writing things for sale, but these three lines of output from every program do tend to get in the way of writing "filter" programs.

As this review was being completed, I attempted to run Supersoft's compiler from my hard disk (which is generally reliable), figuring maybe I was getting invisible disk errors while running from the floppy (and also wanting to be as fair as possible). I got much better results this time, which led me to try rerunning the single-density tests. These also turned out much better than before (see Table 1), so I figured my system had been in error all along. Then I copied one of the other sample programs—the "sieve" program used in the *Byte* magazine benchmarks—hoping to get some more statistics. Guess what? It bombed out again!

Some more experimentation has brought me to the conclusion that somewhere in the Supersoft compiler is a variable that is not initialized properly, and when the compiler is rerun, errors can occur. I ran a memory test and even tried clearing memory before running it again, but no luck. I can't accept the idea that my system has spurious errors that only occur while running one piece of code.

The Supersoft C Compiler is available for $200 from dealers and from:

Supersoft Inc.
P.O. Box 1628
Champaign, IL 61820
(217) 359-2112.

Table 1. Compiling times

Compiler	Pass 1	Pass 2	Assemble	Link/Load	.COM file size	Intermediate files left on disk
Supersoft 1.1 (first run, floppy)	2:05	3:40	1:59	0:23	7K	78K
Supersoft 1.1 (unoptimized, floppy)	2.05	1:54	2:24	0:26	8K	89K
BDS C 1.44 (unoptimized, floppy)	(0:22 total)		n/a	0:13	4K	3K
Supersoft 1.1 (hard disk)	1:28	0:27	0:23	0:07	2K	34K
Supersoft 1.1 (second try, floppy)	2:05	0:38	0:48	0:08	2K	34K

1. Times obtained using 8" single-density, single-sided floppies (except as noted) on a 3MHz Z80-based machine with a 57K CP/M system (52K TPA) and no wait states.
2. "Intermediate files left on disk" refers to the remnants of the compilation process that are neither C source files nor executable. This figure could be significant to someone doing development work on systems with a limited disk capacity.
3. The Supersoft C compiler automatically performs optimization; BDS C automatically doesn't.

HARD DISKS

For Systems Integrators

Whether you're an OEM, system integrator, or end user, when the time comes to add a hard disk unit to your computer you want a building block that offers high performance, quality, and cost effectiveness. The Pragmatic Designs PD-10M, PD-20M, and PD-40M all provide these features and more.

All Pragmatic Designs hard disk sub-systems are designed for use in systems equipped with the CompuPro® Disk II hard disk controller. They can also be used with other OEM controllers which support the popular SA-4000 hard disk interface. Standard features include:

- 10, 20, and 40 Megabyte formatted storage
- 11.7, 23.4, or 47.5 Megabyte unformatted storage
- Fully compatible with CompuPro Disk II controller
- Heavy duty power supply with 110/220V capability
- 19" rack mount configuration available
- 1 Year limited warranty
- Full hard disk system including controller, cables, and software available

Hard disks . . . easy solution. If you're ready to add a full capability industrial grade hard disk sub-system to your computer system then call Jerry Hall at Pragmatic today.

pragmatic designs INC.

Pragmatic Designs, Inc., 950 Benicia Ave., Sunnyvale, CA 94086 408/736-8670 TLX: 171627

CIRCLE 186 ON READER SERVICE CARD

Editor's Note

It is *Microsystems'* policy to send a copy of any review to the vendor of the product for comment. This is done to ensure technical accuracy. When we receive the vendor's comments, we forward them to the author, who, if he agrees with them, will revise his article accordingly. If both vendor and reviewer do not agree, we publish their respective comments, as we are doing in this case.

SuperSoft's Response

We were sorry to hear that Mr. Fiedler had difficulty in using SuperSoft's C compiler. Mr. Fiedler's main problems pertained to disk copying rather than to the quality of the compiler itself. Our C compiler is being used in thousands of installations and in dozens of different disk formats. In addition, SuperSoft uses this compiler for program development on a wide range of systems and disk formats. In all our experience, and in the formal performance reports we have received from our customers, we have yet to hear of another instance of the specific problem Mr. Fiedler had in copying and then using the compiler.

Mr. Fiedler reported erratic performance of the compiler and wisely checked the accuracy of his copy, attempting to establish that his system was faultless. Without further information, we suspect that the original distribution disk sent to Mr. Fiedler by SuperSoft or one of our distributors may have contained a copy flaw, in which case our normal nocharge replacement procedure should have been followed. SuperSoft does everything possible to make sure that customers who have a problem with their original distribution disk receive a new copy at no charge as soon as possible.

There are also several inaccuracies in Mr. Fiedler's review that we would like to address:

1. The review does not correspond to our current version of the product, version 1.2. This explains many of the following inaccuracies. (Also, a new release containing floating point and a direct to REL option is due shortly.)
2. Pass 2 of the compiler performs global as well as peephole optimization.
3. The static and long attributes are partially supported. The library supplied with the compiler has functions for long and double (long float) arithmetic. It also has the most commonly used trigonometric functions.
4. The **#if, #ifdef, #ifndef, #undef, #else, #endif,** and **#line** preprocessor directives are supported. There is no preprocessor directive **#typedef**. There is a C keyword **typedef.**
5. The figures in Table 1 of Mr. Fiedler's article do not correspond to those which we get for the same compilation.
6. SuperSoft C supports a superset, not a subset, of BDS C.
   a. SuperSoft C compiler is a very solid subset of the language, allowing declaration and use of any structure, union, or array (or any combination of those types). This is not true of BDS C.
   b. SuperSoft C is as compatible with UNIX as CP/M allows. This includes a complete set of UNIX-compatible I/O functions, and no sensitivity to CP/M's 128-byte system record size. (Try this with any other CP/M C compiler: open a file, seek 999 bytes, write one byte and close the file. SuperSoft C will change only one byte in the file. This is an absolute requirement for UNIX compatibility.)
   c. BDS C violates certain arithmetic conversion rules regarding the char type. These same rules are scrupulously followed by SuperSoft C.
7. The current list price of SuperSoft's C compiler is $275. Overall, the review seems to imply that BDS C or Q/C is superior to SuperSoft's CP/M C compiler. While BDS C and Q/C are good products, SuperSoft C is better suited to the serious software developer. Many large software firms use SuperSoft C for production programs, and SuperSoft's internally generated products all use the compiler. There are a number of good reasons for this:

1. SuperSoft C, unlike BDS C, does not limit the source file size of programs to be compiled.
2. SuperSoft C programs can be developed under any of their host systems and then ported without source changes to MS-DOS, UNIX, OASIS, and CP/M-86, to name some of the target operating systems. This is certainly not true of BDS C.

We would like to be of timely assistance to you and Mr. Fiedler in this review of SuperSoft C. Our concern is that your review reflect the current version of our product. I will be sending a current version to Mr. Fiedler today. If I can be of any further help, please let me know.

Sincerely,
Richard Balocca
Vice-President
SuperSoft

David Fiedler's Response

After I submitted the above review, I was sent a new copy of the Supersoft C compiler, along with a copy of Supersoft's response to the review. I would like to address the points Supersoft made, and add some comments along with new timings I got when running their latest version.

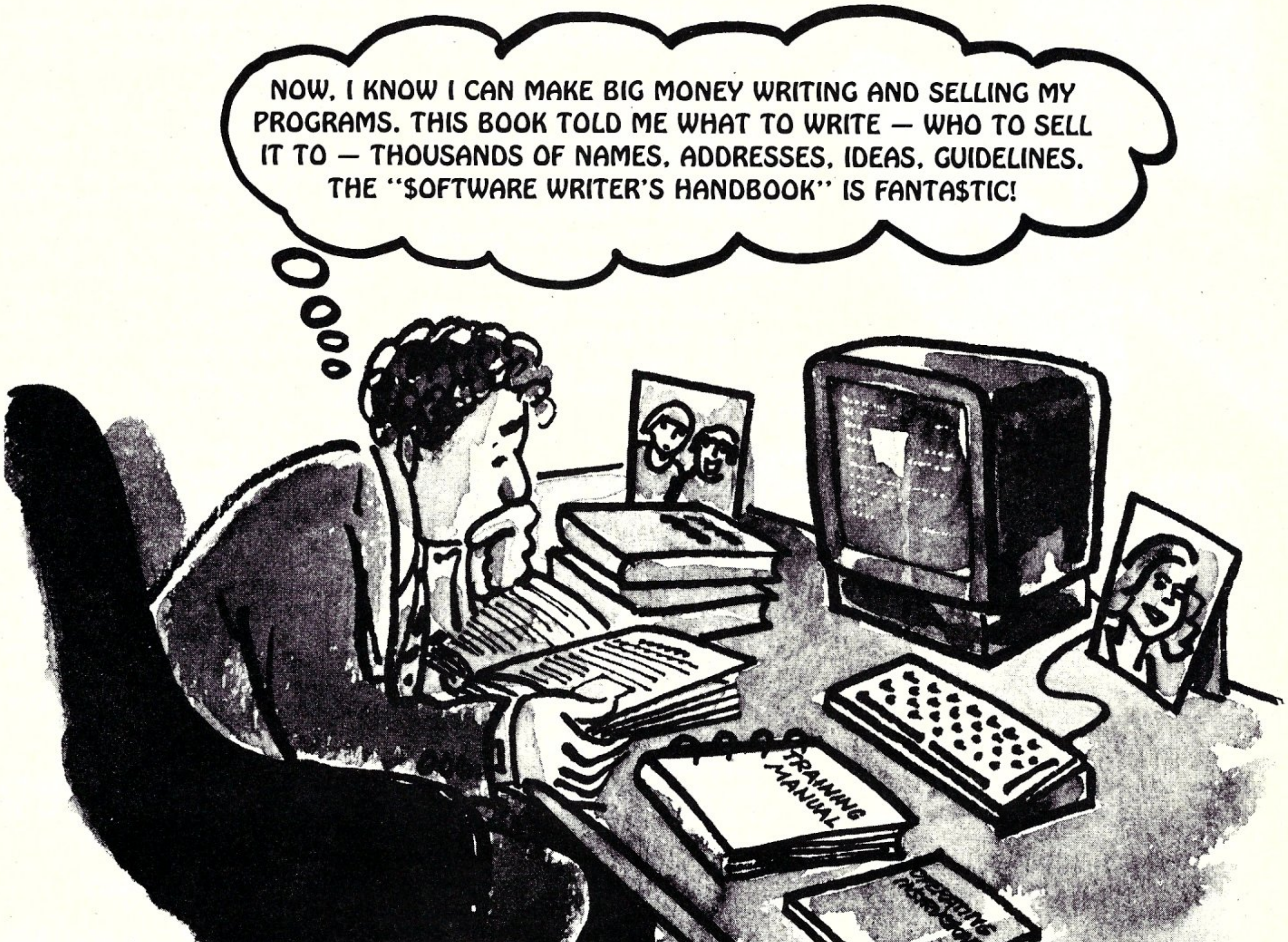
First, when I receive a product for review, *especially directly from the manufacturer,* I can only assume that it is the latest version. I do not appreciate a company's releasing a new version in the time before publication and then complaining that my article is inaccurate, since I was working with what they supplied in the first place. Supersoft's points 1, 2, 3, 4, and 7 all relate to the disparity between versions, and therefore I consider them misleading as criticism, although accurate as they stand.

I would not mention this next point at all, except that I feel the record must be set entirely straight, inasmuch as my own reputation is being called into question by

A good deal of effort has been put into standardizing style, return values, and the like in an attempt to produce a transportable product.

programmer s

READ THIS...



* WHO TO SELL YOUR PROGRAMS TO
* THOUSANDS OF COMPANY NAMES AND ADDRESSES, WITH DETAILED LISTINGS SHOWING:
  (1) WHAT PROGRAMS PUBLISHERS ARE LOOKING FOR
  (2) HOW THEY WANT YOU TO SUBMIT YOUR PROGRAM
  (3) HOW MUCH THEY PAY — AND WHEN!
* 100 CATEGORIES — FROM "ACCOUNTS RECEIVABLE" TO "GAMES" TO "VIDEO CONTROL" PROGRAMS
* HOW TO WRITE CLEAR DOCUMENTATION
* DEBUGGING TECHNIQUES

Reserve Your Copy Today!

Enclose check or money order for $19.95 (No C.O.D.'s) to:

IPF Publications
146 Country Club Lane
Pomona, NY 10970
(914) 354-5585

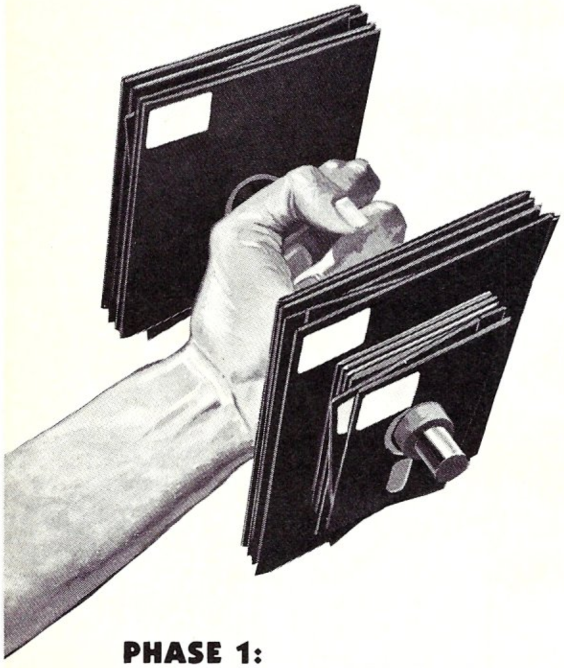
Name ....

Address ....

City............. State............. Zip.............

CIRCLE 37 ON READER SERVICE CARD

WHY YOU NEED MICROSHELL™: NEW POWER FOR CP/M® SYSTEMS



PHASE 1: "LITTLE" FEATURES THAT ADD UP TO BIG TIME-SAVERS

You appreciate MicroShell from the moment you run it. Suddenly your CP/M computer seems much smarter, more powerful, more tolerant of your errors. That's because MicroShell adds features to CP/M normally found only on more powerful operating systems.

You spend a lot of time with your computer. MicroShell makes your time more productive. Take the warm-start after you run a program or when you change disks. MicroShell eliminates it — you can run another program immediately. Change a disk — even in drive A — and a quick command logs it in. No more time lost booting up — no more juggling "system" disks with CP/M on them!

Or how about finding files? Under CP/M you have to know what disk the file is on and then change disks or preface the file name with the disk drive. And if you have a hard disk and are using CP/M user numbers, you also need the user number the file is in. MicroShell automatically searches for files on all disks, or selected disks — your option. And it works even when a program calls for a file. So you can run WordStar, for instance, from any disk or user number and MicroShell finds the necessary WordStar files.

Running several programs in a row? MicroShell lets you put a string of CP/M commands on one line. You end the distraction of entering commands, waiting for execution, then entering the next command.

Make a mistake? With CP/M you have to start over. Add MicroShell and you can edit the last command line (with simple WordStar commands). MicroShell saves the last command string, so you can run through a series several times — or change commands slightly and run again — without retyping.

MicroShell doesn't affect how your CP/M runs. It helps CP/M, but it doesn't get in the way. And your system probably never misses the 8K bytes MicroShell uses because most all popular programs run with 48K. So in a 64K CP/M system, there's plenty of room. For the rare, big program, it's easy to exit MicroShell, run the program and re-enter MicroShell.

You appreciate MicroShell from the start. It has powerful features so easy-to-use that the MicroShell manual tells you "How to Run MicroShell Without Reading the Manual".

PHASE II: NEVER RE-ENTER DATA AGAIN

But please — read all 75 pages of the manual and get to know MicroShell in depth. You appreciate MicroShell more and more when you do the big projects you bought your micro for.

Take the time you turn a spreadsheet analysis into a table for a report. Without MicroShell you have to re-enter data in a file for your word processing software. Why the duplication of effort? MicroShell lets you *redirect* the output of any program to a file — no duplicated keystrokes.

MicroShell captures what appears on your CRT and sends it anywhere. Send data to your printer for hard-copy or save it on disk. Or capture that report your accounting program sends to the printer. MicroShell tames your data-handling jobs — it even "slows down" to feed programs that process input slower than MicroShell sends it.

When you review your work, MicroShell has a "TYP" command that displays 23 lines (an average CRT screen) then stops until you hit a key. If you've ever scanned a long document whipping by at 10 or 20 lines per second, you know you need this feature. Or have you tried to hit control-S in time to stop the display and missed?

For more information about MicroShell, see the following reviews:
Christopher Kern, BYTE, December, 1982
Alan R. Miller, Interface Age, July, 1982
David Fiedler, Microsystems, January, 1983

PHASE III: THE SYSTEM EVOLVES WITH YOU

If you're in computers for the long haul, MicroShell will grow on you. MicroShell helps you teach new users. And it lets you develop new capabilities in a clear and logical way.

MicroShell lets you set up "menus" to make your system "friendly" for non-technical users. And fast — no slow CP/M "submit" files used. As you develop your own software, MicroShell's "pipe" feature lets you grow into ever more sophisticated applications. It links simple programs together by directing the output of one program to the input of another.

Just as MicroShell lets you combine commands on one line, it lets you save any command series in a file. MicroShell reads and executes commands directly from the file — no slow, intermediate steps. For example, you can compile, link and run a program with one command, "compile filename". MicroShell executes all the commands from a file you've built.

And you can call other command files from within a command file to enlarge on what you've built. You can set conditional commands with an "IF" statement. Or you can set command files to call for input from a user and act on it. Before you know it, you have a system much more powerful and friendly than bare CP/M.

SO DON'T WASTE ANY MORE TIME WITH YOUR CP/M SYSTEM — ADD MICROSHELL TO IT TODAY.

ORDER LINE 800-368-3359

Available for all popular 8080, 8085 or Z80, CP/M 2.2 systems on 8" or 5¼" diskette.

Price: $150.00 ($25.00 manual only). Foreign orders add $20.00 for Air Mail shipment.

NEW GENERATION SYSTEMS, inc.

2153 Golf Course Drive
Reston, VA 22091
(703) 476-9143

CP/M®, MP/M™ — Trademarks of Digital Research.
WordStar® — Trademark of MicroPro.

CIRCLE 19 ON READER SERVICE CARD

Supersoft C Compiler continued . . .

Supersoft. Mr. Balocca said he would send me a "current version." The copy I received was marked Version 1.1.37, *not* Version 1.2, as he claimed. Further, I also received a copy of Supersoft's packing list, on which was handwritten "May review: *Send new version*" (emphasis as in original). To me, this implies that the *new*, and not the *current* version was sent—especially because it was for a possible review. I wonder what version customers were receiving at the time?

Mr. Balocca mentions that "functions for long and double (long float) arithmetic" are supplied, and also "the most commonly used trigonometric functions." Nothing that could be remotely considered trigonometric could be found, either on the disk or in the manual. And the only functions that had anything to do with long arithmetic that I could locate were standard 16-bit multiply and divide routines—certainly not "double" as claimed, and not called out as such, as they seemed to be internally used.

Point 5 is a classic example of "begging the question." I can see where my original figures do not correspond to theirs, since running their "latest" version yields times that are *much longer* than on the previous version. The results obtained when compiling their SAMP1.C program, following the *precise* instructions in their manual, are shown in Table 1.

Table 1. Compiling times

Pass 1	Pass 2	Assemble	Link/ Load	.COM file size	Intermediate files left on disk
2:37	3:45	3:49	0:24	8K	220K

To quash any possible objections to the fact that I used the standard CP/M assembler and not a relocating assembler, I will report that only 10K (on the .ASM file) and 20 seconds' compilation time was saved when compiling for use with RMAC or M80. And I was using double-density disks this time! In case it's not obvious, the size of the intermediate files left around almost totally precludes effective use of this compiler with anything less than a hard disk or large-capacity, double-density disk system (in fact, this "sample" compilation is impossible on a single CP/M standard 8″ disk). It can be done if neat juggling of output files and judicious erasing are performed, combined with a fair knowledge of compiler/assembler options.

I'm not surprised to hear that Mr. Balocca feels that Supersoft C is "better suited for the serious software developer" than BDS or Q/C. Its function libraries *are* certainly more standard in terms of the UNIX environment (although different in several respects). I am interested in hearing, however, about the "large firms" that use this compiler for production programs, and I would like to know if these firms are using the native compiler on CP/M (which readers of *Microsystems* would use) or a *cross*-compiler, possibly on a VAX, I know of many CP/M software products that have been developed in C, yet no developer has ever spoken to me of using anything but the BDS, Aztec, C/80, or Whitesmiths compilers. The limitation of source files is true but again begs the question, since large C programs, are, as a rule, developed and compiled in separate modules.

As to the allegation that Supersoft C programs can be "ported without source changes" to operating systems such as UNIX, I will report that I spent a good deal of time investigating this. Neither the SAMP1.C of SAMP3.C program (the "Sieve" benchmark) could be compiled as is when ported along with Supersoft's required function libraries **func.c, crunt2.c, stdio.h,** and **customiz.h** to a 68000-based system running UNIX Version 7. The problem (aside from about two dozen warning messages in each case regarding integer/pointer conflicts) seemed to be with differences between the UNIX and Supersoft compiler in interpreting the rescanning of *included* source files. After editing the files to remove references to Supersoft's functions, the programs did indeed compile and run (although the CP/M-specific functions did not work, as expected).

Finally, a note about the Sieve benchmark. The program Supersoft supplied had been modified to use their "putdec" routine to save memory, rather than using **printf** (this did not interfere with the original algorithm). This program executed on my system in 40 seconds, and took a full 16K for the .COM file.

Options allow the user to include the C source code as comments in the generated assembly language file and select optimization goals.

The Ratfor Preprocessing Language

by A. G. W. Cameron

I have frequently seen Fortran castigated in the pages of microcomputer magazines, sometimes in ignorance, but often with some degree of justification. The lack of good program control structures renders programming relatively difficult; in that respect, among computer languages, probably only Basic is worse. The fact that Basic is beloved of microcomputerists, while Fortran is not, is mainly due to the easier string-handling capabilities of Basic and the fact that it is an interactive language. Yet Fortran is the indispensible number-crunching language of scientific and technical people. The fact that there is only one generally accepted version of Fortran for CP/M-80 and no *acceptable* Fortran for CP/M-86 or MSDOS, in contrast to the many competing packages for other languages, reflects the fact that micros are not yet very useful for number-crunching. That will change as soon as floating-point coprocessors become readily available for the 16- and 32-bit chips. I hope (and expect) that market opportunities will bring forth Fortran packages that will use the internal stacks of those coprocessors more efficiently, thus achieving computation speeds comparable to those on mainframes.

The National Science Foundation recently organized a panel on computation in physics. One of the participants in that panel stated that he did not know what computer language he would be using in the next 10 years, or what its programming structure would be. He knew only one thing: it would be called Fortran!

That tongue-in-cheek remark contains an element of truth about the state of scientific computation. Graduate students tend to be deeply imprinted with Fortran during their training years and are very resistant to the idea of changing to a different language. There is a massive amount of accumulated software in Fortran in all technical fields, and technical people must be prepared to use it. Until recently, Fortran and Cobol were about the only languages you could be sure of finding on most mainframes. Yet Fortran itself is not immune to change. The language was standardized in 1966, but soon developed many extensions. One of the reasons IBM mainframes tend to be very unpopular among scientists is that IBM has refused to let any of these extensions, which frequently add convenience to programming, creep into its Fortran, so that transporting Fortran programs onto IBM mainframes is frequently a pain. Fortran was restandardized in 1978, so that Fortran-77, as the latest version tends to be known, has incorporated some improvements, particularly the IF-THEN-ELSE construct. But most of the criticisms of Fortran-66 remain true of Fortran-77.

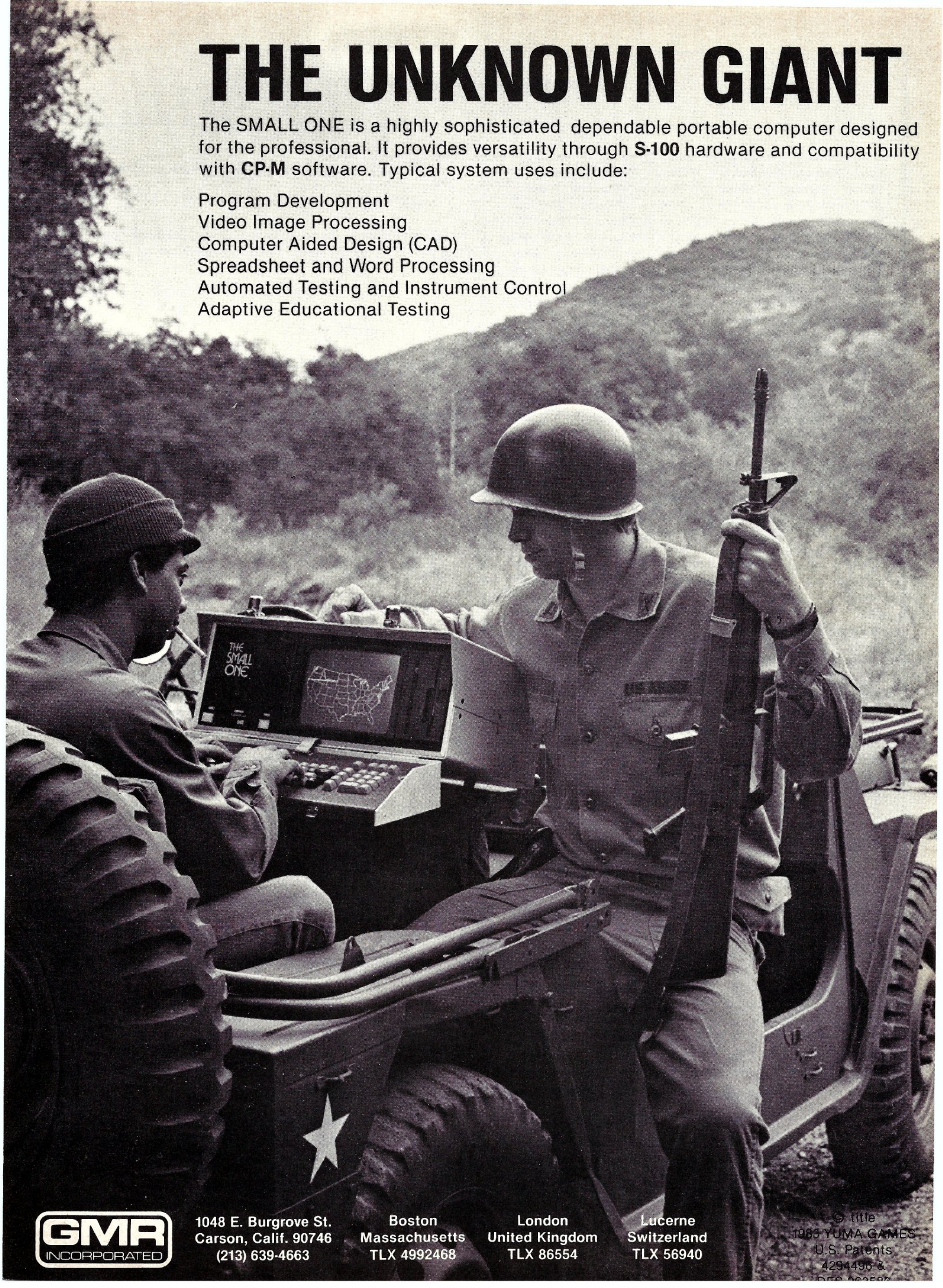
It is therefore interesting that a Fortran preprocessor named Ratfor (for RATional FORtran), which has superior programming flow procedures, has won increased acceptance in the scientific community. It is conceivable that Ratfor, or some language like it, might well become the Fortran of 10 years hence. Since all Ratfor-to-Fortran translations use only valid Fortran-66 statements, the use of Ratfor tends to make Fortran programs more portable between different kinds of computers. That is one of the principle reasons for my current interest in Ratfor, although my original interest was in its superior control structures.

For several years, assisted by a changing group of postdoctoral fellows, I have tried to maximize the value of each number-crunching dollar begged from the government (through the mechanism known as a grant), by investing it in hardware rather than in time on a central mainframe computer. We wound up having linked Data General Nova 3/D and Nova 4/X minicomputers, and a variety of peripherals from different vendors. Each minicomputer offers two active computing grounds, and if each ground is efficiently used to run number-crunching programs, then we can get about the same throughout as a Digital Equipment Corporation VAX 11/780, although our capital investment is much less. Rarely can we be that efficient, since such activities as word processing and program development do not use the CPU as efficiently as number-crunching does. Moreover, it is now evident that our minis are aging, and that the future of cost-effective number-crunching will lie with super-microcomputers. We have gained some efficiency for number-crunching on the minis by off-loading word processing and some of the program development onto ordinary micros. It is for program development that Ratfor is important; it will become more so in the future as we follow upgrade paths on the micros and shift our number-crunching burden to them.

Ratfor was one of several Fortran preprocessors developed to address the problems of poor flow control structures in Fortran. Undoubtedly, the most important event in its development was its use in the book, *Software Tools*, by B.W. Kernighan and P.J. Plauger (Addison-Wesley, 1976). The Software Tools implementations are discussed in an accompanying article in this magazine. My original acquaintance with Ratfor came when we obtained from the publisher (for $25), a magnetic tape containing machine-readable versions of the programs in *Software Tools*, including Ratfor written in Fortran. One of my postdocs, Gordon Weast, converted all the EBCDIC to ASCII, rewrote the primitive routines which interfaced to our operating system on the Novas, and had Ratfor flying in about three days. Fine-tuning and recompilation of Ratfor from Ratfor sources came later. At that point I came to appreciate Ratfor and tended to write new programs in this language, although I continued to change old ones in the original Fortran.

I have seen references to both public domain and commercial versions of Ratfor over the last year or two. Although I do not have direct knowledge of most of these, I judge them to be essentially the *Software Tools* version of the language interfaced to CP/M (much as we did in our minis for a different operating system). Considering the ease of our own effort and the existence of public domain versions, I will leave the reader to judge to what extent these older commercial versions of Ratfor are overpriced.

A. G. W. Cameron, Harvard College Observatory, 60 Garden St., Cambridge, MA 02138.


THE UNKNOWN GIANT
The SMALL ONE is a highly sophisticated dependable portable computer designed for the professional. It provides versatility through S-100 hardware and compatibility with CP-M software. Typical system uses include:
Program Development
Video Image Processing
Computer Aided Design (CAD)
Spreadsheet and Word Processing
Automated Testing and Instrument Control
Adaptive Educational Testing
THE SMALL ONE
GMR INCORPORATED
1048 E. Burgrove St.
Carson, Calif. 90746
(213) 639-4663
Boston
Massachusetts
TLX 4992468
London
United Kingdom
TLX 86554
Lucerne
Switzerland
TLX 56940
© title
1983 YUMA GAMES
U.S. Patents
4294496 &

Meanwhile, a major effort has begun at Berkeley to improve and standardize the Software Tools including the Ratfor language in which they were written, and to transport them to a variety of types of computers. These improvements to Ratfor included additional control structures, inclusion of macro definitions, and improved efficiency of the Ratfor-to-Fortran translation process. Unicorn Systems (which has since incorporated as Carousel MicroTools, Inc.) took on the task of transporting the Tools to CP/M-80, and it is their version of Ratfor that is under review here. However, they have adhered as much as possible to the Software Tools Users Group standard, and so it is really that standard which is being reviewed here.

I have also encountered the standard in another form, since CompuCode undertook to implement the Tools on Data General machines, but did so for Eclipses only. Jon Hanshew, the president of CompuCode, did his best to produce a version for Novas without having one inhouse. So I found myself having to get a few bugs out of his version of Ratfor, then making some changes of my own from the Software Tools version for programming convenience on the Novas. This involved examining in depth both the Ratfor and the Fortran versions of the source code, none of which is well documented internally. An exercise like this really demonstrates how immensely more readable and intelligible Ratfor is than Fortran. The new Software Tools version of Ratfor, considerably longer than the original that we implemented on the Novas, requires that some routines be organized as overlays (the CP/M version is more efficient in the use of space and does not require this.)

The Ratfor translation process operates by scanning the text and looking for keywords that the preprocessor can recognize as meaningful to Ratfor. Statements not containing keywords are assumed to be Fortran and are passed through the translator with only such trivial changes as converting lower case to upper case and postioning the statement properly on the line (Ratfor has a free-form input and does not care if a statement begins in the first column, but of course it must output lines compatible with a Fortran compiler).

I will give a brief description of some of the features of Ratfor not contained in the Fortran language. To quote from *Software Tools:* "The control flow statements of Ratfor are shamelessly stolen from the language C, developed for the UNIX operating system by D.M. Richie." This is not surprising, since most of the authors involved are at Bell Labs, and the Software Tools themselves are modelled on UNIX tools.

One very important piece of notation in Ratfor is the use of braces, " < "and" > ", to surround groups of Fortran or Ratfor statements so they can be treated together as equivalent to a single statement. This is a key step in avoiding the use of GOTOs in the Ratfor language itself. The Ratfor-to-Fortran translation typically introduces GOTO statements and places a labelled CONTINUE following such a pair of braces.

As an example, consider the following Ratfor statement:

```
IF (condition) {statement(s)}
```

The above statements are to be executed if the condition is TRUE. The corresponding Fortran statements became:

```
        IF (.NOT. condition) GOTO label
        statement(s)
label   CONTINUE
```

This Fortran flow structure is sufficiently appealing and I have often found myself using it directly when revising existing Fortran code. A logical extension to the Ratfor construct involves using ELSE ("THEN" is unnecessary):

```
IF (condition)
   {statement(s)}
ELSE
   {statement(s)}
```

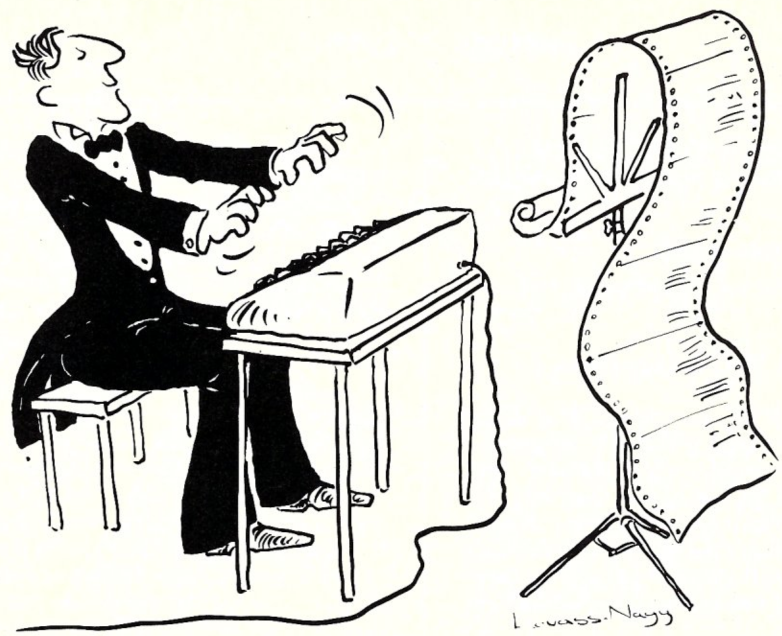
In this case, if the condition is FALSE, control shifts to the ELSE, and the last statement before the ELSE will be a GOTO past the end of the statements following the ELSE.

If the condition represents the various values that a variable may acquire, then alternative pathways may be executed using the construct IF . . . ELSE IF . . . ELSE IF . . . etc. If there are many pathways this can get confusing, even if each step in the chain is equally indented in the listing. Instead, the SWITCH construct may be used:

```
SWITCH (variable or expression)
  {
  CASE value1: {statement(s)}
   CASE value2: {statement(s)}
   ...
   DEFAULT: {statement(s)}
  }
```

Kernighan and Plauger say they prefer an ELSE IF chain to the CASE statement because it is less confusing, but this SWITCH implementation is a Berkeley add-on to Ratfor

The Ratfor translation process operates by scanning the text and looking for keywords that the preprocessor can recognize. Statements not containing keywords are assumed to be Fortran and are passed through the translator with only trivial changes.



The Well-Tempered Cross-Assembler

Before Johann Sebastian Bach developed a new method of tuning, you had to change instruments practically every time you wanted to change keys. Very difficult.

Before Avocet introduced its family of cross-assemblers, developing micro-processor software was much the same. You needed a separate development system for practically every type of processor. Very difficult and very expensive.

But with Avocet's cross-assemblers, a single computer can develop software for virtually any microprocessor! Does that put us in a league with Bach? You decide

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 3 years of actual use. Ask NASA, IBM, XEROX or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on any computer with CP/M* and process assembly language for the most popular microprocessor families.

XASM05	6805	$200 each
XASM09	6809	
XASM18	1802	
XASM48	8048/8041	
XASM51	8051	
XASM65	6502	
XASM68	6800/01	
XASMF8	F8/3870	
XASMZ8	Z8	
XASM400	COP400	
XASM75	NEC 7500	$500

(Coming soon: XASM68K .... 68000)

Turn Your Computer Into A Complete Development System

Of course, there's more. Avocet has the tools you need from start to finish to enter, assemble and test your software and finally cast it in EPROM:

Text Editor VEDIT -- full-screen text editor by CompuView. Makes source code entry a snap. Full-screen text editing, plus TECO-like macro facility for repetitive tasks. Pre-configured for over 40 terminals and personal computers as well as in user-configurable form.

CP/M-80 version $150
CP/M-86 or MDOS version $195
(when ordered with any Avocet product)

ROM Simulator -- ROMSIM by Inner Access eliminates need to erase and reprogram EPROM. Installed in an S-100 host, ROMSIM substitutes RAM for EPROM in external target system. 16K memory can be configured to simulate the 2708, 2758, 2716, 2516, 2732, 2532, 2764, 2564 in either byte or word organization. Avocet's configurable driver makes loading of HEX or COM files fast and easy.

From $495 depending on cabling and RAM installed.

EPROM Programmer -- Model 7128 EPROM Programmer by GTek programs most EPROMS without the need for personality modules. Self-contained power supply ... accepts ASCII commands and data from any computer through RS 232 serial interface. Cross-assembler hex object files can be down-loaded directly. Commands include verify and read, as well as partial programming.

PROM types supported: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 27C32, MCM8766, 2564, 2764, 27C64, 27128, 8748, 8741, 8749, 8742, 8751, 8755, plus Seeq and Xicor EEPROMS.

(Upgrade kits will be available for new PROM types as they are introduced.)

Programmer $389
Options include:
Software Driver Package .... $ 30
RS 232 Cable .... $ 30
8748 family socket adaptor ..... $ 98
8751 family socket adaptor $174

Call Us

If you're thinking about development systems, call us for some straight talk. If we don't have what you need, we'll help you find out who does. If you like, we'll even talk about Bach.

VISA and Mastercard accepted. All popular disc formats now available -- please specify. Prices do not include shipping and handling -- call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research.

AVOCET SYSTEMS INC.
DEPT. 983-M
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TLX 467210

CIRCLE 201 ON READER SERVICE CARD

and personally I would rather SWITCH than ELSE IF.

Here are some more Ratfor conditional control structures:

```
WHILE (condition) {statement(s)}
FOR (initialization; condition; reinitialization)
{statement(s)}
REPEAT {statement(s)} UNTIL (condition)
```

With the REPEAT, the UNTIL is optional. If it is not used, the loop may be exited with a BREAK statement, usually attached to an IF condition, which transfers control beyond the statements. This may also be used with any of the other forms of conditional loops. Similarly, NEXT may be used to transfer control to the beginning of the next iteration of the loop. The FOR statement is a generalization of the Fortran DO, and is more flexible. A trivial example of its use would be:

```
FOR ( i = 1 ; i <= 20 ; i = i + 1 ) sum = sum + x(i)
```

This example also shows a user-friendly convenience: < = is translated into the Fortran .LE., and there are equivalent representations for the other FORTRAN relational operators.

I have observed that the first time a Fortran user sits down to write a program in Ratfor, there will come a moment when he wants to use a GOTO. Strict mental discipline is required at this moment, for it is necessary to pause and decide which Ratfor control structure would be best suited to the purpose at hand. After successfully passing this milestone, there are no withdrawal pangs, and the urge to use GOTOs will not recur. GOTOs are legal in Ratfor, but I have yet to encounter a case in which I would want to use one.

The Software Tools version of Ratfor includes several macros; I will describe only the "define" macro. This takes the form:

```
define (name, replacement string)
```

This is particularly useful for giving generic names in Ratfor text, such as EOF or MAXARGLENGTH, and giving the numerical equivalents of these in a definitions file. Much of the effort in transporting Ratfor to a different system consists of changing numbers in the standard definitions file. This file is named RATDEF. One of the first things I did when I got my Software Tools disks from Unicorn Systems was to look for RATDEF. It was not there. By then I knew enough about the Software Tools to go looking for RATDEF in an archive file of Ratfor sources. There I found it. Ratfor tries to open this file when it is brought up. New users may be puzzled by an error message saying "can't open standard definitions file."

