For the Advanced Computer User

Micro/Systems Journal™

TurboCharge Your 8086/8088 Computer

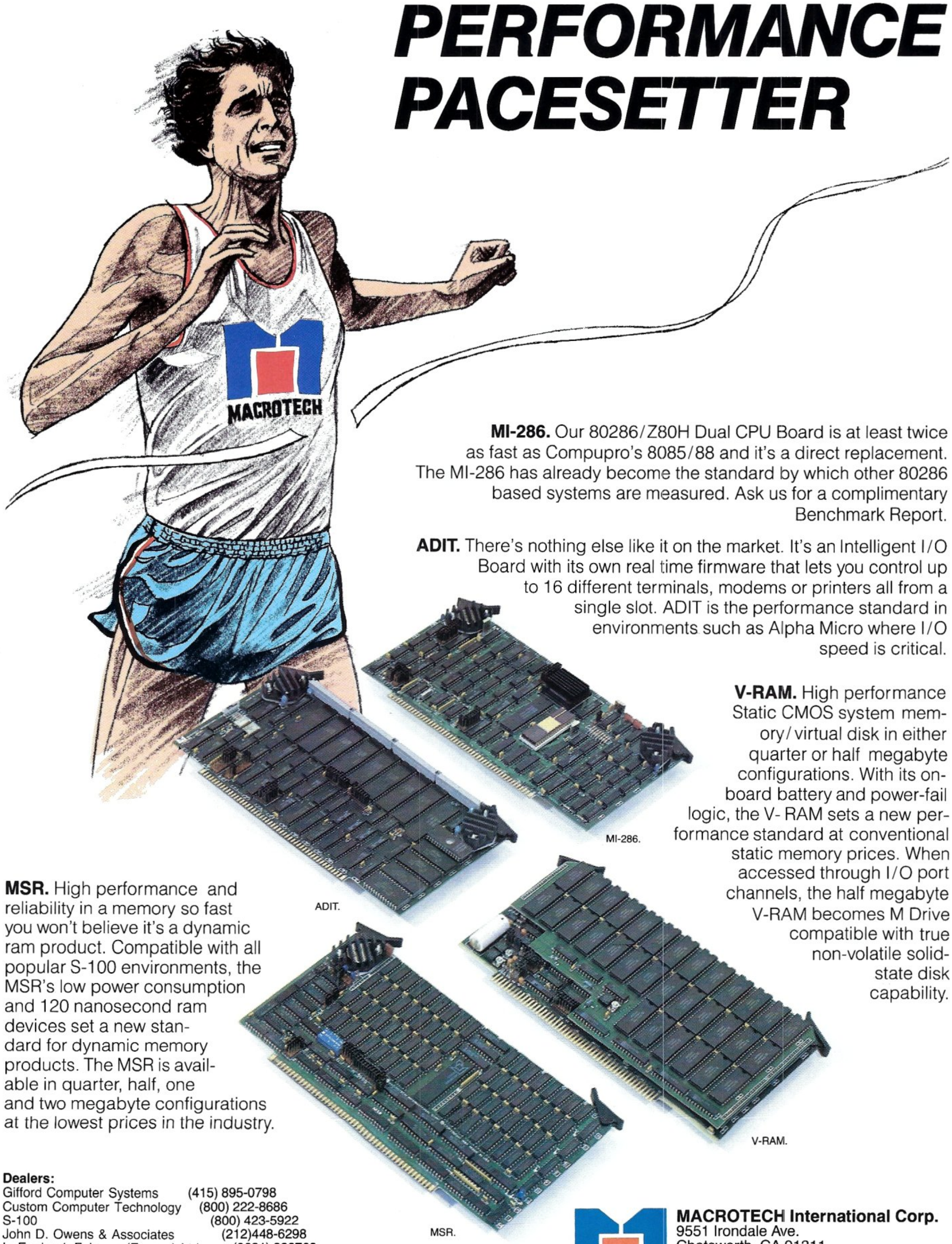
see pages 32-43

Also in this Issue

Complete Table of Contents on Page 3

November / December 1985 Vol. 1 / No. 5 $4.50 U.S.A.

MACROTECH—*STILL THE S-100 PERFORMANCE PACESETTER*



MI-286.

ADIT.

MSR.

V-RAM.

MI-286. Our 80286/Z80H Dual CPU Board is at least twice as fast as Compupro's 8085/88 and it's a direct replacement. The MI-286 has already become the standard by which other 80286 based systems are measured. Ask us for a complimentary Benchmark Report.

ADIT. There's nothing else like it on the market. It's an Intelligent I/O Board with its own real time firmware that lets you control up to 16 different terminals, modems or printers all from a single slot. ADIT is the performance standard in environments such as Alpha Micro where I/O speed is critical.

V-RAM. High performance Static CMOS system memory/virtual disk in either quarter or half megabyte configurations. With its onboard battery and power-fail logic, the V-RAM sets a new performance standard at conventional static memory prices. When accessed through I/O port channels, the half megabyte V-RAM becomes M Drive compatible with true non-volatile solid-state disk capability.

MSR. High performance and reliability in a memory so fast you won't believe it's a dynamic ram product. Compatible with all popular S-100 environments, the MSR's low power consumption and 120 nanosecond ram devices set a new standard for dynamic memory products. The MSR is available in quarter, half, one and two megabyte configurations at the lowest prices in the industry.