Ratfor does have a modest number of error messages. These are reasonably well explained in the Software Tools documentation. However, the explanation of the above error message does not tell you the name of the standard definitions file.

The documentation for Ratfor is part of the documentation for the Software Tools; it has been minimally edited for CP/M by Carousel MicroTools and I find it too terse. There is also a brief tutorial on Ratfor. To get a really good feeling for the usage of Ratfor, the new user would be better advised to work his way through the *Software Tools* book or through the Ratfor sources for some of the Tools themselves. Some of the Software Tools Users Group add-ons to Ratfor are mentioned only in a brief statement of the language syntax, without accompanying examples or further explanation. This means that you have to dig into the Ratfor sources for Ratfor to fully understand what Ratfor does. This sounds totally circular, but it is a tribute to the readability of the language that it works.

The Software Tools, including Ratfor, are in the public domain. Not in the public domain are those primitive routines that interface the language to a particular computing environment, in this case CP/M. Those primitives customized for CP/M are proprietary to Carousel MicroTools. However, Ratfor source code for everything is distributed with the Tools. This is important to the Ratfor user who is interested in program portability. Source code that makes a lot of system calls in a particular environment is not very portable. The progammer should exercise additional restraint and use the standard syntax in the Ratfor environment. Unfortunately, the higher-level primitives, shared by most of the Software Tools, are briefly documented. Little attempt is made to provide an overview of these by comparing similarity of function. Here again it is useful to read the *Software Tools* book carefully.

Ratfor may be a language in transition. A newsletter recently received from Carousel MicroTools reveals that there is a fair amount of sentiment in the Software Tools Users Group for abandoning Fortran as the standard compiler for the Tools, and using a more modern language now that such languages are more commonly available on computing systems. One of the options discussed is the possibility of developing the Ratfor language further and enabling it to output source code for C, Pascal, or other language compilers, as well as Fortran.

In summary, the use of Ratfor as a Fortran preprocessor tames the Fortran language by structuring it and making the source code much easier to read. The Software Tools Users Group version of Ratfor is a very worthwhile upgrade of the language, but is inadequately documented. The Carousel MicroTools implementation of it for CP/M is nicely done. The command file takes 46K of memory. Ratfor is distributed as part of the Software Tools implementation and is not separately priced.

References

B.W. Kernighan and P.J. Plauger: *Software Tools.* Addison-Wesley, Reading, MA, 1976.

Carousel MicroTools (formerly Unicorn Systems), *609 Kearney St., El Cerrito, CA 94530, (415) 528-1300.*

CompuCode, *6147 Aspinwall Road, Oakland, CA 94611.*

The use of Ratfor as a Fortran preprocessor tames the Fortran language by structuring it and making the source code much easier to read.

Z80 Software

SOFTWARE DESCRIPTIONS

TPM (TPM I) - $80 A Z80 only operating system which is capable of running CP/M programs. Includes many features not found in CP/M such as independent disk directory partitioning for up to 255 user partitions, space, time and version commands, date and time, create FCB, chain program, direct disk I/O, abbreviated commands and more! Available for North Star (either single or double density), TRS—80 Model I (offset 4200H) or II, Versafloppy I, or Tarbell I.

TPM-II - $125 An expanded version of TPM which is fully CP/M 2.2 compatible but still retains the extra features our customers have come to depend on. This version is super FAST. Extended density capability allows over 600K per side on an 8" disk. Available preconfigured for Versafloppy II (8" or 5"), Epson QX-10, Osborne II or TRS-80 Model II.

CONFIGURATOR I

This package provides all the necessary programs for customizing TPM for a floppy controller which we do not support. We suggest ordering this on single density (8SD).
Includes: TPM-II ($125), Sample PIOS (BIOS) SOURCE ($FREE), MACRO II ($100), LINKER ($80), DEBUG I ($80), QED ($150), ZEDIT ($50), TOP I ($80), BASIC I ($50) and BASIC II ($100)
$815 Value **NOW $250**

CONFIGURATOR II

Includes: TPM-II ($125), Sample PIOS (BIOS) SOURCE ($FREE), MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), QSAL ($200), QED ($150), ZTEL ($80), TOP II ($100), BUSINESS BASIC ($200) and MODEM SOURCE ($40) and DISASSEMBLER ($80)
$1485 Value **NOW $400**

MODEL I PROGRAMMER

This package is only for the TRS-80 Model I. Note: These are the ONLY CDL programs available for the Model I. It includes: TPM I ($80), BUSINESS BASIC ($200), MACRO I ($80), DEBUG I ($80), ZDDT ($40), ZTEL ($80), TOP I ($80) and MODEM ($40)
$680 Value **NOW $175**

MODEL II PROGRAMMER

This package is only for the TRS-80 Model II. It includes: TPM-II ($125), BUSINESS BASIC ($200), MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), QED ($150), ZTEL ($80), TOP II ($100), ZDDT ($40), ZAPPLE SOURCE ($80), MODEM ($40), MODEM SOURCE ($40) and DISASSEMBLER ($80)
$1445 Value **NOW $375**

BASIC I - $50, a 12K+ basic interpreter with 7 digit precision.

BASIC II - $100, A 12 digit precision version of Basic I.

BUSINESS BASIC - $200, A full disk extended basic with random or sequential disk file handling and 12 digit precision (even for TRIG functions). Also includes PRIVACY command to protect source code, fixed and variable record lengths, simultaneous access to multiple disk files, global editing, and more!

ACCOUNTING PACKAGE - $300, Written in Business Basic. Includes General Ledger, Accounts Receivable/Payable, and Payroll. Set up for Hazeltine 1500 terminal. Minor modifications needed for other terminals. Provided in unprotected source form.

MACRO I - $80, A Z80/8080 assembler which uses CDL/TDL mnemonics. Handles MACROs and generates relocateable code. Includes 14 conditionals, 16 listing controls, 54 pseudo-ops, 11 arithmetic/logical ops, local and global symbols, linkable module generation, and more!

MACRO II - $100, An improved version of Macro I with expanded linking capabilities and more listing options. Also internal code has been greatly improved for faster more reliable operation.

MACRO III - $150, An enchanced version of Macro II. Internal buffers have been increased to achieve a significant improvement in speed of assembly. Additional features include line numbers, cross reference, compressed PRN files, form feeds, page parity, additional pseudo-ops, internal setting of time and date, and expanded assembly-time data entry.

DEVELOPER I

Includes: MACRO I ($80), DEBUG I ($80), ZEDIT ($50), TOP I ($80), BASIC I ($50) and BASIC II ($100)
$440 Value **NOW $150**

DEVELOPER II

Includes: MACRO II ($100), MACRO III ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), BUSINESS BASIC ($200), QED ($150), TOP II ($100), ZDDT ($40), ZAPPLE SOURCE ($80), MODEM SOURCE ($40), ZTEL ($80), and DISASSEMBLER ($80).
$1280 Value **NOW $350**

DEVELOPER III

Includes: QSAL ($200), QED ($150), BUSINESS BASIC ($200), ZTEL ($80) and TOP II ($100)
$730 Value **NOW $300**

COMBO

Includes: DEVELOPER II ($1280), ACCOUNTING PACKAGE ($300), QSAL ($200) and 6502X ($150)
$1930 Value **NOW $500**

LINKER - $80, A linking loader for handling the linkable modules created by the above assemblers.

DEBUG I - $80, A tool for debugging Z80 or 8080 code. Disassembles to CDL/TDL mnemonics compatible with above assemblers. Traces code even through ROM. Commands include Calculate, Display, Examine, Fill, Goto, List, Mode, Open File, Put, Set Wait, Trace, and Search.

DEBUG II - $100, A superset of Debug I. Adds Instruction Interpreter, Radix change, Set Trap/Conditional display, Trace options, and Zap FCB.

6502X - $150, A 6502 cross assembler. Runs on the Z80 but assembles 6502 instructions into 6502 object code! Similar features as our Macro assemblers.

QSAL - $200, A SUPER FAST Z80 assembler. Up to 10 times faster than conventional assemblers. Directly generates code into memory in one pass but also to offset for execution in its own memory space. Pascal like structures: repeat...until, if...then...else, while...do, begin...end, case...of. Multiple statements per line, special register handling expressions, long symbol names, auto and modular assembly, and more! This one uses ZILOG Mnemonics.

QED - $150, A screen editor which is both FAST and easy to learn. Commands include block delete, copy, and move to a named file or within text, repeat previous command, change, locate, find at start of line, and numerous cursor and window movement functions. Works with any CRT having clear screen, addressable cursor, clear to end of line, clear to end of screen, and 80X24.

ZTEL - $80, An extensive text editing language and editor modelled after DEC's TECO.

ZEDIT - $50, A mini text editor. Character/line oriented. Works well with hardcopy terminals and is easy to use. Includes macro command capability.

TOP I - $80, A Text Output Processor for formatting manuals, documents, etc. Interprets commands which are entered into the text by an editor. Commands include justify, page number, heading, subheading, centering, and more.

TOP II - $100, A superset of TOP I. Adds: embedded control characters in the file, page at a time printing, selected portion printing, include/merge files, form feed/CRLF option for paging, instant start up, and final page ejection.

ZDDT - $40, This is the disk version of our famous Zapple monitor. It will also load hex and relocatable files.

ZAPPLE SOURCE - $80, This is the source to the SMB ROM version of our famous Zapple monitor. It can be used to create your own custom version or as an example of the features of our assemblers. Must be assembled using one of our assemblers.

MODEM - A communication program for file transfer between systems or using a system as a terminal. Based on the user group version but modified to work with our SMB board or TRS-80 Models I or II. You must specify which version you want.

MODEM SOURCE - $40, For making your own custom version. Requires one of our Macro Assemblers.

DISASSEMBLER - $80, Does bulk disassembly of object files creating source files which can be assembled by one of our assemblers.

HARDWARE

S-100 — SMB II Bare Board $50. "System Monitor Board" for S-100 systems, 2 serial ports, 2 parallel ports, cassette interface, 4K memory (ROM, 2708 EPROM, 2114 RAM), and power on jump. When used with Zapple ROM below, it makes putting a S-100 system together a snap.
Zapple ROM $35. Properly initializes SMB I/II hardware, provides a powerful debug monitor.

IBM PC — Big Blue Z80 board $595. Add Z80 capability to your IBM Personal Computer. Runs CP/M programs but does not require CP/M or TPM. Complete with Z80 CPU, 64K add on memory, serial port, parallel port, time and date clock with battery backup, hard disk interface, and software to attach to PC DOS and transfer programs. Mfr'd by QCS.
50% Discount on all CDL software ordered at the same time as a Big Blue (and for the Big Blue).

APPLE II — Chairman Z80 $345. Add Z80 capability to your Apple II/II Plus computer. Runs CP/M programs with our more powerful TPM. Includes 64K memory add on (unlike the competition this is also useable by the 6502/DOS as well as the Z80), TPM, QSAL assembler, QED Screen Editor, and Business Basic. Mfr'd by AMT Research.
Apple Special $175. Buy the Apple Z80 Developer at the same time as the "Chairman" and pay only $175 instead of $325.

APPLE Z80 DEVELOPER

Includes: 6502X ($150), MACRO II ($100), MACRO III ($150), QSAL ($200), QED ($150), LINKER ($80), DEBUG I ($80), DEBUG II ($100), ZDDT ($40) and BUSINESS BASIC ($200)
VALUE: $1250 **NOW $325**
$175 when purchased with AMT "Chairman" Board

DISK FORMATS

When ordering software specify which disk format you would like.

CODE	DESCRIPTION
8SD	8" IBM 3740 Single Density (128 bytes/26 sectors/77 tracks)
8DD	8" Double Density (256 bytes/26 sectors/77 tracks)
8XD	8" CDL Extended Density (1024 bytes/8 sector/77 traceks 616K)
5SD	5.25" Single Density (TRS80 Model I, Versafloppy I, Tarbell I)
5EP	5.25" Epson Double Density
5PC	5.25" IBM PC Double Density
5XE	5.25" Xerox 820 Single Density
5OS	5.25" Osborne Single Density
5ZA	5.25" Z80 Apple (Softcard compatible)

TPM INFO When ordering TPM I or II, in addition to Disk Format, please specify one of the following codes:

	CODE	DESCRIPTION
TPM I:	NSSD/H	North Star Single Density for Horizon I/O
	NSSD/Z	North Star Single Density for Zapple I/O
	NSDD/H	North Star Double Density for Horizon I/O
	NSDD/Z	North Star Double Density for Zapple I/O
	TRS80-I	TRS-80 Model I (4200H Offset)
	TRS80II	TRS-80 Model II
	VI8	Versafloppy I 8"
	VI5	Versafloppy I 5.25"
TPM-II:	VII8	Versafloppy II 8" (XD)
	VII5	Versafloppy II 5.25"
	TRS80II	TRS-80 Model II (XD)

Prices and Specifications subject to change without notice.
TPM, Z80, CP/M, TRS80 are trademarks of CDL, Zilog, DRI and Tandy respectively.

ORDERING INFORMATION:

VISA/MasterCard/C.O.D.
Call or Write With Ordering Information....

MasterCard VISA

OEMS:

Many CDL products are available for licensing to OEM's. Write to Carl Galletti with your requirements.

Dealer Inquiries Invited.

For Phone Orders ONLY Call Toll Free...
1-(800) 458-3491 (Except Pa.)
Ask For Extension #15

For information and Tech Queries call
(609) 599-2146

Computer Design Labs

342 Columbus Avenue/Trenton, NJ 08629

CIRCLE 84 ON READER SERVICE CARD

The Software Tools Computing Environment

by A.G.W. Cameron

The January 1983 issue of *Microsystems* featured articles on the UNIX operating system. UNIX is a favorite with programmers because it has a large number of programming tools and supports an elegant programming concept known as **pipes.** The tools for the most part accept a stream of ASCII characters from a "standard input," carry out some operations on them, and put them out to a "standard output." The technical name for such a program is a **filter.** A **pipe** is an operating system command that sets up a coordinated operation involving several tools so that the standard output from one is a standard input to another. See the January issue for a more detailed discussion of these concepts.

UNIX tends not to be a favorite of commercial types who aren't versed in programming techniques. Among other reasons, it is not very generous with verifications of commands to the operating system. In general, when a new prompt appears, you have to assume that the previous command was carried out correctly, for otherwise there would have been an error message. Here, "no news is good news."

As you probably know, a great deal of effort is being expended to transport the UNIX operating system to new computing systems, including the installation of UNIX look-alikes on many of these systems. Not so well known has been an effort to provide programming tools similar to those in UNIX and **pipes** as well, designed to run under a great variety of different operating systems on a wide variety of computers.

Much of this effort has been centered around Berkeley with assistance through a grant from the Department of Energy, but small groups around the country have made contributions as well. As there is now no more grant support, further coordination depends upon volunteer effort through the Software Tools Users Group. There are two users groups for UNIX, one technically oriented and one not, and the Software Tools Users Group has been meeting with one or both of these organizations.

It all started, innocently enough, with the publication of a book, *Software Tools,* by B.W. Kernighan of Bell Laboratories and P.J. Plauger of Yourdon, Inc., a software firm. This book was published in 1976 by the Addison-Wesley Publishing Company, but it is interesting to note that the author's institutions retained the copyright. Also made available through Addison-Wesley, for a nominal administrative charge of $25, was a tape containing the program source material discussed in the book. For this price you copy the tape and then are expected to return it to the publisher.

I first encountered this book a few years ago while browsing through the technical section of the Harvard COOP, a department store in Harvard Square. I bought it on impulse. It gave me a great deal of pleasure as I read it in short sections while I flew around the country to a series of committee meetings.

The book developed a large number of programming tools, modelled on those in UNIX, and presented in a structured programming language. At the time I did not think of these tools as a usable system worth expending the effort to implement on my two Nova minicomputers (except for the Ratfor structured programming language described in an accompanying article in this magazine). Rather, I viewed them as examples developed to demonstrate to the reader a wide variety of good programming practices. A system of practical tools, I thought, should be more elaborate than these examples. Indeed, Kernighan and Plauger gave a lot of practice problems designed to bring about just some of that desired elaboration.

Some of the more important good programming practices emphasized by the authors are these: Keep each program module simple, and try to accomplish essentially one thing in it (although that thing may actually be quite complicated, like the main module of a program which serves to organize the program flow through the other principal modules without going into details at all). Push the details of an algorithm down to quite a low level in the program. Push the details of the interaction with the operating system down to the lowest level of all, and make sure that every interface with a given function goes through just a single primitive routine. In that way the programmer may be only minimally concerned with how his instructions are implemented on a given system. Only these primitive modules need be changed when the tools are transported to a new machine.

Kernighan and Plauger wrote their book for general users, most of whom would not have access to UNIX or to the C programming language, which were primarily restricted to implementations on PDP/11 machines at the time of writing. They needed a programming language that was widely available, and about the only one that would fill the bill was Fortran. But the lack of programming structure made it unacceptable per se. So they developed a structured programming language—Ratfor, modelled on C—that translated input program statements to Fortran. Much of my pleasure in reading their book was in following through the programming examples and learning to appreciate Ratfor.

Meanwhile, the Department of Energy provided a grant to a group at the Lawrence Berkeley Laboratory for a research program on uniform user interfaces in heterogeneous environments. This group then became the focal center of the Software Tools Users Group (STUG). They set out to create a virtual operating system that could be transported to a large number of different machines and made to run under the various native operating systems. It was decided to model the virtual operating system after UNIX. The resulting program, the Software Tools Shell, was created to use the primitive routines incorporated in the Software Tools, which were already modelled on the tools used in UNIX. This meant that the Shell was written in Ratfor. In general, the Tools can be run either directly

A. G. W. Cameron, Harvard College Observatory, 60 Garden St., Cambridge, MA 02138.

under the native operating systems or under the programming environment of the Shell. The intent of the Shell was to provide a familiar environment and familiar tools to a user who moved from one computer to another.

The task of transporting the Software Tools to different machines and operating systems has been carried out by a large number of individuals, many of whom own their own small software firms. The transportation to CP/M was carried out by Unicorn Systems (which has since incorporated as Carousel MicroTools, Inc.). The principals of Unicorn Systems—Deborah Scherrer of the Computer Science and Mathematics Dept., Lawrence Berkeley Laboratory, who was also a principal research worker under the Dept. of Energy grant and a leader in the Software Tools Users Group, and Phillip Scherrer, a solar physicist at Stanford University—continued as principals with Carousel. In general, this CP/M implementation has been very cleverly done.

The Software Tools contain about 60,000 lines of code. The principal task facing the Scherrers was to keep down the space occupied by the Tools to that which would be manageable for a system having only floppy disks, although it is quite evident that the Software Tools would be very practical for a hard disk system. They succeeded in keeping all of the Tools (not counting the Dictionary) to about two-thirds of a megabyte by using a clever trick.

As is always the case in a Fortran program, any executable program draws a lot of service routines from the Fortran libraries. For the Software Tools there are more than 16 kilobytes of these, and they tend to be pretty much the same ones for the different tools. The Scherrers provided several ways for these routines to be supplied. The simplest, and most space-consuming way, was to make the Tools self-sufficient as regular COM files to run directly under CP/M. The second way was to create a special program, RUN, which itself executed under CP/M, and took as an argument the name of the Tool which was to be run. This tool was then stored in a more compact form, with extension .TOL, stripped of the service routines. These service routines are contained in RUN, and the TOL file overlays part of RUN in such a way that it can then use the service routines. The third way is to run the Shell under CP/M; the Shell also contains the service routines, and again the .TOL program is overlaid on part of the Shell and uses its service routines. The COM and TOL versions of a tool can be interconverted using another special program called INSTALL.

If the virtual operating system Shell is used, one wishes to return to it after a Tool is run. The Scherrers modified the terminating service routine **endst** to exit to CP/M in case a Tool aborts on an error. Otherwise, it consults a temporary file to determine what to run next, including the Shell if no other program has been listed to run next. The Shell may take its instructions from a SUBMIT-like file called a **shell script**. This is more general and much better than the CP/M SUBMIT facility. The user may define a "shell environment" which contains search paths to search for any new program name on several disks and for several user numbers. The Shell makes it easier to use the user numbers, and normally the user is expected to specify them in shell commands.

One problem with all of this is the speed of execution involved in using the Shell. The Scherrers have done their best to improve the speed by making use in the Shell of part of memory as a cache to reduce the number of necessary disk accesses. I have run the Software Tools on two micros: an Otrona Attache and an Ithaca Intersystems S-100 bus system with a regular Cache BIOS. On both systems, running a series of Tools under the Shell gives satisfactory speed, and the Shell prompt ('%') returns quickly enough. However, on the Otrona, calling for the execution of a CP/M program while under the Shell causes a rather annoying delay before the Shell returns. The delay is very much reduced on the Ithaca Intersystems machine. I would judge that many people who want to run a mixture of CP/M programs and Tools would prefer to do so from CP/M, since this eliminates these delays, and **pipes** would not be useful in these circumstances anyway. The Shell would probably be the preferred mode when using just the Tools.

There are too many Tools (60) to mention more than a few of them. Most of the Tools are intended for text manipulation of some kind. Some of the Tools carry out an elaborate series of operations; others do just a single simple thing. One useful example of the former is the AR Tool, which maintains archive files. If you are writing a book, it is very convenient to be able to maintain the various chapters as separate files, but to be able to concatenate them into a single file for housekeeping purposes. AR does this, and allows you to move the resulting file around as a single entity. There are a number of commands associated with the AR Tool which allow you to replace files in the archive, extract them, delete them, print them, or print a table of the archive contents, or even salvage a damaged archive. For program development, AR is also very convenient for holding all of the subroutines that may be associated with a main Fortran program.

An example of a very simple tool is CRT, which copies a file to the terminal. It does so 22 lines at a time, then pauses and prints a message telling you to hit the carriage return to get the next set of 22 lines, or to hit q to quit. That is also a very useful facility, especially as the last command at the end of a **pipe**.

However, some of the major Tools are very crude and unsophisticated compared to other programs which are available to run on CP/M machines. There is a desk calculator, DC, which can perform only integer arithmetic. When I want to use a calculator, I usually want to use real numbers, and I would generally be better off using an interactive Basic for a desk calculator. The Software Tools editor, ED, does not support a screen display, and therefore it will not be most people's choice for a text editor. Furthermore, the editing commands, though meeting minimum requirements to do the job are extremely crude compared to those available in the command mode of a first-class editor such as PMATE. There is a tendency in the Software Tools community to invoke ED and input commands to it within shell scripts, but I suspect that few

The CP/M implementation of the Tools is very cleverly done, although their sheer massiveness creates a certain awkwardness that is hardware dependent.

users accustomed to better editors will be willing to learn the ED commands for that purpose alone.

There is a FIELD Tool available to manipulate fields of data in files. If you want to do much of that sort of thing, you are probably already using something like dBASE II, which is far better. Similarly, the FORMAT Tool (sometimes called ROFF), which is a text output formatter, will be a great disappointment to those wishing to have decent text output. It is extremely limited in its output options and achieves justification by inserting whole spaces into text, a practice I consider hideous. There are much better tools available under UNIX itself, so I am surprised that these have not been copied. Under CP/M some programs are appearing, such as SCRIBBLE, which support true proportional spacing, but as yet these are very limited, since they still make it very difficult to put what you want where you want on the page. However, they are a great improvement on FORMAT.

A SPELL Tool is available to find spelling errors in text files, using a 42,000-word dictionary. I have not tried to use this. The Scherrers warn in their manual that this dictionary was the first one distributed by the STUG and that it contains some errors and words specific to the development sites. The current STUG dictionary is about twice as long, but it has not been included in the CP/M Tools package for obvious reasons of space. It is also obvious that the CP/M user has a variety of other spelling packages available that are more sophisticated than SPELL, such as THE WORD PLUS, which can in some cases also criticize your style and indicate hyphenation.

The manual distributed with Software Tools for CP/M consists of the standard descriptions of the Software Tools that are being provided to all of the sites, with editing to indicate items specific to the CP/M version, and a section particularly written to describe the CP/M implementation. I consider it to be barely adequate for an experienced user. I recommend that the user also read the *Software Tools* book and a good book on UNIX and UNIX-related tools. The general inadequacy of the documentation may be presently limiting the popularity of the Software Tools among potential users.

The manual is also contained on the 15 single-sided single-density 8″ floppy disks on which the Software Tools arrive. There is a MAN Tool designed to print out part of the manual for the user as a help device. I suspect this will not easily be feasible for anyone without a hard disk. I can keep the Tools on a double-sided double-density 8″ floppy disk on my Intersystems micro, but on the Otrona, which has smaller floppies with 360K available on each disk, I have to divide selections of the Tools between two floppies, one primarily for text manipulation purposes, and one primarily for program development purposes. I suspect that users with still smaller floppy capacities will be limited to using the Tools singly, and will not be tempted to implement the Shell. I do not have the dictionary, the manual, or the Ratfor sources normally on-line.

In summary, the Software Tools is an excellent concept which has a lot to offer users at several different levels. The

Software Tools continued . . .

CP/M implementation of the Tools has been very cleverly done. Nevertheless, the sheer massiveness of the Tools software creates a certain awkwardness for the CP/M implementation, the extent of which is hardware dependent. Nearly everyone has something to gain from the Tools. However, some of the major Tools are inferior to the excellent CP/M software which is otherwise available. The cost of the Tools, at $249 without sources and $349 with sources, is modest considering the large amount of software involved. The package can be ordered from: **Carousel MicroTools, Inc.** (formerly called Unicorn Systems), *609 Kearney St., El Cerrito, CA 94530, (415) 528-1300.*

REFERENCES

B.W. Kernighan and P.J. Plauger: *Software Tools.* Addison-Wesley, Reading, MA, 1976.

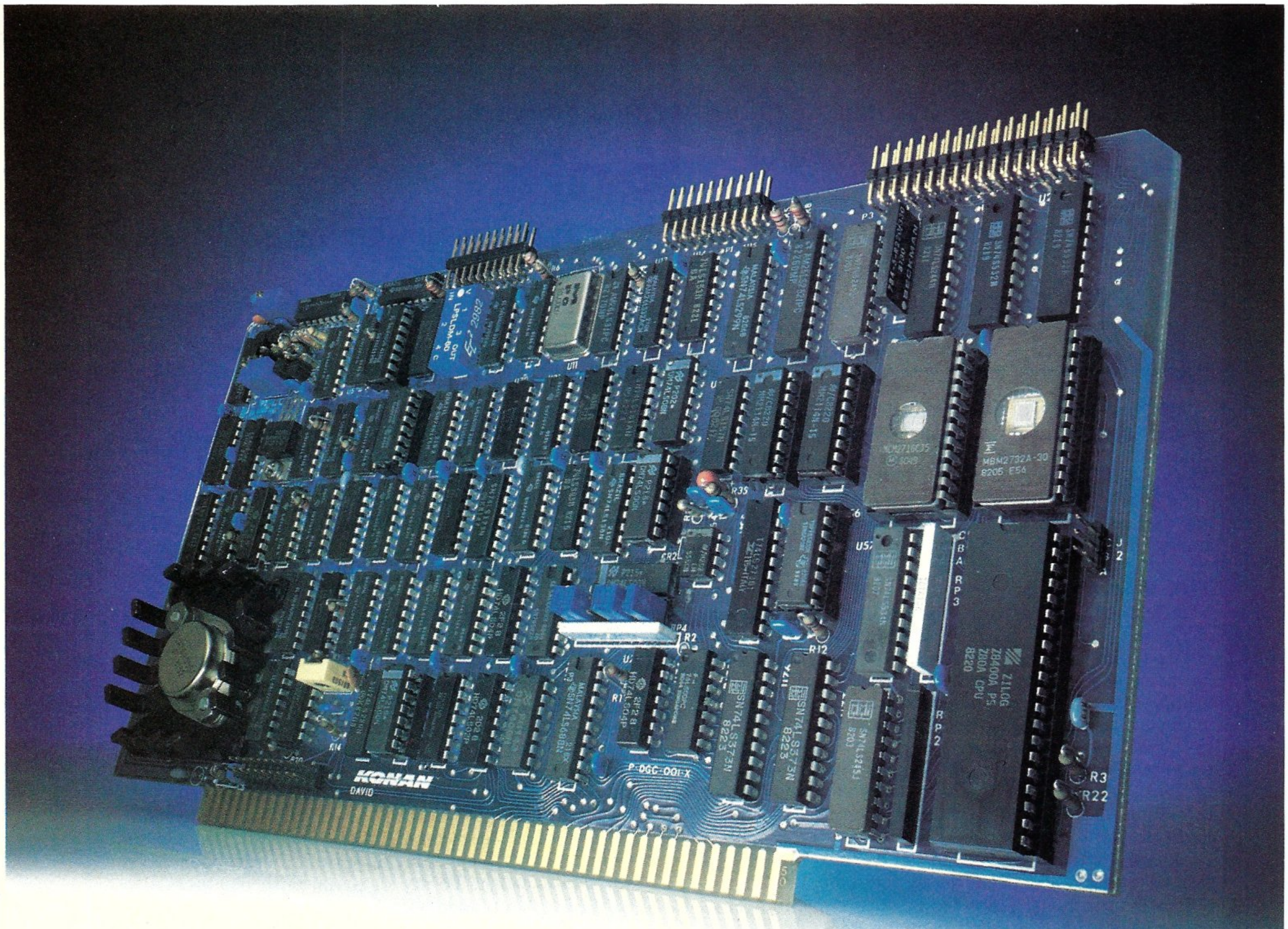
Software Tools Users Group, *1259 El Camino Real #242, Menlo Park, CA 94025.*

UNIX is a trademark of Bell Telephone Laboratories, Inc., Murray Hill, N.J.

PMATE is a trademark of Phoenix Software Associates, Ltd.

dBASE II is a trademark of Ashton-Tate, 10150 W. Jefferson Blvd., Culver City, CA 90230.

THE WORD PLUS is a trademark of Oasis Systems, San Diego, CA.


KONAN
DAVID

MICROSYSTEMS REVIEWS THE

Micronix

A Morrow Designs operating system combining the best features of CP/M and UNIX

by David Gewirtz

CP/M is primarily a single-user, single-tasking operating system. UNIX is primarily a multiuser, multitasking system. The Micronix operating system, available on the Morrow Decision 1 computer, provides access to the wealth of CP/M software along with the power and programmability of multiuser UNIX.

The Decision 1

The Decision 1 from Morrow is a Z80 S-100 based machine with some truly powerful hardware features.

The system I worked with had 320K RAM. Using 24-bit extended addressing, the Z80 is able to access the entire memory address space. The hardware-provided memory management support allows multiple users to share memory space by allocating memory for use by the CPU in 4K increments as needed. Each 4K segment has memory protection attributes, so that the operating system can prevent users from accessing memory that has not been allocated to them.

A hardware-trapping mechanism has also been implemented to allow the operating system complete control over user operations. Traps can be set up in a number of operating system controlled configurations, or disabled entirely. In addition to preventing users from disturbing each other's work, the trapping mechanism prevents users from corrupting such system resources as the disks, memory, and I/O facility.

The Morrow system came configured with one $5^1/_4''$ hard-sectored floppy (North Star compatible) using the Morrow DJ/DMA floppy controller. The majority of work was done with a 16MB 5″ hard disk that used the Morrow HDC/DMA hard disk controller.

To round everything out, the Decision 1 comes with three RS-232C serial ports, a parallel port, a plug/port for an 8″ Shugart-compatible drive, a real-time clock, and progammable interrupt controller.

Some more detailed hardware information about the Decision 1 machine can be found in an article by Ernest Mau in the April 1983 issue of *Microsystems.*

UNIX-like Micronix

Micronix, as its name implies, is a micro version of the UNIX operating system. It supports many of the now-famous UNIX features, including multiple users, multiple tasks, a hierarchical file system, UNIX system call similarity, file protection, many software tools, I/O redirection, pipes, and an on-line reference manual.

One of the most valuable facets of Micronix is a facility called **upm. Upm** is a CP/M simulator. By providing an environment with comparable BIOS and BDOS calls running native on an interrupt-driven Z80 processor, many of the nationally sold CP/M software packages can run without modification under Micronix and **upm.** As proof, this entire article was written using WordStar while logged in as user 'david' in Micronix.

When a customer purchases the Decision 1 with Micronix, a number of CP/M-based packages are included. These packages are the same as those available on the popular Morrow Micro Decision and include WordStar from MicroPro, Microsoft BASIC, BaZic, Personal Pearl, the LogiCalc spreadsheet, and the Correct-it spelling checker. I have found that WordStar, Microsoft BASIC, and Correct-it work under Micronix **upm** without fail. I haven't been able to properly test Personal Pearl, LogicCalc, and BaZic. I have tested both the Aztec and BDS C compilers and have found both to work perfectly. For variety, I tried to run Microsoft's MultiPlan and a BDS C based program called ALIAS, which I built to aid me when typing long command lines. ALIAS makes extensive use of a number of CP/M-specific features to create a pseudo-name for a command line, and MultiPlan uses an awful lot of just plain crunching power for files and processor-bound work. To my surprise, both products also worked without fail in **upm.**

Why Micronix?

With other microcomputer operating systems, including MP/M and OASIS, already established commercially, Micronix's raison d'être is firmly entrenched in the hardware of the Decision 1. Because Morrow is most familiar with the S-100/Z80 hardware environment and the CP/M software environment, it makes sense for the company to produce its first high-end machine based on that technology. In order to complement the proliferation of the smaller Micro Decisions, it also makes sense to give the Decision 1 multiuser capability. The available multiuser Z80-based operating systems, OASIS and MP/M, are inferior to UNIX. Because of the limited capabilities of both MP/M and OASIS, those manufacturers who have chosen to make CP/M available to multiple users, including TeleVideo, Gifford, and OSM, have each seen the need to build a custom form of a CP/M-compatible multiuser OS. Micronix, with its similarity to the UNIX environment, takes the proprietary CP/M-like multiuser operating system one step further. By simulating UNIX with its named hierarchical directories, user file protections, and login capabilities, as well as providing a CP/M "engine," Micronix offers the best of both worlds for the Z80 processor.

Micronix in detail

Micronix is a hard disk based operating system (though I did get a minimal system to boot on a floppy). As such, it has a large capacity for file storage. One of the most wonderful features of UNIX—a hierarchical file system—is beautifully simulated by Micronix.

Under CP/M 2.2, files are organized in either drives or

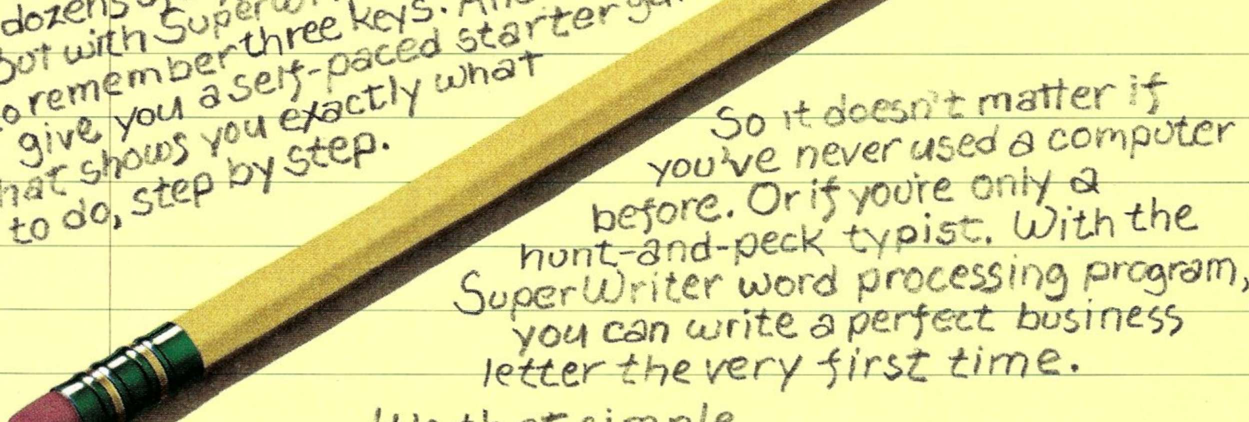
David Gewirtz, One Captain Drive, No. 457, Emeryville CA 94608

You're looking at the only word processor that's as easy to use as SuperWriter.™

If you're like most people, you don't want to make a career out of word processing. When you have to write a simple letter or memo, all you want to do is roll up your sleeves and get down to business.

And if you're buying a personal computer, now there's a new software package that makes things just that simple.

SuperWriter.™



Write your first letter in 10 minutes flat.

With some word processing software, you have to learn dozens of different commands. But with SuperWriter, you only have to remember three keys. And we give you a self-paced starter guide that shows you exactly what to do, step by step.

So it doesn't matter if you've never used a computer before. Or if you're only a hunt-and-peck typist. With the SuperWriter word processing program, you can write a perfect business letter the very first time.

It's that simple.

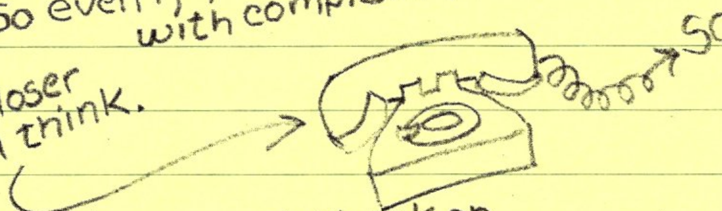
Everything you need for only $295.00 — even a spelling che(c)ker.

We're talking about a really complete package here. You get a full-function word processor, our 20,000-word Super SpellGuard™ spelling checker, and a form letter generator. All for $295.00 suggested retail. To get all these functions any other way, you might have to spend $900.00 or more.

You never knew your letters could look so good.

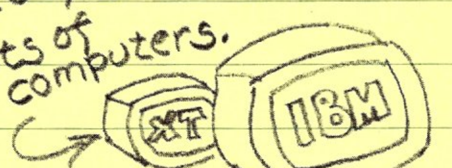
We've built all kinds of professional features into SuperWriter. Like proportional spacing. Boldface highlighting. Memo formatting. And more. So even if you're a complete beginner, you'll wind up with completely professional results.

Help is closer than you think.

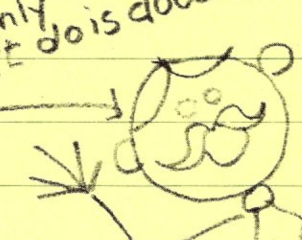


If you get stuck on a problem with any other word processing software, you'll probably have to go back to the store for help. But if you have any questions at all about SuperWriter, all you do is pick up the phone. We've got a full-time crew of customer service experts on call, ready to help you out.

Works on lots of personal computers.



About the only thing it can't do is doodle.



Right now, SuperWriter is available for the IBM PC and the new IBM PC XT. Of course, it takes full advantage of their features—full color, hard disk and more. By June 15, we'll have SuperWriter ready for CP/M, CP/M-86 and MS/DOS operating systems, which means you can use it on most of the popular personal computers.

So stop by your nearest computer or software store soon and ask about SuperWriter. In just 10 minutes, you can learn everything you ever need to know about word processing. And see why every other system is too complicated for words.

SuperWriter™
SORCIM®

2310 Lundy Ave.
San Jose, CA 95131
(408) 942-1727

users, with each drive having 16 users. Each drive is assigned a unique letter, from A through P, and each user assigned a unique number from 0 through 15. A file can be found in an area specified by the drive/user combination. A file can be specified by logging into a specified user area (with the user command) and then by preceding the file name with a drive specification:

```
A>USER 5
A>TYPE B:FILE
Now is the time for all men...
A>[]
-----------------------------------
Typing the file FILE in
User 5, Drive B
```

The Micronix file system

The Micronix file system is based on a tree structure, in which each directory is given a name. The top directory of the tree is called the "root directory" and is symbolized with '/' (slash). Directories (and files) under the root directory may have any name at all. Figure 1 shows a sample Micronix file system.

By using named hierarchical directories such as the one depicted in Figure 1, it is possible to organize groups of files clearly. In this case, I built a directory with my name on it called **david.** Since one of my projects was a review of Micronix, I created a directory "under" **david** called **review.** In the review, I was working with benchmark programs, final versions of those programs, and the text of the articles, so I created three directories under **review** called **text, work,** and **finals.** One file in **finals** is **benchmark.c.** In addition, I could have put a directory as well as files under **finals** if I needed to. Using these hierarchical directories, large projects can be easily organized.