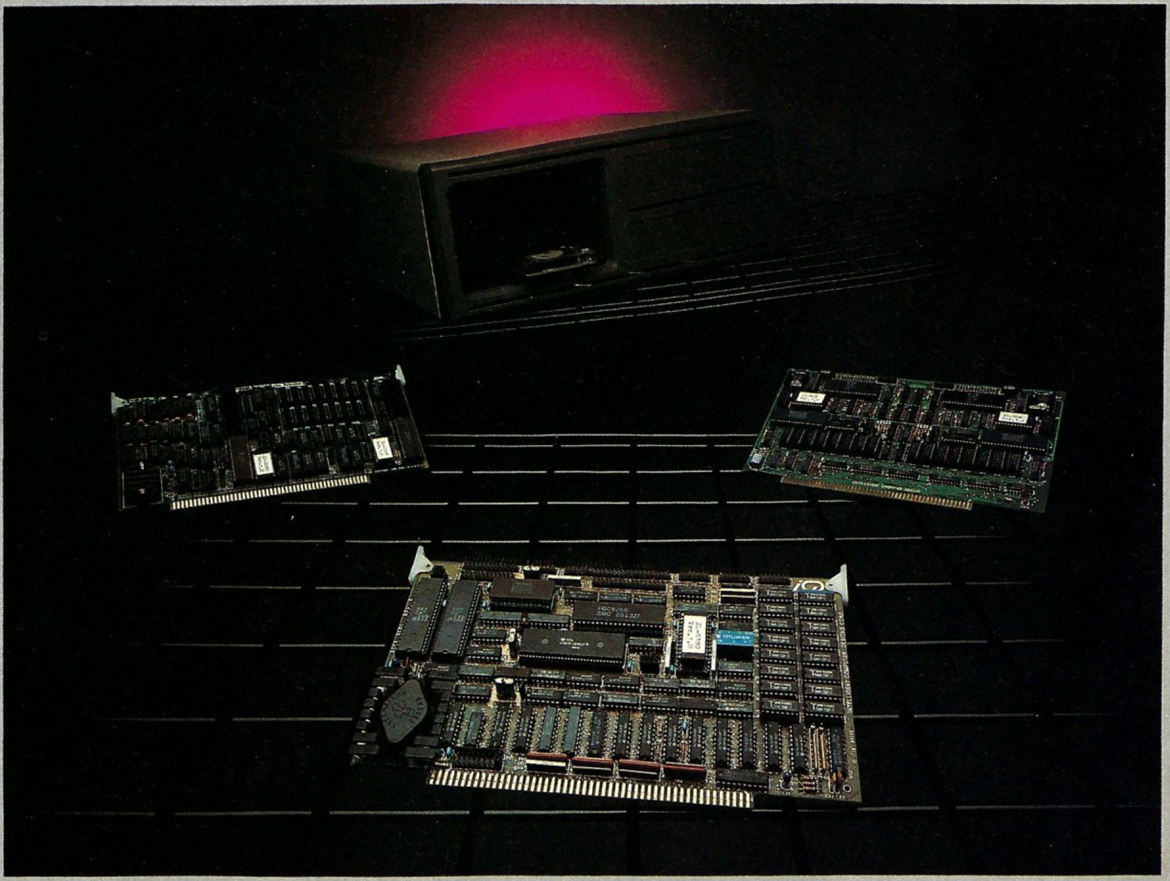
Dealers:

Gifford Computer Systems	(415) 895-0798
Custom Computer Technology	(800) 222-8686
S-100	(800) 423-5922
John D. Owens & Associates	(212)448-6298
In England; Fulcrum (Europe) Ltd.	(0621) 828763

Macrotech dealers also include most Compupro Systems Centers, Heathkit Electronic Centers and Alpha Micro Dealers.

MACROTECH International Corp.
9551 Irondale Ave.
Chatsworth, CA 91311
(800) 824-3181 • in Calif. (818) 700-1501
Telex: 9109970653

Intelligent Ideas Come From Intelligent Minds



Intelligent Computer Designs XL series S-100 products offer the latest in state-of-the-art designs.

Our slave products, the 8-bit XL-DLZ80 and the 16-bit XL-DL88, feature two slaves on one card, enabling system integrators double the number of users previously possible.

The XL-M180, a high-performance system master for single or multi-user applications, complete with hard disk, streaming tape and floppy capabilities introduces something new. An onboard Local Area Network with up to 5Mbps transfer speed. With it, interfacing other XL-M180's or even IBM-PC's is a snap!

If you think ICD has solutions for your systems' needs, write or give us a call.

XL-DL88

2 completely independent slaves
8mhz 70108 processor
128k to 512k memory expansion
4 serial communication channels
DMA transfers to/from system master
CP/M 86, MS-DOS, PC-DOS, TurboDOS compatible

XL-M180

High performance
HD64180 uPD
512k memory with burst refresh
5Mbps local area network
4 bi-directional parallel ports
2 serial ports
1 clocked serial port
Pipelined architecture
Z SYSTEMS O/S available

Hard disk controller/Streaming tape option
All digital floppy disk controller
Extended instruction set
Programmable instruction set
20 level interrupt structure
Over 80 I/O ports
Software programmable baud rates

XL-DLZ80

2 Z80H 8mhz processors
½ megabyte memory
2 190k local drives
4 serial ports
Draws only 700 millilamps
No wait states
CP/M, MP/M, TurboDOS compatible

All ICD products have a one-year part/labor warranty.

9252 Garden Grove Boulevard, Garden Grove, CA 92644 • 714 638-3622/Telex 31265

Teleteknology

One Success After Another.