It is possible to "move around" in the directory tree by using the change directory (**cd**) command. I can move down to the **text** directory by giving the following commands:

```
% cd /
% cd users
% cd david
% cd review
% cd text
% pwd
/users/david/review/text
% []
```

In the above example, the system prompt is the UNIX percent sign prompt instead of the familiar A> given by CP/M. The **pwd** (Print Working Directory) command is used to display the current directory as a "path". It is possible (and more common) to get to the directory **text** under the directory **review** by using **cd** to specify the path in the following manner:

```
% cd /users/david/review/text
% []
```

The previous directory (the one directly above the current directory) is specified by the symbol '..' (dot-dot in UNIX vernacular). To move from the **test** directory to the finals directory, the following command could be used:

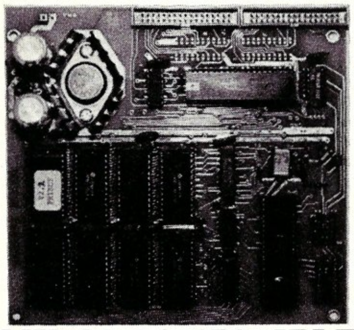
```
% pwd
/users/david/review/text
% cd ../finals
% pwd
/users/david/review/finals
% []
```

Micronix, like UNIX, allows individual disks to have their own file system. For example, you may have one hard disk containing the root directory and its branches. Micronix can configure floppy disks (or other hard disks) to also have their own file systems. By "mounting" a

```
                                /
          +---------------------+---------------------+
          :                     :                     :
         bin                  users                  cpm
     +----+----+           +----+----+          +-----------+
     :    :    :           :         :          :           :
     ls   cp   mv        david     ellen     wordstar   multiplan
                        +----+                  +
                        :                       :
                     review                   ws.com
                  +------+-------+
                  :      :       :
                text   finals   work
        +---------+             +----+
        :         :                  :
   outline.txt  art.txt          benchmark.c
```

Figure 1. *An example of a directory tree.*

Dirt Cheap.



MX-80 COMPATIBLE PRINT BUFFER

$49.50 + $2.50 SHIPPING

System consists of: Bare board, software in ROM and complete documentation.

- Uses popular 2K or 8K byte wide RAMs for expansion to 14K or 56K.
- Automatic adjustment for RAM size.
- Diagnostics and status indicators supported.
- Parallel interface (Centronics and Epson compatible).
- Clear, copy and pause switches supported.
- Requires only 7 non RAM ICs: (8085, 8155, 74LS373, 74LS138, 74LS32, 74121, 74LS14)
- Small 6" x 6" size.

ADCOM SYSTEMS
P.O. BOX 345 • MILFORD, OHIO 45150
(513) 831-1561

VISA MasterCard

Ohio Residents Add 5.5% • Add $3.00 For C.O.D. Orders

CIRCLE 7 ON READER SERVICE CARD

GREAT FEELING
#MBC 1250

1 MB per drive. 640 x 400 pixel graphics. 15 function keys. Sanyo-Graphic-Basic included. Shown: MBC1250, $2995. MBC1200, single drive,
$2495.

YOUR CUSTOMER GETS $2,000 OF FREE SOFTWARE WITH EVERY SANYO COMPUTER YOU SELL.

SANYO DESK-TOP COMPUTERS
"It's a great feeling"

SANYO
DESK-TOP BUSINESS COMPUTERS BY
SANYO BUSINESS SYSTEMS CORP.
51 Joseph St., Moonachie, N.J. 07074 (201) 440-9300 (800) 526-7043

CIRCLE 22 ON READER SERVICE CARD

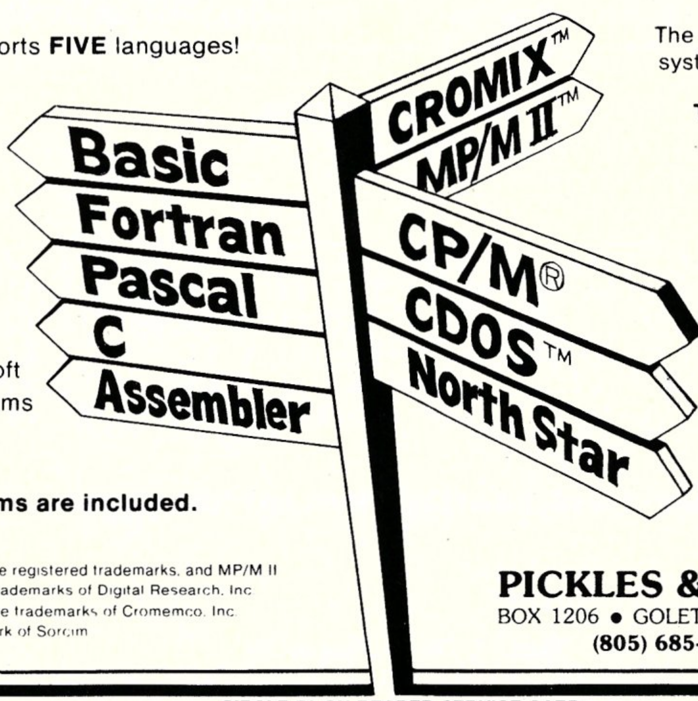
THE P&T IEEE-488 BUS GOES TO THEM ALL!

The P&T-488 interface enables you to use your S-100 computer and any of these operating systems and languages to communicate with 488 equipment.

The P&T-488 supports **FIVE** languages!

- Basic:
  Microsoft
  CBasic 2*
  Cromemco
  North Star
- Pascal:
  Pascal/M™
  Pascal/MT+™
- Fortran: Microsoft
- C: Quality Systems
- Assembler

Sample Programs are included.



The P&T-488 supports **5** operating systems, **2** of which are multiuser!

The P&T-488 includes useful utilities!

- Interactive **bus monitor** aids setting up test equipment.
- **Self test** checks the interface for proper operation

The P&T-488 is **complete!** Interface, manual, programs on disk, 18" cable and connector mounting hardware are all included for $450 (domestic, FOB Goleta).

★ CP/M and CBasic 2 are registered trademarks, and MP/M II and Pascal/MT+ are trademarks of Digital Research, Inc.
★ CDOS and CROMIX are trademarks of Cromemco, Inc.
★ Pascal/M is a trademark of Sorcim

PICKLES & TROUT®
BOX 1206 • GOLETA • CA 93116
(805) 685-4641

CIRCLE 70 ON READER SERVICE CARD

floppy's file system onto that of the root directory, the file system on the floppy can be treated as if it were just another part of the hierarchy originating with the root.

Multiuser and multitasking

When the Micronix system is powered up, it comes up in "single-user mode." The user can interact with full system with privileges as the 'root' (or super) user through the console terminal. When the console 'root' user types the character CONTROL-D (the UNIX End-of-File character), the single-user shell (user console command processor) exits, and the multiuser system is started.

The Micronix system can have up to 20 independent concurrently executing processes. A process (or task) is a separate element of programming. The single-user Micronix system uses a minimum of three tasks. The **System** task provides the actual supervisory control over the system. The **Init** task is used to start up individual terminal tasks. The **-sh** task is the Micronix shell.

Once single-user mode is exited, multiuser mode is started up. **Init** spawns a process for each terminal specified in the terminal configuration file **/etc/ttys.** The **/ect/ttys** file specifies the baud rate for each terminal port, and whether the terminal can be used to log in:

```
% cat /etc/ttys
ttyA  9600    login
ttyB  9600    login
ttyC  1200
% []
```

The /etc/ttys file is used to configure terminal baud rates and login capabilities.

UNIX version 7 uses a slightly different format than Micronix in the **/etc/ttys** file, and UNIX System III uses the file **/etc/inittab** instead of **/etc/ttys.**

When the user sits down to the Micronix multiuser system, the **/etc/banner** file, followed by the login prompt (see Figure 2), is presented.

The **/etc/banner** file can be modified or eliminated. In the case of the system I used, the line "MICROSYSTEMS MAGAZINE EVALUATION SYSTEM" was added to the end of the file.

In response to the "Name:" login prompt, the user must type in his or her user name. Depending on the configuration of the password file **/etc/passwd,** the system will ask the user for his password. When the password is typed, it is not echoed (displayed), protecting the sanctity of password access.

The **/etc/passwd** file contains information about the user. The file consists of any number of password records, each with seven fields:

```
name (login name)
encrypted password
numerical user ID
numerical group ID
comment about user
home directory
command interpreter program name
```

The name field specifies who the user is. When I log in as **'david'**, the record with the name field of **'david'** is selected to describe the actions to be taken. A password can be placed in the password field by running the **passwd** program. The user and group ID fields are used when determining the protections and access privileges of the user. Two of the most important fields of the password file are the home directory field and command interpreter name field.

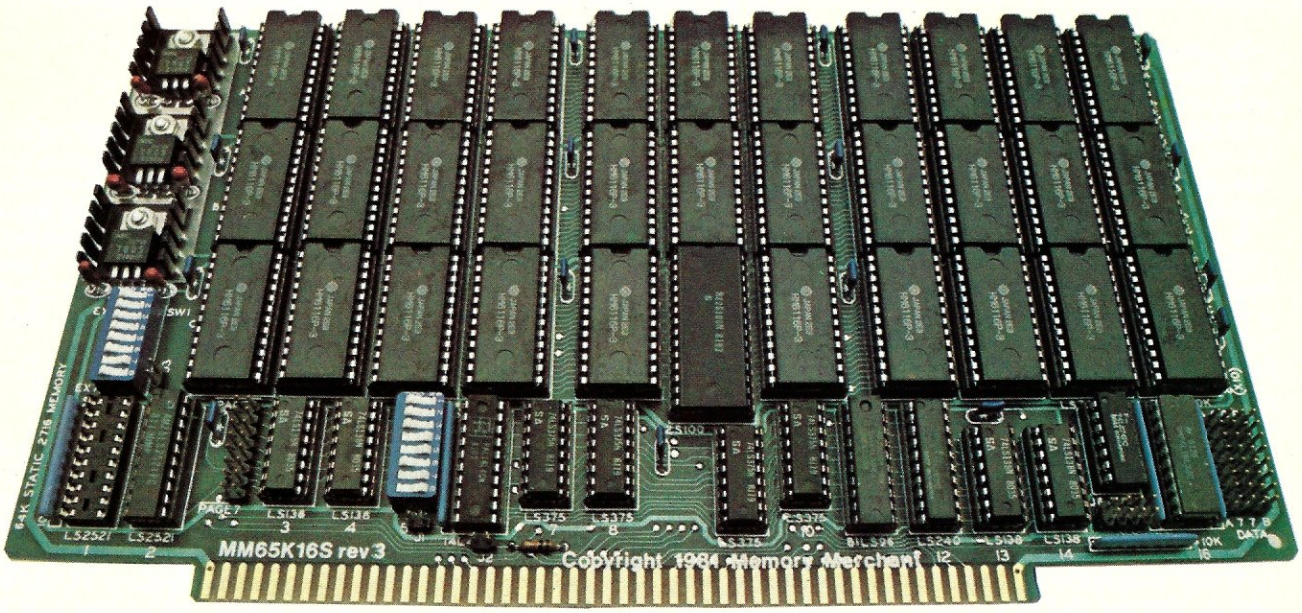
When I log in as **'david'** and type my password, I am placed in the directory **/a/david.** This is called my "home directory". The home directory for user **'jim'** is **/a/jim,** for **'guest'** it is **/a/guest,** and for **'demo'** it is **/a/demo.** The home directory does not necessarily have to correspond to the user name.

CP/M has only one command interpreter, the Console Command Processor. But in UNIX and Micronix, placing

```
                         MORROW DESIGNS
                           DECISION I


  __   __    _   ______   ______   ______   __   _    _  __  _
 / /\ / /|  //|  /    /|  /    /|  /    /|  / /\ //|  //| //\ //
||\\//|| |  || | ||||||/  |||||| |  |||||| |  ||\\ || |  || | \\  //
|| |   || |  || | || |     ||  || |  ||  || |  || \\|| |  || |  \\//\
|| |   || |  || | || |___  ||||||/  ||  || |  ||   || |  || |   \\  \
|| |   || |  || | ||/   /| || \\ \  ||  || |  ||   || |  || |  //\\  \
||/    ||/   ||/  ||||||/  ||  \\/  ||||||/   ||   ||/   ||/  //  \\/


              MICROSYSTEMS MAGAZINE EVALUATION SYSTEM


Name:
```

Figure 2. *The* **/etc/banner** *file.*

64K STATIC RAM MEMORY



S-100 STATIC MEMORY BREAKTHROUGH

Finally, you can buy state-of-the-art S-100/IEEE 696 static memory for your computer at an unprecedented savings.

Memory Merchant's memory boards provide the advanced features, quality and reliability you need for the kind of operational performance demanded by new high-speed processors.

Completely Assembled.

These memory boards are not kits, nor skeletons — but top-quality, high-performance memories that are shipped to you completely assembled, burned-in, socketed, tested and insured with one of the industry's best warranties.

Superior Design & Quality.

Memory Merchant's boards are created by a designer, well known for his proven ability in advanced, cost-efficient memory design. Innovative circuitry provides you with highly desired features and incredible versatility.

Only first-quality components are used throughout, and each board is rigorously tested to assure perfect and dependable performance.

No Risk Trial.

We are so convinced that you will be absolutely delighted with our boards that we extend a no-risk trial offer. After purchasing one of our boards, you may return it (intact) for any reason within 15 days after shipment and we will refund the purchase price (less shipping).

NEW S-100 PRODUCTS COMING SOON:

* DUAL 8/16 BIT CPU BOARD
* 128K 8/16 BIT STATIC RAM
* 256K 8/16 BIT DYNAMIC RAM

$629.

48K PARTIALLY POPULATED $519.
32K PARTIALLY POPULATED $409.

64K RAM, MODEL MM65K16S

- 64K × 8-bit
- Speed in excess of 6 MHz
- Uses 150ns 16K (2K × 8) static RAMS
- Ultra-low power (435 Ma. max. — loaded with 64K)
- Bank Select **and** Extended Addressing
- A 2K window which can be placed anywhere in the 64K memory map
- Four independently addressable 16K blocks organized as:
  — Two independent 32K banks **or**
  — One 64K Extended Address Page **or**
  — One 48K and one 16K bank for use in MP/M[1] (option)
- Each 32K bank responds independently to phantom
- 2716 (5V) EPROMS may replace any or all of the RAM
- Field-proven operation in CROMEMCO CROMIX* and CDOS*.
- Compatible with latest IEEE 696 systems such as Northstar, CompuPro, Morrow, IMS, IMSAI front panel, Altair and many others.

OEM and DEALER inquiries invited.

Memory Merchant™
14666 Doolittle Drive
San Leandro, CA 94577
(415) 483-1008

CIRCLE 98 ON READER SERVICE CARD

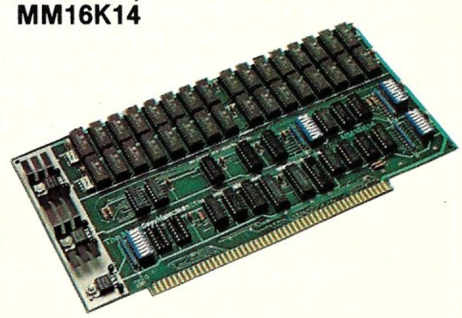
FULL TWO-YEAR WARRANTY.

The reliability of our boards, through quality-controlled production and proven performance, has enabled us to extend our warranty to a full two years. That's standard with us, not an option. This includes a 6-month exchange program for defective units.

Shipped direct from stock.

All Memory Merchant's boards are shipped direct from stock, normally within 48 hours of receipt of your order. Call us at (415) 483-1008 and we may be able to ship the same day.

16K RAM, Model MM16K14



16K STATIC RAM $169.

16K × 8 Bit
Bank Select & Extended Addressing
Four independently addressable 4K blocks
One 4K segment equipped with 1K windows
Uses field-proven 2114 (1K × 4) RAMS
Low Power (less than 1.2 Amps)
Runs on any S-100 8080, 4 MHz Z-80 or 5 MHz 8085 system.

Prices, terms, specifications subject to change without notice.

*Cromix and CDOS are trademarks of CROMEMCO.
[1] MP/M is a trademark of Digital Research

a file name in the command interpreter field of the password file causes that program to be treated as the command interpreter. When user **'david'** logs in, the program **sh** in the directory **/bin** (or **/bin/sh**) is used as the command shell. When user **'jim'** logs in, the program **/bin/upm** (the CP/M simulator) is run with a number of arguments specifying default drives.

In addition to allowing multiple users, Micronix (like UNIX) allows programs to run simultaneously. As an example, if you are compiling a large program using Whitesmith's C compiler, it will take quite a while to compile. By compiling the program in background mode (by placing the character '&' at the end of the command line), the C compilation can run while some other task is being executed in foreground.

When a task is placed in background, the system displays the process number and issues a prompt. Later, when the process finishes, a message is displayed.

File protections

In UNIX (and faithfully implemented in Micronix) protections have been set up to allow a user read, write, and execute access to a file. These three access permissions can be granted to the owner of the file (usually the person who created it), a specific group of users, or everyone. By getting a long listing of a Micronix directory entry, those protections can be displayed as shown in Figure 3.

```
% ls -l
rw-r-----    1 david      7680 Jun 16 15:38 a.txt
rw-r-----    1 david     16640 Jun 16 20:22 b.txt
rwxr-xr-x    1 david     20322 Jun 19 10:14 rsieve
% []
```

Figure 3. *File protection modes.*

In the above example, the **ls** (List) command was used with the long (**-l**) option to show the protection modes assigned to the file. The protection modes for the files **a.txt** and **b.txt** say that I have read and write access to the files; that everyone else in my group can read them but not change them; and that the rest of the world can't touch them at all. On the other hand, the **rsieve** program can be executed by everyone, read (and hence copied) by everyone, but only I can change it. This is a short diagram describing the long display shown in Figure 4.

```
owner  world  owner          date changed
 / \    / \     :             /----+---\
 rwxrwxrwx 1 david    23453 Jun 19 22:14 rsieve
    \ /     :           :                  :
   group    :          size            file name
          links
```

Figure 4. *Interpretation of directory entries.*

Size represents the size of the file. An actual file can have a number of names in directories that all refer to the same file. A new user may want to call the file **/cpm/ws** 'WordStar' and locate it in his own directory. He can make a "link" which will link the name 'WordStar' with the file **/cpm/ws**. The 'links' entry in the file description above shows the number of links to a file. This, too, is available in Micronix as well as in UNIX.

I/O Redirection and pipes

UNIX is often lauded for its I/O (input/output) redirection capability. Usually, when a program is run, some output appears on the screen of the terminal and some input is received from the keyboard. By redirecting the input or output of a program, the information normally destined to the "standard output" of the terminal screen may be sent to a file or some other device. The same applies to input; input normally taken from the "standard-input" (keyboard) may instead be taken from a file.

In addition, to redirect I/O from terminals to files (and vice versa), UNIX allows the user to "pipe" the output of one program into the input of another.

Micronix, like UNIX, supports I/O redirection and pipelines. In order for these mechanisms to work, UNIX and Micronix treat all I/O similarly. The input from a device is the same as that from a file or from a program. The same is true for output. This feature, though difficult to get used to, is one of the most powerful features of UNIX. With I/O redirection, many small "tools" can be built to accomplish specific tasks such as sorting a list or counting lines. These tools can be used together as building blocks with the help of redirectable I/O, creating many new powerful tools.

Shell programs

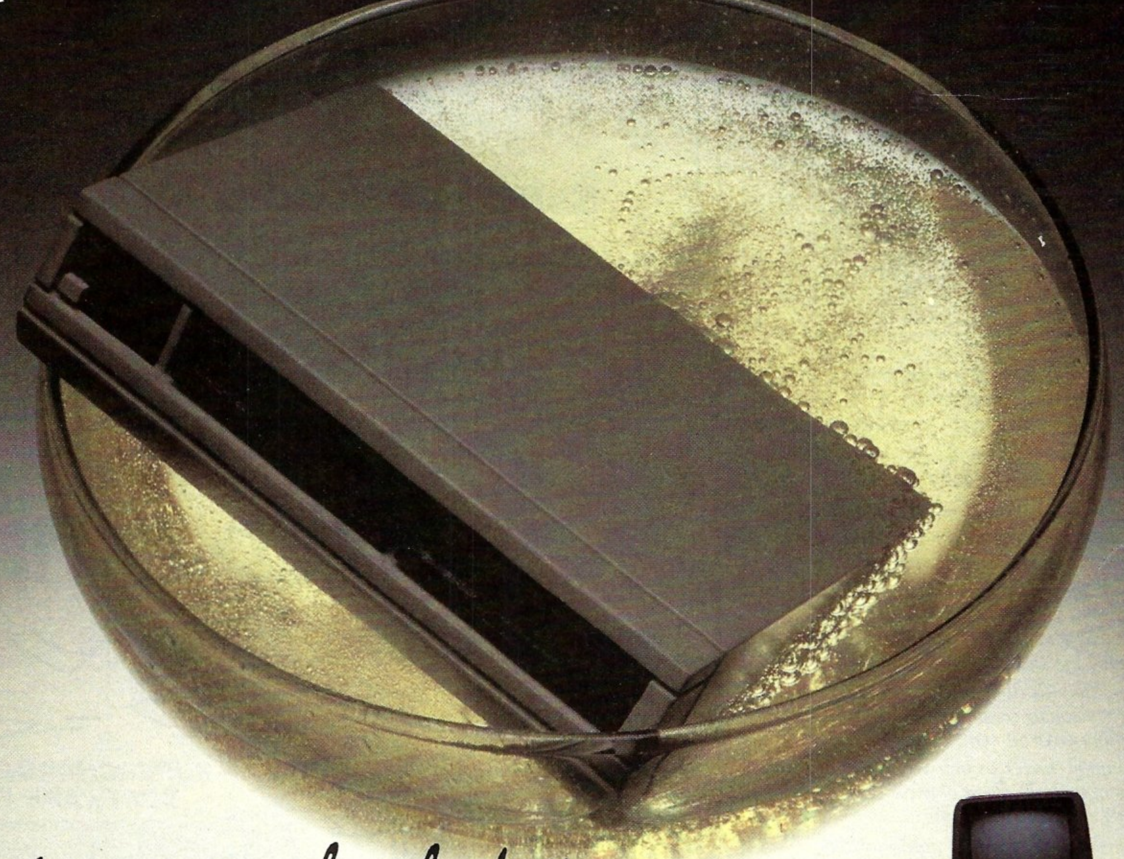
We saw before that the user can interact with Micronix through a shell program. There have been two traditional shell programs available to UNIX users, the Bourne Shell (**/bin/sh**) and the Berkeley C Shell (**/bin/csh**). Micronix has its own shell, which provides many of the features of both UNIX shells while leaving out some features that may be even more important.

The interactive capacity of the Bourne shell allows the user to redirect I/O, use pipes, execute background tasks and execute commands with arguments in a way similar to CP/M. These capabilities exist and work quite properly in the Micronix shell. The C Shell enhances the Bourne shell by allowing a complex "history" mechanism whereby previous commands can be recalled, modified, and then re-executed. Micronix can recall a previous command (though not in the same way as the C shell), but does not allow modification. It also doesn't recall commands in the same way as the C shell. The C Shell also has a powerful **alias** command with argument substitution, which Micronix duplicates (minus the argument substitution).

Small programs can be built from commands placed together in a file. These programs are called "shell programs" or "shell scripts." Much of the power of shell programs comes from the control constructs found in the Bourne and C shells but missing from Micronix. Also, Micronix appears to have a limit to the number of lines the shell will process from a shell program. A number of the benchmark shell programs I wrote had to be split up into more than one file after about 25 lines so they could run.

Micronix does offer two additional shells that complement its programmer (UNIX-like) shell. One is **upm**, the program that simulates CP/M. The other is a "menu-

Micronix, as its name implies, is a micro version of the UNIX operating system.



shell," which provides an easier interface to the beginning user by allowing him to choose the action to be performed by selecting a numbered item from the menu. I have seen only a preliminary version of the menu shell; the market version may behave slightly differently.

Micronix software tools

Micronix has a number of software tools available to the programmer, though fewer than UNIX.

Some UNIX tool/commands are slightly different on Micronix. The UNIX **chown** (change owner) doesn't exist. Instead, the command **owner** is used, with a slightly different syntax. A Micronix program called **far** accesses floppies. Most UNIXes with floppies use a program called **ter**, again with a slightly different syntax.

C and Pascal

Micronix, like UNIX, uses C-based system calls. Morrow offers Whitesmith's C and Pascal compilers for $500 above the basic Micronix cost.

Documentation

Micronix comes with both hard copy manuals and on-line documentation in the form of file-based manual entries. The content of the Micronix manual is an interesting cross between incomplete tutorial and hacker-terse. The first section of the manual, called "Installation and Operation," describes booting up, configuring the terminal (**/etc/ttys**), printer configuration, and setting the time. The next section is 12 pages of "Maintenance and Administration," describing in a reasonably clear manner such things as message of the day, terminals, custom banners, login records, passwords, group accounts and file ownership.

Following this is a section called "Tutorials" (all 16 pages' worth). It describes logging in, passwords, logging out, mail, and so forth. The interesting thing is that you have to know all of this stuff in order to deal with the previous section. Next, in "Tutorials," is a relatively clear section on **upm**, the Micronix CP/M emulator. Next are three unclear pages on mounting devices (some device references are not accurate).

Unfortunately, these three manual sections do not contain enough information for the UNIX beginner to gain anything from them, nor will anyone with a good understanding of UNIX find them helpful.

The remaining sections of the manual follow what has become the UNIX cryptic standard. Each command is listed in alphabetical order with a short usage description. These pages make just enough information available to use them, and no more.

UPM: The Micronix CP/M emulator

Upm runs CP/M programs. **Upm** will run nearly all CP/M programs, except those which direct I/O to processor ports and those which execute interrupts. Most commercially available CP/M programs will work, but some public domain programs, such as the MODEM program, will not work because of the direct I/O traps.

Upm executes CP/M programs "interactively" and "directly." Interactive **upm** looks and acts like CP/M. Giving the command **upm** to Micronix will cause an interactive **upm** to start up:

```
% upm
Morrow Designs upm 3.2
55361 Bytes free

A: -> ./

A>[]
```

Micronix uses an ingenious strategy to enable the Micronix file system of named directories to work with the CP/M file system based on floppy disks. A CP/M disk is equated to a Micronix directory, so that in the above diagram, drive A is actually the current directory. (UNIX uses the symbol ./ to refer to the current directory).

Protections are enforced when trying to log into a directory in which the user does not have write access.

Micronix commands can be executed directly from interactive **upm**, though they tend to be rather slow. If an exclamation point is the first character on the line, anything following it will be treated as a command to Micronix. Likewise, CP/M programs can be executed directly from the Micronix shell.

Overall, I was highly impressed with the linkage between Micronix and the CP/M simulator. It is clear and straightforward, and very easy to get used to.

Upm can be executed in direct mode with command line arguments to set up drive/directory relationships and start up programs. The command line "**upm a:/cpm b:./ b: a:ws**" will assign drive A to '**/cpm**' and drive B to '**./**;.' It will then enter **upm**, log in drive B, and execute **A:WS.**

Upm is also supposed to handle output to the list device. The command argument "LST:!/bin/1pr" will cause list output to be piped through the program **/bin/1pr**. Try as I might, I've never been able to get this particular part of **upm** to work. Instead, I've had to move the text files to be printed to an 8″ floppy, then print them on Pandora, my S-100 computer system.

Upm is a close simulation of CP/M, but it's not perfect. For example, **upm** doesn't support multiple user numbers and the CP/M USER command. CP/M's **STAT** program doesn't work; **upm** has its own version of **STAT**, which shows the amount of disk space left. Unfortunately, it doesn't handle **$sys** and **$r/o** files. It also doesn't support other of the **STAT** options including **VAL:** and **DSK:.**

When **upm** is executed, Micronix "forks" a new process, so for each copy of **upm** running there are at least two processes. When two users are typing into WordStar, they can easily out-type the program by entire words. There appears to be a 25-50% reduction in response time when the second user logs into Micronix and powers up **upm.**

UNIX compatibility

Because UNIX has been around for years, it has evolved into many different versions.

For quite some time, the most common Bell version available was called UNIX version 6; Micronix claims to

By simulating UNIX and CP/M, Micronix offers the best of both worlds for the Z80 processor.

be version 6 compatible. Unfortunately, version 6 is rather old. In doing this review, I was unable to find a version 6 system to compare with Micronix.

In examining Micronix, it would be foolish to consider discarding it because it is not compatible with the latest versions of UNIX. Micronix is a Z80 version of UNIX, and as such should be treated with a great deal of reverence. The questions I asked were: How close to the "spirit" of UNIX is Micronix? Does it act the same? Are directories and files treated the same way? Are the shell and shell commands similar? Do most of the tools behave the same way? The answer to all of the above questions is an unqualified *yes.* Micronix is very similar to UNIX and is quite faithful to the "spirit" of UNIX.

Benchmark Tests

A set of simple benchmarks was put together to get a feel for the speed differences between UNIX and Micronix.

The Processor Intensive Tests consisted of register- and nonregister-based versions of the Sieve program originally printed in *Byte* magazine. The Disk Intensive Tests created a large file, printed the last few lines of the file, and sorted the file. [*Editor's note: Lack of space prevented inclusion of the programs and shell scripts for the benchmarks. They are available on request from Microsystems.*]

To test the multiuser capabilities of Micronix, the concurrent tests were run. The most arduous test of all was running two concurrent sorts simultaneously.

The final set of tests (Operating System Intensive Tests) was used to see how much overhead existed in the operating system. Since nearly all programs running under an operating system use system calls, a test such as this, which examines how long the operating system takes in a call, is quite valid.

These tests were performed on the Decision 1 running Micronix (hardware cost about $6500), a Digital Equipment Corporation VAX 11/750 (no users logged in; hardware cost about $250,000), and a representative 68000-based UNIX System III system (hardware cost about $20,000) using Sun memory management architecture running UniSoft's UniPlus+. Both the Micronix system and the UniPlus+ 68000 system used the same type of 5¼" mini-Winchester hard disks, while the VAX used very fast Fujitsu disks.

All tests were run using the identical set of shell scripts. Timing was accomplished by executing the program **date** before and after each test. To ensure that concurrent processes had finished, the program **wait** was used before **date,** so that all processes finished running before the time was printed. All tests were run with only one user logged in and each machine in multiuser mode.

As can be seen from Table 1 (benchmark execution times), Micronix is quite a bit slower than both the VAX and the 68000 system. Although this was to be expected in the processor intensive tests, it is something of a surprise to see such a difference in speed on the disk intensive tests between the 68000 system and Micronix.

Conclusions

Micronix is quite faithful to the "spirit" of UNIX and allows most CP/M-based programs to run under its CP/M simulator **upm**. In most cases, the speed degradation normally found in a multiuser system was not terribly severe, and the system functioned as expected in a multiuser environment.

Micronix is an operating system that will live the life of the Decision 1 computer. As a method of allowing multiple users to run CP/M-80 software simultaneously while making some of the better features of UNIX available, Micronix is excellent. As an engineering feat, I respect the accomplishment of Gary Fitts (the guiding force behind Micronix) and the others at Morrow. Micronix is a shining example of what fine engineering talent can produce to run on the tiny Z80 processor.

Editor's note: A beta test version was the subject of this review. The current version (1.6) incorporates many corrections and enhancements, some of which address the problems noted in Dave's review.

TABLE I

TEST	VAX 4.1bsd	68000 UniPlus+ System III	Decision One Micronix
CATEGORY 1			
Sieve	00:00:06	00:00:11	00:00:35
Register Sieve	00:00:04	00:00:07	00:00:28
CATEGORY 2			
Largefile Creation	00:00:05	00:00:21	00:01:05
Tail of Largefile	00:00:00	00:00:04	00:00:44
Sort of Largefile	00:00:38	00:00:58	00:27:19
CATEGORY 3			
2 Conc. Rsieve	00:00:07	00:00:14	00:00:53
3 Conc. Rsieve	00:00:10	00:00:19	00:01:24
CATEGORY 4			
2 Conc. Sorts	00:01:06	00:33:04	00:54:28
CATEGORY 5			
Function Calls	00:00:19	00:00:24	00:00:42
System Calls	00:02:50	00:02:49	00:15:43

With I/O redirection, "small" tools can be created and used as building blocks for powerful new tools. Micronix is very similar to UNIX and is faithful to the "spirit" of UNIX.

Upgrading Older S-100 Computers to the CompuPro Dual Processor

by David W. Bray

Upgrading your 2 MHz S-100 computer to a 16-bit processor may not be as difficult as you might think. This article describes how to upgrade an IMSAI 8080 (and other S-100 computers) that has a 2 MHz 16K-chip Expandoram I memory board to incorporate a CompuPro 8085/8088 dual processor board. This modification allows the 8085 to run at 4 MHz and the 8088 to run at 5 MHz. This article also discusses a simple memory management technique that allows the 8088 to have its reset vector at FFFF0 and a small amount of RAM memory at that location. With this modification the S-100 computer can be expanded to any desired amount of memory without concern about memory allocation for the 8088 reset vector.

My computer is an ISMAI 8080 with an Expandoram I (64K), a homebrew I/O board, a North Star double-density disk system; it had the original IMSAI 8080 processor board. I wanted to upgrade the processor. The obvious choice would have been a Z80 board, but with all the activity in 16-bit processors these days it somehow didn't seem right to upgrade to a faster processor that was of an old vintage. So I decided that I would purchase the CompuPro dual processor board. I knew I was taking a big chance that the Expandoram I would not work with the 8085/8088. I figured if worst came to worst, I would have to buy a new memory board. Murphy's Law prevailed: after purchasing the board, my worst fears were realized. The dual processor board did not work with the Expandoram I. However, there is a happy ending to the story.

The CompuPro dual processor board has both an 8085 and an 8088 processor. On power-up the 8085 processor is activated, and the 8088 is in the hold state. The user can switch to the 8088 by the execution of an IN instruction. This will activate the 8088 within four clock cycles. The IN instruction actually causes the processor board to toggle between the two processors.

The processor board allows the user to select the speed of the 8085 to be either 2 or 8 MHz. The 8088 is always running at 8 MHz. Since the board is switched to the 8085 processor on power-up and I could run it at 2 MHz, I felt that there was a chance that I could modify the Expandoram I to get it to run at at least 2 MHz with the 8085. That is what I did first.

Expandoram modifications

My Expandoram I was an early Revision A board. After some research I found that later revisions of the board had modifications to allow jumper changes for an IMSAI 8085 processor. Having made these modifications, the Expandoram I did work at 2 MHz with the 8085 processor. For those readers who have an early version of the Expandoram I, these modifications are shown in Figure 1.

Although that was a beginning, I still had some difficulties to overcome. The North Star disk interface is memory mapped and uses the READY bus signal to put the processor in a wait state until it has obtained data. The Expandoram uses the S-100 WAIT signal (pin 27) to recognize that the processor is in a wait state and therefore must refresh the memory by its own means. This is all well and good except that the IEEE-696 standard, which the dual processor board uses, does not define the WAIT bus signal. Therefore, it is necessary to create such a signal external to the dual processor board. This is easy to do, and a circuit for creating WAIT is shown in Figure 2. The two ready signals PRDY and XRDY must both be sensed, since the IMSAI front panel uses XRDY to indicate that the front panel is holding the processor in a wait state. The signals pWR* and pDBIN* synchronize the ready signals so that the generated wait signal is nearly the same as would have been produced by the 8080. The signal pDBIN* (the inversion of pDBIN) is generated in the memory management circuit discussed below. I built this circuit on an S-100 wirewrap board, along with the memory management circuit.

The next challenge was to get the 8088 working. Careful study of the timing diagram of the dual processor board revealed that if I could get the memory to work at 4 MHz, then it was likely that the 8088 would work at 5 MHz. Through some more research (i.e., reading a lot of adver-

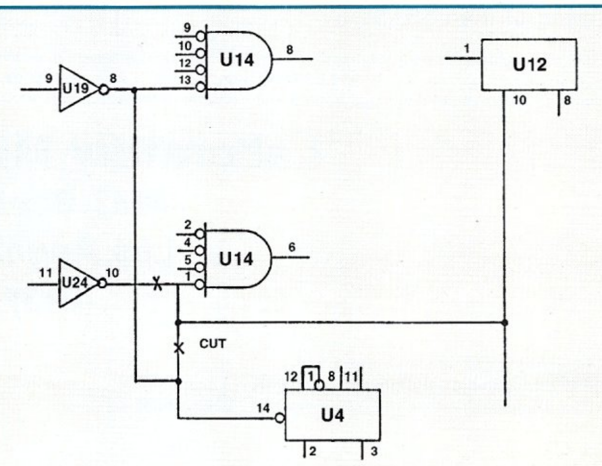


Figure 1. Expandoram modifications.

Dr. David W. Bray, Dean, Educational Computing System, Clarkson College of Technology, Potsdam, NY 13676.

Table 1. IMSAI nonstandard S-100 bus pins

Pin	IEEE	IMSAI	Implementation
20	GROUND	UNPROTECT	Do not remove GROUND from 85/88 board; disconnect pin 20 on front panel
21	NDEF	SINGLE STEP	Front panel produced
24	Φ	Φ_2	Φ from 85/88
25	pSTVAL	Φ_1	sSTVAL from 85/88
27	Reserved	PWAIT	WAIT circuit produced
28	Reserved	PINTE	**GROUND**
53	GROUND	SENSE SWITCH DISABLE	Not used
55	DMA0*	CHASSIS GROUND	Remove GROUND
66	NDEF	Unused	Special Phantom
69	Reserved	PROTECT STATUS	Not used
70	GROUND	PROTECT	Front panel GROUND
71	Reserved	RUN	Front panel produced
98	ERROR*	SSTACK	**GROUND**

NDEF = Not to be defined.

tisements) I found that J.E.S. Graphics of Tulsa, OK offered a kit to upgrade the Expandoram I to 4 MHz for $10. I ordered the kit and installed it in about 10 minutes. I then charged the crystals on the dual processor board to 8 MHz for the 8085, and to 15 MHz for the 8088, resulting in 4 MHz and 5 MHz processor speeds respectively. Much to my pleasure the old ISMAI was up and running with the dual processor board. The J.E.S. Graphics kit warned that it might not work without additional modifications, and they gave hints as to how to get the memory board to work if it didn't work right away. Mine worked the first time (where was Murphy?), and it has been running reliably now for over six months.

The IMSAI front panel

The CompuPro processor board has provisions to work with the IMSAI (and other) front panels. The IMSAI front panel does work with the modified Expandoram and the dual processor board with one exception. The deposit feature will not work. To obtain a reliably operating front panel at 4 MHz, a small modification to the front panel itself was necessary. When the processor board was running at 4 MHz, sometimes when I would enter an address with the EXAMINE switch, the actual address latched by the panel would be one number greater than the one I entered. This was fixed by inserting a 100 ohm resistor on the front panel between pin 9 of U13 and pin 6 of U15. (*Note: the resistor must be close to pin 6 of U15 so that it is on the U15 side of the wire before it connects to C8.*)

To complete the modification, there are a few S-100 signal connections that should be attended to, to be sure the panel is enabled and to make the front panel lights look about the same as before. Table 1 lists S-100 bus definitions of the IEEE standard (and therefore the dual processor board) that differ from that of the IMSAI. It also lists the bus signals as they are implemented. Those marked with ** should be done to make the front panel operate and to have the lights as they were. These ground (or removal of ground) connections simply provide signals that the IMSAI front panel really would like to have which the IEEE-696 standard does not provide. Except for the removal of the ground from pin 20, they are not necessary but probably are worth the small amount of effort, since they can be placed on the same board as the WAIT signal circuit.