Call For Reduced Pricing

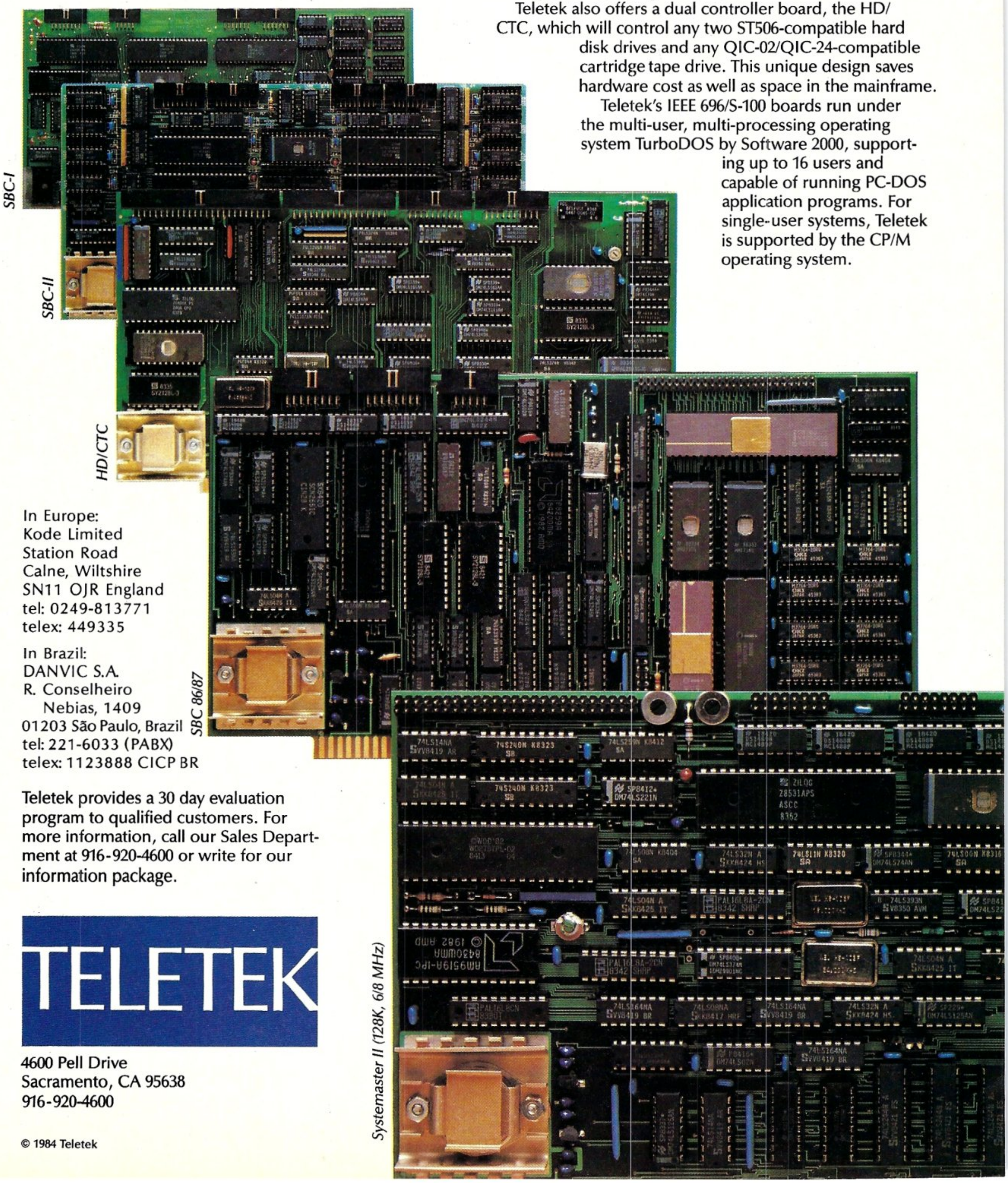
Since 1968 Teletek has been a leader in the design and manufacture of single board computers, controllers, memory boards and interface boards.

Teletek offers five distinct single board computers (SBCs), each with its own unique features, to meet the varied needs of the system integrator. Based on the 8086 16-bit and Z80 8-bit microprocessors, Teletek's SBCs will run at 4, 5, 6, or 8MHz and are available with up to 512K of onboard dynamic RAM. The SBC 86/87 also offers an optional 8087 math coprocessor for numeric intensive applications.

Teletek's Systemaster II provides two RS232C serial ports and two Centronics-compatible parallel ports or may be optionally configured to provide a SCSI interface or an IEEE-488 interface to support many laboratory testing and measuring instruments.

Teletek also offers a dual controller board, the HD/CTC, which will control any two ST506-compatible hard disk drives and any QIC-02/QIC-24-compatible cartridge tape drive. This unique design saves hardware cost as well as space in the mainframe.

Teletek's IEEE 696/S-100 boards run under the multi-user, multi-processing operating system TurboDOS by Software 2000, supporting up to 16 users and capable of running PC-DOS application programs. For single-user systems, Teletek is supported by the CP/M operating system.



In Europe:
Kode Limited
Station Road
Calne, Wiltshire
SN11 OJR England
tel: 0249-813771
telex: 449335

In Brazil:
DANVIC S.A.
R. Conselheiro
Nebias, 1409
01203 São Paulo, Brazil
tel: 221-6033 (PABX)
telex: 1123888 CICP BR

Teletek provides a 30 day evaluation program to qualified customers. For more information, call our Sales Department at 916-920-4600 or write for our information package.

TELETEK

4600 Pell Drive
Sacramento, CA 95638
916-920-4600

For the Advanced Computer User

November/December 1985
Vol. 1 No. 5

Micro/Systems Journal™

STAFF

Publishers
Sol & Lennie Libes

Editors
Sol & Don Libes

Associate Editor
Susan Libes

Contributing Editors
A.G.W. Cameron
David Carroll
Ian Darwin
Nelson T. Dinerstein
Dave Hardy
Ken Jackson
Henry Kee
Steve Leon
Bruce Ratoff
William Wong

Editorial Assistant
Lenore Melillo

Advertising
Lennie Libes

Circulation/Administration
Lennie Libes
Diane Overholser

SUBSCRIPTION RATES	**1 Yr.**	**2 Yrs.**
U.S.	$20	$35
Canada & Mexico	$28	$52
Europe & So. America	$35	$64
Asia, Africa, Australia & Near East	$39	$70

ADVERTISING RATES: Available on request. Call (201) 522-9347 or write to: M/SJ, Box 1192, Mountainside, NJ 07092.

CHANGE OF ADDRESS: Please send old label and new address to: Micro/Systems Journal, Box 1192, Mountainside, NJ 07092.

CORRESPONDENCE: Address all letters to : M/SJ, Box 1192, Mountainside, NJ 07092. The editor may also be contacted via MCI Mail (SLIBES).

Micro/Systems Journal (ISSN #8750-9482) is published bimonthly by Libes Inc., 995 Chimney Ridge, Springfield NJ 07081. Second Class postage is paid at Springfield, NJ and at additional offices. **POSTMASTER:** send address changes to MICRO/SYSTEMS JOURNAL, PO Box 1192, Mountainside NJ 07092.

IN THIS ISSUE

FEATURE ARTICLES

PRODUCT REVIEWS

DEPARTMENTS

AUTHORS: Micro/Systems Journal is always seeking good articles. Please write or call first to see if we are interested in the subject. Please do not send the article unless we ask for it.

If you are interested in reviewing hardware or software please write telling us your interests, your background and include a sample of your writings.

Send a stamped self-addressed business size envelope for a copy of our Author's Guide.

TRADEMARK ACKNOWLEDGEMENTS

Macintosh	Apple Computer Company
UNIX	AT&T
GEM, CP/M	Digital Research Inc.
Concurrent DOS	Digital Research Inc.
8088, 8086, 80286	Intel Corporation
IBM PC, PC/XT, AT	International Business Machines
MS-DOS, XENIX	Microsoft
GW Basic	Microsoft
68000, 68010,68020	Motorola

Editor's Page

by Sol Libes

FUTURE DIRECTIONS FOR MICROSOFT'S OPERATING SYSTEMS

Operating System (OS) software is the silent foundation of all computer systems. At its most elemental level, this software enables the loading and running of application programs as well as communications between the program, keyboard, display and peripheral devices, e.g. printers. Operating-system software designed for use with a disk-type, software-storage device is called a "Disk Operating System" or DOS. DOS software usually has additional features, such as the ability to show the user what is available on the disk.

Microsoft is, without a doubt, the leading supplier of microcomputer OS and DOS software, having five OS and DOS packages currently in use by microcomputer-system manufacturers. They are called packages because in addition to the OS or DOS program they also contain utility programs which the OS or DOS can use to add functions to the OS or DOS (e.g. format a disk).

Microsoft's five OS and DOS packages include, of course, PC-DOS and MS-DOS for use on the IBM-PC and IBM-PC compatibles. Microsoft has also created XENIX (a Unix, multi-user, multi-processing DOS), MSX (for low-cost, Z80, cassette-based, home-computer systems), and MSX-DOS (for low-cost, Z80, disk-based, home-computer systems). Microsoft has also created two MS-DOS extensions: MS-Windows (an MS-DOS environment for applications' software which uses bit-mapped graphics and a mouse) and MS-Net (a networking extension for MS-DOS). Microsoft thus has the broadest line of microcomputer, operating-systems' software currently available.

At a recent National Computer Conference, Steve Ballmer, Vice President of Marketing at Microsoft, gave some insight into Microsoft's system-software strategies. The following discussion is based, in large measure, on his talk.

SOFTWARE & SYSTEM COMPATIBILITY

Microsoft is attempting to provide upward compatibility among all its operating systems, something no other system software producer has ever really attempted. Microsoft has adopted a philosophy that all new versions of operating systems must be able to run software written for earlier versions....this is called "upward" compatibility.

Microsoft is also attempting to provide compatibility between operating systems. Their greatest concern appears to be compatibility between MS-DOS, PC-DOS, and XENIX. Thus, they have attempted, wherever possible, to provide the same functionality, hierarchical file system, and the same memory management scheme in each DOS.

Microsoft claims they already have source-code compatibility between systems. Thus applications' software written in a Microsoft, high-level language can be compiled for either PC-DOS, MS-DOS, or XENIX. They are also working on binary-level compatibility so that applications' software can run under either operating system without compilation or modification.

Microsoft is also promising networking compatibility between users of different DOS kernels to guarantee software reliability. This means that what one user does on his system does not affect data on another system in the network.

DATA RELIABILITY

As systems become more complex with multi-tasking and networking capabilities data, reliability becomes increasingly more difficult. Thus a protection system is needed to insure that data is not altered inadvertently.

For example, current 8088- or 8086-based systems do not provide any hardware protection, allowing any program to take command of the system at any time and do anything that it wants. Thus, in a networking system, there is no way of certifying that data at any one workstation is reliable.

The Intel, 80286 microprocessor includes on-board protection. Thus, Microsoft will, in all likelihood, introduce multi-tasking DOS exclusively for 80286-based systems with data reliability features.

OPERATING SYSTEM PHILOSOPHIES

MSX and MSX-DOS are Microsoft's home-computer versions of OS and DOS, respectively. They are very small, single-user, single-tasking systems with very friendly user interfaces. They are currently being used by several Japanese, personal-computer manufacturers in their consumer-oriented systems. These systems are being exported to countries other than the U.S. The price wars that have occurred in the U.S.-consumer, computer marketplace have made them hesitate.

PC and MS-DOS are Microsoft's office-worker versions of DOS. They are also single-user systems. Although the current versions of MS-DOS and PC-DOS are single-tasking systems, the next versions are expected to be capable of multi-tasking. Thus, a user will be able to handle communications, electronic mail, or networking tasks in the background. These new versions are also expected to have an advanced easy-to-use interface.

XENIX has been designed by Microsoft to serve as a multi-user operating system for transaction-processing in a small business environment (data-base access). It provides a customizable user shell so that software developers can create turnkey systems to hide the OS from the user altogether.

LOOKING TO THE FUTURE

Microsoft expects the Intel 8086, 80286, and 80386 (32-bits) to be the dominant microprocessors. As discussed earlier, the 286 includes hardware, memory-protection logic to insure data integrity in multi-processing systems. There is no doubt that the 386 will include even more powerful memory-management features.

The likelihood is that the new versions of PC-DOS and MS-DOS will require more memory. Thus, we can expect that the typical memory size on the new 286-based systems will be in the megabyte range. Considering that we will have in memory the DOS kernel, the networking support, a window manager, and applications such as mail programs, integrated packages (Symphony, etc.), talking about kilobyte sized memory will become a thing of the past.

The current IBM-PC with its slow, 8088-based processor, 640K sized memory, and primitive, memory-management system, will be inadequate for the next generation of desk-top office workstations.