Memory management

There is a small problem in managing memory for the two processors if more than 64K of memory is to be used. The 8088 processor will address 1 megabyte of memory with its 20-bit address bus. The 8085 will addresse 64K with its 16-bit address bus. The IEEE-696 standard specifies a 24-bit address bus that will address 16 megabytes. To coordinate these differences in addressing capabilities the CompuPro dual processor board has a simple memory management register that allows the user to latch the high-order 8 bits of the IEEE-696 24-bit address bus when the 8085 is in use, or to latch the upper 4 bits when the 8088 is active. At first thought this seems like a nice solution, allowing the user to incorporate as much memory in his or her computer as desired. However, there is a problem with this. Before we discuss this in detail, we need to understand how the 8085 and

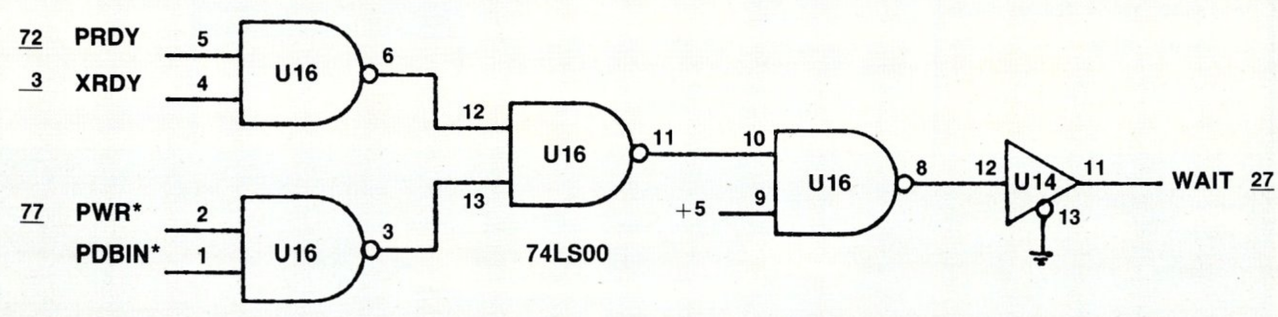


Figure 2. Wait signal.

with *LOW, LOW* Prices

CALIFORNIA RESIDENTS SAVE 6% SALES TAX

8 YEARS (1976) EXPERIENCE IN COMPUTER MAIL ORDER BUSINESS

PRINTERS

DIABLO
620 (25CPS/SERIAL) 940.00
630 (40CPS/MULTI-IF) 1,925.00

C, ITOH
Pro-writer I (8510A) Par 120 CPS 365.00
Pro-writer I (8510A) Serial 120 CPS 529.00
Pro-writer II Parallel-15" 669.00
8600 (180CPS) Par or Serial-
18 PIN 90 CPS LTR 1,099.00

JUKI
6100-18CPS/Diablo Compatible Par 565.00

NEC
NEC7715 (w/Diablo Emulation) 2,075.00
NEC8023A (100CPS-Par-Graphics) 419.00

MANNESMAN-TALLY
160L (160CPC-40CPS LTR 10") 635.00
180L (160CPS-40CPS LTR 15") 869.00

OKI-DATA
Microline 82A (SER & PAR-120CPS 10") 399.00
Microline 83A (SER & PAR-120CPS 15") 629.00
Microline 92 (PAR-160CPS-LTR-10") 519.00
Microline 93 (PAR-160CPS-LTR-15") 899.00
Microline 84P (PAR-200CPS-LTR-15") 969.00
Microline 84S (SER-200CPS-LTR-15") 1,059.00

STAR MICRONICS
Gemini 10 (new version)
(PAR-100CPS-10") 299.00
Gemini 15-(15" carriage) 439.00

SILVER REED
EXP 55OP-17CPS Daisy Wheel-PAR 670.00
EXP 550S-17CPS Daisy Wheel-Serial 690.00

TOSHIBA
P-1350 (192CPS-120CPSLTR
PAR or Serial) 1,500.00
P-350 (100 CPS-PAR-15") 399.00

PRINTER ACCESSORIES

DIABLO
Diablo 620-Uni-Direct Tractor 110.00
Diablo 620-Sheet Feeder 720.00
Diablo 630-Bi-Direct Tractor 275.00
Diable 630-Tractor Sound Cover 29.00
Diablo 630-SGL Sheet Feeder 775.00

JUKI-6100
Serial Interface 55.00
Bi-Dir Tractor 125.00
Uni-Dir Tractor 90.00

MANNESMAN-TALLY
Factory Quick Tear 230.00
Auto Front Feed 1,150.00

NEC
Thimbles 15.00
Vertical Tractor 219.00
Horizontal 152.00
Bi-Dir. Tractor 330.00
Cut Sheet Feeder 1,100.00

OKI-DATA
84 Cut Sheet Feeder 515.00
82/92 Tractor 55.00
Serial Intf. w/2K Buffer (For 82A) 125.00
Serial Intf. w/2K Buffer (For 92 & 93) 110.00
Okigraph I 82A or 83A Graphics ROM 35.00
Okigraph II 82A or 83A Disk for Apple 55.00

STAR-MICRONICS
Serial Intf. Bd 70.00
Serial Intf. Bd w/2K Buffer 120.00
Commodore 64 Intf. Bd. 72.00

TOSHIBA
P-1350 Cut Sheet Feeder 1,125.00
P-1350 Bi-Dir. Tractor 250.00

APPLE/FRANKLIN ACCESSORIES

ASTAR
RF Modulator 19.00

ORANGE MICRO
Grappler & Graphics Interface 121.00
Grappler & 16K Buffer Exp. 179.00

TG PRODUCTS
Joy Stick 39.00
Paddles 29.00

QUADRAM
E-RAM 80 (80COL64K) 135.00

PRACTICAL PERIPHERALS
Microbuffer II 16K PAR or Serial 230.00
Microbuffer II 32K PAR or Serial 259.00
Snapshot 99.00

EVERTEK
EV16K (16K Add-on Memory) 49.00

MICROTEK
Dumpling 64K-D 235.00
Dumpling GX-P 75.00
PAR BD 61.00
Memory 128K 299.00

VISTA
A800 Floppy Controller for 8" Drives 300.00
A-800-1 Cable 27.00

QT/COMPUTIME
CCA-A Clock/Calendar 100.00

OUR SALTIEST DEALS

IBM Add-On Drive
* Direct Drive *
* Quiet (Teac Equiv.) *
*320 K/48 TPI *
* 5ms Track to Track *
REMEX RFD 480 .... **239.00**

S-100 BOARD SET
Z80CPU (Big Z)/64K Mem/Floppy
Controller (Double D) **150.00**

IBM ACCESSORIES

MAYNARD
Floppy Controller 155.00
Floppy Controller (Serial) 225.00
Floppy Controller (PAR) 205.00

QUADRAM
Quadboard 64K 289.00/459.00
Quadboard II 64K/256K 289.00/459.00
Quad Link (Allows IBM PC to use Apple II
Software) 589.00

AST
Combo Plus (SPC 64K) 269.00
Mega Plus (SC, 64K) 269.00
I/O Plus (SC) 114.00

KEYTRONICS
Enhance your PC-with a superior keyboard .... CALL

64K UPGRADE KIT
$48.00

MICRO-TEK
IBM 64K w/Parity 239.00
IBM 128K w/Parity 369.00
IBM 256K 639.00

MULLEN COMPUTER

TB-4A Extender BD w/Logic Probe **80.00**
ICB-10 8 Channel Low Voltage
Controller BD **180.00**
ZB-1 ZIF Extender Test BD **130.00**

SYSTEMS

Apple II E Starter **1,625.00**
Eagle PC (16BIT) (1-320 K FD
64 RAM) **1,795.00**
Franklin 1000 .... **919.00**
Franklin 1200 Starter **1,625.00**
Kay Pro II **1,495.00**
Sanyo MB-100 .... **1,589.00**
Televideo TS-803 .... **1,995.00**
Toshiba T-100 .... **1,895.00**

TRY US FOR OTHERS

DISK SUB ASSEMBLY

OUR BEST BUYS

8" SUB ASSEMBLY
DDS + O 2EA SS/DD Siemens
FD100-8 Drives w/Cabinet **595.00**
DDS + 2 2EA DS/DD Mitsubishi
M2894-63 w/cabinet **1,075.00**
DDS + 4 2EA DS/DD 8" Thinline
Drives w/Cabinet **1,150.00**
Specify—Vertical or Horizontal Cabinet

5¼" SUB ASSEMBLY
DDS + 5 1EA SS/DD Shugart
SA400 Drive **200.00**
DDS + 6 2EA SS/DD Shugart
SA400 Drive **369.00**

TERMINALS

TELEVIDEO
TV910 .... **560.00**
TV925 .... **715.00**
TV950 .... **925.00**
TV970 .... **1,095.00**
RG1000/TV60 Graphics
Upgrade for 925/950 .... **1,100.00**

WYSE
WYSE-100 .... **725.00**
WYSE-300 (Color) **1,125.00**

MasterCard

Retail Sales
123 East 200 South
Salt Lake City, Utah 84111

The Great Salt Lake Computer Company, Inc.
1-801-363-3314

Mail Orders
P.O. Box 3150
Salt Lake City, Utah 84110

VISA

CIRCLE 39 ON READER SERVICE CARD

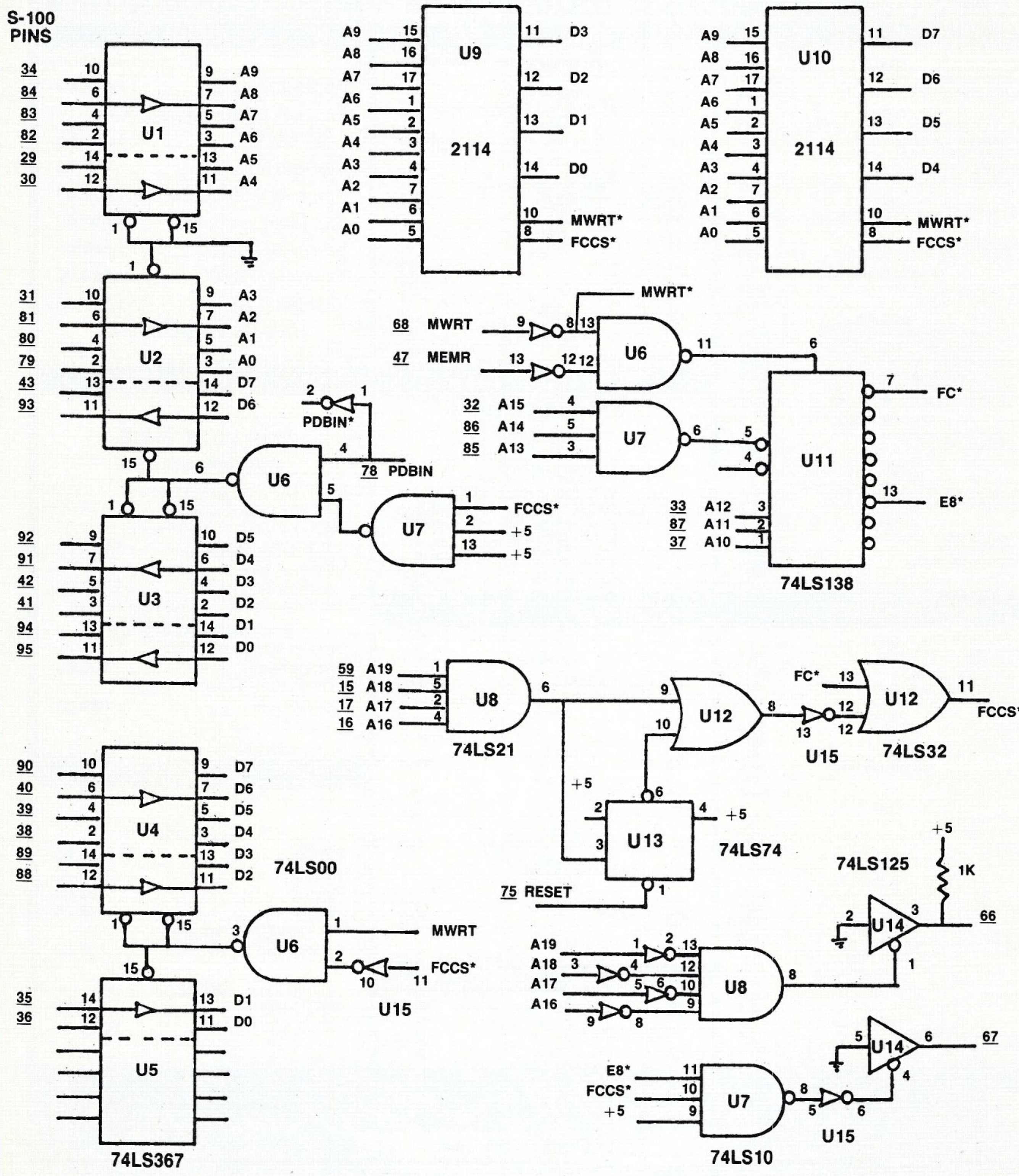


Figure 3. Memory management.

8088 go about initializing themselves at power-up.

The 8085, upon power-up, clears its internal program counter to zero and then begins executing the instructions found at that address. Most systems have a circuit (or ROM at memory address 0) that will provide the necessary instructions to boot the computer up from disk or ROM. On the other hand, the 8088 expects to find its initial instructions at memory address FFFF0. These are the very last 16 memory locations of the 1 MB of memory that it will address.

The dual processor board has a special power-up circuit that activates the 8085 at power on, and then upon user request, as mentioned above, it will deactivate the 8085 and activate the 8088. The user can select whether the newly activated processor is to continue from the last address executed when it was deactivated or whether it is to restart as if it were powered up each time. No matter which option is chosen, the very first time each processor is activated it must start at its predefined memory address. If there is only 64K in the system, there is no problem. The 8085 will start at address 0, as is the case of any 8080, 8085, or Z80 system. The 8088 will send out the address FFFF0, but since there is only 64K, the high-order F will be ignored, and the actual address accessed for initial instructions is FFF0. Since the 8085 must have been previously activated, part of its initializing sequence could be to place the 8088's initial instructions at FFF0. In case there is no memory at FFF0, and a ROM must be placed there to get the 8088 going.

A problem exists if there is more than 64K of memory in the system. In this case it would appear, at first thought, that there must be a ROM at FFFF0, since it is not possible for the 8085 to place instructions for the 8088 anywhere outside of its 64K address space. The 8085 cannot put instructions at FFFF0, since if an address past 64K is created by setting the high-order 8 bits of the 24-bit address with the memory management register, the 8085 will be accessing memory that does not contain any instructions. Setting of the high-order bits in effect switches the memory bank. The switch applies to instructions as well as data to be written to memory.

There is a solution. Rather than placing a ROM at FFFF0, Figure 3 shows a circuit that allows 1K of RAM to be located from 0FC00 to 0FFFF upon power-up and therefore accessable by the 8085. As soon as the 8088 is activated, this 1K of RAM is switched so that it is located from FFFC0 to FFFFF. It remains at that location until the computer is powered down. The advantage of this over a ROM is that the memory below FFFF0 can be used as a stack for the 8088. It is a large stack space and is far from conflict with any other memory in most systems. The circuit of Figure 3 not only provides this 1K of memory and causes this memory address change, but also produces a signal to disable the Expandoram I when memory greater than 64K is accessed.

Referring to Figure 3, upon power-up the 1K of RAM overlays the normal memory at 0FC00 because the decoder U11 produces a low signal at FC* and, because of U13, a low signal at FCCS* whenever that address is accessed. During this access the Expandoram memory at 0FC00 is disabled by the signal from U7, causing the memory phantom to be (low) active. However, after the 8088 is activated, the full Expandoram 64K is available because the signal from U8 pin 6 sets U13, which allows FCCS* to be low only when the address range is greater than F0000. Thus the 1K RAM then becomes accessable only from FFC00 to FFFFF as determined by U11. When any address outside of 00000 to 0FFFF is accessed the Expandoram must be disabled. Since it does not use the new S-100 standard, a signal must be sent to it to cause it to be disabled. This is accomplished by creating a disable signal placed on S-100 bus pin 66 (nonstandard). The circuit of U8/U14 creates this extended address phantom. To allow the Expandoram to use this signal, one more change to the Expandoram is required. The Expandoram I decoder U7 has pin 6 tied to +5 volts. Disconnect this and connect pin 6 of U7 to pin 5 of U19 (which is connected to S-100 pin 66). The phantom memory signal of U7/U14 goes low, disabling any memory controlled by it whenever the memory at FFFC0-FFFFF is active, or on power-up the same memory at 0FFC0 is active, or whenever the North Star disk is active (E800-EBFF). Be sure that the Expandoram board phantom signal is connected to S-100 pin 67, since it is required.

The notation used in Figure 3 shows S-100 bus pins underlined; e.g., <u>78</u> is pDBIN. In constructing the circuit, all leads with the same name are connected together. For example, U1 pin 1, U9 pin 15, and U10 pin 15 are labeled A9 and are therefore connected together. The same is true of U3 pin 10, U14 pin 3, and U10 pin 13. Table 2 lists the ICs required for the circuits of Figures 2 and 3.

Table 2. Integrated circuits required

2 - 74LS00	U6, U16
1 - 74LS04	U15
1 - 74LS10	U7
1 - 74LS21	U8
1 - 74LS32	U12
1 - 74LS74	U13
1 - 74LS124	U14
1 - 74LS138	U11
5 - 74LS367	U1, U2, U3, U4, U5
2 - 2114	U9, U10 (Static memory)

Conclusions

The addition of the dual processor board was well worth it. The cost of a Z80 board would have been nearly as much, but with much less ultimate capability. The conversions were easy, and the old IMSAI moves right along at 4 MHz. It is a real pleasure to get computations done in one-half the time of the old processor. I haven't used the 8088 very much yet, but I do know that it works reliably. The only software that I have developed for the 8088 is a monitor, similar to the North Star Monitor. It is very handy, since I can look at or modify any memory location in its 1 MB address space. I am sure that as time goes on the 8088 will see more action.

David W. Bray is Dean of the Educational Computing System at Clarkson College of Technology, where he oversees the Clarkson personal computer project. He is engaged in research for both the Rome Air Development Center and the General Electric Company in the design and simulation of high-throughput computer systems and the development of fault detection software aids. Prior to joining Clarkson College, Dr. Bray worked for GE on designing special-purpose computer languages and other projects.

A MICROSYSTEMS TUTORIAL

Relocating Assemblers and Linkage Editors: Part 1

by Andrew L. Bender

Assemblers that generate relocatable programs are becoming increasingly available. The purpose of this tutorial is to explain why you should consider using these assemblers and when to use them, as well as exactly how these assemblers work. Also considered is how the binary output of these assemblers is used by a linkage editor, and how to develop software with these programs. I assume that the reader has had some contact with assemblers, but I do not assume that he or she knows how an assembler works, and so that topic will be discussed as necessary. For the sake of appealing to the greatest number of readers, I will use the CP/M ASM assembler as a model of an absolute assembler and Microsoft's M-80 assembler as a relocating assembler.

Relocatable programs consist of modules, all of which start at location 0000H, and in which all branch addresses point to targets that are inside the module or are symbolically defined as "external." A linking editor can then concatenate any number of such modules, in any order, using information contained in the modules to change symbolic, relative addresses into absolute addresses. This procedure not only allows a programmer to build up a library of functional subroutines for performing certain tasks, but also allows programmers to share these subroutines with one another without regard to how they operate internally. Once written, properly documented, and tested, a program can be put into the library so that it can be obtained anytime it is needed. If the user needs an ASCII-to-binary conversion program, it would be necessary only to look up the library catalog to see how to use it, and then incorporate a call for this program into the program that needed it. The program library avoids the need for constant "reinventing of the wheel." Smaller, faster assembling or compiling and better understood programs result when all components are written as individual, functional, subprograms.

Relocatable binary modules may be used with other relocatable binary modules as long as they are all in the same relocatable binary format. Thus, relocatable binary modules generated by a compiler may be linked not only with each other, but also with the output of assemblers and with precompiled or assembled library subroutines. This allows a certain degree of interchange of programs at the relocatable binary level, so that several different programs may be connected together and used together. The programmer can write most of an application in a high-level language such as Fortran or PL/I, and those sections which required better access to the computer facilities could be written in assembly language. All of the resulting relocatable binary modules could then be linked together by a linkage editor to produce an absolute binary load file or load module. This load module or absolute binary load file is the familiar .COM file in CP/M.

Andrew L. Bender, M.D., Neurological Services, Inc., 336 Center Avenue, Westwood, NJ 07675

It is unfortunate that the trend for everyone to go his own way in software has not been better controlled. When Microsoft prepared the MITS-DOS program (in late 1976 to mid-1977), they designed a relocating assembler and linking loader that were the prototypes for M80 and L80. All of Microsoft's software that produced relocatable output did so in this format.

At about the same time, Computer Design Labs (at that time called Technical Design Labs) had gone their own way in designing a relocatable binary format that was not interchangable with that of Microsoft. This meant that Microsoft software could not be intermixed with CDL software at the relocatable binary module level. Digital Research did make the output of their RMAC assembler and PL/I-80, as well as CB-80, compatible with Microsoft's relocatable binary format. The result of this compatibility is that Microsoft and Digital Research 8080 software can be intermixed at the relocatable object code level. Not to be left out, MicroPro designed their Super Sort package to interface with the Microsoft relocatable format.

Nonetheless, vendors still produce software with other relocatable formats. While the Microsoft format has its problems, these other vendors have not done anyone a service by supplying users with non-interchangeable relocatable formats. The value of being able to interchange software at this level far outweighs most advantages gained by these nonstandard formats. My advice to these vendors is write a converter that will make their relocatable format look like Microsoft's, or one that will make Microsoft's look like theirs. If the vendors don't do it, then you, the user, will either have to write it yourself or be denied the advantage of being able to use the growing library of programs out there.

An assembler that produces relocatable programs is sometimes referred to as a relocatable assembler. This term is a bit ambiguous because it makes us feel that the assembler itself may be capable of being positioned anywhere in memory; but that is not what is meant. A more correct term is "relocating assembler." A relocatable *program* is one which may be positioned anywhere in memory. If the relocating assembler is also a relocatable program, then it can be positioned anywhere in memory—and that is where the ambiguity in the name arises.

Relocation is the ability to position a program anywhere in memory and have it work properly. Relocation may be of two types: hardware relocation and software relocation. To be sure, both types may be mixed so that a program can be relocated both by software and then by hardware.

Let us look first at hardware relocation, because it is easier to understand. Imagine that the address space of a computer has been extended by some sort of hardware register. Although the computer can only directly address 64K of memory (because it has only 16 address lines) there is, perhaps, a page register that points to a particular page on which all 64K of that page may be found. By changing the page address, another 64K page will be addressed, and so on. Because the instructions in the programs written for this computer do not specify which page they are on, but only where on the page they are located, programs can be placed on any page and will run properly as long as they are not moved around on that page. Both large computers and micros make use of this feature, which is called segmentation or paging.

Software relocation is quite different. The principle may be the same, but the method by which it is accomplished is different. The remainder of this installment will deal with the construction of software relocatable programs and the way in which assemblers or compilers produce these programs. The standard assembler, which produces an absolute program—that is, a program that is not relocatable—keeps a table of symbols that it produces by examining the source program. This first examination of the source program is usually called "pass 1." During this pass, the labels in the program are assigned address values by looking at the lengths of each instruction in the program, the size of each storage reservation, the length of constants in the program, and any explicit instructions to the assembler (by means of EQU directives) as to the value to be assigned to a given symbol. As each label is encountered, it is placed in a table together with the explicit or computed value.

After the first pass has been completed, the source program is read again. This reading of the source program is usually called "pass 2." It is during this second pass of the assembly process that the binary output of the assembler is generated. During pass 2, each time a label is encountered as an address or constant in the source code, the symbol table is consulted. If the label is found in the table, the associated value is substituted for the symbolic address or constant where it appeared in the source program. If the label is not found in the symbol table, then the assembler reports an "undefined" error, usually printing the letter "U" next to the offending line of the assembly listing. As a double check, when each label definition is encountered during pass 2, a check is made to make sure that the value of the label during pass 2 is the same as the value at the end of pass 1. If this is not the case, the assembler places the letter P ("phase error") alongside each line containing the label. A variety of other assembly errors might be caused by a phase error. The most common cause of this error is the use of the same label at two different points in the assembly—i.e., multiple definition of the label.

In order to tell an absolute assembler where to position a program in memory, we use an ORG statement. This specifies an address by which all of the addresses in the program are to be biased. Let us look at a sample program to understand relocation:

```
Ø1ØØ                          ORG    Ø1ØØH
Ø1ØØ 312ØØ1       START:      LXI    SP,STACK
Ø1Ø3 11ØEØ1                   LXI    D,MSG
Ø1Ø6 ØEØ9                     MVI    C,9
Ø1Ø8 CDØ5ØØ                   CALL   BDOS
Ø1ØB C3ØØØØ                   JMP    BOOT
Ø1ØE 48454C4C4FMSG:           DB     'HELLO$'
ØØØ5 =            BDOS        EQU    5
ØØØØ =            BOOT        EQU    Ø
Ø114                          DS     12
Ø12Ø Ø2ØØ         STACK:      DW     2
Ø122                          END    START
```

Note that although this is an absolute assembly, all of the addresses of the various operands, with exception of the addresses BDOS and BOOT, are all relative to each other. That means that no matter where this program is ORG'ed, all of the addresses will be the same relative distance apart, and the addresses of BOOT and BDOS will not change. BOOT and BDOS are said to be *absolute symbols.* An absolute symbol is one whose value will not change no matter where the program is placed in memory. The other symbolic addresses are said to be *relocatable symbols.* The address values corresponding to these symbols will change if the program is moved to a new location in memory. Let us reassemble our sample program for a different origin and verify that this is exactly what happens:

```
Ø2ØØ                          ORG    Ø2ØØH
Ø2ØØ 312ØØ2       START:      LXI    SP,STACK
Ø2Ø3 11ØEØ2                   LXI    D,MSG
Ø2Ø6 ØEØ9                     MVI    C,9
Ø2Ø8 CDØ5ØØ                   CALL   BDOS
Ø2ØB C3ØØØØ                   JMP    BOOT
Ø2ØE 48454C4C4FMSG:           DB     'HELLO$'
ØØØ5 =            BDOS        EQU    5
ØØØØ =            BOOT        EQU    Ø
Ø214                          DS     12
Ø22Ø Ø2ØØ         STACK:      DW     2
Ø222                          END    START
```

Notice that all of the addresses in the program except BDOS and BOOT have been increased by 0100H.

The relocatable assembler uses certain information supplied by the programmer to determine which symbols are absolute. Other symbols are considered to be relocatable unless the programmer supplies special information about those symbols that may indicate other special uses or attributes. A relocating assembler places information in the symbol table as to the relocation characteristics of each symbol recorded.

```
ØØØØ'   31 ØØ2Ø'              START:  LXI    SP,STACK
ØØØ3'   11 ØØØE'                      LXI    D,MSG
ØØØ6'   ØE Ø9                         MVI    C,9
ØØØ8'   CD ØØØ5                       CALL   BDOS
ØØØB'   C3 ØØØØ                       JMP    BOOT
ØØØE'   48 45 4C 4C           MSG:    DB     'HELLO$'
ØØ12'   4F 24
ØØØ5                          BDOS    EQU    5
ØØØØ                          BOOT    EQU    Ø
ØØ14'                                 DS     12
ØØ2Ø'   ØØØ2                  STACK:  DW     2
                                      END    START

Symbols:
BDOS    ØØØ5    BOOT    ØØØØ    MSG     ØØØE'   STACK   ØØ2Ø'
START   ØØØØ'
```

Note the quote mark after each relocatable value. Absolute values are not marked, meaning that the assembler recognizes that they should not be changed. As the assembler generates the binary program file from the source program, it adds the relocation information so that the linkage editor can determine which values should be changed and which should not. In order to run the above program under CP/M in a standard system (one "ORG'ed" at 0100H), the value 0100H would have to be added to every address marked with a quote mark. The resulting program would look exactly like our first sample program.

The CP/M utility MOVCPM is a special type of program relocator containing both the program to be moved (a relocatable image of CP/M) and a bit map indicating which bytes should have the relocation bias added to them

Relocation is the ability to position a program anywhere in memory and have it work properly, and it may be of two types: hardware relocation and software relocation.

and which should not. In MOVCPM, it is not possible to move in less than full page increments. The bit map can be constructed by assembling the program once on one page address and reassembling it at an address one page higher. Both outputs are processed by subtracting the programs byte by byte from each other. Every time there is a difference in a page address it will equal one, and this can be output in a table. It is then just as trivial to write a relocator which will move every byte into memory at its correct address by adding the bias to each byte in which the corresponding entry in the bit map contains a one. The reason for using a page increment is related to the addressing hardware of the 808X series microprocessors. Since the 16-bit address registers are composed of two 8-bit registers, it is convenient to organize the address space into 256 pages of 256 bytes each. Because programs are generally longer than one page, from a practical standpoint it is only meaningful to relocate pages. The algorithm for page relocation used in MOVCPM is given by Kildall (1978).

Since computer memory is usually segregated into different areas with special attributes, there has to be a way to deal with the positioning of programs in these different areas. As an example, we might allocate our address space as follows:

LOW MEMORY 00000to 24000 is Data Storage RAM
MIDDLE MEMORY 24001 to 48000 PROGRAM ROM
EMPTY ADDRESS SPACE HERE
VIDEO DISPLAY MEMORY 6000 to 64000 AREA

And suppose this layout represents a fixed widget controller for which we are going to have to write some software. We would probably want to put our program code in the ROM region, the data into the RAM region with our stack, and the video display routines and refresh memory at the top of the address space. Using an absolute assembler, we just place an appropriate ORG directive at the head of each program section of data definition section. But suppose we are using a relocating assembler, which makes all of the code start at zero? How does one control all of this stuff? The answer is simple. The relocating assembler contains provisions for dividing up the program into areas which can be treated differently by the linkage editor. These areas are called "sections" or "segments." Some programmers may call them "banks."

Different assemblers have fixed names for both the data and program sections of a program. The data section might be called "DSEG", "DSECT", "DATA" or some other similar name. The program section might be called "CSEG," "CSECT," "PROG," or "CODE." These segments, as we will call them, are characterized separately in the symbol table of the assembler. All symbols defined following a "CSEG" command have that fact recorded in the assembler symbol table, and a similar link is recorded for those symbols defined following a "DSEG" command. In most assemblers, the appearance of a new segment name causes the current segment to end and the new segment to begin. Here is the sample program with the code and data segregated into the appropriate segments:

```
0000'                             CSEG              ;CODE SEGMENT
0000'   31 000C"         START:   LXI     SP,STACK
0003'   11 000E'                  LXI     D,MSG
0006'   0E 09                     MVI     C,9
0008'   CD 0005                   CALL    BDOS
000B'   C3 0000                   JMP     BOOT
000E'   48 45 4C 4C      MSG:     DB      'HELLO$'
0012'   4F 24
0005                     BDOS     EQU     5
0000                     BOOT     EQU     0
0014'                             DSEG              ;DATA SEGMENT
0000"                             DS      12        ;ONLY THE STACK
000C"   0002             STACK:   DW      2
                                  END     START
Symbols:
BDOS    0005    BOOT    0000    MSG     000E'   STACK   000C"
START   0000'
```

Notice that when references are made to the DSEG region, the double-quote mark appears next to the value of the label. The DSEG area can be located at any point in the absolute program by informing the linkage editor where you would like the DSEG area to begin. The same applies to the CSEG area marked with the single-quote mark. We have now created a program with two separate areas that can be positioned separately by the linkage editor; we could, however, ignore this division and the program would still be processed properly by the linkage editor.

A different use for the two different segments would be the familiar case of those macro instructions which define a message to be displayed and the instructions that perform the output operation. The sequence might be:

```
MSG:  MACRO    A
      LOCAL    B,C
      MVI      C,9
      LXI      D,B
      CALL     BDOS
      JMP      C
B:    DB       0DH,0AH,A,'$'
C:    DS       0
      ENDM
```

Each time this macro is expanded, a 3-byte jump instruction is generated. Say there are 200 such macro calls in a program. This consumes 600 bytes of storage, and the program may no longer fit into the storage allocated to it. An easy way to avoid this is to use CS'∃G and DSEG commands:

```
      MSG:  MACRO   A
            LOCAL   B
            CSEG    ;; DONT ASSUME ANYTHING
            MVI     C,9
            LXI     D,B
            CALL    BDOS
            DSEG    ;; GENERATE MESSAGE IN DSEG AREA
B:          DB      0DH,0AH,A,'$'
            CSEG    ;; PUT CODE SEGMENT COUNTER BACK ON
            ENDM
```

Relocatable binary modules generated by a compiler may be linked not only with each other, but also with the output of assemblers and with precompiled or assembled library subroutines.

Here, generating the message in the DSEG area displaces it in the absolute program; after linkage, the message and its code are no longer at the point where the source code appears. The jump instruction is eliminated, saving three bytes on each macro expansion. Let's examine the source code of this program to see how the macro is called.

```
BDOS:   EQU      5          ;CP/M ENTRY
BOOT:   EQU      Ø          ;WBOOT ENTRY
MSG:    MACRO    A
        LOCAL    B
        CSEG
        MVI      C,9
        LXI      D,B
        CALL     BDOS
        DSEG
B:      DB       ØDH,ØAH,A,'$'
        CSEG
        ENDM
START:           LXI        SP,STACK
        MSG      'HELLO '
        MSG      'TEST OF '
        MSG      'CSEG/DSEG'
        JMP      BOOT
        DSEG
        DS       12
STACK:           DW         Ø
        END      START
```

The assembled program is shown below. When a plus sign (+) appears to the left of the symbolic code it means that that code was generated by the macro processor of the M80 assembler. Note the appearance of the CSEG and DSEG directives, the way the locations change and the double-quote flags which appear in every DSEG section, as opposed to the single-quote flags in the CSEG section:

```
ØØØ5                         BDOS     EQU    5          ;CP/M ENTRY
ØØØØ                         BOOT     EQU    Ø          ;WBOOT ENTRY
                             MSG      MACRO  A
                                      LOCAL  B
                                      CSEG
                                      MVI    C,9
                                      LXI    D,B
                                      CALL   BDOS
                                      DSEG
                             B:       DB     ØDH,ØAH,A,'$'
                                      CSEG
                                      ENDM
ØØØØ'   31 ØØ2C"             START:   LXI    SP,STACK
                                      MSG    'HELLO '
ØØØ3'                   +             CSEG
ØØØ3'   ØE Ø9           +             MVI    C,9
ØØØ5'   11 ØØØØ"        +             LXI    D,..ØØØØ
ØØØ8'   CD ØØØ5         +             CALL   BDOS
ØØØB'                   +             DSEG
ØØØØ"   ØD ØA 48 45     +    ..ØØØØ:  DB     ØDH,ØAH,'HELLO ','$'
ØØØ4"   4C 4C 4F 2Ø     +
ØØØ8"   24              +
ØØØ9"                   +             CSEG
                                      MSG    'TEST OF '
ØØØB'                   +             CSEG
ØØØB'   ØE Ø9           +             MVI    C,9
ØØØD'   11 ØØØ9"        +             LXI    D,..ØØØ1
ØØ1Ø'   CD ØØØ5         +             CALL   BDOS
ØØ13'                   +             DSEG
ØØØ9"   ØD ØA 54 45     +    ..ØØØ1:  DB     ØDH,ØAH,'TEST OF ','$'
ØØØD"   53 54 2Ø 4F     +
ØØ11"   46 2Ø 24        +
ØØ14"                   +             CSEG
                                      MSG    'CSEG/DSEG'
ØØ13'                   +             CSEG
ØØ13'   ØE Ø9           +             MVI    C,9
ØØ15'   11 ØØ14"        +             LXI    D,..ØØØ2
ØØ18'   CD ØØØ5         +             CALL   BDOS
ØØ1B'                   +             DSEG
ØØ14"   ØD ØA 43 53     +    ..ØØØ2:  DB     ØDH,ØAH,'CSEG/DSEG','$'
ØØ18"   45 47 2F 44     +
ØØ1C"   53 45 47 24     +
ØØ2Ø"                   +             CSEG
```

Assemblers continued . . .

```
001B'   C3 0000                        JMP     BOOT
001E'                                  DSEG
0020"                                  DS      12
002C"   0000                  STACK:   DW      0
                                       END     START

Macros:
MSG

Symbols:
BOOT    0000    STACK   002C"   START   0000'
```

Notice that separate location counters are maintained by the assembler for the CSEG and DSEG sections. The information as to which values are associated with which location counter is maintained in the symbol table. During the second pass of the assembly, when the binary information is being output, the appropriate location counter information will also be output with each value to which it pertains so that the linkage editor will be able to use it.

Some assemblers allow for more than just a code and data counter. This and other features will be discussed in the next installment [of this tutorial.]

Reference:

Gary Kildall, "A Simple Technique for Static Relocation of Absolute Machine Code." *Dr. Dobbs Journal;* Vol. 3, Number 22, February, 1978; pg. 10.

Andrew L. Bender is a practicing physician specializing in medical neurology. He owns two antique S-100 systems and spends some of his free time writing articles about them.

Sleuthing WordStar Files with Pascal

WordStar meets Pascal in a recursive article

by Warren Lambert

A sophisticated text editor like WordStar invisibly marks certain bytes, such as "soft" carriage returns. A Pascal program, called WSSLEUTH can "reveal-print" files with invisible marks to help the user investigate problems, or just understand WordStar files better. WSSLEUTH examines each byte in a file, and writes it to a printer to reveal hidden features.

WordStar

WordStar, like Visicalc, is a popular classic among microcomputer programs. As the Cadillac of word processing programs, it transforms the humble 8-bit general-purpose computer into a useful office tool. After reading science-fiction author Jerry Pournelle's enthusiastic paean to WordStar in 1980, I bought it, and after two years of using WordStar almost daily, I haven't changed my mind about its usefulness and reliability. WordStar has quirks, but it has never ever turned on me with delusions of Pacman and gobbled up any of my writing.

One of WordStar's most civilized features is its ability to mark the spaces and carriage returns it adds as "soft," then to go back later when the text is revised and delete them without disturbing spaces or carriage returns entered by the user. The secretary who uses WordStar to type letters may not care how a text editor marks bytes, but the programmer who uses WordStar as a utility program will want to know exactly what WordStar is doing. This article demonstrates an easy way to reveal the details of any WordStar file with a Pascal program called **WSSLEUTH.**

Pascal

While Basic is the universal microcomputer language, it doesn't include recent advances in the pedagogy of programming—ideas such as structured programming in self-documenting modules. Trials with North Star extended Basic, a good Basic, revealed two serious problems: short variable names such as "X3" were confusing, and it was difficult to structure programs so that the layout visually revealed their Logic. Further, Basic and 5¼″ minifloppies share the fault of originality: everybody in California invented a new proprietary format incompatible with his competitor's, preventing users from sharing disks or programs. The differences among Basics make it hard to use textbooks to learn advanced techniques, such as disk I/O or subroutines, because the examples in the book may not run until you know enough not to need the book.

Pascal was introduced by Jensen and Wirth (1974) to make available "a language suitable to teach programming as a systematic discipline based on certain fundamental concepts . . . (page 133)." Studies of computer science students suggest that freshmen starting with Pascal rather than Basic became better programmers as juniors and seniors; from the beginning they learned methodical habits. Personally, I found that my beginner's programs in North Star Basic started to bog down if they exceeded 50 lines, so I was reluctant to invest too much energy in Basic, especially when the programs wouldn't run with other Basics without being born again in a tedious conversion experience. With Pascal, if the programmer sticks to standard features, rather than machine-specific goodies, a program should run in practically the same form on any computer, and the system-specific parts can be isolated in easy-to-find procedures. The universities have gobbled up Pascal, and given publish-or-perish pressures, many fine Pascal textbooks are available.

Warren Lambert, Ph.D., 5908 Lyons View Drive, Knoxville, TN37919.

JRT Pascal

I bought JRT Pascal in order to learn structured programming. I had tried North Star Pascal (USCD Pascal with input/output for the Horizon), but N* Pascal was isolated in its own operating system, unable to read or write CP/M files without fundamental utility programs that I couldn't write or buy from N*. When respected reviewers (Edelson, 1982; Miller, 1982) said that JRT Pascal was worth at least $29.95, I jumped at the chance to buy it.

JRT Pascal worked fine, but the feature that really warmed my heart was the "RESET" statement:

```
RESET(DUMP_FILE,FILENAME,BINARY,4096);
```

In standard Pascal, "Reset" means "open a file for reading." The endearing feature is "BINARY," which means "read the ASCII [American Standard Code for Information Exchange] file literally from CP/M, just as a CP/M "TYPE" statement would." The "4096" sets a buffer of 4096 bytes, so you don't torture your minifloppies, or go mad and hock your new car for a miniwinny. The ability to read CP/M files instead of being limited to the System files of Pascal means that JRT Pascal (and probably most CP/M Pascals) can read text files created by WordStar. This ability to read and write CP/M files is a crucial feature, since you can still use your regular editor, utility programs, and anything else you've kludged up outside Pascal. Thus all your expensive resources—CP/M utilities, favorite editor, programs written and purchased—all pool their strengths rather than bickering with each other.

Studying WordStar files

The WordStar manual tells you what to do, but it doesn't always explain what WordStar is doing when magic features such as "word wrap" are at work. But the evidence for WordStar's behavior is recorded in any WordStar text. Studying text is a useful way for WordStar users to learn exactly what the editor is doing with your files. Below, a Pascal program "WSSLEUTH" will analyze WordStar files; but first, let's look at the sample text with a CP/M

"TYPE" statement, which causes the file to be read and sent to the console or printer (see Listing 1).

When TYPED, the WordStar text looks naive and innocent of any fancy digital footwork. The control characters, such as ^S to underline the abstract, are invisible, as are the "soft hyphens" after 'Word-' and 'micro'. But with DDT (Dynamic Debugging Tool, a standard CP/M utility), we can see if there's more in a WordStar file than meets the eye. The results, shown in Listing 2, appear when you run DDT while the CP/M ' ^P' command sends output to the printer.

DDT flushed out some strange characters. For example, DDT printed bytes other than periods as ".", because they weren't in the printable range from A . . Z, a . . z, etc. For example, in the title, the word "WORDS" in the ninth DDT line between "SLEUTHING" and "FILES" [186H to 18FH] is entirely unprintable according to DDT, even though it looked perfectly normal when it was TYPEd.

To find out what happened to the word 'WORDS' in Listing 2, let's examine 'WORDS' in printable ASCII characters, hexadecimal numbers, ASCII-decimal numbers, and binary numbers.

The four different ways of saying 'WORDS' appear in Table 1. The rows called 'normal' refer to standard 7-bit ASCII way of writing the letters 'W,O,R . . . ,' and the 'marked' rows refer to the characters actually used by

Table 1. Words expressed in common number systems and WordStar marked text

TABLE I
'WORDS' EXPRESSED IN COMMON NUMBER SYSTEMS AND WORDSTAR 'MARKED' TEXT

Number system	Way of saying 'WORDS'				
ASCII normal	W	O	R	D	S...
Hexadecimal version					
Normal hex	57	4F	52	44	53
Marked hex	D7	CF	D2	C4	D3
Difference (hex)	80	80	80	80	80
ASCII tens version					
Normal ASC(CHR)	87	79	82	68	83
Marked ASC(CHR)	215	207	210	196	211
Difference (ten)	128	128	128	128	128
Binary version					
Normal	01010111	01001111	01010010	01000100	01010011
Marked	11010111	11001111	11010010	11000100	11010011
Difference	10000000	10000000	10000000	10000000	10000000

WordStar, according to the DDT listing. The binary version reveals most clearly what WordStar did: to mark certain bytes for future reference, it extended 7-bit ASCII to mark the eighth bit, which becomes 1 rather than 0 on marked text. The really clever aspect of this programmer's trick is the fact that normal ASCII-coded devices, such as terminals or printers, display these marked bytes quite normally, since such devices only look at seven bits [0 . . 127]. Thus WordStar is able to remember which space it added (e.g., by automatic justification) as opposed to "hard" features chosen by the author, while retaining WordStar's cardinal feature: what you see on the screen is what you get on paper.

Pascal as WordStar Sleuth

While DDT is helpful in exploring a WordStar file, its outputs are hard to read. A utility program that prints WordStar texts revealingly could be useful in studying special problems, or for just getting to know WordStar's behavior better. I needed to study Pascal program texts because a clever text editor like WordStar may be too smart for one's own good. Out of habit, I unthinkingly hit ' ^B' [reform the line] while editing a Pascal program. The resulting code looked perfectly normal, but the Pascal compiler gave some very flakey error messages. The reason, I finally learned after munging* my program into oblivion, was that the word 'PROGRAM' had a marked letter and was unrecognizable to the compiler, which evidently looked at all eight bits; yet my 7-bit ASCII terminal and printer showed 'PROGRAM' as a normal word. Once discovered, the problem could be fixed with the standard utility program, PIP (*filename.ext-filename.ext[Z]), which has a 'Z' option to Zero the eighth bit on all bytes in a file.

JRT Pascal could do practically anything you want—not only display files intelligently but also modify them. Listing 3 shows a program "WSSLEUTH," which reads any file and sends it to the printer after formatting certain bytes, such as carriage returns, line feeds, and WordStar "marked" bytes.

The main point of WSSLEUTH appears in the CASE statement in the "revealprint" procedure, which examines each byte in the file and writes it to the printer, so as to reveal hidden (nonprinting) features. WSSLEUTH could be changed easily into a byte-oriented utility program merely by changing the CASE statement as needed and by writing the output to a new file rather than the printer. This is easy in Pascal, which views the printer as a Logical file.

Listing 4 shows the first 1024 bytes of Listing 1"revealprinted." Control characters appear in standard control format; e.g., 'control-S' appears as ' ^S.' (The ^Q is a user-modifiable WordStar control, in this case telling a NEC Spinwriter to use its second character set; ^W turns the second set off.) Marked bytes, such as soft spaces (added by WordStar for justification) or end-of-words, are underlined to show that they were marked. Subtracting 128 sets bit 7 to zero ($2^7 = 128$). Soft hyphens were a surprise, since I expected marked hyphens. Actually, they are ASCII 31s (unprintable characters, hex 1F or binary 00011111) and not hyphens at all, which explains why they are invisible when you TYPE them. When WordStar shows soft hyphens as reverse video hyphens, a clever computer program is being deceptively ingenuous, humoring the user again.

Marked bytes are underlined by WSSLEUTH, and nearly all the carriage returns in the example are underlined. The original text was typed with "wordwrap," and most of the carriage returns were entered by WordStar as soft carriage returns; they appear as '[CR]' in the output of WSSLEUTH. A normal [or "hard"] carriage return is '[CR]', without the underline. The last letter of most words was marked by WordStar, which is why the word "WordStar" in the title is non-ASCII-128: WordStar marked the spaced letters as ends of words.

A utility program in interpreted Pascal, such as JRT, is

**Mung, the recursive verb: Mung means "mung until no good."*

much slower than machine code, especially with the long, easy-to-write but slow-to-run case statement. However, the program WSSLEUTH will drive a NEC Spinwriter 7710 faster than its limit of 55 characters per second on a 4 MHz Z80 computer, so the extra speed of machine language or elegant programming would offer no practical benefit. A more machine-efficient version of WSSLEUTH was written without the long CASE statement, and it ran in two-thirds of the time used by the version in Listing 3 (which took 24 seconds for 1K with the printer turned off). With a medium-speed printer, the faster version took 34 seconds, while the slower but easier to read version took 36 seconds; the trickier code saved only 2 seconds.

Conclusion

WSSLEUTH offers an easy way to inspect WordStar or similar files. Such a program could help to discover problems in files produced by WordStar, or help the user learn exactly what the editor is doing with the files it creates.

References

Edelson, R. "Pascal for CP/M Systems." *Creative Computing*, 1982, 8 (11), 113-116.

Jensen, K., and Wirth, N. *Pascal, User Manual and Report*. New York: Springer-Verlag, 1974.

JRT Systems. *JRT Pascal Users Guide*. San Francisco: JRT Systems, 1982.

Miller, A. R. "JRT Pascal, Version 2." *Interface Age*, 1982 7 (10), 118-121.

Petix-Marney, V. [Ed.]. *Some Common Pascal Programs*. Berkeley: Osborne/McGraw, 1982.

Listing 1. Printing a WordStar text with CP/M "TYPE" statement.

```
B>
B>
B>
B>
B>TYPE RECURSIV.DAT
.CW 8
.PO 30
.LH 11
.PL 48
.MT 5
..

.he                 W O R D S T A R   &   P A S C A L

             S L E U T H I N G   W O R D S T A R   F I L E S
                           W I T H   P A S C A L

                                   by

                         Warren Lambert,Ph.D.


                WordStar meets Pascal in a recursive article.


      Abstract.  A sophisticated text editor like  Word
      Star invisibly marks certain bytes, such as "soft"
      carriage  returns.   A  CP/M  Pascal  program  can
      "reveal-print"  files  to help  the user investigate
      problems   or   just  to   understand   WordStar   files
      better.

   WORDSTAR>, like VisiCalc>, is a popular classic among micro
computer programs.  As the Cadillac> of word processing programs,
it  transforms  the humble 8-bit general purpose computer into  a
useful office tool.   After reading science fiction author  Jerry
Pournelle's enthusiastic paean to WordStar in 1980,
B>
B>
B>
B>
B>
```

Listing 2. Revealing hidden details of a WordStar text with DDT.

```
B>
B>C:DDT B:RECURSIV.DAT
DDT VERS 2.2
NEXT  PC
0500 0100
-D100,500
0100 2E 43 57 20 38 0D 0A 2E 50 4F 20 33 30 0D 0A 2E .CW 8...PO 30...
0110 4C 48 20 31 31 0D 0A 2E 50 4C 20 34 38 0D 0A 2E LH 11...PL 48...
0120 4D 54 20 35 0D 0A 2E 2E 20 0D 0A 0D 0A 2E 68 65 MT 5.... .....he
0130 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0140 20 13 57 20 4F 20 52 20 44 20 53 20 54 20 41 20  .W O R D S T A
0150 52 20 20 26 20 20 50 20 41 20 53 20 43 20 41 20 R  &  P A S C A
0160 4C 13 0D 0A 0D 0A 20 20 20 20 20 20 20 20 20 02 L.....         .
0170 04 11 53 20 4C 20 45 20 55 20 54 20 48 20 49 20 ..S L E U T H I
0180 4E 20 47 20 20 20 D7 20 CF 20 D2 20 C4 20 D3 20 N G    . . . . .
0190 D4 20 C1 20 D2 20 20 20 46 20 49 20 4C 20 45 20 . . .   F I L E
01A0 53 0D 0A 20 20 20 20 20 20 20 20 20 20 20 20 20 S..
01B0 20 20 20 20 20 20 20 20 20 D7 20 C9 20 D4 20 C8          . . . .
01C0 20 20 20 D0 20 C1 20 D3 20 C3 20 C1 20 4C 17 02    . . . . . L..
01D0 04 0D 0A 0D 0A 20 20 20 20 20 20 20 20 20 20 20 .....
01E0 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
01F0 20 20 20 20 62 79 0D 0A 0D 0A 20 20 20 20 20 20     by....
0200 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
0210 57 61 72 72 65 6E 20 4C 61 6D 62 65 72 74 2C 50 Warren Lambert,P
0220 68 2E 44 2E 20 20 20 0D 0A 0D 0A 0D 0A 20 20 20 h.D.   ......
0230 20 20 20 20 20 20 20 57 6F 72 64 53 74 61 72 20        WordStar
0240 6D 65 65 74 73 20 50 61 73 63 61 6C 20 69 6E 20 meets Pascal in
0250 61 20 72 65 63 75 72 73 69 76 65 20 61 72 74 69 a recursive arti
0260 63 6C 65 2E 0D 0A 0D 0A 0D 0A A0 A0 A0 A0 A0 A0 cle.............
0270 A0 02 04 41 62 73 74 72 61 63 74 2E 02 84 A0 20 ...Abstract....
0280 13 C1 20 73 6F 70 68 69 73 74 69 63 61 74 65 E4 .. sophisticate.
0290 20 74 65 78 F4 20 65 64 69 74 6F F2 20 6C 69 6B  tex. edito. lik
02A0 E5 A0 20 57 6F 72 64 1F 8D 0A A0 A0 A0 A0 A0 A0 .. Word.........
02B0 A0 53 74 61 F2 20 69 6E 76 69 73 69 62 6C F9 20 .Sta. invisibl.
02C0 6D 61 72 6B F3 20 63 65 72 74 61 69 EE 20 62 79 mark. certai. by
02D0 74 65 73 AC 20 73 75 63 E8 20 61 F3 20 22 73 6F tes. suc. a. "so
02E0 66 74 A2 20 8D 0A A0 A0 A0 A0 A0 A0 A0 63 61 72 ft. ........car
02F0 72 69 61 67 E5 A0 20 72 65 74 75 72 6E 73 13 AE riag.. returns..
0300 A0 20 20 13 C1 A0 20 43 50 2F CD A0 20 50 61 73 .  ... CP/.. Pas
0310 63 61 EC A0 20 70 72 6F 67 72 61 ED A0 20 63 61 ca.. progra.. ca
0320 EE 20 8D 0A A0 A0 A0 A0 A0 A0 A0 22 72 65 76 65 . ........"reve
0330 61 6C 2D 70 72 69 6E 74 A2 A0 20 66 69 6C 65 F3 al-print.. file.
0340 20 74 EF 20 68 65 6C F0 20 74 68 E5 20 75 73 65  t. hel. th. use
0350 F2 20 69 6E 76 65 73 74 69 67 61 74 E5 20 8D 0A . investigat. ..
0360 A0 A0 A0 A0 A0 A0 A0 70 72 6F 62 6C 65 6D F3 A0 .......problem..
0370 20 6F F2 A0 20 6A 75 73 F4 20 74 EF A0 20 75 6E  o.. jus. t.. un
0380 64 65 72 73 74 61 6E E4 A0 20 57 6F 72 64 53 74 derstan.. WordSt
0390 61 F2 A0 20 66 69 6C 65 F3 20 8D 0A A0 A0 A0 A0 a.. file. ......
03A0 A0 A0 A0 62 65 74 74 65 72 13 2E 0D 0A 0D 0A 20 ...better......
03B0 20 20 20 20 02 04 11 57 4F 52 44 53 54 41 52 02     ...WORDSTAR.
03C0 04 3E 17 AC 20 6C 69 6B E5 20 56 69 73 69 43 61 .>.. lik. VisiCa
03D0 6C 63 11 04 3E 04 17 AC 20 69 F3 20 E1 20 70 6F lc..>... i. . po
03E0 70 75 6C 61 F2 20 63 6C 61 73 73 69 E3 20 61 6D pula. classi. am
03F0 6F 6E E7 20 6D 69 63 72 6F 1F 8D 0A 63 6F 6D 70 on. micro...comp
0400 75 74 65 F2 20 70 72 6F 67 72 61 6D 73 AE 20 20 ute. programs.
0410 41 F3 20 74 68 E5 20 43 61 64 69 6C 6C 61 63 11 A. th. Cadillac.
0420 3E 97 20 6F E6 20 77 6F 72 E4 20 70 72 6F 63 65 >. o. wor. proce
0430 73 73 69 6E E7 20 70 72 6F 1E 67 72 61 6D 73 AC ssin. pro.grams.
0440 20 8D 0A 69 F4 A0 20 74 72 61 6E 73 66 6F 72 6D  ..i.. transform
0450 F3 A0 20 74 68 E5 20 68 75 6D 62 6C E5 20 38 2D .. th. humbl. 8-
0460 62 69 F4 20 67 65 6E 65 72 61 EC 20 70 75 72 70 bi. genera. purp
0470 6F 73 E5 20 63 6F 6D 70 75 74 65 F2 20 69 6E 74 os. compute. int
0480 EF A0 20 E1 20 8D 0A 75 73 65 66 75 EC 20 6F 66 .. . ..usefu. of
0490 66 69 63 E5 20 74 6F 6F 6C AE A0 20 20 41 66 74 fic. tool..  Aft
```

```
04A0 65 F2 20 72 65 61 64 69 6E E7 20 73 63 69 65 6E e. readin. scien
04B0 63 E5 20 66 69 63 74 69 6F EE 20 61 75 74 68 6F c. fictio. autho
04C0 F2 A0 20 4A 65 72 72 F9 20 8D 0A 50 6F 75 72 6E .. Jerr. ..Pourn
04D0 65 6C 6C 65 27 F3 20 65 6E 74 68 75 73 69 61 73 elle´. enthusias
04E0 74 69 E3 20 70 61 65 61 EE 20 74 EF 20 57 6F 72 ti. paea. t. Wor
04F0 64 53 74 61 F2 20 69 EE 20 31 39 38 30 AC A0 1A dSta. i. 1980...
0500 C3 .
-
```

Listing 3. WSSLEUTH: a Pascal program to reveal CP/M texts.

```
(* ========================================================================== *)
(*                              W S S L E U T H                               *)
(*                                                                            *)
(*  Author:    Warren Lambert,  5908 Lyons View Drive,  Knoxville TN 37919,   *)
(*             615:584-1561 X7724.                                            *)
(*                                                                            *)
(*                                                                            *)
(* This program in JRT Pascal 2.0 reads a WordStar text file from CP/M disk,  *)
(* and  prints it with WordStar's invisible features and control  characters  *)
(* marked as follows:                                                         *)
(*                                                                            *)
(*      control characters       (e.g. control-B)         become ^B           *)
(*      marked bytes             (e.g. soft CR's)         are underlined      *)
(*      carriage returns         (i.e. 0D_H)              become [CR]         *)
(*      marked carriage returns  (i.e. 8D_H)              become _[CR]        *)
(*      line feeds               (i.e. 0A_H)              become [LF]         *)
(*      marked line feeds        (i.e. 8A_H)              become _[LF]        *)
(*                                                                            *)
(*                            Instructions                                    *)
(*      Execute    WSSLEUTH.   It    will   ask   for  a  file  to  read  (e.g. *)
(* C:RECURSIV.DAT).  Enter the file name and carriage return.  Test on short  *)
(* files,  as WSSLEUTH has no built-in interruptions.  When the program asks  *)
(* whether to continue and print another file, answer yes or no as directed.  *)
(*                                                                            *)
(*                        System specific features                            *)
(*      Features that might not run with other Pascals appear in caps in the  *)
(* program,  and  most are isolated in procedures,  such as  "console_only,"  *)
(* which could be changed easily.  Some details follow:                       *)
(*      JRT Pascal permits names_including_the_underscore_character,  and an  *)
(* extended   "ELSE"   in the case statement;  these reasonable  features  are *)
(* unwirthy of standard Pascal.    "CLOSE" file is not standard, but it is OK *)
(* in UCSD Pascal.                                                            *)
(*                                                                            *)
(* ========================================================================== *)

program wssleuth;
                                                   {          COMMENTS          }
var     byte        : char;
        wordstar_file : file of char;
        filename    : array [1..14] of char;


procedure console_only;                            {procedures   console_only    and }
begin                                              {listdevice_only   use   JRT  Pascal }
     SYSTEM(nolist);                               {options to send the output to the }
     SYSTEM(cons);                                 {console,   printer or both.  other }
end;                                               {Pascals    may   require   different }
                                                   {routines.   NOTE CAPS ON JRT PROCS }

procedure listdevice_only;                         {JRT Pascal version; make your own }
begin
     SYSTEM(nocons);
     SYSTEM(list);
end;

procedure open_file;                               { JRT features: BINARY reads file  }
begin                                              { literally without EOL.. 4096 is a }
     write('enter file name : ');                  { buffer of 4096 bytes             }
     readln( filename );
     RESET( wordstar_file, filename,BINARY, 4096);
end; { open_file }


function NotAgain: boolean;                        {procedure   NotAgain   is    from }
var      user_response:char;                       {Osborne's   Some  Common   Pascal }
begin                                              {Programs, page 228.               }
    console_only;
    write(chr(12),'would you like another run? (y/n)  ');
    readln(user_response);
    while not (user_response in ['Y','y','N','n'] ) do
        begin
            write('Type y for yes, or n for no: ');
            readln(user_response)
        end;
    NotAgain := user_response in ['N','n']
end; {NotAgain}


procedure revealprint(byte: char);                 {procedure    revealprint    sends }
begin                                              {checked&altered   bytes   to   the }
     listdevice_only;                              {printer.    The   case  statement }
     if ord(byte) > 127 then                       {assumes  standard  ASCII  values. }
        begin
        write('_',chr(08));                        { underline&backspace              }
        byte := chr( ord(byte)-128)                { unmark marked bytes              }
        end;
     case byte of
        chr(01)    :write('^A');                   chr(14)    :write('^N');
        chr(02)    :write('^B');                   chr(15)    :write('^O');
        chr(03)    :write('^C');                   chr(17)    :write('^Q');
        chr(04)    :write('^D');                   chr(19)    :write('^S');
        chr(06)    :write('^F');                   chr(20)    :write('^T');
        chr(07)    :write('^G');                   chr(22)    :write('^V');
        chr(08)    :write('^H');                   chr(23)    :write('^W');
        chr(09)    :write('<--TAB-->');            chr(24)    :write('^X');
        chr(10)    :write(byte,'[LF]');            chr(25)    :write('^Y');
        chr(13)    :write('[CR]',byte);            chr(31)    :write('-',chr(08),'^');
        ELSE       :write(byte);
     end;
     console_only;
end; {revealprint}


procedure crtprint(byte: char);
begin
     console_only;
     write(byte);
end; {crtprint}

begin {main}
  repeat
     begin
     open_file;
     while not eof(wordstar_file) do               { Until end of file, read a byte   }
        begin                                      { and  send  it  unaltered to the  }
        read(wordstar_file; byte);                 { CRT  and  "reveal print" it on   }
        crtprint(byte);                            { the printer.                     }
        revealprint(byte);
        end;
```

Sleuthing WordStar Files continued . . .

```
     CLOSE(wordstar_file);
     listdevice_only;
     writeln( chr(12));
     console_only;
     end
  until NotAgain;
end. {wssleuth}
```

Listing 4. Output from WSSLEUTH: the first 1024 bytes of this article.

```
.CW 8[CR]
[LF].PO 30[CR]
[LF].LH 11[CR]
[LF].PL 48[CR]
[LF].MT 5[CR]
[LF].. [CR]
[LF][CR]
[LF].he                ^SW O R D S T A R  &  P A S C A L^S[CR]
[LF][CR]
[LF]          ^B^D^QS L E U T H I N G  W O R D S T A R  F I L E S[CR]
[LF]                         W I T H  P A S C A L^W^B^D[CR]
[LF][CR]
[LF]                              by[CR]
[LF][CR]
[LF]                     Warren Lambert,Ph.D.   [CR]
[LF][CR]
[LF][CR]
[LF]          WordStar meets Pascal in a recursive article.[CR]
[LF][CR]
[LF][CR]
[LF]________^B^DAbstract.^B^D_ ^SA sophisticated text editor like  Word^[CR]
[LF]________Star invisibly marks certain bytes, such as "soft" [CR]
[LF]________carriage  returns^S_   ^SA  CP/M  Pascal  program  can [CR]
[LF]________"reveal-print"  files to help the user investigate [CR]
[LF]________problems  or  just to  understand  WordStar  files [CR]
[LF]________better^S.[CR]
[LF][CR]
[LF]      ^B^D^QWORDSTAR^B^D>^W, like VisiCalc^Q^D>^D^W, is a popular classic among micro^[CR]
[LF]computer programs.  As the Cadillac^Q>^W of word processing programs, [CR]
[LF]it  transforms  the humble 8-bit general purpose computer into  a [CR]
[LF]useful office tool.  After reading science fiction author  Jerry [CR]
[LF]Pournelle's enthusiastic paean to WordStar in 1980,
```

Cross Check

by Scott Nowell

Nothing is as disconcerting as calling a file up with your editor and finding a piece of it garbled beyond recognition. Upon examining your text file, you might recognize some of the prompts from one of your favorite programs embedded among endless control characters. It may occur to you that you are seeing pieces of two different files that have been accidently merged.

What happens is that the disk allocation map becomes damaged, either in memory or on the disks themselves. This could be the result of a program crash, a memory failure (the disk allocation map is in memory at all times), changing disks during a disk write operation, or even changing a disk without typing control-C (used to tell CP/M that you changed disks and the disk allocation map must be reloaded) before continuing with anything else.

With a floppy disk the problem may not be too severe; after all, you did back the files up just before you compiled the new version of the program, didn't you? Although you might be able to recover most of the files, you do not know if the problem will come right back. If the problem is the result of two files trying to occupy the same block on the disk, then the files may well appear to self-destruct with no warning. With a Winchester disk, the problem may be magnified by the huge number of files on the disk at any time, most of which were probably backed up sometime within the last year. Still, you don't know which files are merged. If you could view the disk allocation map for the files, you might be able to decide which file owned most of the disk space and would be easiest to recover. Enter Cross Check.

Cross Check checks the disk allocation table of all the files on the disk against all of the other file disk allocation tables on the disk. If any duplications are found, either between files or between blocks within a file, the Cross Check program displays the names of the files and the disk allocation tables on the screen. After looking at the display, you can decide which file is in the worst shape and, Murphy's Law notwithstanding, save the most important one with the aid of a disk sector editor such as DUMP22 from the CP/M User's Group. The important point is to delete one of the files or disk block references so that the conflict no longer exists.

Crosschk operation

To run the program, type "crosschk," followed by a carriage return. Cross Check asks you if the file names should be displayed. If you want to see the name of each file as it is being checked, answer "Y". The number of the file being checked is displayed if the filename is not, but the number is only an indication of how many files have been checked.

Cross Check then prompts you for the name of the drive to be checked. After the letter is entered (upper or lower case), the driver is selected, and a list of information about the drive is displayed:

```
Address of disk parameter block header:
Address of disk parameter block:
Disk offset:
Sectors per track:
Number of tracks to read:
Max number of directory entries:
Max number of sectors to read:
Total files to be checked:
```

Cross Check then loads the entire directory into memory and starts cross-checking the disk allocation blocks. If any disk blocks are found to be allocated to more than one file, Cross Check displays the names of both files along with the extent number, the number of the user area they appear in, and the disk allocation map for both files. The display format is:

```
CROSSCHKDOC  Ext: 0 User: 2 Map: 0023 0024 0025 0000 0000 0000 0000 0000
XASM     COM  Ext: 1 User: 0 Map: 0304 0305 030A 0120 0025 0000 0000 0000
```

The first file listed is the one being checked against all other entries, and the second file is the one it is currently being compared to. The number of entries in the map is variable, based upon the capacity of the disk drive. Drives with up to 255K will have 16 map entries. Each map entry points to a 1K block of disk space. If the drive has more than 255K of space, the map will show only eight entries, but they are wordsized values (as in the example above), capable of pointing to up to 65,535 blocks of 2K, 4K, 8K or 16K.

If no disk allocation block duplications are found, then the disk being tested is okay and doesn't have any blocks belonging to more than one file.

Program description

The Cross Check program is written in C (Listing 1) and was compiled with the BDS C compiler. When compiled with the E switch, the program takes about 7K of space and reserves up to 32K of data space for the directory. The -E switch tells the compiler the location of the variables in memory (allowing direct addressing of the variables), reduces the amount of generated code, and gives a slight increase in speed of execution. To compile the program, use the command line:

```
CC1 CROSSCHK.C -E2000
```

The compiled program places the directory array in memory, starting at hexadecimal 2000. The source program is set up for 1024 directory entries.

The structures defined at the beginning of the program can be used in other programs that need direct access to CP/M disks. The address of the DPH (disk parameter block header) structure is obtained by doing a direct disk-select operation via the system BIOS. The select disk BIOS call will return the address of the DPH in the HL register pair of the CPU. The DPB (disk parameter block) address is the fifth item in the DPH structure.

The BIOS call function in the BDS C library does not return to the "C" program the value returned in HL by the BIOS call. Instead, it returns the value contained in the Accumulator. In order to get the value returned in HL, a new function was required. The new function is a modification of the BIOS call function supplied in the DEFF2.ASM file with BDS C. This new function is written in assembly language and must be separately assembled using MAC from Digital Research. The function

Scott Nowell, Future Dimensions Inc., 421 Amherst St., Nashua, NH 03063

(Listing 2) called HBIOS() adds a third parameter to the function call and returns, to the C code, the value returned in HL by the BIOS routine. The third parameter is put in the DE register pair before the BIOS call is made. The HBIOS function is used for the sector translation and disk-select BIOS calls.

To link the compiled code into a runnable program, the command line

```
CLINK CROSSCHK HBIOS -L STDLIB2 STDLIB1
```

is used. The two libraries, STDLIB1 and STDLIB2, must be named in reverse order to resolve all of the external declarations without having to link the entire modules.

Conclusion

The program is very helpful when used in conjunction with a disk test program like BADLIM from BLAT Software. BADLIM does a super job of disk testing and tells you if any files contain damaged sectors, but it doesn't tell you what user area the files are in. This is a problem if you have the same filename in more than one user area. BADLIM creates a file in user area 15 called BSBSBSBS.BSB. This read-only system file contains all the bad blocks on the disk. Cross Check properly identifies the names of the files that have had blocks allocated into the BSBSBSBS.BSB file. It has proven to be invaluable on my Winchester disk system in finding crashes of the directory caused by new programs gone awry.

Ray Cote originally wrote the program in Pascal MT+ while we were debugging a new BIOS for a Winchester disk system. Our main goal at that time was to get the program running as soon as possible without any particular regard for speed. I converted the program to C while making some changes in the user interface of the program. The Cross Check program could probably be rewritten again to execute faster by taking advantage of the C language's pointer mechanism. However, it is the type of utility that is not run very often, so speed of execution was secondary in the original design. The compare routine from the original Pascal source was converted to C as literally as possible so that differences in operational speed could be checked. The compare routine is completely memory bound, and should readily show up speed differences between the Pascal and C code. I was surprised that there was no perceptible difference in speed between the Pascal and C.

Scott Nowell is the owner of Future Dimensions, a small consulting firm in southern New Hampshire specializing in hardware and software design for microprocessor-based products. He has designed micro-based systems ranging in size from simple controllers using single-chip computers to complete word processing systems with multiuser capability.

```
/*
        Functions numbers for CP/M bios calls
*/

#define CLDBOOT 0
#define WRMBOOT 1
#define CONSTAT 2
#define CONIN   3
#define CONOUT  4
#define LIST    5
#define PUNCH   6
#define READER  7
#define HOMDSK  8
#define SELDSK  9
#define SETTRK  10
#define SETSEC  11
#define SETDMA  12
#define DSKREAD 13
#define DSKWRT  14
#define LISTST  15
#define SECTRN  16

#define NULL    0
#define FALSE   0
#define TRUE    -1

/*
        File Control Block structure.

        Each directory entry is set up in this format.
*/

        struct dir {
                char    et;             /* user #, valid fcb if not 0xE5 */
                char    fn[11];         /* file name */
                char    ex;             /* extent */
                char    s1;             /* CP/M system use */
                char    s2;             /* CP/M system use */
                char    rc;             /* record count for extent */
                union {
                        int     w[8];

                                char    b[16];
                                } alloc;        /* disk allocation */
                        } directory[512];

        /*
                Disk Parameter Block Header structure:

                This structure is contained in the system BIOS and contains
                pointers to descriptive properties of the selected disk.
                There is a seperate DPH for each disk drive in the system.
        */

                struct dph {
                        int     *trans_tab;     /* sector translation table */
                        int     b1,b2,b3;       /* three words of zero for bdos */
                        int     *dirbuf;        /* address of directory buffer */
                        struct  dpblock *dpb;   /* pointer to disk parameter block */
                        int     *csv;           /* pointer to directory check sum */
                        int     *alv;           /* pointer to block allocation table */
                        } *dpbase;

        /*
                Disk Parameter Block structure:

                This structure contains information describing the selected
                disk drive.  One or more per system, dependent on number and
                type of drives.
        */

                struct dpblock {
                        int     spt;    /* sectors per track */
                        char    bsh;    /* data allocation block shift factor */
                        char    blm;    /* block shift mask */
                        char    exm;    /* extent mask */
                        int     dsm;    /* drive storage capacity */
                        int     drm;    /* number of directory entries */
                        char    al0;    /* number of reserved directory blocks */
```

Cross Check loads the entire directory into memory, then cross-checks the disk allocation blocks.

```
                char    all;
                int     cks;    /* directory check vector size */
                int     off;    /* number of reserved system tracks */
                } *dpbptr;

        int     dir_point[1024];        /* pointer to 1024 directory entries */
        int     master_word, test_word; /* temporaries for the compare routine *
        int     total_files;    /* number of valid fcb's in directory */
        char    name[30];       /* temp. for file names */
        int     drivenum;       /* number of drive selected A: = 1 */
        char    ch;             /* temp. */
        int     alloc_cnt;      /* number of units in disk map = 8/16 */
        int     words;          /* boolean for disk map = words/bytes */
        int     display;        /* boolean for name display on/off */

/*      -----------------------------------------------------------------
        Load Directory:

        Read the directory from the user selected drive into the
        directory structure array.  The number of entries is
        determined by the information in the disk parameter block.

*/

load_directory()
{
        int     sector;
        int     new_sect;
        int     max_sect;
        int     dma_adr;
        int     track;
        int     numtrk;

        dma_adr = directory;    /* a pointer to an array of fcb */
        bios(SETTRK,2);
        if (!(dpbase = HBios(SELDSK, drivenum, NULL))) {
                Printf("Can't Select %c:",drivenum + 'A');
                return(FALSE);
                }
        Printf("Address of disk parameter block header: %x\n",dpbase);
        dpbptr  = dpbase->dpb;
        Printf("Address of disk parameter block: %x\n",dpbptr);
        max_sect = (dpbptr->drm + 1) / 4;
        if (!(numtrk  = max_sect / dpbptr->spt))
                numtrk = 1;

/*
        If the data allocation block shift factor is greater than 3,
        then the disk contains more than 256K.  When the disk contains
        more than 256K the allocation bytes in the file control blocks
        are word pointers (ie. 0 - 65K blocks of data block size).

        The "word" variable is used to switch from byte sized allocation
        blocks to word size.  The allocation count tells how many blocks
        are allocated per fcb.
*/

        words = (dpbptr->bsh > 3);      /* indicates alloc. map is in words */
        alloc_cnt = (dpbptr->bsh > 3) ? 8 : 16;


        Printf("Disk offset: %d\n",dpbptr->off);
        Printf("Sectors per track:  %d\n",dpbptr->spt);
        Printf("Number of tracks to read: %d\n",numtrk);
        Printf("Max number of directory entries: %d\n",dpbptr->drm+1);
        Printf("Max number of sectors to read: %d\n",max_sect);

/*
        Do the actual read of the directory.
*/

        for (track = 0; track < numtrk; ++track) {
                bios(SETTRK,dpbptr->off + track);
                for (sector = 0; sector < dpbptr->spt; ++sector) {
                        bios(SETDMA,dma_adr);
                        if (dpbase->trans_tab)
                            new_sect = HBios(SECTRN,sector,dpbase->trans_tab);
                        else
                            new_sect = sector;
                        bios(SETSEC,new_sect);
                        bios(DSKREAD,0);
                        dma_adr += 128;
                        }
                }
        Puts("Finished reading directory.\n");
        return(TRUE);
        } /* read_dir */

/*      -----------------------------------------------------------------
        Compare:

                This is the actual compare routine.  It starts with
        the first disk allocation map entry for the first directory
        entry and compare it to all other disk allocation map entries,
        including the others for the same file.  If a match is found
        the filename, extent, user number, and disk map for the entry
        being checked and the one it collided with are displayed on
        the screen.

                The routine works from the beginning toward the end,
        and therefore runs through entries faster with each one it
        checks, because it doesn't need to recheck the entries pre-
        ceeding the current one.

*/

compare()
{
        int     index,index_2,index_3,index_4,index_5;
        int     point,point_3;

        Puts("\n");
        for (index = 0; index < total_files; ++index) {
          point = dir_point[index];
          Printf("%4d ",index);
          if (display) {
            Movmem(directory[point].fn,name,11);
            name[11] = 0;
            Printf("%s   Ext: %d   User: %2d   Recs: %2xH\n",&name,
                directory[point].ex,
                directory[point].et,
                directory[point].rc);
            }
/*
        loop through 8/16 allocation words at each fcb.
*/
          for (index_2 = 0; index_2 < alloc_cnt; ++index_2) {
/*
        if non-zero, compare master word to all other words.
*/

            if (words)
                master_word = directory[point].alloc.w[index_2];
            else
                master_word = directory[point].alloc.b[index_2];

            if (master_word) {
              for (index_3 = index; index_3 < total_files; ++index_3) {
                point_3 = dir_point[index_3];
                for (index_4 = 0; index_4 < alloc_cnt; ++index_4) {

/*
        Test the master_word against the one pointed to by point_3.

 */


            if (words)
                test_word = directory[point_3].alloc.w[index_4];
            else
                test_word = directory[point_3].alloc.b[index_4];

                  if (test_word) {
                    if ((master_word == test_word) &&
                        !((index_4==index_2) && (index==index_3))) {

/*
```

```
        match found, print filename and block numbers of both files
*/

                Printf("\n        Pointer Collision\n\n");
                for (index_5 = 0; index_5 < 11; ++index_5)
                  PutCh(directory[point].fn[index_5] & ~128);
                Printf(" Ext: %d User: %2d Map: ",
                          directory[point].ex,
                          directory[point].et);
                if (words) /* allocation map is words */
                  for (index_5 = 0; index_5 < alloc_cnt; ++index_5)
                    Printf("%4x ",directory[point].alloc.w[index_5]);
                else       /* allocation map is bytes */
                  for (index_5 = 0; index_5 < alloc_cnt; ++index_5)
                    Printf("%2x ",directory[point].alloc.b[index_5]);
                PutChar('\n');

/*
        Now display the file we matched
*/

                for (index_5 = 0; index_5 < 11; ++index_5)
                  PutCh(directory[point_3].fn[index_5] & ~128);
                Printf(" Ext: %d User: %2d Map: ",
                          directory[point_3].ex,
                          directory[point_3].et);
                if (words) /* allocation map is words */
                  for (index_5 = 0; index_5 < alloc_cnt; ++index_5)
                    Printf("%4x ",directory[point_3].alloc.w[index_5]);
                else       /* allocation map is bytes */
                  for (index_5 = 0; index_5 < alloc_cnt; ++index_5)
                    Printf("%2x ",directory[point_3].alloc.b[index_5]);
                PutChar('\n');
                PutChar('\n');
                }       /* if master_word == ...        */
              }       /* if test_word                 */
            }       /* for index_4                  */
          }       /* for index_3                  */
        }       /* if (master_word)             */
      }       /* for index_2                  */
    }       /* for index                    */
}       /* Load directory               */

/*      ------------------------------------------------------------

        Cull Directory:

                Build an index of all of the valid file control
        blocks in the directory.  Entries starting with 0xE5 are
        invalid.

*/

cull_directory()
{
        int     index;

        Puts("Finding all valid directory entries.\n");
        total_files = 0;
        for (index = 0; index <= dpbptr->drm; ++index) {
                if (directory[index].et != 0xE5) {       /* valid entry */
                        dir_point[total_files] = index;
                        total_files += 1;
                        }
                }
        Printf("Total files to be checked: %d\n",total_files);
}

/*      ------------------------------------------------------------    */

Main()
{
        Puts("Display files? (y/n): ");
        ch = ToUpper(getchar());
        if (ch=='Y') {
                display = TRUE;
                Puts("\bYes\n");
                }
          else {
```

```
		display = FALSE;
		Puts("\bNo\n");
		}

	Do {
		Puts("Crosscheck what drive: ");
		ch = Toupper(getchar());
		Printf("\b%c",ch);
		} while ( ch < 'A' || ch > 'P' );
	drivenum = ch - ('A');
	Printf("\nLoading directory from drive: %c\n",
			drivenum + ('A'));
	if (load_directory()) {
		cull_directory();
		Puts("Starting directory cross check.\n");
		compare();

Puts("Cross check completed.\n");
		}
	}

/*	-------	End of Crosscheck ------------------------------------		*/
```

Listing 2. The HBIOS() function

```
;
;	Modified bios call routine for BDS C.  The original for
;	this routine is from the BDS C - DEFF2.ASM file.
;

	maclib	cmac
	maclib	bds


	direct
	define	hbios
	enddir


;
; HBios:		for the sector translation routine
;
;	bios(n,c,d)
;
; Call to bios jump table routine n, with BC set to c, DE set to d.
; return value returned in hl to caller.
; n=0 for boot, n=1 for wboot, n=2 for const, etc.
;
	prelude hbios

	call arghak
	push b
	lhld base+1		;get addr of jump table + 3
	dcx h			;set to addr of first jump
	dcx h
	dcx h
	lda arg1		;get function number (1-85)
	mov b,a			;multiply by 3
	add a
	add b
	mov e,a			;put in DE
	mvi d,0
	dad d			;add to base of jump table
	push h			;and save for later
	lhld arg2		;get value to be put in BC
	mov b,h			;and put it there
	mov c,l
;
	lhld arg3		;get value to be
	xchg			;and put it there
;
	reloc <lxi h,>,hrtadd	;where call to bios will return to
	xthl			;get address of vector in HL
	pchl			;and go to it...
hrtadd:	pop b			;all done. return value is in HL
	ret			;and return to caller

	postlude hbios
```

CP/M to North Star DOS File Transfer

by David Yates

Most microcomputer users soon encounter problems with their systems. That is, the initial solution is not in the manuals or literature. The problem to which I refer here was that I had written a substantial quantity of educational software in Basic on a Commodore machine, but wished to use it on my Horizon when I returned home to Australia.

The initial step was to link the Commodore 8032 to a Texas Instruments cassette-based terminal and LIST the program to it. The terminal read the ASCII character string from the IEEE-488 interface and stored it on cassette. This cassette was then read by a high-speed reader connected to a serial port on a North Star Horizon. The ASCII file produced was stored by a CP/M-based program as a text file on disk under CP/M, and the disks were easily transported to Australia.

My intention was to modify the programs to run under both Microsoft Basic and North Star Basic. (Although the former is very similar to their Commodore Basic, I prefer the latter for a number of reasons). The use of WordStar allowed the addition of line feeds at the end of each line, as well as the global "correction" of several features of the original programs. This allowed their loading and possible running under Microsoft Basic, since it accepts ASCII files. To convert the programs to a North Star compatible form under North Star DOS was a little more difficult, because North Star Basic relies on tokenized Basic files.