Microsoft is also expected to introduce a new version of PC/MS-DOS for the 8088 and 8086 systems currently in use. Expect this new version to include a more robust, memory-management system that provides for relocation of code, swapping of code, and virtual memory.

Microsoft is also expected to make the new versions of MS-PC/MS-DOS more modular. The new versions of the operating systems are expected to be dynamic. Modules will be moved in and out of memory as they are needed. As a form of dynamic linking, sometimes called "late binding", essentially any piece of the operating system will be available at run time. This is expected to allow binary compatibility between PC/MS-DOS and XENIX by linking in the appropriate OS, service library at run-time.

Microsoft has indicated it has already begun initial development of versions of MS-DOS and XENIX for Intel, 80386-based systems. Intel has already furnished preliminary data on the device to key users and is promising samples shortly with limited production to begin in early '86. Considering that Microsoft is already at work on operating systems for the device, it appears that the 386 versions of PC/MS-DOS and XENIX will be released in early 1986.

News, Views & Gossip

by Sol Libes

GOSSIP

IBM is expected to change the motherboard in the AT soon. Look for the new board to contain 640K of RAM.....gone will be piggy-backed chips. Also expect some new VLSI chips to shrink the chip count. And a System V version of Xenix for the AT is due very early next year.....Big Blue is finally expected to introduce its laptop machine in the first half of '86. Expect it to use a CMOS 80286, 3.5'' drives, and back-lit screen......IBM is reportedly asking floppy disk drive vendors to quote on 3.5'', double-sided drives storing 1.6Mbytes unformatted and 1.2Mbytes formatted.

Digital Research has reportedly sold a significant minority interest (estimated at about 12%) to Northern Telecom Inc., a Nashville Tennessee telecommunications company for about $6 million. It is also rumored that Motorola will take another ''minority'' interest in DRI to help it through its hard times. Olivetti is already an investor in DRI. NT is using GEM in their 68000-based Meridian computers and Motorola sells DRI's languages for the 68000. DRI reported approximately $40 million in gross revenue for its last fiscal year, down from the previous year's $55 million.

Borland International, of Turbo-Pascal fame, seeking to enlarge its business, is expected to shortly introduce Basic, Assembler, C, Modula-2, and Pascal for the Commodore Amiga. SideKick-Plus and a word-processing package are expected early next year. Borland, which started strictly as a mail-order vendor, now gets over 70% of its revenue from dealers.

Intel has disclosed that it is hard at work developing the 80486 microprocessor which will be upward compatible with the 80386, 80286, and 8086 microprocessors.

Texas Instruments demoed the prototype of a new graphics IC at a recent conference. Called the TMS34070, it integrates 3 DACs, a video processor, ROM, and RAM. A color look-up table allows the displaying of 16 colors from a 4,096 pallette on each line that can be changed line by line for an 800 by 600 pixel display. A similar chip is expected from INMOS.

Early next year **Kodak** is expected to begin sampling a 9-Gbyte, table-top size, optical-disk system. Although the systems from Sony, Hitachi, and NEC have higher capacities, they are much larger in size.

Sony is reported to be developing 1.6 and 2Mbyte versions of their popular 3.5'' floppy disks. The units, which feature 135 tracks/inch and 17.4Kbits/inch, are expected to be in production some time next year.

AT NOT DOING WELL

Sales of the IBM AT are reportedly not doing well and many dealers are already offering significant discounts and special deals on them, only a little over a year since its introduction.

It appears to be a repeat of the PC/XT story. IBM did not anticipate the great demand for the unit. Hence, there was a severe shortage of units when it was introduced. IBM upped production substantially at the same time that the clone makers started shipping. The result.....a glut of ATs and AT-clones.

A further problem for IBM has been reports of reliability problems with the machine, the high price of the system, and compatibility problems with the PC/XT. This has resulted in a ''window of opportunity'' for the AT- and XT-clone makers.

In the meantime, IBM dealers are crying the blues, claiming that IBM is flooding an already saturated market with PC's and XT's. They claim that large-volume dealers, who are forced to take at least 400 systems/month in order to maintain an additional 8% discount, are discounting (typically 28-30%) or dumping on the grey market (which despite IBM's attempts to stamp it out is still flourishing). IBM is also appointing more large-volume dealers and forcing dealers to take a product mix that includes portables and PCjrs which have accumulated in IBM's warehouses. These IBM tactics are squeezing many of the smaller independent dealers out of business.

APPLE NEWS

There are rumors that Apple is hard at work on two new products - a portable ''Flat-Mac'' and a ''Color-Mac'' using the 68010 processor. There are also rumors that an ''open-Mac'' with slots for plug-in cards and detailed technical info will be introduced shortly.

In the meantime, sales of the Macintosh are lagging despite rebates ($150 and $250), customer sweepstake contests (grand prize $100,000), promotional giveaways (free bicycle with a specific system configuration), additional bulk discounts, and salesmen incentives ($50 per system sold). And, Lotus' Jazz, an integrated software package, has not boosted sales as expected. The package has proved difficult to use on a machine designed to be easy to use.

Apple has still not penetrated the corporate market where IBM now reigns supreme. Here the largest applications are word processing, spreadsheet use, and data-base management. Thus, the lack of an internal, hard-disk system, slow speed of operation, inability to upgrade the system internally, as well as network with IBM systems has proved to be a handicap.

Although the Mac has achieved some success in the home market, it now has competition from Atari and Commodore who are now offering Mac features at Apple-II prices.

ZCPR NEWS

The ZCPR3 printed manual is finally available. All users of ZCPR and those interested in running it on their Z80-based systems should get a copy. It's $19.95+$4 S&H and is available from: Echelon Inc., 101 First St, Los Altos CA 94022, tel:(415)948-3820.

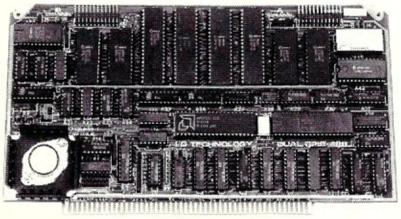
I think that Echelon is doing a terrific job with ZCPR3. All those users of CP/M who have been deserted by Digital Research's failure to upgrade CP/M-80, now have Echelon providing a host of upgrades, advanced software-development tools and advanced utilities. Echelon has also introduced an upgraded version of CP/M for systems using the new Z80-compatible, Hitachi-HD64180 microprocessor (see last issue of MS/J; we expect to review this operating system in an upcoming issue).

Another thing that impresses me about Echelon is their user support. They publish a ''Z-News'' newsletter twice a month providing a great deal of information to ZCPR users as well as support a number of RBBS (Remote Bulletin Board Systems) around the country. These RBBS's are referred to

INPUT/OUTPUT TECHNOLOGY, INC.

25327 Avenue Stanford, Unit 113, Valencia, CA 91355 • (805) 257-1000

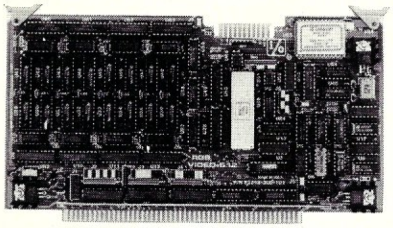
Uncompromising Additions to your S-100/IEEE-696 BUS



DUAL GPIB-488 INTERFACE BOARD

A Stand-Alone, Independently Controlled Dual Channel IEEE-488 I/O Processor. Interface Activity Modes for Controller-in-Charge, Controller Assigned or Terminal Bus Slave, and all Interface Functions are handled transparent to Host System CPU through an on-board CPU and DMA controller. User Friendly operation.

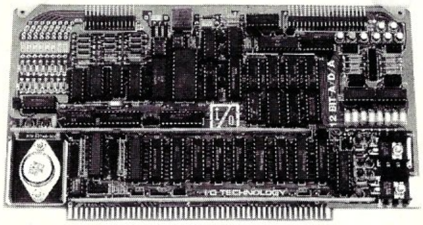
A&T, P/N 52748-800-102



RGB COLOR GRAPHICS BOARD

Programmable resolution up to 512 x 512 pixels with 4 local video planes and on-board graphics processor. Color mapper allows 16 colors from a palette of 4096. Light pen input. Plus more ...

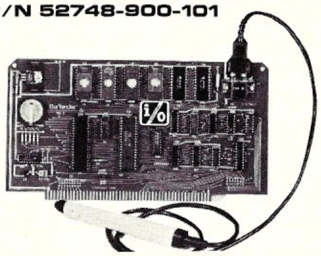
A&T, P/N 52748-300-101



12-BIT A-D-A CONVERTER BOARD

8 Channel A-D: 12 microsec. Conversion, 50KHz Sample Rate, Programmable Gains, Offset and Diff./Single Modes. **8 Channel D-A:** 2 microsec. Settling, Bipolar V or Unipolar I Output. Programmable Reference levels, Dual-Ported Channel Refresh RAM. **16/8-Bit Data Transfers** via I/O or Memory Mapped

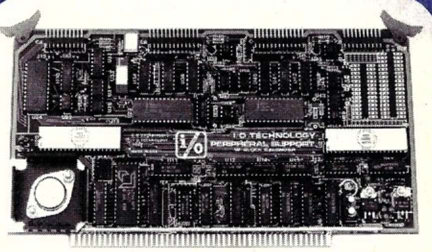
A&T, P/N 52748-900-101



BAR CODE PROCESSOR BOARD

The BarTender is a stand-alone I/O Processor that reads and prints most common Bar Codes. Includes bi-directional reading, wand interface, clock/calendar with battery. Extensive documentation and software.

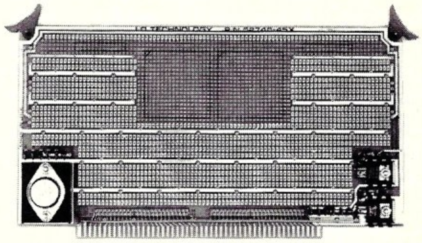
A&T,52748-500-101 Without Wand
A&T,52748-500-201 With Wand



PERIPHERAL SUPPORT BOARD

Two Serial SYNC/ASYNC Ports with RS-232, TTL or Current Loop Outputs, three 8-Bit Parallel Ports, three Timers, Real Time Clock/Calendar and Response Programmable Interrupt Controller. Small Proto Area with +5 and ±12v.

A&T, P/N 52748-150-101



MULTI-PURPOSE PROTOTYPING KIT

Industrial Quality with Plated-Thru holes for Wire-Wrap or Solder projects. Complete with +5, ±12v Regulators, Bus Bar, Filter Capacitors, and Manual.

P/N 52748-450

ALSO AVAILABLE: MULTI-FUNCTION I/O BOARD, SMART PROTOTYPING KIT, 128Kx8/64Kx16 STATIC RAM MODULE

CIRCLE 10 ON READER SERVICE CARD

SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE.

as "Z-Nodes". Two of the Z-Node operators are developing a full data-link, satellite-relay function to network Z-Nodes. For more information on Z-Nodes, contact Echelon or log into one of the Z-Nodes. Here are some of the Z-Node phone numbers:

(415)489-9005 CA (Z-Node Central)

(915)755-3342 TX

SERVICING AN IBM-PC

Users who service their own PC or clone might be interested in obtaining a copy of the Howard W. Sams "Computerfact" package ($39.95 at most electronic-parts suppliers). It includes complete schematics, circuit-board photos with all components identified, and troubleshooting instructions. It covers the PC motherboard, power supply, keyboard, display, floppy-disk controllers, and several other optional cards.

Here is another service hint. The PC's power supply is enclosed in its own case. If your power supply fails, try opening the case (with power plug unplugged from the wall) and check the slow-blow fuse inside the case. It is mounted in a fuse holder. Incidentally, this fuse is not an IBM-replaceable item. If you bring your PC/XT/AT into an IBM service center for repair and they find the power supply is defective they replace it without checking the internal fuse (a replacement power supply is $290!).