Two commercially available packages allow the conversion of an ASCII disk file to a memory disk file coded using standard North Star tokens and in a "runnable" form. One of these ("Matchmaker" by the SOHO group) provides an ALOAD enhancement to the modified North Star Basic running under CP/M. This allows the loading of an ASCII file from disk and effectively directs the character input routine so that the interpreter accepts input from a named disk file rather than the normal keyboard input routine. Once loaded, the program is a standard North Star Basic file. It may be written back to disk as such or may be saved again as an ASCII file. As an aside, it is worth noting here that the ALOAD enhancement provides an undocumented "batch" facility for North Star Basic. Most direct mode commands to the interpreter may be incorporated in the ASCII file either before or after the text of the program. If RUN is included after the program, the ALOAD will result in the program being loaded and executed. In addition, a file containing input data for a program may be prepared using a text editor, "RUN" placed as the first line in the file, and the file saved to disk as "datafile." With an appropriate program in memory, the command ALOAD "datafile" may be given, and the datafile will be read in. On encountering RUN in this file, the interpreter executes the program in memory and as it requires data, will take it from the datafile (until it is all read) rather than from the keyboard. After the datafile is exhausted, normal keyboard input resumes. These details are provided here as possibilities offered by a feature not even suggested by the authors of "Matchmaker" in their manual.

The other commercially available product to allow the use of North Star Basic under CP/M is the Infosoft "NS BASIC Interface Program." Included with the package is a program called NSENTER, which produces a North Star Basic file under CP/M from an ASCII program file. This file can then be loaded by the modified NSBasic. A third product, baZic by Micro Mikes, though a slightly enhanced North Star "look-alike" running under CP/M, apparently uses only "standard" North Star tokenized files.

There are numerous other features of these two packages, both good and bad, but these will not be discussed here. The point is that they each offer a painless way to convert an ASCII program file to a standard North Star Basic program file that could be run under either of the modified versions of North Star Basic under CP/M.

A particular problem may become apparent, for example, only after using WordStar and one of the modified North Star Basics under CP/M. Although the use of these software products together seems straightforward enough, the substantial drawback encountered is that there is a large amount of time involved in swapping between the word processor and Basic, and in the loading and saving of files. Without resorting to other time-consuming methods, it is not possible to automatically generate line numbers while preparing a program with WordStar. In addition, although the software environment described above allows the use of WordStar or other editors to modify programs, I could not use some of the other software tools I have attached to the floating point version of Basic running under North Star DOS. These tools, without which I feel almost lost, include N*BUS and N*SORT by SZ Systems. At the time, these were not available for use under CP/M, although by now both are available to run with baZic. There are also occasional problems with both versions of the modified North Star Basic under CP/M, which cause me to hesitate in using either regularly.

The answer to my problems was to transfer the tokenized Basic files from the CP/M disk onto a North Star DOS format disk. This is relatively simple for smaller files (less than 30 blocks), but for larger files it is complicated because it involves "cutting and pasting." The method is also open to operator error. At the time, I was not able to find a program to perform the task, and so decided to write one. As my Basic is far superior to my assembly language expertise, the program was written in North Star Basic (and has since been compiled using Allen Ashley's COMPILER for North Star). Recently, Micro Mikes has produced COPYALL, which transfers files either way between CP/M and North Star DOS.

The problem centered around two issues: The first was the finding and deciphering of the CP/M directory by the Basic program. The second was the use of the directory information to read the file, one byte at a time, and transfer it to a DOS file.

The position of the CP/M directory was easy to find.

Dr. David Yates, Botany Department, University of Queensland, St. Lucia, 4067, Queensland, Australia

Logical CP/M Sectors

	10	11	12	13	14	15	16	80	81	82	83	84	85
Physical North Star Sectors	40	42	44	51	53	60		320	322	324	331	333	
	45	47	49	56	58	65		325	327	329	336	338	
	41	43	50	52	54	61		321	323	330	332	334	
	46	48	55	57	59	66		326	328	335	337	339	

Table 1a—*Examples of the correspondence of physical North Star DOS records, with logical CP/M sectors for CP/M 2.2 on a Quad capacity North Star Horizon.*

Logical CP/M Sectors

	P	P+1	P+2	P+3	P+4
Relative Physical North Star Sectors	0	2	4	11	13
	5	7	9	16	18
	1	3	10	12	14
	6	8	15	17	19

Table 1b

*Relative positions of physical North Star DOS records in a group of 20 records corresponding to five CP/M logical sectors commencing at sector P, where P is a multiple of five. The absolute address of any DOS record corresponds to the sum of the relative value and P*4.*

There are references to the first two tracks on a disk in the manuals supplied with CP/M.

The approach used in getting ready to perform the transfer involves the creation, on the CP/M disk, of a DOS file of type 0, commencing immediately after the DOS directory area and occupying the complete disk. This may be accomplished with the DOS command

CR CPM,2 1392 4.

This creates a type 0 file called CPM on disk drive 2, which covers the entire disk (except the NS Directory). It does not write anything to the file. It is assumed at this stage that the CP/M file or files of interest exist on the disk in drive 2. At this point, Basic is executed. For the next step, it is necessary to have a disk in drive 1. This will be a NS DOS format disk with sufficient space available to take the file to be transferred from the CP/M disk.

Now, a rather curious step is necessary. A Basic program (*any* program) that is at least as large as the CP/M file of interest is NSAVED onto the target disk in drive 1. The reason for this step will be covered below, but at present we can assume that the reason for this is to reserve space on the disk. The name under which the file is saved should correspond to the name required for the file being transferred.

When these steps have been completed, CPMTODOS can be run. As it is listed here, it opens the appropriate files and then proceeds to read the CP/M directory from "CPM." This directory can contain as many as 64 entries of 32 characters each.

As may be seen from Listing 1, the first 8K of the file "CPM" are ignored, as this contains bootstrap and other information and is of no significance in the present context.

The directory data of initial interest includes the filenames. The user must select one of these for transfer to the DOS file. The remainder of the CP/M directory contains details of the size and location of the various portions of the file of interest. At this point, some of the differences between CP/M and DOS file-handling become apparent. DOS files occupy contiguous disk space, while CP/M files do not necessarily do so. This means that once one sector has been read from the CP/M file, it is necessary to "go and look for" the next sector. It will not normally reside next to the previous one. For example, a further complication occurs under DOS: the 20th record actually occupies the 20th record position on the disk, while under CP/M there is a "skewing factor" to be accounted for. The skewing factor is used to speed up disk access, since it allows the processing of information after the reading of one 512-byte record and before another. If logical sectors were to be contiguous, then a second one might be passed over by the read head before the first one was to be stored or processed.

In addition, one CP/M sector is equivalent to four (non-contiguous) DOS records. As a result of these complications, it is necessary to calculate which DOS physical records correspond to a particular CP/M logical sector. The program presented here will work for standard Lifeboat CP/M for D/Q North Star. A different skewing factor would require a modification only of the "sector equivalents" in the program.

The procedure used is as follows: assuming the CP/M logical sector required is P, then one must locate the first relevant DOS physical sector. This may be done with the expression

P1 = INT (P/5) *5*4

where P1 is the first DOS record represented in a group of 5 CP/M sectors. It is sufficient to note that with the skewing factor used, a group of 5 CP/M logical (and sequential) sectors corresponds to a group of 20 DOS records, and that the sequence of the DOS records required to be read in order to sequentially access the information in the CP/M file is as represented in Tables 1a and 1b. On the basis of the value of the difference between P and INT(P/5)*5, the appropriate first DOS (relative to the start of the 5 CP/M sectors) for the CP/M sector P may be calculated using the information in Table 1b. This relative value is added to the value of the first DOS record for the 5 CP/M sectors to obtain the starting address, in DOS records, of the CP/M sector of interest.

An example of this procedure is in order. If the CP/M directory entry indicates that the next sector of interest is 83, then the expression

INT (83/5)*5*4 (=320)

tells us that the group of 20 DOS records containing that sector commences at DOS record 320. The difference between 83 and INT(83/5)*5 is 13, and this indicates that the 13th DOS record in the group of 20 is the one corresponding to the start of CP/M sector 83. According to the sector equivalent table (Table 1b), the 13th DOS record relative to the start of the group of 5 CP/M sectors is in fact 11. The first absolute DOS record in CP/M sector is thus 320+11 or 331. The table also indicates that the three successive DOS physical records completing the CP/M sector are 336 (320+16), 332 (320+12) and 337 (320+17).

The end result is that the program locates the successive but noncontiguous portions of the CP/M file and transfers them to a contiguous North Star DOS file. In the context of transferring Basic programs, there is sense in this being a Type 2 file.

While it is possible for CPMTODOS to calculate the size of the file required to accommodate the transferred CP/M file and then to CREATE a file of the appropriate name, size, and type, there is a drawback in doing this. When a Basic program file is loaded by North Star Basic, not only is the disk directory checked for the existence of the desired file, but the interpreter also checks to see the actual size of the program to be loaded. The directory as listed by CAT (or by LI under the operating system) displays the amount of space allocated for the file, not the amount actually used. The actual size of the program is usually somewhat less than the space allocated. This actual size is also recorded in the directory on the disk, but it is not displayed or simply available to the user. It is, however, used by Basic in loading a program file. If a Basic file is CREATed (as opposed to NSAVEd) in Basic or under DOS, the byte in the directory entry containing the true size information remains unchanged (from ASCII 32 or " " on a freshly initialised disk). Unless by chance this byte is equal to or larger than the number of blocks actually occupied by the program on the disk, the interpreter, in trying to load the Basic program file, will respond with a PROGRAM TOO LARGE or MEMORY FULL ERROR. While it is possible to access the DOS directory from Basic and so build an appropriate directory entry for the file being transferred, this would add a further complication to the program. Instead, the device referred to earlier is used. Prior to commencing the transfer procedure, a Basic file (any Basic file) is NSAVEd onto the target disk under the name required to hold the transferred CP/M file. The important requirement is that this file be larger than the file to be transferred, so that the "true size" byte in the DOS directory is such that all the transferred files may be read by Basic. Basic does not mind if the "true size" byte is too large, but doesn't like it if it is too small.

Having transferred the desired Basic file to the DOS format disk, it is a simple matter to LOAD it into memory and RUN it. Providing that the syntax is standard North Star Basic, the program should run. SAVEing the program back to disk will update the directory to include the true size of the file.

While the procedure outlined above is most useful for transferring program files from CP/M to DOS format disks, it may be used for any other filetype just as successfully. For example, assembler source and other test files can be transferred between editors just as easily.

CP/M to North Star DOS file transfer

```
100 LINE#0,80,0
110 REM     'CPMTODOS'    TO READ CPM DISK
120 PRINT
130 PRINT
140 PRINT "THIS ROUTINE READS THE DIRECTORY AND CONTENTS OF A CPM DISK"
150 PRINT "AND ALLOWS THE TRANSFER OF SELECTED FILES TO NORTH STAR DOS FORMAT"
160 PRINT
170 PRINT" IT DOES NOT YET COPE WITH FILES OF MORE THAN ONE EXTENT ALTHOUGH"
180 PRINT" THIS WOULD NOT BE DIFFICULT TO INCLUDE"
190 PRINT
200 PRINT" WRITTEN BY DAVID YATES, UNIVERSITY OF QUEENSLAND"
210 PRINT"                          ST. LUCIA, 4067"
220 PRINT"                          QUEENSLAND, AUSTRALIA"
230 PRINT
240 PRINT
250 PRINT
260 PRINT
270 PRINT"                    PLEASE WAIT"
280 REM SET UP FOR 32 FILES MAXIMUM
290 M=32
300 DIM A(M,32),P(256),N(20)
310 B$=CHR$(8)+CHR$(8)
320 B$=B$+B$+B$
330 F1$="CPM,2" \REM NAME OF DOS FILE ENCOMPASSING CPM DISK CONTENTS
340 IF FILE(F1$)<>0 THEN GOTO 1290 \REM DIRECTORY FILE DOES NOT EXIST
350 F2$="TARGET"\REM NAME OF DOS FILE TO WHICH SELECTED CPM FILE IS TRANSFERRED
360 GOSUB 1010   \REM SET UP SECTOR EQUIVALENTS
370 S=8192       \REM FIRST ADDRESS AFTER CPM DIRECTORY
380 OPEN#1%0,F1$
390 WRITE#1%S,NOENDMARK
400 REM READ CP/M DIRECTORY ENTRIES
410 FOR F=1 TO M
420 IF F=17 THEN WRITE#1%512*21,NOENDMARK
430 FOR K=1 TO 32
440 READ#1,&A(F,K)
450 NEXT
460 NEXT
470 REM CP/M DIRECTORY READ
480 REM CP/M DIRECTORY READ
490 REM
500 REM NOW PRINT OUT CP/M DIRECTORY
510 PRINT
520 PRINT"        CP/M FILE DIRECTORY"
530 PRINT"        -------------------"
540 PRINT"              FILE                SECTOR AND OTHER"
550 PRINT"NO USER NAME                        INFORMATION"
560 FOR F=1 TO M
570 IF A(F,1)=229 THEN GOTO 650 \REM A NON-ENTRY
580 PRINTF,"> ",TAB(4),
590 PRINT#Z9,A(F,1),TAB(8),
600 FOR K=2 TO 32
610 A=A(F,K)
620 IF K<=12 THEN PRINT#Z9CHR$(A), ELSE PRINT#Z9 ,TAB(K*3-20),A,
630 NEXT
640 PRINT#Z9
650 NEXT F
660 F=F-1 \REM THIS MANY FILES IN CPM DIRECTORY
670 PRINT
680 PRINT "DO YOU WANT A FILE LISTING ALONE   . . . . . . . 1  "
690 INPUT "                OR A FILE TRANSFER AND LISTING . . 2  ",A
700 IF A=2 THEN F8=1 ELSE F8=0
710 INPUT "WHICH FILE . . . . . . . ?",I
720 IF F8=0 THEN GOTO 780
730 REM CHECK IF DESTINATION FILE  EXISTS
740 T=FILE(F2$)
750 IF T=-1 THEN PRINTCHR$(7)," DESTINATION FILE ",F2$," DOES NOT EXIST"
```

the friendliest cat around

BOBCAT

only $25.00

the **MASTER DISK CATALOG** program for CP/M File names

- creates, adds, updates, and deletes a filename catalog
- provides four report formats and three search routines
- numbers disks and provides titling in catalog
- prints disk contents for disk jacket
- 24 (22 OSBORNE) page MICROGUIDE* documentation
- 8" CP/M SSSD
- OSBORNE for single or double density
- 94K (78K OSBORNE) of really friendly compiled PL/I

REQUIREMENTS - CP/M 1.4 and up
- 56K and up memory
- two or more drives

U.S. residents .... $25.00 U.S.
Canadian residents .. $25.00 Cdn (Ont. residents add $1.75 pst)
Other countries ...... $30.00 U.S.
all post paid

NAME, NUMBER, EXP DATE
(MC FOUR EXTRA DIGITS)
Bank drafts, cert. checks, money orders, no personal checks please

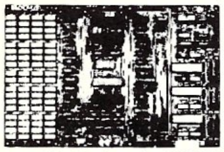
R&L Micro Consulting Services
6 Lipstan Ave.
Nepean, Ont.
K2E 5Z3
(613) 225-7904

R&L THE HOME OF THE BOBCAT

*MICROGUIDE is a (tm) of R & L Micro Consult Svs. CP/M is a tm of Digital Research. OSBORNE is a tm of Osborne Computer Corp.

CIRCLE 74 ON READER SERVICE CARD

The Mega Super Computer



Free Computer with a Purchase of a RAM Disk

- ★ Z80B Running at 5mhz, ★ Versatile CPM, ★ Math chip 9511 or 9512 AMD, ★ 48 Plus Ports on Mega Expander BUS, ★ CTC, DMA, ★ 2 Parallel Ports—with hand shaking, ★ Serial Ports—with or without hand shaking runs 150 to 19K2 Baud. Runs most terminals, printers, and modems.
- ★ NOW! CPM 3.0.
- ★ Hard Disk Interface hooks directly to Priam Drives.
- ★ Floppy Disk Controller—Handles Single Density IBM compatible disks and Dual Density 1 or 2 sided supports. 8" or 5¼" in various combinations—3 Drives equal over 4 Mega Bytes of Storage. WD2797.
- ★ 512K Bytes of 64K D-rams with parity configurable as a HIGH SPEED ELECTRONIC DISK or 8 banks of 64K for multi-user or countless other applications.
- ★ **All this on a state-of-the-art 4 layer card—with accurate documentation—10" x 15".**
- ★ Prices: Bare Board with Documentation $399
  - 64K Base System A & T $895
  - 512K Base System A & T $1595
  - CPM 2.2 System A & T $125
  - 512K Base Kit $1299
  - Manual $25
  - MPM $325

MEGA CO.
2318 S. Park Street, Madison, WI 53713 (608)255-7400

CIRCLE 47 ON READER SERVICE CARD

CP/M.D.

Diagnose and cure disk problems quickly and easily. Recover erased files, retrieve "bad sector" files, lock out bad sectors. CP/M.D. is a "doall" CP/M utility with dozens of menu-driven fast-acting functions. Requires 2 drives, 32K, CP/M 1 or 2.

SIMPL-SIMON SOFTWARE **$29.95**

Add $2.25 per order for shipping. Send check for prompt shipment. Ga. residents add 4% sales tax. Specify CP/M 8", Superbrain, or Xerox 5.25". CP/M is a trademark of Digital Research, Inc.

Write for information about CP/M database system, $89.95; word processing, $88.90; other inexpensive CP/M software.

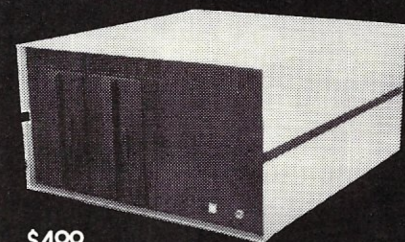
TELEPRINT, INC.
PO Box 10B, Sylvania, Ga. 30467

CIRCLE 271 ON READER SERVICE CARD

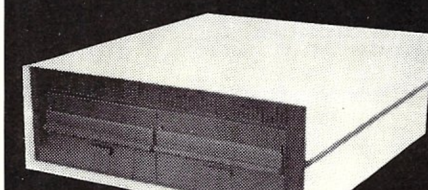
Main/Frames
Main/Frames

from
$175

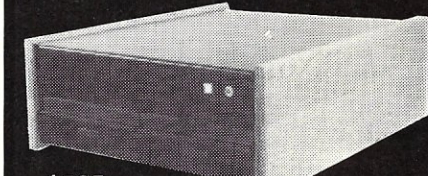
- 70 Models of Enclosures
- Assembled and tested
- Quasi-Coax Motherboards
- Power Supply
- Card cage and guides
- Fan, line, cord, fuse, power & reset switches



$499
8" Floppy Main/Frame



$275
8" Disc Enclosure



$525
Phase/80 8" Floppy Main/Frame

$370
Slim Line 8" Floppy Main/Frame

Write or call for our brochure which includes our application note: "Making micros, better than any ol' box computer"

INTEGRAND
8620 Roosevelt Ave. • Visalia, CA 93291
209/651-1203
We accept BankAmericard/Visa and MasterCharge

CIRCLE 49 ON READER SERVICE CARD

NEW!!! THE ELECTRIC MOUTH*

for S100, Elf II, Apple, TRS-80 Level II*

From $99.95 kit

Now — teach your computer to talk, dramatically increasing the interaction between you and your machine.

That's right: the ELECTRIC MOUTH actually lets your computer talk! Installed and on-line in just minutes, it's ready for spoken-language use in office, business, industrial and commercial applications, in games, special projects, R&D, education, security devices — there's no end to the ELECTRIC MOUTH's usefulness. Look at these features:

- ★ Supplied with 143 words/letters/ phonemes/ numbers, capable of producing hundreds of words and phrases.
- ★ Expandable on-board up to thousands of words and phrases (just add additional speech ROMs as they become available).
- ★ Four models, which plug directly into S100, Apple, Elf II and TRS-80 Level II computers.
- ★ Get it to talk by using either Basic or machine language (very easy to use, complete instructions with examples included).
- ★ Uses National Semiconductor's "Digitalker" system.
- ★ Includes on-board audio amplifier and speaker, with provisions for external speakers and amplifier.
- ★ Adds a new dimension and excitement to programming; lets you modify existing programs and games to add spoken announcements of results, warnings, etc.
- ★ Installs in just minutes.

Principle of Operation: The ELECTRIC MOUTH stores words in their digital equivalents in ROMs. When words, phrases, and phonemes are desired, they are simply called for by your program and then synthesized into speech. The ELECTRIC MOUTH system requires none of your valuable memory space except for a few addresses if used in memory mapped mode. In most cases, output ports (user selectable) are used.

Spoken Material Included

one	eighteen	at	dollar	inches	number	ss	c	t
two	nineteen	cancel	down	is	of	second	d	u
three	twenty	case	equal	it	off	set	e	v
four	thirty	cent	error	kilo	on	space	f	w
five	forty	400hertz tone	feet	left	out	speed	g	x
six	fifty	80hertz tone	flow	less	over	star	h	y
seven	sixty	20ms silence	fuel	lesser	parenthesis	start	i	z
eight	seventy	40ms silence	gallon	limit	percent	stop	j	
nine	eighty	80ms silence	go	low	please	than	k	
ten	ninety	160ms silence	gram	lower	plus	the	l	
eleven	hundred	320ms silence	great	mark	point	time	m	
twelve	thousand	centi	greater	meter	pound	try	n	
thirteen	million	check	have	mile	pulses	up	o	
fourteen	zero	comma	high	milli	rate	volt	p	
fifteen	again	control	higher	minus	re	weight	q	
sixteen	ampere	danger	hour	minute	ready	a	r	
seventeen	and	degree	in	near	right	b	s	

"Elf II" and "The Electric Mouth" are reg. trademarks of Netronics R&D Ltd. "Apple" is a reg. trademark of Apple Computer Inc. "TRS-80 Level II" is a reg. trademark of Tandy Corp.

Continental U.S.A. Credit Card Buyers Outside Connecticut
CALL TOLL FREE 800-243-7428
To Order From Connecticut Or For Technical Assistance, Etc.,
Call (203) 354-9375

NETRONICS R&D LTD. Dept MS
333 Litchfield Road, New Milford, CT 06776

Please send the items checked below:

- ☐ **S100 "Electric Mouth" kit $99.95**
- ☐ **Elf II "Electric Mouth" kit $99.95**
- ☐ **Apple "Electric Mouth" kit $119.95**
- ☐ **TRS-80 Level II "Electric Mouth" kit..... $119.95**

Add $20.00 for wired & tested units. All plus $3.00 postage & insurance. Conn res. add sales tax.

Total Enclosed $ ________

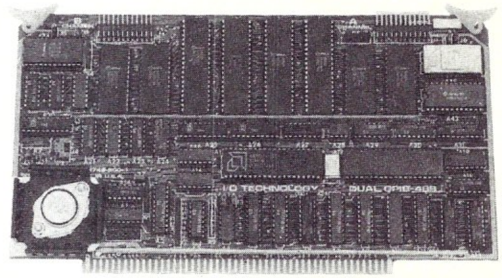
☐ **Personal Check** ☐ **Cashier's Check/Money Order**
☐ **Visa** ☐ **Master Charge (Bank No.** ________**)**
Acct. No. ________
Signature ________ **Exp. Date** ________
Print Name ________
Address ________
City ________
State ________ **Zip** ________

CIRCLE 155 ON READER SERVICE CARD

NO COMPROMISE ON P³*

I/O TECHNOLOGY S-100/IEEE-696 PLUG-INS

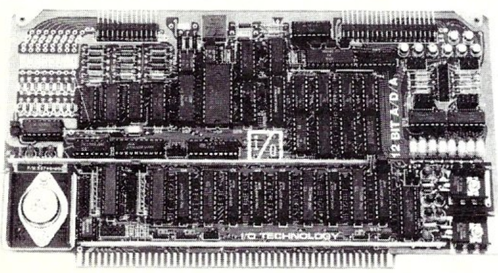
DUAL GPIB-488 INTERFACE BOARD



The **Dual GPIB-488,** is a Stand-Alone, independently controlled Dual Channel IEEE-488, 1978 Interface Controller. Interface Activity Modes such as Controller In-charge, Controller Assigned or Terminal Bus Slave and all Interface Functions, their Sub-Functions including Extended Functions, are handled by an On-Board 5MHz Processor and DMA Controller. 500K Byte Data rates are easily achieved with minimum host processor overhead.

Assembled and tested, P/N 52748-800-100 $650.00

12-BIT A-D-A CONVERTER BOARD

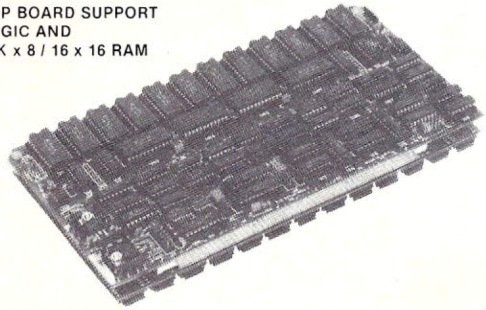


8 Channel A-D: 12 microsec. Conv. time, Programmable gain & Offset Voltage control, Diff./Single Voltage or Current Input.
8 Channel D-A: 2 microsec. Settling time, Bipolar V / Unipolar I Output, Programmable Reference levels, DUAL-PORTED Refresh RAM. 16 or 8-bit Data Transfers via Program I/O or Memory Mapped I/O, Extended Addressing and much more.

Assembled and tested, P/N 52748-900-100 $430.00

128K x 8 / 64K x 16 CMOS STATIC RAM MODULE

TOP BOARD SUPPORT LOGIC AND 32K x 8 / 16 x 16 RAM

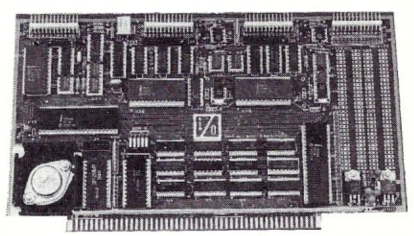


BOTTOM BOARD 96K x 8 / 48K x 16 RAM

150 nsec. Access, 2716 compatible RAM devices, Extended Addressing, Programmable wait cycles, Write protect and Bank select. Battery back-up capability. No wait cycles with fast 16-Bit Processors. The module's "Piggy-back" arrangement provides high density @ low input power yielding an improved MTBF and space utilization.

Assembled and tested, P/N 52748-650-128 $825.00

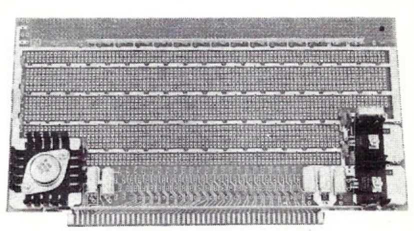
MULTI-FUNCTION I/O BOARD



Assembled and tested, P/N 52748-100-101 $325.00

Two Serial SYNC/ASYNC RS-232, TTL or Current Loop with XTAL controlled Dual Baud Rate Generator, Four Parallel Ports (Input, Output or Bi.), Three 16-Bit timers, 8-level Interrupt Controller and large Proto area with +5V, ±12V for custom applications.

PROTOTYPING BOARD KIT



Can be used for wire-wrap or solder prototyping projects. Comes complete with +5, ±12V Regulators, Heat sinks, 2 Bus Bars, Filter Capacitors and Manual.

Kit P/N 52748-400 $59.95

*PERFORMANCE, POWER, PRICE

New additions to our line of fine products are always on the drawing board. Please watch for the introduction of these new items as they become available.

POST OFFICE BOX 2119
CANYON COUNTRY, CA 91351
(805) 252-7666

I/O TECHNOLOGY ®

CANYON COUNTRY CALIFORNIA

VISA MasterCard

Calif. residents add 6½ % sales tax.
U.S. Domestic Price, F.O.B. Factory.
Prices and Specifications subject to change without notice.

CIRCLE 34 ON READER SERVICE CARD

```
760 IF T=-1 THEN STOP
770 OPEN#7%T,F2$
780 FOR B1 =17 TO 31
790 P=A(1,B1)
800 IF P=0 THEN EXIT 950 \REM THERE IS NO MORE TO THIS FILE
810 P=A(I,B1)
820 Q=FNP(P)
830 FOR P8=P6 TO P6+3
840 Q=S+(P1+N(P8+1))*512
850 WRITE#1%Q,NOENDMARK   \REM POINT IN SOURCE FILE
860 FOR P =1 TO 512
870 READ#1,&A
880 PRINT CHR$(A),
890 IF F8=1 THEN WRITE#7,&A,NOENDMARK
900 IF B=13 AND A<>10 THEN PRINT \REM PUT IN LINE FEEDS IF NECESSARY
910 B=A
920 NEXT \REM CHARACTER
930 NEXT \REM NORTH STAR DOS RECORD
940 NEXT \REM CP/M SECTOR
950 PRINT
960 PRINT " END OF FILE HAS BEEN TRANSFERRED"
970 PRINT CHR$(7)
980 IF F8=1 THEN WRITE#7,&1,&1,&1
990 CLOSE#7
1000 END
1010 REM SETUP SECTOR EQUIVALENTS
1020 N(1)=0
1030 N(2)=05
1040 N(3)=01
1050 N(4)=06
1060 N(5)=02
1070 N(6)=07
1080 N(7)=03
1090 N(8)=08
1100 N(9)=04
1110 N(10)=9
1120 N(11)=10
1130 N(12)=15
1140 N(13)=11
1150 N(14)=16
1160 N(15)=12
1170 N(16)=17
1180 N(17)=13
1190 N(18)=18
1200 N(19)=14
1210 N(20)=19
1220 RETURN
1230 DEF FNP(P)   \REM RETURN FIRST NORTH STAR RECORD
1240 P1=INT(P/5)*5*4   \REM START OF OVERALL NORTH STAR SECTOR GROUP
1250 P2=P*4        \REM POSITION OF FIRST OF FOUR NORTH STAR DOS RECORDS TO READ
1260 P6= (P2-P1)
1270 RETURN P6
1280 FNEND
1290 FOR K=1 .TO 12
1300 PRINT CHR$(7)
1310 NEXT
1320 PRINT
1330 PRINT
1340 PRINT "FILE    ", F1$," MUST EXIST AS A TYPE 0 FILE ON THE CPM DISK",
1350 PRINT " IN DRIVE 2"
1360 PRINT
1370 PRINT
1380 PRINT" EXIT TO DOS AND TYPE:-"
1390 PRINT"                          CR CPM,2 1392 4 D"
1400 PRINT
1410 PRINT " AND THEN RUN THIS PROGRAM AGAIN"
1420 PRINT
1430 PRINT
1440 PRINT
1450 PRINT
1460 END
```

North Star DOS as a CP/M .COM File

Run the North Star DOS as a CP/M program with a North Star-to-CP/M file transfer utility

by Allen Ashley

The following patch to North Star DOS 5.2 (ORG. 100H) allows that operating system to run as a command file under CP/M. The patch is given in the first listing, while a North Star-to-CP/M transfer utility is given in the second listing.

The specific system configuration addressed by the patch consists of North Star DOS 5.2 personalized for your system, and North Star CP/M 2.2 as distributed by North Star. If your version of CP/M is not from North Star, you will have to verify the locations of the two disk primitives and modify the patch program accordingly. The patch consists of three jumps: the first is to warm start entry of the DOS, and the others direct two disk operations done by the DOS to the corresponding operations in BDOS. The two disk primitives are found at locations 175H and 2ABH respectively in NS DOS 5.2, and at calculated positions DISKP1 and DISKP2 in the BDOS. Verify the code at the locations of DISKP1 and DISKP2 before executing the resulting file.

The code at 175H, and again at DISKP1 should read:

```
PUSH PSW
PUSH H
MOV A,C
```

The code at 2ABH, and again at DiskP2 should read:

```
MVI D,1
CALL XXXXX
```

Follow these steps to effect the patch:

1. Place the DOS 5.2 disk in drive 1 and boot your system.
2. Remove the North Star disk, replace it with your version of CP/M, and re-boot the system.
3. Save the original version of the DOS as a .COM file:

```
SAVE 13 NDOS.COM
```

4. Use your CP/M editor to create the small patch program, being sure to replace the value of MSIZE with that corresponding to your system.
5. Assemble the patch file with ASM to create PATCH.HEX. Note the locations of DISKP1 and DISKP2 during assembly.
6. Use DDT to bring in the NDOS file for patching:

```
DDT NDOS.COM
```

7. Disassemble the code at locations 175H and DISKP1 to verify that they are identical.
8. Disassemble the code at locations 2ABH and DISKP2 to verify that they are identical. If your version of CP/M does not reveal identical code in these locations, then you must search out the proper locations for DISKP1 and DISKP2, and go back to step 4.
9. Bring in the patch with DDT:

```
IPATCH.HEX
R
```

10. Press control/c to re-boot, and save the file:

```
SAVE 13 NDOS.COM
```

11. Execute NDOS and verify that you can get a directory from any North Star disk.
12. I have appended the file transfer utility to NDOS.COM and changed the first jump to 0E00H instead of 128H.

File transfer utility

The second listing presents a program that allows the transfer of files from a North Star DOS disk in drive 1 to a CP/M disk in drive B:. You may create the utility as a Type 1 file under NDOS, or append it to NDOS.COM. In either case you boot CP/M and execute NDOS created above. Remove the CP/M disk from drive A: and replace it with a North Star disk. Place a new CP/M-formatted disk in drive B:.

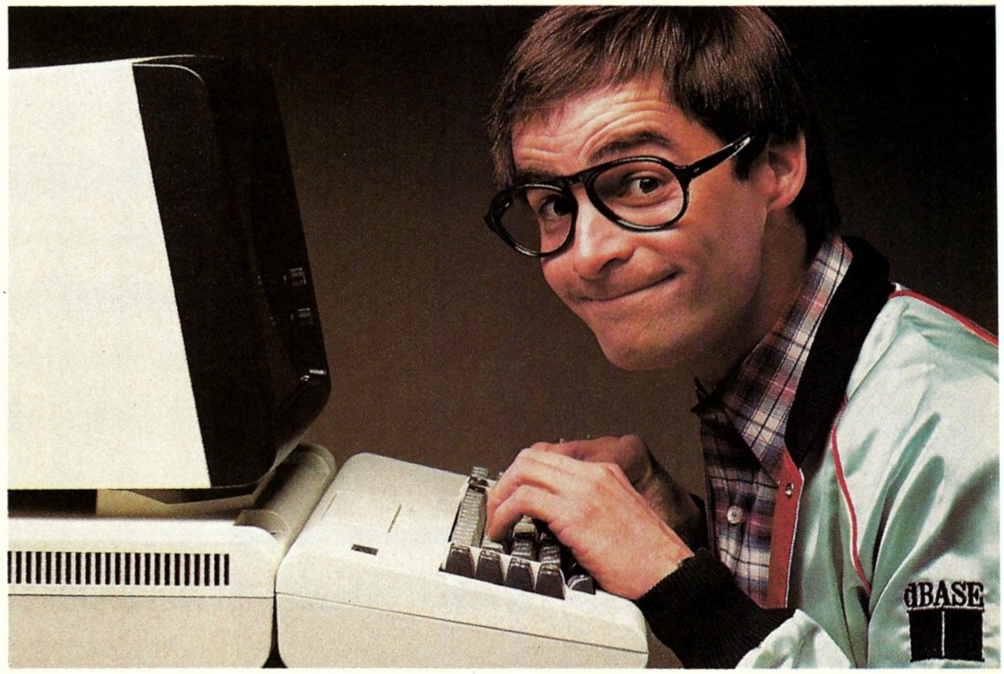
The transfer utility allows you to designate the extent to be assigned to the subsequent CP/M files. The default extent is assigned as .REL consistent with the PDS relocatable object format. You may reset the default extent by typing:

```
.EXT
```

where EXT is the new extent, which will hold until another default is designated. Thereafter type the names of files to be transferred and await the menu to signify completion. You should transfer your files in groups—say, the source code first with extension .ASM—then change extension to allow the transfer of data files and object files.

Allen Ashley, 395 Sierra Madre Villa, Pasadena, CA 91107.

All you dBASE II™ hotshots are about to get what you deserve.



You've written all those slick dBASE II programs.

Business and personal programs. Scientific and educational applications. Packages for just about every conceivable information handling need.

And everybody who sees them loves them because they're so powerful, friendly and easy to use.

But that's just not good enough.

Uh-uh.

Because now you can get the gold and the glory that you really deserve.

Here's how.

We've just released our dBASE II RunTime™ application development module.

And it can turn you into an instant software publisher.

The RunTime module condenses and encodes your source files, protecting your special insights and techniques, so you can sell your code without giving the show away.

RunTime also protects your margins and improves your price position in the marketplace. If your client has dBASE II, all he needs is your encoded application. If not, all you need to install your application is the much less expensive RunTime module.

We'll tell the world.

With your license for the dBASE II RunTime module, we provide labels that identify your program as a dBASE II application, and you get the benefit of all the dBASE II marketing efforts.

We'll also provide additional "how to" information to get you off and running as a software publisher sooner.

And we'll make your products part of our Marketing Referral Service. Besides putting you on our referral hotline, we'll publish your program descriptions and contact information in *dBASE II Applied*, a directory now in computer stores world-wide.

Go for it.

But we can't do any of this until we hear from you.

For details, write RunTime Applications Development, Ashton-Tate, 10150 West Jefferson Boulevard, Culver City, CA 90230.