TRON IS COMING

TRON (The Real-time Operating Nucleus) is an operating system being developed by a group, headed by Professor Ken Sakamura, at the University of Tokyo (considered the MIT of Japan). TRON has already been licensed by NEC, Mitsubishi, Matsushita, Xerox Japan, and Canon.

TRON is expected to be a full-blown, enhanced operating system specifically designed to take advantage of 32-bit microprocessor chips such as the Motorola 68020. TRON is planned as an operating-system nucleus contained in a single ROM with other ROMs containing different system and application interfaces. TRON will feature fast real-time response and task-switching. NEC is expected to be the first to introduce a TRON system, sometime next year.

NOTEWORTHY BULLETIN BOARDS

Dan Wise's "Cleveland County BBS" (Shelby NC) has 45 different sections including S-100, CP/M, and UNIX/XENIX and callers can chat with "Willard" (artifical intelligence). The BBS runs 7 days 24 hrs, and there is no time limit. Call (704)482-8012.

Mike Connick (Reston VA) runs a BBS devoted to C at (703)476-9459.

RANDOM BITS

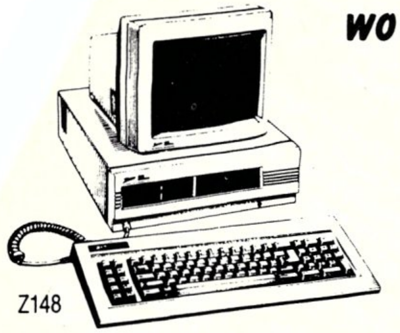
The **Forth Interest Group** (FIG) will hold the "Forth Modification Laboratory" conference at the Asilomar Conference Grounds, Pacific Grove CA, Nov 29-Dec 1, to discuss and share new proposals to enhance the Forth Language. For more info, call (408)277-0668 or write FIG, Box 8231, San Jose CA 95155.

Teletek (4600 Pell Dr., Sacramento CA 95838; (916)920-4600) is now offering an MS-DOS emulator (non-hardware specific) which runs under TurboDOS. Graphics capability is being explored.

Intel reports that it has begun shipping versions of the 80286 which run at clock speeds of 10-12.5Mhz.

Intercontinental Micro Systems has acquired the manufacturing rights to **Musys'** S-100 product line.

STARTER SYSTEM SPECIALS



WOW! Z-148-21 Only **$1195.00**
one drive, 128K

FCC-SYS148-02 only **$1595.00**
148, 2 drives
640K RAM
1220 or 1230 Monitor

FCC-SYS158-02 only **$2099.00**
158, 2 drives
640K RAM
1220 or 1230 monitor

WOW! POWERFUL WINCHESTER SYSTEMS

FCC-SYS158-11 only **$2475.00**
158, one floppy
11 meg half high winchester
640K RAM
1220 or 1230 monitor

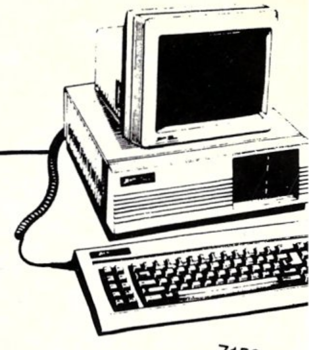
With ZVM-135 color monitor only **$2899.00**

FCC-SYS158-25 only **$2849.00**
158, one floppy
25 meg half high winchester
640K RAM
1220 or 1230 monitor

With ZVM-135 color monitor only **$3299.00**

ASHTON·TATE™

Framework. For Thinkers.™



WOW! FREE FRAMEWORK!

with purchase of 36 meg SuperSystem!
Z158, one floppy
36 Meg winchester, only **$3875.00**
full height, 40ms access time
640K RAM
ZVM-135 color monitor

UTOCAD™

a full-line AutoCAD dealer
custom quotes

AUTO CAD SYSTEM

Complete workstation with plotter and digitizer

Z-158 8 Mhz
one floppy, 25 meg H/H internal
8087-2 8 Mhz
ZVM-135 monitor *only* **$9599.00**
AutoCAD w/ADE-1, 2, & 3
GTCO 12 x 12 4 button digitizer
DMP-41 or 42 plotter w/cable

ZVM-136 w/ Techmar Graphics Master
640 x 400 Color, add $ 785.00

WOW! Printers with above systems:

Epson LX-90,	*only $259.00*
Epson FX-85,	*only $399.00*
Toshiba P-1340,	*only $596.00*
Epson FX-185,	*only $549.00*
Toshiba P-351	*only $1225.00*

We also have the two hot new machines from Zenith, the 171 and the 200! Call for our latest price and specials, as well custom configurations on the 200, and the new 158.

Call for pricing on tape backup systems and cartridge winchester

ST COMPLETE
IPATIBLE LINE

TH | data systems

ED SALES AND SERVICE

First Capitol Computer

1-800-TO-BUY-IT (800-862-8948)
Orders and quotes
1-314-946-1968
Technical support and order status

FIRST CAPITOL — THE ZENITH EXPERTS!
Order direct from First Capitol Computer.
All prices pre-paid (cash, check, Mastercard or Visa).
Add 2% Shipping charge.
Missouri residents add 6.225% sales tax.

1106 First Capitol Dr.
St. Charles, MO 63301
(314) 946-1968

Call for purchase order terms, government, and quantity pricing.

We will meet or beat any verifiable advertised price on stock Zenith Data Systems CPU's, from any legitimate dealer.

there is mail......

Letters to M/SJ

We welcome your letters with comments, compliments, criticism and suggestions. We read them all and publish the most noteworthy, even if they are critical of us. We do not have the staff to answer all letters personally. And all letters become the property of M/SJ and may be subject to editing. Further, we do not print letters that do not include a name and address.

Please send your letters to: Micro/Systems Journal, Box 1192, Mountainside NJ 07092.