Or better yet, just call (213) 204-5570. And get what you deserve today.

dBASE™ II

ASHTON·TATE

CIRCLE 13 ON READER SERVICE CARD

```
MAKRO ASSEMBLER AMA.2

0000            ; PATCH TO RUN NSDOS5.2 AS .COM FILE UNDER NS-CP/M
0000            ;
0000            ;FOLLOWING DESIGNATES YOUR MEMORY SIZE IN KBYTES
0000            ;
0030            MSIZE       EQU       48               ;CHANGE THIS TO SUI
T YOUR SYSTEM
0000            ;
C000            SIZEM       EQU       MSIZE*1024
0000            ;
0000            ;LOCATION OF DISK PRIMITIVES IN CP/M BDOS
0000            ;
B3D3            DISKP1      EQU       0B3D3H+SIZEM-48*1024
B3BF            DISKP2      EQU       0B3BFH+SIZEM-48*1024
0000            ;
0000            ;JUMP TO NDOS WARM START
0000            ;
0000                        ORG       100H
0100  C32801                JMP       128H
0103            ;
0103            ;PATCH DISK PRIMITIVES
0103            ;
0103                        ORG       175H
0175  C3D3B3                JMP       DISKP1
0178            ;
0178                        ORG       2ABH
02AB  C3BFB3                JMP       DISKP2
```

```
MAKRO ASSEMBLER AMA.2

0000            ; 'NORTH STAR TO CP/M TRANSFER'
0000            ;
0100            DOS         EQU       100H
010D            COB         EQU       DOS+0DH
0110            DCI         EQU       DOS+10H
011C            DLOOK       EQU       DOS+1CH
0125            DLIST       EQU       DOS+25H
0122            DCOM        EQU       DOS+22H
0128            WARM        EQU       DOS+28H
0000            ;
0000            CPORG       EQU       0
0005            ENTRY       EQU       CPORG+5
005C            FCB0        EQU       CPORG+5CH
0013            DELF        EQU       19
0016            CREATE      EQU       22
0015            WRITE       EQU       21
0010            CLOSE       EQU       16
0000            ;
000D            CR          EQU       0DH
000A            LF          EQU       0AH
0000            ;
0000                        ORG       DOS+0D00H
0E00  313411    OUTR        LXI       SP,STACK
0E03  214610                LXI       H,MESSG
0E06  CD950E                CALL      PLOOP
0E09  CDE20E                CALL      CI
0E0C  FE1B                  CPI       1BH
0E0E  CA0000                JZ        CPORG
0E11  FE03                  CPI       3
0E13  CA2801                JZ        WARM
0E16  DAA00E                JC        DOCAT
0E19  213411                LXI       H,INBUF
0E1C  77                    MOV       M,A
0E1D  CDDA0E                CALL      CO
0E20  23                    INX       H
0E21  0608                  MVI       B,8
0E23  CDE20E    INLOOP      CALL      CI
0E26  FE03                  CPI       3
0E28  CA000E                JZ        OUTR
0E2B  FE08                  CPI       8
0E2D  CAC00E                JZ        BAK1
0E30  FE7F                  CPI       7FH
0E32  CAC00E                JZ        BAK1
0E35  77                    MOV       M,A
0E36  CDDA0E                CALL      CO
0E39  23                    INX       H
0E3A  FE0D                  CPI       0DH
0E3C  CA480E                JZ        INDONE
0E3F  FE2C                  CPI       ','
0E41  CA480E                JZ        INDONE
0E44  05                    DCR       B
0E45  C2230E                JNZ       INLOOP
0E48            INDONE
0E48  3E0A                  MVI       A,0AH
0E4A  CDDA0E                CALL      CO
0E4D  360D                  MVI       M,0DH
```

```
MAKRO ASSEMBLER AMA.2
0E4F  213411                LXI       H,INBUF
0E52  7E                    MOV       A,M
0E53  FE2E                  CPI       '.'
0E55  CA780E                JZ        DEFEXTENT
0E58  FE20                  CPI       20H
0E5A  CA000E                JZ        OUTR
0E5D  FE0D                  CPI       0DH
0E5F  CA000E                JZ        OUTR
0E62  3E01                  MVI       A,1
0E64  CD1C01                CALL      DLOOK
0E67  D2E60E                JNC       GOTSOURCE
0E6A  21AA0E                LXI       H,SOUR
0E6D  CD950E    BEEP        CALL      PLOOP
0E70  3E07                  MVI       A,7
0E72  CDDA0E                CALL      CO
0E75  C3000E                JMP       OUTR
0E78  213511    DEFEXTENT   LXI       H,INBUF+1
0E7B  7E                    MOV       A,M
0E7C  32830F                STA       C1+1
0E7F  328410                STA       CC1
0E82  23                    INX       H
0E83  7E                    MOV       A,M
0E84  32860F                STA       C2+1
0E87  328510                STA       CC2
0E8A  23                    INX       H
0E8B  7E                    MOV       A,M
0E8C  32890F                STA       C3+1
0E8F  328610                STA       CC3
0E92  C3000E                JMP       OUTR
0E95  7E        PLOOP       MOV       A,M
0E96  CDDA0E                CALL      CO
0E99  23                    INX       H
0E9A  7E                    MOV       A,M
0E9B  B7                    ORA       A
0E9C  C2950E                JNZ       PLOOP
0E9F  C9                    RET
0EA0  3D        DOCAT       DCR       A
0EA1  6F                    MOV       L,A
0EA2  3E01                  MVI       A,1
0EA4  CD2501                CALL      DLIST
0EA7  C3000E                JMP       OUTR
0EAA  0D0A      SOUR        DB        CR,LF
0EAC  534F5552              DB        'SOURCE FILE ERROR',CR,LF,0
      43452046
      494C4520
      4552524F
      520D0A00
0EC0  78        BAK1        MOV       A,B
0EC1  FE08                  CPI       8
0EC3  CA000E                JZ        OUTR
0EC6  04                    INR       B
0EC7  2B                    DCX       H
0EC8  3E08                  MVI       A,8
0ECA  CDDA0E                CALL      CO
0ECD  3E20                  MVI       A,20H
0ECF  CDDA0E                CALL      CO
0ED2  3E08                  MVI       A,8
0ED4  CDDA0E                CALL      CO
```

```
MAKRO ASSEMBLER AMA.2
0ED7  C3230E                JMP       INLOOP
0EDA  C5        CO          PUSH      B
0EDB  47                    MOV       B,A
0EDC  AF                    XRA       A
0EDD  CD0D01                CALL      COB
0EE0  C1                    POP       B
0EE1  C9                    RET
0EE2  AF        CI          XRA       A
```

North Star DOS continued . . .

```
0EE3  C31001                JMP    DCI
0EE6  E5        GOTSOURCE   PUSH   H
0EE7  CD760F                CALL   MAKECPM
0EEA  E1                    POP    H
0EEB  5E                    MOV    E,M
0EEC  23                    INX    H
0EED  56                    MOV    D,M
0EEE  23                    INX    H
0EEF  4E                    MOV    C,M
0EF0  23                    INX    H
0EF1  46                    MOV    B,M
0EF2  23                    INX    H
0EF3  7E                    MOV    A,M
0EF4  E680                  ANI    80H
0EF6  320B11                STA    TYPE
0EF9  210002                LXI    H,200H
0EFC  17                    RAL
0EFD  DA020F                JC     DOUBLE
0F00  2601                  MVI    H,1
0F02  220711    DOUBLE      SHLD   SECTOR
0F05  EB                    XCHG
0F06  220911                SHLD   DISKAD
0F09  60                    MOV    H,B
0F0A  69                    MOV    L,C
0F0B  220511                SHLD   SIZE
0F0E  2A0511    READ1       LHLD   SIZE
0F11  7C                    MOV    A,H
0F12  B5                    ORA    L
0F13  CAC10F                JZ     DONEREAD
0F16  2B                    DCX    H
0F17  220511                SHLD   SIZE
0F1A  3A0B11                LDA    TYPE
0F1D  3C                    INR    A
0F1E  4F                    MOV    C,A
0F1F  0601                  MVI    B,1
0F21  78                    MOV    A,B
0F22  2A0911                LHLD   DISKAD
0F25  23                    INX    H
0F26  220911                SHLD   DISKAD
0F29  2B                    DCX    H
0F2A  114811                LXI    D,BUFFR
0F2D  CD2201                CALL   DCOM
0F30  DA000E                JC     OUTR
0F33  3A0811                LDA    SECTOR+1
0F36  87                    ADD    A
0F37  214811                LXI    H,BUFFR
0F3A  F5        WRITL       PUSH   PSW
0F3B  E5                    PUSH   H
0F3C  EB                    XCHG
0F3D  0E1A                  MVI    C,26
MAI FO  ASSEMBLER AMH.2
0F3F  CD0500                CALL   ENTRY
0F42  115C00                LXI    D,FCB0
0F45  0E15                  MVI    C,WRITE
0F47  CD0500                CALL   ENTRY
0F4A  B7                    ORA    A
0F4B  C25B0F                JNZ    WRITERR
0F4E  E1                    POP    H
0F4F  118000                LXI    D,128
0F52  19                    DAD    D
0F53  F1                    POP    PSW
0F54  3D                    DCR    A
0F55  C23A0F                JNZ    WRITL
0F58  C30E0F                JMP    READ1
0F5B  21610F    WRITERR     LXI    H,WRM
0F5E  C36D0E                JMP    BEEP
0F61  0D0A4350  WRM         DB     CR,LF,'CP/M WRITE ERROR',CR,LF,0
      2F4D2057
      52495445
      20455252
      4F520D0A
      00
0F76  213411    MAKECPM     LXI    H,INBUF
0F79  115C00                LXI    D,FCB0
0F7C  CDE80F                CALL   FCB2E
0F7F  216500                LXI    H,FCB0+9
0F82  3652      C1          MVI    M,'R'
0F84  23                    INX    H
0F85  3645      C2          MVI    M,'E'
0F87  23                    INX    H
```

North Star DOS continued . . .

```
0F88  364C       C3        MVI   M,'L'
0F8A  061E                 MVI   B,30
0F8C  23         ZLOOP     INX   H
0F8D  3600                 MVI   M,0
0F8F  05                   DCR   B
0F90  C28C0F               JNZ   ZLOOP
0F93  215C00               LXI   H,FCB0
0F96  E5                   PUSH  H
0F97  3602                 MVI   M,2
0F99  EB                   XCHG
0F9A  0E13                 MVI   C,DELF
0F9C  CD0500               CALL  ENTRY
0F9F  D1                   POP   D
0FA0  0E16                 MVI   C,CREATE
0FA2  CD0500               CALL  ENTRY
0FA5  3C                   INR   A
0FA6  C0                   RNZ
0FA7  21AD0F               LXI   H,OPENM
0FAA  C36D0E               JMP   BEEP
0FAD  0D0A4350   OPENM     DB    CR,LF,'CP/M OPEN ERROR',CR,LF,0
      2F4D204F
      50454E20
      4552524F
      520D0A00
0FC1  115C00     DONEREAD  LXI   D,FCB0
0FC4  0E10                 MVI   C,16
0FC6  CD0500               CALL  ENTRY
0FC9  3C                   INR   A
0FCA  C2000E               JNZ   OUTR
MAFPO ASSEMBLER AMA.2
0FCD  21D30F               LXI   H,CLOSM
0FD0  C36D0E               JMP   BEEP
0FD3  0D0A4350   CLOSM     DB    CR,LF,'CP/M CLOSE ERROR',CR,LF,0
      2F4D2043
      4C4F5345
      20455252
      4F520D0A
      00
0FE8  EB         FCB2E     XCHG
0FE9  E5                   PUSH  H
0FEA  010020               LXI   B,2000H
0FED  CD1F10               CALL  BLBUF
0FF0  E1                   POP   H
0FF1  13                   INX   D
0FF2  1A                   LDAX  D
0FF3  1B                   DCX   D
0FF4  FE3A                 CPI   ':'
0FF6  C2FF0F               JNZ   UNIT0
0FF9  1A                   LDAX  D
0FFA  D640                 SUI   40H
0FFC  77                   MOV   M,A
0FFD  13                   INX   D
0FFE  13                   INX   D
0FFF  23         UNIT0     INX   H
1000  012008               LXI   B,820H
1003  CD2610     GNAME     CALL  GTUPPR
1006  CA3410               JZ    FXTNT
1009  DA1B10               JC    NAMND
100C  77                   MOV   M,A
100D  23                   INX   H
100E  05                   DCR   B
100F  C20310               JNZ   GNAME
1012  CD2610     TERMNM    CALL  GTUPPR
1015  CA3410               JZ    FXTNT
1018  D21210               JNC   TERMNM
101B  3E03       NAMND     MVI   A,3
101D  80                   ADD   B
101E  47                   MOV   B,A
101F  05         BLBUF     DCR   B
1020  F8                   RM
1021  71                   MOV   M,C
1022  23                   INX   H
1023  C31F10               JMP   BLBUF
1026  1A         GTUPPR    LDAX  D
1027  13                   INX   D
1028  FE2E                 CPI   '.'
102A  C8                   RZ
102B  FE21                 CPI   21H
102D  D8                   RC
```

North Star DOS continued . . .

```
1030  3F                    CMC
1031  D0                    RNC
1032  91                    SUB    C
1033  C9                    RET
1034  CD1F10     FXTNT      CALL   BLBUF
1037  0603                  MVI    B,3
1039  CD2610     XTNTLP     CALL   GTUPPR
103C  DA1F10                JC     BLBUF
MAFRO ASSEMBLER AMA.2
103F  77                    MOV    M,A
1040  23                    INX    H
1041  05                    DCR    B
1042  C23910                JNZ    XTNTLP
1045  C9                    RET
1046  0D0A4E4F   MESSG      DB     CR,LF,'NORTH STAR TO CP/M COPY',CR,LF
      52544820
      53544152
      20544F20
      43502F4D
      20434F50
      590D0A
1061  0D0A                  DB     CR,LF
1063  534F5552              DB     'SOURCE=1.........'
      43453D31
      2E2E2E2E
      2E2E2E2E
      2E
1074  43502F4D              DB     'CP/M DEST=2'
      20444553
      543D32
107F  0D0A                  DB     CR,LF
1081  0D0A                  DB     CR,LF
1083  2E                    DB     '.'
1084  52         CC1        DB     'R'
1085  45         CC2        DB     'E'
1086  4C         CC3        DB     'L'
1087  3D204445              DB     '= DEFINE EXTENT FOR FILES',CR,LF
      46494E45
      20455854
      454E5420
      464F5220
      46494C45
      530D0A
10A2  5E433D42              DB     '^C=BACK TO NDOS',CR,LF
      41434B20
      544F204E
      444F530D
      0A
10B3  5E413D43              DB     '^A=CATALOG OF DRIVE 1 TO CRT ',CR,LF
      4154414C
      4F47204F
      46204452
      49564520
      3120544F
      20435254
      200D0A
10D2  5E423D43              DB     '^B=CATALOG OF DRIVE 1 TO LIST',CR,LF
      4154414C
      4F47204F
      46204452
      49564520
      3120544F
      204C4953
      540D0A
10F1  4553433D              DB     'ESC=BACK TO CPM',CR,LF,CR,LF
      4241434B
      20544F20
MAFRO ASSEMBLER AMA.2
      43504D0D
      0A0D0A
1104  00                    DB     0
1105  0100       SIZE       DS     2
1107  0100       SECTOR     DS     2
1109  0100       DISKAD     DS     2
110B  00         TYPE       DS     1
110C  2700                  DS     40
1134             STACK
1134  1300       INBUF      DS     20
1148             BUFFR
```

Add a Rescue Key to Your System

by Loren Amelang

Remember last time your system took a random walk down memory lane? Return didn't do anything; control-C had no effect. Those words you said as you reached for the reset button probably didn't help, either. Or how about when you were using the debugger and, for want of a comma, sent your processor off to what was intended as a breakpoint? Instead of reset, reboot, disk select, user number, debugger, program load, and whatever manipulation was needed to set up the test you muffed, wouldn't you prefer a single press of the rescue key and an instant return to the debugger prompt? Maybe you'd enjoy interrupting an editing task to use the hex number manipulations of your monitor—if you could return to the edit easily and find everything as you left it. Or how about a tool for exploring the inner workings of a program, giving you the program counter and register contents anytime you push the button, even if you've lost normal control of the system? Once you install the rescue key, the possibilities are endless.

I set out to write a trivial CP/M program to call my ROM-based monitor. When it occurred to me that an interrupt could cause a jump to a specific location, the rescue key was born. Since I have an S-100 bus system, the obvious first level was to wire a momentary contact switch to the INT* line, pin 73. Pulling this line to ground would cause the program counter to be pushed onto the stack, and send the processor to memory location 38H for its next instruction. I'd simply initialize 38H with a jump to my monitor ROM . . . and a press of the rescue key would cause an instant jump to my monitor. Or, if I was in DDT, pressing the rescue key would produce the DDT prompt, since DDT initializes 38H with its own jump vector.

If you have no monitor to jump to, Alan R. Miller's book, *8080/Z-80 Assembly Language* (Wiley, 1981), describes construction of both 8080 and Z80 monitors and gives complete listings for many different functions. Unfortunately, his monitors have no breakpoint and register-save capabilities, but his book does introduce building blocks for adding most any fantasy you can think of to your monitor. It features an appendix of 8080 and Z80 instructions, cross-referenced and sorted both by alphabetic mnemonics and numerical opcodes. It tells you precisely which flags are affected by which instructions, and even includes a handy table for figuring how many "pages" to specify when using the CP/M SAVE command!

One of the motivations for this project was to create a debugging tool that would not alter low or high memory locations, rearrange the stack, or overwrite the TPA when it was loaded. I chose to dedicate the top 4K of memory to my monitor and debug routines, leaving CP/M untouched in the rest of memory. Since the ROM version of my monitor uses low memory for jump vectors and puts its stack in the CP/M system area, it would have to be modified. The monitor's Move memory command would allow me to copy the ROM into RAM at a lower address, but in order to SAVE my copy to CP/M disk for later reloading in the top 4K of memory, I'd have to boot CP/M. Knowing that my boot loader overwrites the TPA when it loads, and that DDT also overwrites up to 1400H, I had to move the ROM image twice—once with the monitor Move command into mid-memory, and then with DDT once CP/M was booted.

I used the Substitute command to key in a short initialization and relocator routine at 100H, the beginning of the TPA (DDT's Assemble command wouldn't put in the Z80 instructions I wanted to use). Then I moved down the monitor image, carefully changing references to low memory so that it would store all jump vectors and stack above itself in the dedicated 4K top memory. With 38H initialized to the monitor's breakpoint entry point, pressing the rescue key would display the program counter, save all register contents, and produce the monitor prompt. But if the software-controlled interrupt enable happened to be DISabled when I pressed the rescue key, nothing at all happened! Seems every silver lining has a cloud somewhere . . .

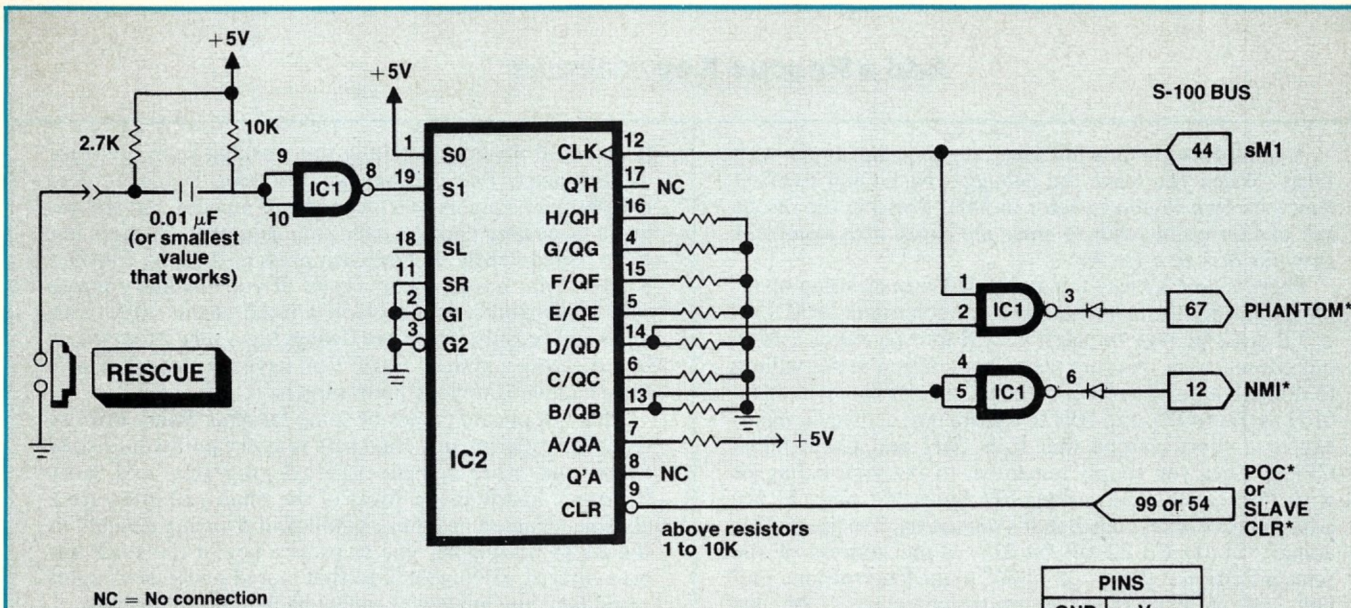
With an 8080 processor, you're limited to a sometimes rescue key—which is still better than none at all, especially when all it costs is one momentary switch and a bit of code doctoring. If you have an 8085, you're in luck: the TRAP input causes a nonmaskable interrupt to address 24H every time. Besides, TRAP is both edge- and level-sensitive; it must go high and remain high to be acknowledged, and will not be recognized again until the next time it goes low and then high. It has built-in immunity to switching glitches and noise. Note that servicing the TRAP interrupt will disable the software-maskable interrupts, but that the 8085 stores the interrupt enable status so you can put everything the way it was before jumping into your monitor.

My Z80 also has a nonmaskable interrupt, but to my dismay I found it jumps to 66H, right in the middle of CP/M's default file control block! How could I ever keep 66H initialized to a jump vector? The answer came from knowing why debuggers so often use location 38H for their re-entry jump vector. The machine code for ReSTart to 38H is OFFH—all ones—the result you get if you read nonexistent memory! So if 66H could be made to disappear by means of PHANTOM, perhaps a Z80 nonmaskable interrupt would then jump to 38H.

This was going to take a bit of hardware. I really hate the smell of burning components and the permanence of hardware mistakes, but I love what the rescue key allows me to do, so I pressed on. My design is mainly an outgrowth of the chips I found in my junk box. I'm sure there are cleverer ways to mechanize this idea, but this one takes only two common ICs, and works. If you can find room on an existing board anywhere in your system, you can steal power and support, and simplify construction. I built mine on a surplus S-100 card by itself, which meant investing a 7805 +5V regulator and adding the bypass capacitors they like to see ahead of and behind themselves.

The LS299 (Figure 1) is an 8-bit shift register with parallel load and master reset. The S-100 sM1 signal, pin 44, is used as clock, so that every instruction fetch causes the 299 to shift. Input line A is pulled up, and lines B-H are pulled

Loren Amelang, Box 24, Philo, CA 95466-0024



		PINS	
		GND	V_{CC}
IC1	74LS00	7	14
IC2	74LS299	10	20

Figure 1. Rescue key added circuitry.

down along with the serial data input. Mode select 0 (SB) is held high; so as long as mode select 1 (S1) is low each clock pulse will shift another Low into the 0 position of the register. When one section of the LS00 presents an inverted version of the rescue key line to S1, pulling it high, the parallel load function puts a High into position A and Lows into B-H. To prevent switch noise or bounce from repeating the parallel load after the cycle has begun, the switch is AC coupled into the LS00. Thus only the momentary pulse stored in the capacitor triggers the load. Pick the smallest capacitor value that will reliably trigger a parallel load in order to minimize the chance of glitches.

Once the High is in postion A of the shift register, each instruction fetch by the CPU will shift it up a step. When it reaches position B, it causes another section of the LS00 to pull the S-100 NMI* line, pin 12, active low. This sets an internal latch in the Z80; at the beginning of the last T-state of the present memory cycle, the latch is sampled and the interrupt response begins. The first M1 fetch of the interrupt ignores the instruction it reads and substitutes the interrupt routine. M2 and M3 are used to push the program counter to the stack. The next M1 cycle reads 66H, so we use position C of the LS299 to begin PHANTOM*. Since the ReSTart from 38H has the same M-cycles as the NMI, if we leave PHANTOM* low until the fourth M1 fetch it will prevent the program counter from being pushed to the stack by the ReSTart. Since the stack pointer still gets decremented, however, you'll still have to POP the extra space off the stack in order to get to the "real" program counter that NMI pushed. It seemed clearer to me to use a third section of the LS00 to NAND output C of the 299 with sM1 and use the result to pull PHANTOM* (pin 67) low only during the M1 fetch. With the rescue key released, the incoming Lows shift the High on out of the 299, and we're back to steady state.

In order to prevent spurious Highs in the 299 from ReSTarting the processor while it is being powered up, you'll have to tie SLAVE CLR* (pin 54) or POC* (pin 99) to the CLR line of the 299. To make the rescue key handy to use, I mounted it on my terminal. I used EIA line 11, which along with 18 and 25 is unassigned, to run the rescue signal along the serial cable to the CPU card. From there I jumpered it to S-100 pin 66, which is also not defined. (Pins 21 and 65 are officially not defined, but I found my system uses them.) With this arrangement there are no dangling wires, and the card with the interrupt circuit may be in any slot. The diodes shown on the NMI* and PHANTOM* lines let the LS00 simulate the open-collector output of an LS03, which I didn't happen to have on hand.

The first software task is to initialize 38H. At 100H I entered "3E C3 32 38 00 21" followed by <the address of my interrupt target> and then "22 39 00". To move the monitor from the loaded image at the bottom of memory to the dedicated 4K at the top, I entered "01" followed by <the number of bytes in the monitor image>, "11 00 F0" to set the move target, "21 80 01" to begin moving code at 180H, and "ED B0", the Z80 LDIR instruction. Listing 1 has an assembly listing of this code. If you have an 8080 or 8085, this bit of code will have to be expanded; you might want to use your assembler. I then entered a jump into my modified monitor's cold start routine to set up its stack and jump targets, having added jump back to here afterward. (Beware of CALLing a cold start routine that rearranges the stack; it will destroy your return address!).

This is a very powerful tool! It doesn't pose the physical dangers of a chainsaw or a pistol, but the anguish of writing garbage all over your directory tracks is not to be risked lightly.

Add a Rescue Key continued . . .

A final detail in this initialization code has to do with DDT. When you leave the debugger by GOing to 00, it leaves its own re-entry vector in 38H. Pressing the rescue key at that point *tends to erase the entire directory track from the disk in drive* A!

There is no easy way to patch DDT into cleaning up after itself, since the exit is simply a GO command to 00. Later I'll describe how to patch CP/M into resetting 38H to our chosen value on every warm boot. If you're not willing to do that, a quick answer is to initialize memory location 0FH to 00 (NOP), and 10H to a jump into dedicated memory to a short routine that fixes 38H and warm boots CP/M. Then you simply remember to exit your debugger with "GF", a GO to location 0FH. Since "G" and "F" are adjacent on the keyboard, that's not too hard to learn. This added "21 00 C3 22 0F 00 21", <the address of the reinitialization code>, "22 11 00" to the load routine. End your code with "C3 00 00" to return control to CP/M once the monitor is loaded.

Just above the monitor image, you'll need code to reinitialize 38H and warm boot CP/M: "3E C3 32 38 00 21", <the address of your interrupt receiving code>, "22 39 00 C3 00 00". And you need the actual code to which the interrupt jumps. At this point you have to adjust the stack. The RST instruction is equivalent to a 1-byte CALL, and has pushed a return address into the stack. You have to pop this off the stack and throw it away to uncover the "real" return address pushed into the stack by the NMI. In order to avoid altering the main Z80 registers, I used the EXX instruction, popped the unwanted address, and then EXXed it away into the alternate register. I then jumped to my monitor's breakpoint entry target, from which is saved all registers and warm boots for itself. When you're satisfied with the rescue key's destination, SAVE the memory image to disk.

Create a CP/M system which is 4K smaller than your total RAM. While you're at it, here's a chance to use the "Autoload" function. Patch in the name you SAVEd above, and make sure that your BIOS cold boot and warm boot routines end with a jump to the beginning of the CCP, an address ending in "00". If your BIOS boot routines jump to CCP+3 (an address ending in "03"), you'll have to change the low-order byte of the jump address from "03" to '00" to enable autoload. To prevent repetitive cycling of autoload, your monitor relocating code must end by jumping to "<CCP+3>". Your monitor will then be automatically loaded along with CP/M, and reloaded with every warm boot! This solves the problem of reinitializing 38H after exiting your debugger, but it creates a problem of overwriting the TPA on every warm boot, thus interfering with your chance to SAVE a memory image. The ultimate solution will require the BIOS cold boot routine jumping to "<CCP>" for autoload, and a separate warm boot routine that fixes 38H and jumps to "<CCP+3> to avoid autoload. I'll stick with "GF . . ."The Digital Research Application Note on "Autoload" was reprinted in the July/Aug 1982 issue of *Microsystems.*

Before you load up and gleefully press your new key, I hope you'll consider carefully this warning from the voice of experience: *This is a very powerful tool!* It doesn't pose the physical dangers of a chainsaw or a pistol, but the anguish of writing garbage over your directory tracks is not to be risked lightly. Before each press of the key, consider whether you have backup copies of your memory image and your disks. If you don't need them, eject the disks—especially if you are debugging a new program or investigating a system crash! You have no idea what may have happened to your jump targets.

Make duplicate copies of some familiar disks, and use them for experiments. Find safe re-entry points into your editor and other console-oriented programs, and write them in a handy place. Most of the times you interrupt a console-oriented program, you'll find it sitting dumbly in the BIOS waiting for you to press a key; if the stack has been altered, GOing back to that stored program counter could take you anywhere except back to your program.

Get familiar with the status registers of your disk controller, so that when you see "BDOS ERR ON B: BAD SECTOR" you can jump in and find out exactly where the problem is. You could even include "disk doctor" routines in your monitor image, reformat a sector on the spot, and jump back to the BIOS to try again without your application program knowing there had been a problem! I imagine you're already way ahead of me with your own ideas to incorporate, so I'll wish you happy experimenting! ■

```
F800 =          ITG:     EQU     0F800H  ;CHANGE TO ADDRESS OF YOUR
                ;                         INTERRUPT TARGET
0C00 =          CNT:     EQU     0C00H   ;Number of bytes in your monitor
F000 =          MONAD:   EQU     0F000H  ;Final address of your monitor
0180 =          MONSRC:  EQU     0180H   ;Place to start moving from
                ;
0100                     ORG     100H
                ;Initialize the interrupt vector
0100 3EC3       INIT:    MVI     A,0C3H  ;put a jump
0102 323800              STA     0038H   ;at 38H
0105 2100F8              LXI     H,ITG   ;store your int. target address
0108 223900              SHLD    0039H   ;at 39H & 40H
                ;
                ;Move the monitor
010B 01000C     MONMV:   LXI     B,CNT   ;Set the count of bytes to move
010E 1100F0              LXI     D,MONAD ;Set destination pointer
0111 218001              LXI     H,MONSRC ;Set source pointer
0114 EDB0                DB      0EDH,0B0H ;Z80 LDIR instruction does the move
0116 C300F0              JMP     MONAD   ;Jump to monitor's cold start address.
```

I wanted to create a debugging tool that would not alter low or high memory locations, rearrange the stack, or overwrite the TPA when it was loaded.

ERG/68000

MINI-SYSTEMS

- ☐ Full IEEE 696/S100 compatibility

HARDWARE OPTIONS

- ☐ 8MHz, 10MHz or 12MHz 68000 CPU
- ☐ Memory Management
- ☐ Multiple Port Intelligent I/O
- ☐ 64K or 128K STATIC RAM (70 nsec)
- ☐ 256K/512K or 1MB Dynamic RAM, with full parity (150 nsec)
- ☐ 5¼″-8″ DID, D/S floppy disk drives
- ☐ 5MB-40MB hard disk drives
- ☐ Full DMA Disk Interface
- ☐ SMD Interface
- ☐ ¼″ tape streamer
- ☐ 10 to 20 slot backplane
- ☐ 20 or 30A amp power supply
- ☐ Desk top or Rack mount cabinets

SOFTWARE OPTIONS

- ☐ 68KFORTH[1] systems language with MACRO assembler and META compiler, Multi-user, Multi-Tasking
- ☐ Fast Floating Point package
- ☐ Motorola's MACSBUG
- ☐ Unix[2]Type Operating System with C,PASCAL, FORTRAN 77, 68K-BASIC[1], CIS COBOL[4], RDBMS
- ☐ CP/M-68K[3] O/S with C, Assembler, 68K-BASIC[1], 68KFORTH[1], Z80 EMULATOR[1], FORTRAN 77
- ☐ Unix[2] System III with C, etc.
- ☐ VED68K[1] Screen Editor

Trademark [1]ERG, Inc.
[2]BELL LABS [3]Digital Research
[4]Micro Focus

30 day delivery
with valid Purchase Order
OEM prices available
For CPU, Integrated Card Sets
or Systems.

Empirical Research Group, Inc.
P.O. Box 1176
Milton, WA 98354
206-631-4855

ERG SOFTWARE AVAILABLE FOR:

CP/M-68K[3] O/S: 68K FORTH[1], 68K-BASIC[1]
VED68K[1] Screen Editor,
Z80 Emulator[1], FORTRAN 77

Motorola KDM Board: 68KFORTH[1]

Motorola MVME 110: 68KFORTH[1]

NEW PRODUCTS IN DEVELOPMENT

68000/16032 Processors

For The

VME Bus

10MHz ERG/VME 16032
/w 16082 MMU $1995
/w 16081 FPU $2495

12 MHz ERG/VME 68000
/w 68451 MMU $2495
/w 16081 FPU $2695

ERGFORTH[1] (68KFORTH[1]) Kernal and Compiler in resident ROM. Complete ERGFORTH[1] Systems Language available for both processors. Call for additional software availability.

AVAILABLE: 68000 —Now
16032 — August

Trademark [1]ERG, Inc. [3]Digital Research

Empirical Research Group, Inc.
P.O. Box 1176
Milton, WA 98354
206-631-4855

Book Review

by Chris Terry

For a good many years, both CP/M and the C language suffered from a dearth of *good* tutorials and reference manuals. Then, in 1981, came Thom Hogan's *Osborne CP/M User's Guide* (Osborne/McGraw-Hill, $12.99), which was a landmark in thoroughness and clarity for the CP/M operator (see the review in the Nov/Dec 1981 issue of *Microsystems*).

The C language was even worse off: until this year the only books from which to learn the language were Ritchie & Kernigan's reference book, *The C Programming Language* (Prentice-Hall, 1978) and C.T. Zahn's *C Notes* (Yourdon Press, 1979). The Ritchie & Kernigan book is, of course, the complete definition of the language and constitutes the standard. Zahn's different point of view is illuminating, but the book is in no sense a tutorial for the beginner. This year, however, three new books on C have appeared to fill the tutorial gap, and several on UNIX.

We hope to review at least some of these books later in depth; meanwhile, we want to draw attention to their availability.

C tutorials

The C Primer, *by Les Hancock and Morris Krieger.* Osborne/McGraw-Hill, 1982. 235 pp., $14.95.

This book is directed to people who know a little (though not much) about programming, but nothing at all about C. It assumes a UNIX environment, but is full of examples and admirably clear explanations that make it immaterial whether you are using Small C on a portable or full UNIX on a VAX—the explanations still make a lot of sense. The chapters on Arrays, Structures, and Pointers are particularly clear, and shed a great deal of light on the topics that cause most difficulty to the newcomer. The authors also give one of the clearest explanations of recursion that I have ever come across with a good example (computing a factorial number). The chapter on Input/Output and Library functions, shows clearly how to format strings and numbers (Ritchie & Kernighan are very terse in this area).

Learning to Program in C, *by Thomas Plum.* Plum Hall, 1 Spruce Ave., Cardiff NJ 08232, 1983. 372 pp., $25.

Plum Hall, Inc., is a training and consulting firm specializing in the C language and UNIX; this book is the textbook used in their courses on C. It assumes only some acquaintance with computers, not with any other programming language. The emphasis is on the portability of C programs and the suitability of C for real-time applications. There are copious examples and (as one would expect in a textbook) problems for the student; a large case-study problem illustrates important guidelines for software design. The answers to the problems are contained in an appendix. The book is based on several years' experience in teaching the courses.

C Programming Guide, *by Jack Purdum.* Que Corporation, 7960 Castleway Drive, Indianapolis, IN 46250, 1983. 250 pp., $17.95.

This book is directed to readers who have some knowledge of another programming language (especially Basic); many of the examples show the same function programmed in Basic and in C side by side for comparison. It is particularly suitable for users of personal computers, since examples in the first five chapters can be compiled with C compilers ranging in price from $19.95 (Small C) to $50 (Software Toolworks). An appendix lists 13 compilers running under CP/M or on the IBM PC. This book has more detail on disk file operations and file I/O than the other two, which is especially helpful in the CP/M environment. There is also a good chapter on Common Mistakes and Program Debugging.

CP/M tutorials

Inside CP/M, with CP/M-86 and MP/M2, *by David E. Cortesi.* Holt, Rhinehart & Winston, 1982. 371 pp., $25.95.

This is probably the best CP/M tutorial and reference book to appear so far. It has excellent guides to using CP/M and the associated utilities, including the Digital Research macroassembler, MAC. The tutorial on macros describes the creation of macros for commonly used operations, and their inclusion in a macro library. There is also accurate and detailed information on disk I/O operations and the manner in which CP/M stores data on the disks. CP/M and MP/M function calls are listed and explained very clearly.

CP/M Simplified, *by Jeffrey R. Weber.* Weber Systems, Inc., 8437 Mayfield Rd., Cleveland OH 44026, 1982. 316 pp., $13.95.

No prior knowledge of computers is assumed. An operator's and programmer's guide for users of a Radio Shack Model II running CP/M.

Mastering CP/M, *by Alan R. Miller.* Sybex, Inc., 2344 Sixth Street, Berkeley, CA 94710, 1983. 398 pp., $16.95

Another good, comprehensive programmer's guide to CP/M. Assembly language and macros are used throughout to illustrate various CP/M operations and to build a useful macro library. A fair knowledge of assembly language is essential.

Soul of CP/M, *by Mitchell Waite and Robert Lafore.* Howard W. Sams & Co., Inc., 4300 West 62nd St., Indianapolis IN 46268, 1983. 391 pp., $18.95.

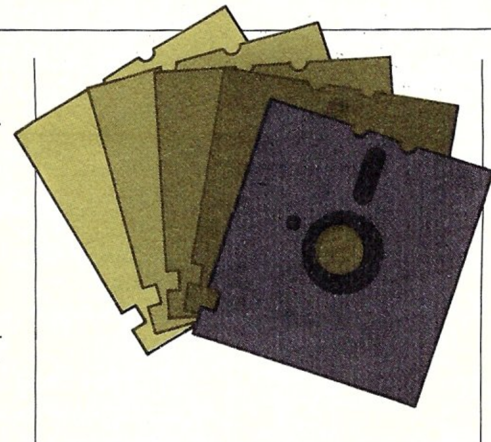
Aimed at Basic or other high-level language programmers who need to call assembly language subroutines, who handle disk records in a manner not accessible to the high-level language, or who just use assembly language routines to add speed to the programs. Examples accumulate to form a library of useful routines. Well written, enhanced by attractive illustrations, diagrams and flowcharts.

CP/M Assembly Language Programming, *by Ken Barbier.* Prentice-Hall Inc., Englewood Cliffs NJ 07632, 1983. 226 pp., $12.95 paperback; $19.95 hardcover.

This book assumes that the reader has no previous CP/M or assembly language experience. The author teaches the fundamentals of assembly language programming in the CP/M environment, using ASM, the standard 8080 assembler supplied with CP/M. Particularly valuable are the tips on how to preserve the user's environment.

Software Directory

Program name: ZSTEM, Z100 Smart Terminal Emulator
Hardware system: Zenith Z100 running ZDOS
Minimum memory: 128K Bytes
Language: 8088 Assembler code
Description: ZSTEM is a powerful, flexible, user-friendly asynchronous communications package for the Z100. ZSTEM adds terminal functionality for local or remote connection to a mainframe, another microcomputer, or a time-sharing service (e.g., SOURCE or CompuServe). ZSTEM supports parallel and serial printers and bidirectional disk file transfers. Fully user configurable: 45.5-38,400 baud, data/parity/stop bits, flow control protocols, half or full duplex, programmable softkeys/macros (any key can represent any string of characters for auto dialing, sign-on procedures, etc.), and temporary or permanent configuration save. Written in 8088 assembler code under ZDOS for fast, reliable operation.
When Released: April 1983
Price: $98.95 U.S.
Included with price: ZSTEM and documentation on 5 ¼″ disk with comprehensive manual.
Available from:
KEA SYSTEMS LTD.
Department ZMD-1
#311-811 Beach Avenue
Vancouver, B.C.
CANADA V6Z 2B5



604-687-2744
CIRCLE #306 ON READER SERVICE CARD

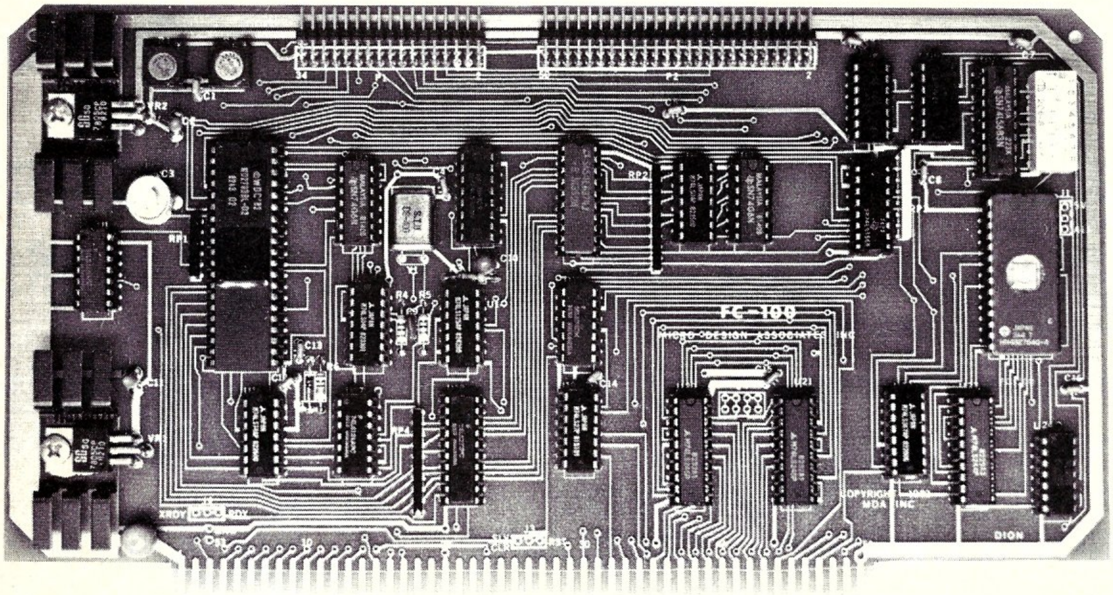
Program Name: MAILER for mailing list management
Hardware system: 8080/Z80, CP/M 2.2, CRT with addressable cursor
Minimum memory: 64K
Language: Object Code (Source PL/I 80)
Description: Complete mailing list management in a simple-to-use, versatile, stand-alone package which can interface with popular word-processing packages as well as dBASE II. A screen form makes data entry and update fast and easy and it automatically checks the state code, zip and phone. Record access is virtually instantaneous. The plain English selection menus and prompts eliminate confusion while costly mistakes are avoided with entensive error trapping features. The User's Guide is in tutorial form and includes an index. No programming skills at all are needed.

Mailer can read data in your existing address file as well as create files for use in letters, listings, etc. The records for output can be selected by many criteria. Labels are formatted to print 1, 2 or 3 across. Files can be scanned, sorted (name or zip), merged and purged.
When released: March 1983
Price: $100 includes SSSD 8″ disc and manual; inquire about other disc formats.
Available from:
dealers, or:
Maurizi Associates
1344 Fitch Way
Sacramento, CA 95825
916-486-2993
CIRCLE #312 ON READER SERVICE CARD

Program name: microSUB:MATH
Hardware system: CP/M Fortran compilers
Description: Engineers and scientists can now include numerical methods in their computer programs using a library of subroutines for microcomptuers. microSUB:MATH is a library of Fortran subroutines, supplied in both single and double precision, to perform tasks ranging from simple linear interpolation to

THE ULTIMATE DISK CONTROLLER

STATE OF THE ART



ALL SIGNALS IEEE 696

$299
Assembled and Tested

- Runs 5¼" and 8" drives simultaneously
- On board extended address bus driver (16 Meg)
- On board shadowable ROM (up to 8K)

- 8 bit or 16 bit data bus
- Low power consumption
- Compatible with all 1771, 179X & 279X software
- I/O mapped (does not take up any memory space)
- Very fast - runs at maximum transfer rate (.5 Meg B)
- Comes with free bios & instructions for your CP/M
- Reads & writes formats of the following systems: IBM 3740, System 34, IBM PC, Cromemco, Televideo, Morrow, Osborne, Kaypro, Lobo, Xerox TRS 80 models 1, 2, 3, 4, 12, 16 and most other soft sectored systems.
- Comes with format program & other utlities
  CP/M 2.2 Bios - Free
  CP/M 2.2 w/Bios - $150.00
  CP/M 3.0 w/Bios - $350.00

TO ORDER CALL:
MICRO DESIGN ASSOCIATES, INC.
25 S. 8th st. (314) 442-7714
Columbia, MO 65201

CIRCLE 85 ON READER SERVICE CARD

polynomial root finding and solving eigen value problems.
microSUB:MATH gives the user maximum flexiblility in writing programs. Subroutines, written in Fortran, are supplied as relocatable object files ready to be linked with the users program.

Available from:
foehn consulting
Box 5123
Klamath Falls, OR 97601
503-884-3023

CIRCLE #310 ON READER SERVICE CARD

Program name: Modula-2 Compiler
Hardware system: CP/M
Minimum memory: 60K
Description: Modula-2 is the advanced new programming language which evolved from Pascal. The module concept is the major enhancement. This allows programmers to develop large programs composed of small modules which are kept in a program library. Because each module is divided into a definition part and an implementation part, any module may be modified

Software Directory continued . . .

without affecting its interface to the rest of the program. Unlike earlier languages, Modula-2 does type checking between separately compiled program sections.

Modula-2's syntax is similar to Pascal's, but more systematic, making it easier to learn. Modula-2 has low-level facilities which allow direct access to the computer hardware and allow circumvention of strong type checking. This new language has multiprogramming features—signals, monitors, process creation.

The concept of procedure variables allows new possibilities of program control.

Modula-2 is the latest language designed by Nicklaus Wirth, who created Pascal in 1970.

Available: October 1

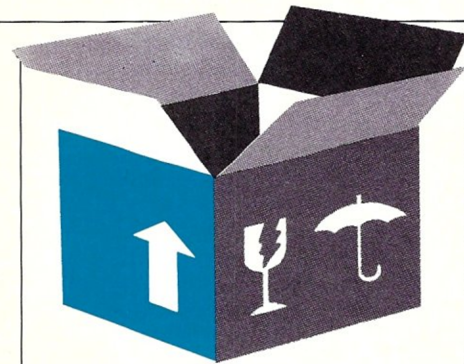
Price: suggested price, $100

Available from:

JRT Systems
45 Camino Alto
Mill Valley, CA 94941
415-388-9670

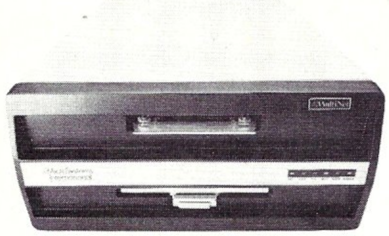
CIRCLE #313 ON READER SERVICE CARD

New Products



MULTINET Multiuser System

MicroSystems International, MA has announced a multiuser, multiprocessor system, MULTINET, that accommodates up to 8 users, each with his own processor and memory. A basic 2-user system with master processor, 2-user processors 5.25″ 20 MB Winchester disk, 13.4 MB cartridge tape drive, 1.2 MB floppy disk, distributed processing operating system and CP/M costs only $9,995. MULTINET system is easily expanded by plugging in additional S-100 processor-memory cards, one per user. MULTINET allows the intermixing and simultaneous operation of 8-bit (Z-80) and 16-bit (8086/8087) user processors within a single system.



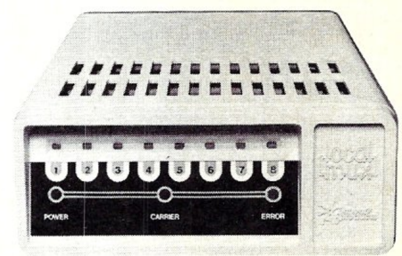
Each user runs his own CP/M program or share programs and data with other users. The heart of the MULTINET system is a distributed processing system that resides in the master processor and allows each user to have a dedicated processor while handling the sharing and management of common services such as disk files, tape and printers. Complete file/record updating, record locking and powerful automatic print spooling are provided. MULTINET offers a one-year full warranty for the entire system.

MicroSystems International
12 Mercer Rd.
Natick, MA 01760
617-655-9595

CIRCLE #300 ON READER SERVICE CARD

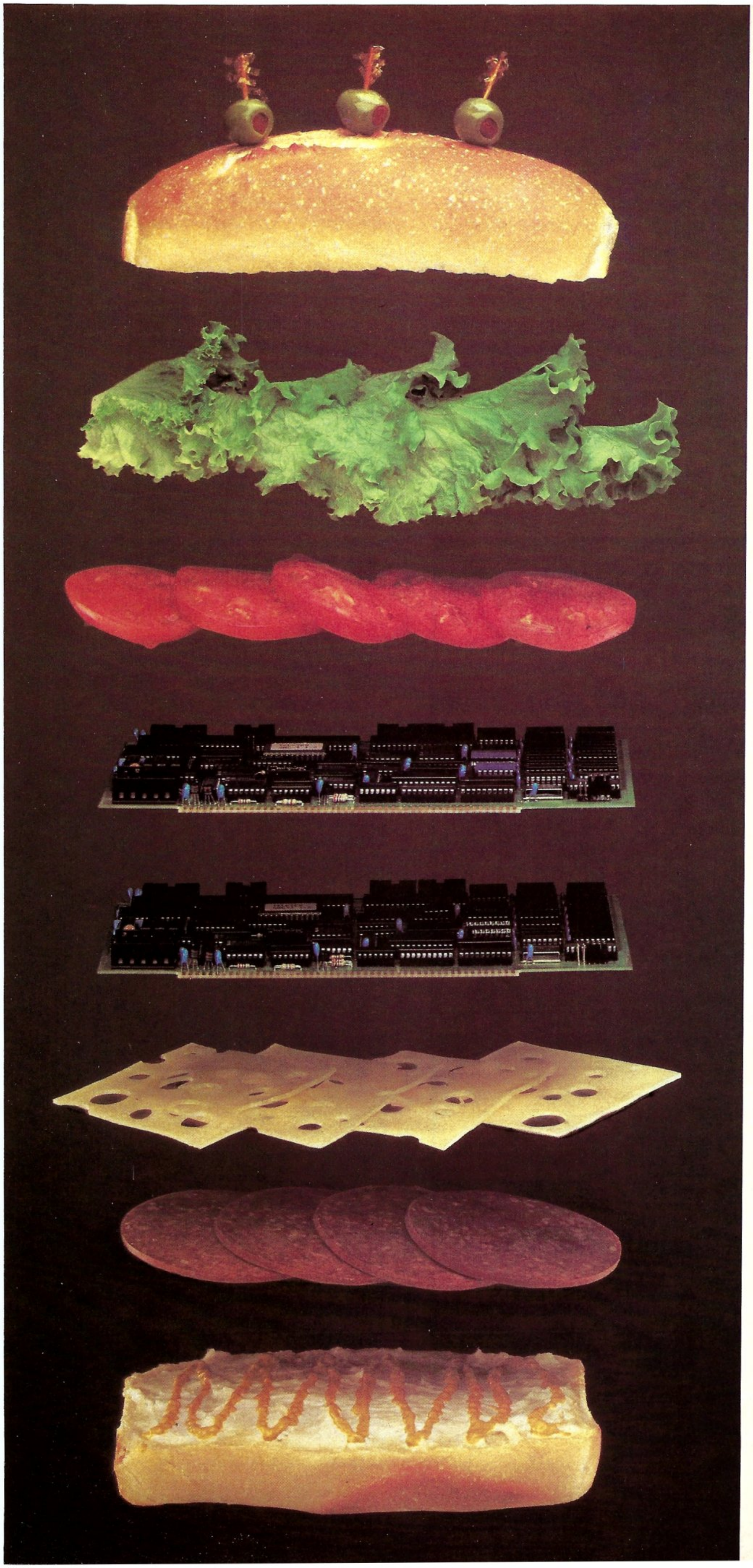
LocalMux Area Multiplexer

LocalMux is a combination multiplexer and short-haul modem that provides a cost-effective means of communicating up to 8 full-



duplex asynchronous or synchronous lines over 2 twisted-pair cables. Transmission range is dependent on characteristics of communication line. With 24-gauge twisted-pair wire at a terminal rate of 19,200 baud, the range is typically up to 500 feet. Distances of up to 2 miles are achievable using user-selectable fall-back speeds of 1/2, 1/4 and 1/8 maximum speeds. Additionally, the fall-back speeds can be used to compensate for marginal communication lines. Ports are completely transparent to character length, code, parity, stop bits, protocol and encryption. Data is communicated directly through at full speed with no flow control interposed. LocalMux has no transmission delay as well as no aggregate data limitations.

Price: $850
Network Products
P.O. Box 13239
Research Triangle Park, NC 27709
919-549-8210
CIRCLE #304 ON READER SERVICE CARD



Teletek's New Combo Could Make You A Hero!

The SBC-II could be just the right ingredient for your latest concoction. The newest member of Teletek's family of multi-user, multi-processing S-100 products, the SBC-II essentially combines, or "sandwiches" two Teletek SBC-I's into one board. The SBC-II provides the capability to support two users from one standard size IEEE-696/S-100 slave board.

The SBC-II maintains full performance for each user with an independent CPU (Z80A or Z80B), 64K RAM, Serial I/O, and FIFO communications port to the system master. The system integrator benefits by getting complete support for two users for the price of one board.

TurboDOS and MDZ operating systems will support combinations of SBC-I's and SBC-II's offering system design efficiency and flexibility never before possible.

If you're hungry for value and efficiency, order an SBC-II from Teletek. You'll love every byte.

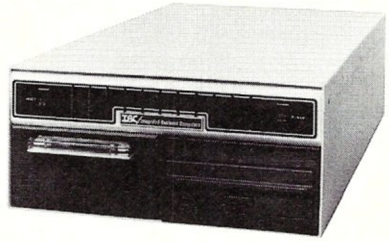
TELETEK

4600 Pell Drive
Sacramento, CA 95838
(916) 920-4600
Telex 4991834 TELETEK
Dealer inquiries invited.

CIRCLE 174 ON READER SERVICE CARD

Superfast Micro

The Ensign is an extremely fast 5-processor microcomputer system. The main CPU is a Motorola MC 68000 running at 8MHz with no wait states using memory management and ECC. Two additional CPUs handle all serial I/O for up to 32 users, thereby freeing the main CPU from communications overhead. The fourth CPU supervises all disk and tape I/O. The fifth CPU is used for memory management. The Ensign supports the OASIS-16 and UNIX operating systems. The Ensign is available in either a desk-top or rack-mounted cabinet and supports up to 8 MB of main memory



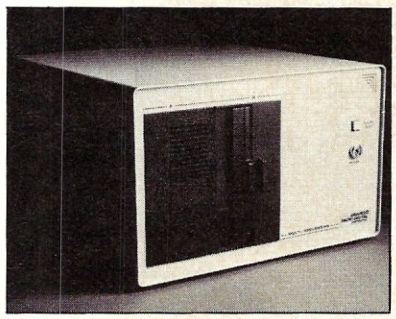
(up to 512 KB per user), a capacity of over 1000MB of SMD disk memory, and cartridge or 9 track, reel-to-reel magnetic tape units. The Ensign sells for $25,000 for a 16-port system with 1 MB of memory, 85 MB disk storage, 1 MB of floppy disk storage and a Centronics port.

IBC/Computers Integrated Business
21592 Marilla St.
Chatsworth, CA 91311
213-882-9007

CIRCLE #301 ON READER SERVICE CARD

Multiuser S-100 System

Advanced Digital Corporation has announced its "SUPER-SYSTEM," S-100-based computer using a Master-Slave concept to provide each user with his own dedicated Z80 CPU. By giving each user his own memory and I/O, the system all but eliminates the degradation in response times usually found in traditional configurations. The "SUPER-SYSTEM" can be configured as a single-user system running on the Advanced Digital's single-board computer, the SUPER-QUAD, which acts as the Master processor and communicates directly to the disks and peripherals. SUPER-QUAD includes a Z80-A CPU; 64K of bank-select dynamic RAM, 2K or 4K of shadow EPROM: a 5.25" or



8" floppy disk controller; 2 serial and 2 parallel I/O ports; and a real-time clock. The board meets full IEEE-696 specifications and runs with CP/M, MP/M, CP/Net or TurboDos. The basic system also includes a power supply and chassis, 8-slot motherboard and 2 Shugart 801R single-sided drives. Delivered with the CP/M operating system, it lists for $2,675. OEM discounts are

VEDIT
CUSTOMIZABLE
FULL SCREEN EDITOR

The Professional's Editor for Program Development Word Processing Source Code Translations

Widely acclaimed as an editor, VEDIT has evolved to be much more. Only VEDIT offers the combination of a versatile full screen editor integrated with a powerful command language. For the first time you'll be able to perform complex, yet useful, text manipulations that are virtually impossible with other editors or word processors. Plus, its customizability and hardware support ensure that VEDIT will be perfectly matched to your individual needs and to any microcomputer you are ever likely to own.

With two modes of operation, VEDIT never compromises its speed or ease of use for its power and sophistication. As one reviewer (Bradford Thompson, BYTE) wrote: 'If this review gives you an appetite for simplicity while editing, then VEDIT is well worth considering.' Its command language, based on TECO, is virtually a text oriented programming language, allowing command macros to be created, loaded and saved on disk. Yet its simplicity allows even a novice to perform tasks beyond the capabilities of any word processor.

VEDIT cuts programming time in half - with multiple file handling, macro capability and special features for Pascal, PL/1, 'C', Cobol, Assembler and other languages. And it can help with source code translations (example ZILOG to/from INTEL translator macros are included). A complete line of translators will be available by the year's end.

Word processing is a snap with word wrap, paragraph and print functions. Command macros free you from tedious search/replace operations. Hundreds of search/replace on dozens of files can be performed by VEDIT without waiting or intervening.

VEDIT easily configures to your favorite keyboard layout. Use any function or cursor keys you wish. It optimally supports nearly every 8080, Z80 and 8086 computer.

Go ahead and expect a lot from VEDIT. Its performance and our support will satisfy your most exacting needs.

To order, please specify your 8080, Z80 or 8086 microcomputer, operating system and disk format.

COMPARE VEDIT'S FEATURES

- True Full Screen Editing
- Horizontal scrolling
- Edit files one disk in length
- Automatic Disk Buffering
- Compact (only 16K) and Fast
- Display of line and column #
- Set/Goto text markers
- 'Undo' key to restore line
- Automatic Indent/Undent
- Adjustable Tab positions
- Repeat function key
- Text Move and Copy
- 10 Scratchpad Buffers
- Load/Save buffers on disk
- Powerful command macros
- Directory display
- Edit additional (small) files simultaneously
- Insert another disk file
- Unlimited file handling
- Recovery from 'Full Disk'
- Change disks while editing
- Word wrap, format paragraph
- Simple Printing
- 150 page indexed manual
- Startup command file
- Menu driven installation
- Program CRT function keys
- Support newest CRT terminals
- Support smart CRT functions
- Flexible Memory Mapped support
- Customizable keyboard layout

IBM PC, Displaywriter ° Zenith Z100 and Z89 ° NEC APC ° DEC Rainbow and VT180 ° Televideo 802
TRS-80 I, II and 16 ° Xerox 820 ° Apple II Softcard ° SuperBrain ° NorthStar
MP/M ° CP/M-86 ° MP/M-86 ° Concurrent CP/M-86 ° Cromix ° Turbo DOS ° MSDOS ° PCDOS

VEDIT - Disk and Manual
8080, Z80 or IBM PC $150
CP/M-86 or MSDOS $195
Manual only $18

CP/M and MP/M are registered trademarks of Digital Research Inc. Apple II is a registered trademark of Apple Computer, Inc. MS-DOS and Softcard are trademarks of Microsoft. TRS-80 is a trademark of Tandy Corporation. IBM is a trademark of International Business Machines.

CompuView
PRODUCTS, INC.

1955 Pauline Blvd., Suite 200 ° Ann Arbor, Michigan 48103 ° (313) 996-1299

CIRCLE 35 ON READER SERVICE CARD

available. The Slave boards provide each user with his own dedicated CPU, 4 serial and 2 parallel interface ports, 2K (expandable to 4K) of EPROM and 64K expandable to 128K of bank-switchable RAM for the additional cost of just $650.

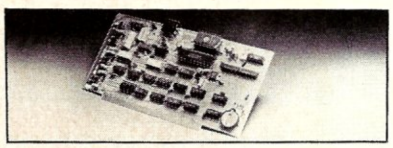
Advanced Digital Corporation
12700-B Knott Ave.
Garden Grove, CA 92641
714-891-4004

CIRCLE # 303 ON READER SERVICE CARD

S-100 EPROM Programmer and Calendar/Clock Board

TimeEPROMmer (TP-100) is unique in that the calendar/clock and the programmer are both on one board and independently addressable. The EPROM Programmer supports 2708, "16, "32, "32A and "64 type single voltage EPROMs. Port mapping is implemented so that no memory space is being used. Software control of voltages allows remote exchange of EPROMs via extender harness (not included).

CP/M software is executable from menu and allows read, verify, programming, disk transfers, etc. Personality headers are used for device



compatability with future EPROMS up to 28 pins. The calendar/clock has a socketted lithium battery and CMOS LSI implementation and allows the capability to time stamp sign-on and monitor activities with time domain.

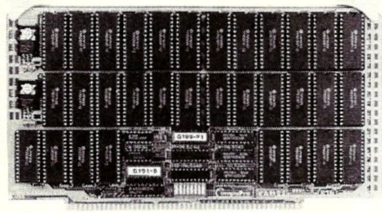
Includes software and documentation. Bare board: $75; kit: $195; A&T fully guaranteed: $295.

Optronics Technology
PO Box 81
Pittsford, NY 14534
716-377-0369

CIRCLE # 302 ON READER SERVICE CARD

High-Speed, Low-Power RAM Board

A high-speed (12MHz), low-power (less than 4 watts) static RAM S-100 board compatible with both 8- and 16-bit computer systems has been introduced by CompuPro. The new RAM 22 provides up to 256K of storage addressable as either one 256K x 8 or one 128K x 16 block. List price: $2,495, 18-month end-user limited warranty; a 36-month warranty plan is available on the Certified System version, priced at $2,695.



CompuPro
3187C Airway Ave.
Costa Mesa, CA 92626
714-641-3579

CIRCLE # 305 ON READER SERVICE CARD

Correction

The price for the Dual Systems System 83/80 was erroneously quoted in the August issue as $2,990. The correct price s $20,990.

Microsystems Mart

I WILL BEAT ANY COMPETITOR'S PRICE PROVIDED IT IS NOT BELOW MY COST.
TRY TO BEAT THESE IC PRICES:

DYNAMIC RAM		
64K	200 ns	$5.35
64K	150 ns	5.35
64K	120 ns	5.90
16K	200 ns	1.40
EPROM		
2764	250 ns	$6.35
2732	450 ns	4.05
2716	450 ns	3.25
2532	450 ns	4.60
STATIC RAM		
6264LP-15	150 ns	$60.00
6116P-3	150 ns	5.00
2114	200 ns	1.45
Z80A FAMILY		
CPU, CTC, or PIO		$3.39
DART		7.99
DMA or SIO/0		12.50

MasterCard/VISA or UPS CASH COD
Factory New, Prime Parts

Prices 6/1/83

MICROPROCESSORS UNLIMITED
24,000 South Peoria Ave.
BEGGS, OK. 74421
(918) 267-4961

Prices subject to change. Call for volume prices. Subiect to available quantities. Shipping & Insurance extra. Cash discount prices shown.

S-100 COLOR GRAPHICS MUSIC SYNTHESIS DUAL I/O PORTS

SINGLE BOARD DESIGN USES ONLY 22 IC'S

COLOR GRAPHICS

- -TMS 9918A Video Display Processor
- -On board RF Modulator
- -256 x 192 pixel resolution
- -NTSC composite video connector
- -16 unique colors
- -1 text and 3 graphic modes
- -16K of display RAM (4116's)
- -32 sprites for 3D simulation

COMPLEX SOUND GENERATION

- -AY-3-8910 Prog. Sound Generator
- -16 level amplitude control
- -3 programmable channels
- -Noise generator
- -Produces 3-note chords
- -Envelope shape/cycle control
- -Tones from 33.5Hz to 125KHz
- -Special sound effects

The AMUSIGRAF system is offered as an affordable package which includes the following:

- -Bare S-100 P.C. Board
- -AY-3-8910 data manual
- -TMS 9918A data manual
- -Complete user's documentation
- -CP/M* compatible 8" disk with test and demo software

ORDER by mail through: FORESIGHT TECHNOLOGY Write or call for more information

ORDER by phone through: ADCOM MKTG. SERVICES (513) 831-1561

AMUSIGRAF SYSTEM
Introductory offer
Visa $95.00 Master Card
Add $3.00 for C.O.D.
Ohio res. add 5.5% sales tax
Send certified check or money order for immediate delivery
Personal checks allow 2 wks.

* CP/M is a trademark of Digital Research

FORESIGHT TECHNOLOGY
1016 Huffman Ct.
Cincinnati, Ohio 45231

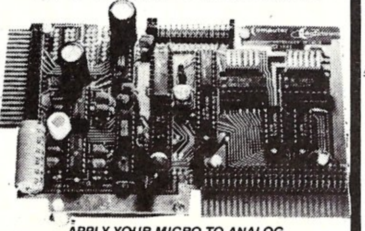
ISIS ⟷ CP/M®

Full bi-directional file transfer capabilities are provided in the ISIS-CP/M utilities package. Written in machine language and running under CP/M, these utilities permit the CP/M user to read or write files direct to/from an ISIS Diskette. They will run under any version of CP/M without regard to diskette density. The complete package is $250.00 including user's manual. Write for free brochure on other CP/M software.

CP/M is a registered trademark of Digital Research
ISIS is a trademark of Intel Corporation

SOUTHERN COMPUTER SYSTEMS, INC.
P.O. Box 3373A
Birmingham, AL 35255
(205) 933-1659

A/D 8 channels + D/A 8 channels



APPLY YOUR MICRO TO ANALOG SIGNAL PROCESSING & CONTROL

Connects to μ processor buss of most Z80 & 6502 computers including Timex, TRS-80, Apple, CBM 64. Our pin rearrangement feature eases interfacing.

FAST: 200,000 samples per sec with Z80A.

EASY TO PROGRAM: One PEEK chooses the channel, starts conversion & gets data. Routines provided.

SOFTWARE AVAILABLE: FFT in Z80, 256 points in one second; storage scope for Timex.

LOW COST: Assembled & tested $195. Cable $15. Please call or write for free catalog.

Computer Continuum
301 Sixteenth Avenue
San Francisco, CA 94118 (415) 752-6294

Automatic Micro

Smartboard™ RC100 - Remote Control Relay Interface
Reliable, flexible control of up to 16 bi-stable relay switches.

- S-100 compatible
- 16 channels relay control & status
- All inputs/outputs optically isolated
- Onboard 8035 CPU and 4 Status/Command registers allow easy control & communication with main CPU
- Relay state unaffected by power outages
- Zero voltage switching for extended relay contact life
- Port or Memory mapped addressing
- Fully Documented
- External 2716 or 2732 EPROM allow custom programming for application flexibility (Designed for GE RR8 remote control relays)

RC100 Remote Control Relay Interface $495.00
RR8 relay (ea) $17.95
DB-25 connector cable (ea) $24.95

Dealer inquiries invited!

Automatic Micro, Inc.
P.O. Box 20007
Columbus, Ohio 43220

CUSTOMIZE YOUR KEYBOARD

We maintain a large stock of blank keytops that can be custom engraved for:

★ **Video Display Terminals** ★
★ **Point of Sale Terminals** ★
★ **& Computers** ★

Replacement keytop kits for Word Processing Software packages configured for popular terminals.

OEM PROTOTYPES AND PRODUCTION RUNS

— Call or Write —

ARKAY ENGRAVERS, INC.
2073 Newbridge Road
Bellmore, New York 11710
(516) 781-9859 (516) 781-9343

VT100 PERFORMANCE FOR H/Z19

Get VT100 performance at fraction cost of VT100: $95 for HT100 firmware. Totally software compatible with VT100. Only limitations: normal and reverse video character attributes only, no programmable tabs, no smooth scroll, and no 132 column display.

You get:

- ANSI and DEC VT100 Escape Sequences
- VT52 Escape Sequences
- VT100 Style Setup Mode
- VT100 Graphics Characters
- Control Character Display Mode
- APL Char Set with Overstrike ($5 Option)

HT100 firmware extensively tested on DEC's video editors (EDT/KED) for several years. Included: 3 ROMs, Reference Manual, Installation Guide, APL keycap labels.

For information or ordering:

MicroFirm, Inc.
Rockefeller Center Station P.O.
P.O. Box 1276
New York, N.Y. 10185

Make check payable to: MicroFirm, Inc.
$95 for HT100, add $5 for APL Option
New York State residents add applicable tax.

8086/8087/8088 CROSS SOFTWARE PACKAGES

1 C cross compiler for the 8086. All facilities of the complete C language, including floating point for the 8087, are supported. Optionally, memory can be allocated for use with the 8088. Output is symbolic assembly language. The compiler is suitable for use in porting UNIX to the 8086.

2 Cross assembler/linker/librarian/downline loader for the 8086. Assembler input is an extension to that used by Intel. Loader output is a file in standard Intel hex format.

3 Simulator/debugger for the 8086. Capabilities include display, breakpoints, interpretive execution, as well as many others.

Host System: PDP-11 running RT-11, RSX-11M, UNIX/V6, UNIX/V7; or VAX-11 running VMS, UNIX/32V.

For additional information:

Advanced Digital Products, Inc.

1701 21st Ave. S., Suite 222 • Nashville, TN 37212
Phone (615) 383-7520

Intel is a trademark of Intel Corporation; UNIX and UNIX/32V are trademarks of Bell Laboratories; RT-11, RSX-11M, PDP-11, VMS, and VAX-11 are trademarks of Digital Equipment Corporation.

LIQUIDATION

Cash-pressed OEM is liquidating un-used **5.33 megabyte hard disk drives** for **$380 EACH**. These are Shugart SA1002 high reliability, 8" Winchester technology drives sold as-is in the original factory sealed double-packed shipping containers. Western Digital controller card for this drive $350. CP/M software with S-100 host schematic $75. Manual set for drive $10, shipping $12. Call **LIQUIDATORS** at 803-877-9828 or send check to 105 S. Main St., Greer, SC 29651.

classified ads

CLASSIFIED RATES: Per Word, 15 Word Minimum. REGULAR: $1.00. EXPAND-AD: $1.50. GENERAL INFORMATION: Prepayment discounts available. Payment must accompany order except credit card — Am. Ex., Diners, MC, VISA (include exp. date) — or accredited ad agency insertions. Copy subject to publisher's approval; must be typewritten or printed. First word set in caps. Advertisers using P.O. Boxes MUST supply permanent address and telephone number. Orders not acknowledged. They will appear in next available issue after receipt. Send order & remittance to: Classified Advertising, MICROSYSTEMS Magazine, 1 Park Avenue, New York, NY 10016. Direct inquiries to: Kenneth L. Novotny (212) 725-3927.

SOFTWARE

CP/SIM—Simulates CP/M on VAX/VMS, use CP/M programs on a VAX PROTO SYSTEMS, 1238 Josephine St., Berkeley, CA 94703. (415) 420-9579.

FOR THE OSBORNE 1. Four great games. Chess, Checkers, Backgammon and an all new adventure, Castle of Ohme. All are written in assembly language. $30 for all 4. Send check to: M-J Software, P.O. Box 5266, Richardson, TX 75083-0660.

ULTIMENU FOR DOS simplifies IBM's, PC-DOS to meet your needs. $24.95. DB/DC Software Associates, P.O. Box 4695, Manchester, NH 03108.

COMPUTER EQUIPMENT/SUPPLIES

COMPUTER PAPER—Save $$$: Lowest single carton prices. Free samples. Bob's charts, 405 East Third, Long Beach, CA 90802.

FOR SALE

FOR SALE IMS-8000 Z80, I/O 2S+1P, (2) 8″ Shugart 801, (4) 16K Static Ram, super reliable, $2800. Philip Van Ostrander, POB 15842, Philadelphia, PA 19103. (215) 462-4416.

ADVERTISERS INDEX

Four Answers To Your S-100, Multi-User Problems.

Intercontinental Micro Systems makes everything you need for S-100 bus multi-user systems, networks or single user systems.

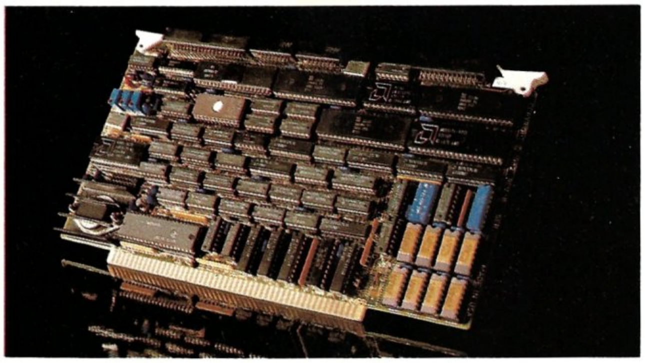
At a price that won't break your budget.

Quite simply, our single board computers, slaves, 256K memories and personality boards let you build a system *now*, not later. The hardware works, the software works, and the prices are what you'd expect from a company that uses the most advanced design, software and production techniques to keep costs down.

What you won't expect is the almost awesome sophistication of Intercontinental Micro System's products.

So stop messing around with multiple sourcing, hardware integration problems and software nightmares. Come to Intercontinental Micro and get it all — price, performance and delivery.

Read the specs, then call, write or circle the bingo number below. We'd be glad to send more information and help solve your S-100, multi-user system problems.

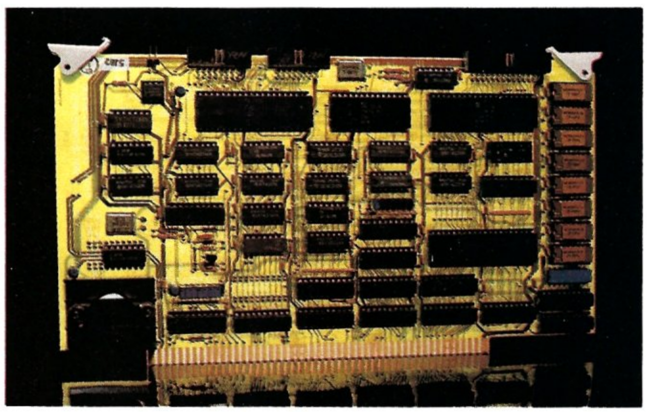


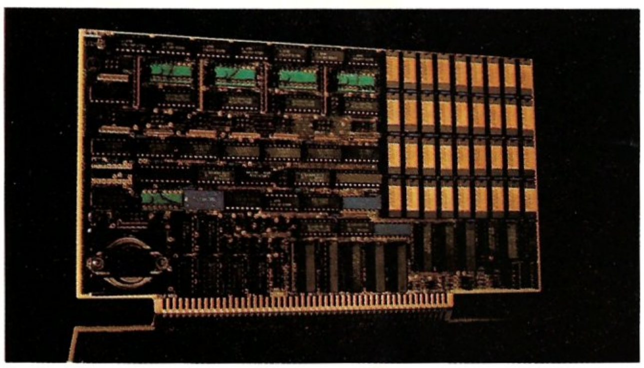
1 CPZ-48000 SINGLE BOARD COMPUTER.

□ IEEE 696.1/D2 S-100 compliance. □ Z80A,™ 4MHz Operation. □ Floppy disk controller (FDC). Single or double sided. Single or double density. 8" or 5¼". □ Two synchronous or asynchronous serial I/O channels (SIO). □ Two parallel I/O channels (PIO). □ Four channel DMA controller. □ 64K on board RAM. □ Memory management unit (MMU). Addresses up to 16 megabytes of system memory. □ Eight Vectored priority interrupts. □ Provisions for 2K or 4K onboard EPROM. □ Software selectable baud rates. □ IBM Bisync, HDLC, SDLC and other protocols. □ CP/M,™ MP/M,™ and TurboDOS™ operating systems available. □ Turbo-Disk® implementation included.

2 CPX-MX SLAVES.

□ IEEE 696.1/D2 S-100 compliance. □ Compatible with CPZ-48000 SBCP, any Z-80A based CPU with extended address capability or 16 bit based CPUs complying with IEEE 696.1/D2 bus specification. □ Z-80B™ 6MHz (CPS-6X) or Z80A 4MHz (CPS-4X) operation. □ Two synchronous (CPS-MS) or asynchronous (CPS-MA) serial I/O ports. □ TurboDOS™ & CP/NET™ compatible. □ Master confiscation of slave memory for diagnostic purposes. □ Two parallel I/O ports; eight data bits + 2 handshake lines per port. □ 64 Kbytes of onboard dynamic RAM. □ Master/slave memory-to-memory transfers under DMA control @ 571 Kbyte/sec transfer rate when used with CPZ-48000 SBCP. □ Software selectable baud rates. □ Usable as an intelligent I/O processor in single user system.



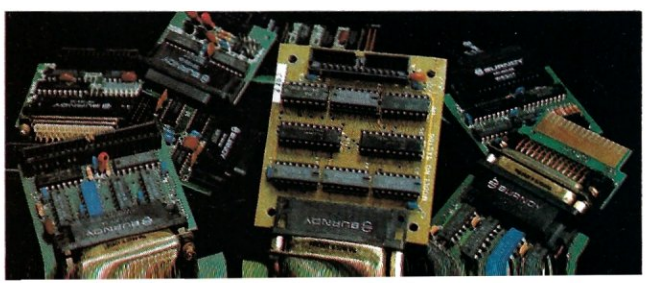


3 256KMB-100 256K MEMORY.

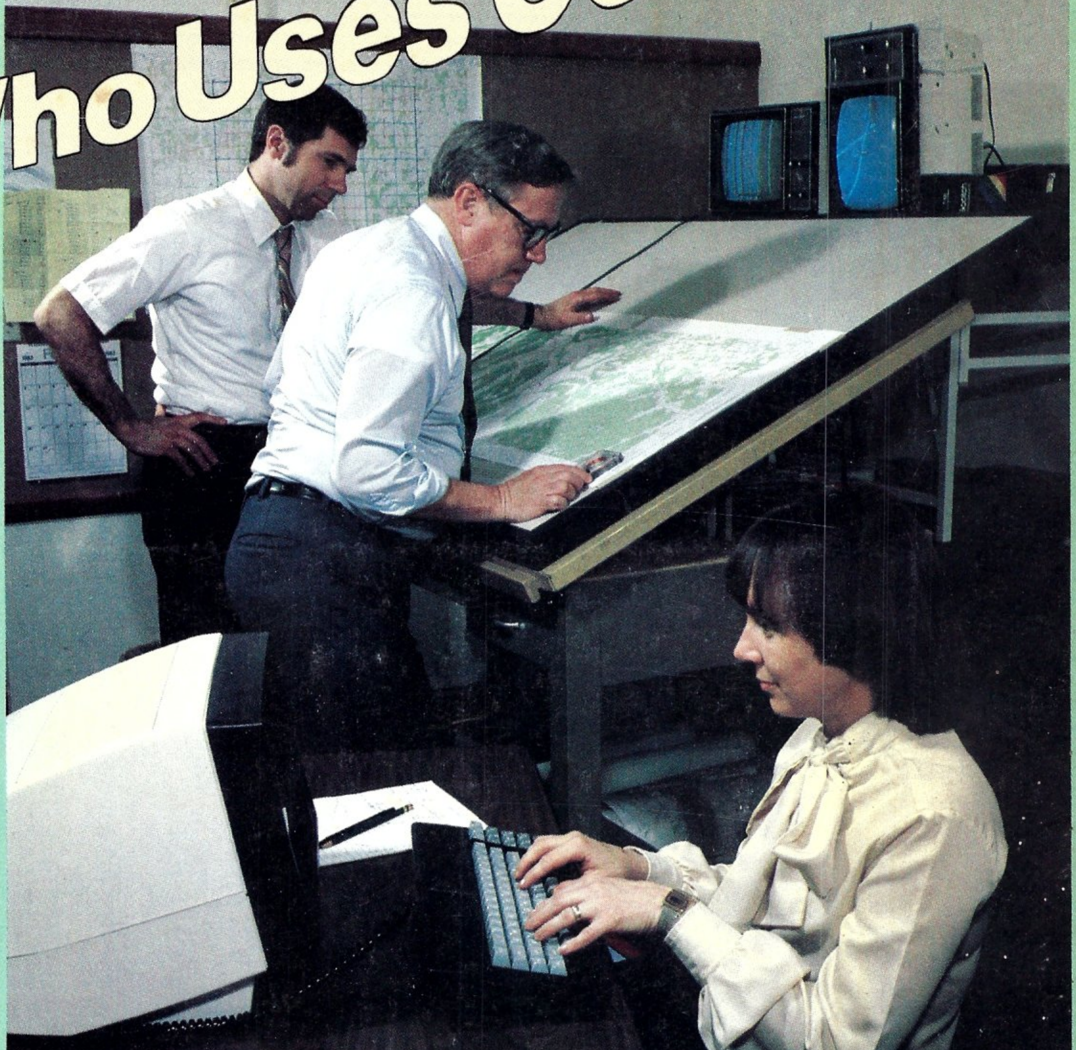
□ IEEE S-100 bus, spec 696.1/D2 compliance. The 256KMB-100 is compatible with most IEEE S-100 board products now on the market. □ Linear addressable to 2 megabytes. □ 225 nano-second access time, maximum, 160 nano-seconds, typical. □ 295 nano- second read-write time, minimum. □ Bank selectable 16K increments. □ I/O port address bank selection. □ Configures for phantom deselection. □ Parity error detection, visual and/or interrupts. □ Bank selection compatible with CROMIX,™ CP/M2.2,™ MP/M,™ Alpha Micro, and other major systems.

4 PERSONALITY BOARDS.

□ Centronics printer. □ 8 inch floppy disk. □ 5¼ inch floppy disk. □ RS232 serial communications. □ Synchronous/asynchronous modem. □ Priam smart/smart E hard disk. □ Long distance serial communication (2000 ft @ 9600 baud). □ Shugart Associates Systems Interface (SASI). □ Clock/calendar. □ Konan David,Jr.™ hard disk. □ Archive tape drive.



Who Uses CompuPro?



BOREXCO finds oil. It sounds simple but Borehole Exploration Corporation relies heavily on a sophisticated system of computer-generated, two- and three-dimensional magnetic models.

Another state-of-the-art system — from **CompuPro** — digests the megabytes of sensitive measurements that are recorded during BOREXCO's survey flights over hundreds of miles of terrain.

CompuPro's speed, reliability and cost-effectiveness won the respect of BOREXCO's engineering director, Dr. Brad Rehm. "I looked at others and I could not get this level of performance at several times the price," Dr. Rehm said.

Also, the **CompuPro** has trimmed BOREXCO's timesharing costs on a big minicomputer by as much as $7,000 a month. All of that without a single breakdown in two years.

In a complex world, **CompuPro** delivers performance, quality and reliability. For business, scientific and industrial computing solutions, contact a **Full Service CompuPro System Center**; call (415) 786-0909 for location.