Dear Mr. Libes,

''Structured Programming with M80'' (July/August '85) has some interesting constructs. However, preferring BASIC to PASCAL as a structured language, I have enclosed an INSTR and ON n GOSUB ¦ GOTO, which could probably be modified to IN [list] and sort of a version of CASE. Oh yes, in the REPEAT code on page 32, it certainly will; BC is the counter, but HL is being checked for zero. LDIR is a repeat and even CPI can be used if followed by JP PE for a test loop or zeroing a block. Both take up quite a bit less space than the code presented.

The previous issue had an interesting article on Loadable BIOS drivers for CP/M. Although in SLCLR, on page 70, ANA A ! JP SL50 should have probably been a JZ SL50. In debugging CCP replacements a number of techniques were developed to allow the code to be dropped into RAM without altering the disk image. Since the loader takes less than one record it was easier to include it in the code rather than as a separate program. As a small example, A Z80 register dump is enclosed. It contains a few useful routines including the HEX routine, macros to do page relocatable code, a Z80 cpu test, and an exit for those who try to type a .COM file. The code was assembled using a conditional restart as the trap (RST 38h). If this is to be used with a vanilla version of DDT or ZSID it is recommended that the restart number be changed. A 8088 version of INSTR is also included.

L. Barker

Regretfully we did not have the space available to publish the code enclosed with this letter. We suggest readers who wish copies contact Mr. Barker at Box 135, Chicago IL 60690.

Dear Mr. Libes:

Shame on you for printing Charles Prohaska's *patch* in the July/August issue! The binary-to-ASCII-hex conversion that he *fixed* dates back (at least) to '77 or '78, when an 8080 version appeared as a ''byte-saving programming trick'' in Dr. Dobbs Journal. It works perfectly as originally written.

Either Mr. Prohaska has a very sick processor in his system, or he never actually ran his code, but condemned it based on his (mis)understanding of the written source. The relevant instructions are:

```
ADD A,90H
DAA
ADC A,40H
DAA
```

Contrary to his assertion, binary 1010 (for example) is correctly converted to ASCII ''A''. This first addition gives hex 9A (decimal 100). The decimal-adjust leaves 0 in the accumulator and *sets the carry flag*, as is necessary for multi-byte BCD additions. The following add-with-carry therefore gives not 40, as he claims, but 41.

Bill Snyder, Richardson TX

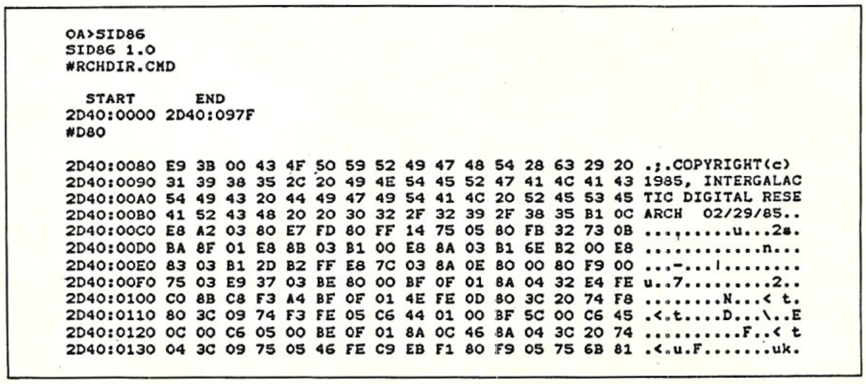
We would also like to thank Patrick O'Leary, County Kildare Ireland and L. Barker, Chicago IL for also picking up this error and pointing it out to us. -

Dear Sol:

Has DRI gone to outer space?

Perhaps some readers might enjoy the dump (shown in the box below) of the Digital Research Concurrent PCDOS 4.1 embedded copyright message.

Alex K. H. Soya, Melbourne Beach FL



```
OA>SID86
SID86 1.0
#RCHDIR.CMD

  START      END
2D40:0000 2D40:097F
#D80

2D40:0080 E9 3B 00 43 4F 50 59 52 49 47 48 54 28 63 29 20 .;.COPYRIGHT(c)
2D40:0090 31 39 38 35 2C 20 49 4E 54 45 52 47 41 4C 41 43 1985, INTERGALAC
2D40:00A0 54 49 43 20 44 49 47 49 54 41 4C 20 52 45 53 45 TIC DIGITAL RESE
2D40:00B0 41 52 43 48 20 20 30 32 2F 32 39 2F 38 35 B1 0C ARCH  02/29/85..
2D40:00C0 E8 A2 03 80 E7 FD 80 FF 14 75 05 80 FB 32 73 0B ...,.....u...2s.
2D40:00D0 BA 8F 01 E8 8B 03 B1 00 E8 8A 03 B1 6E B2 00 E8 ............n...
2D40:00E0 83 03 B1 2D B2 FF E8 7C 03 8A 0E 80 00 80 F9 00 ...-...|........
2D40:00F0 75 03 E9 37 03 BE 80 00 BF 0F 01 8A 04 32 E4 FE u..7.........2..
2D40:0100 C0 8B C8 F3 A4 BF 0F 01 4E FE 0D 80 3C 20 74 F8 ........N...< t.
2D40:0110 80 3C 09 74 F3 FE 05 C6 44 01 00 BF 5C 00 C6 45 .<.t....D...\..E
2D40:0120 0C 00 C6 05 00 BE 0F 01 8A 0C 46 8A 04 3C 20 74 ..........F..< t
2D40:0130 04 3C 09 75 05 46 FE C9 EB F1 80 F9 05 75 6B 81 .<.u.F.......uk.
```

Sol:

When a magazine really works, it speaks so personally and appropriately to its readers that they make it their own, taking personal offense at the less-effective articles and taking pride in the good ones. Such magazines don't belong to their publishers or editors but to their devoted readers. The staff are the caretakers.

So the gut reaction of us readers when Ziff-Davis cancelled Computers and Electronics and Microsystems was that they had no right. In a legal sense they certainly had every right, but in a more important sense I think the gut reaction is the correct one. They had no right.

Thanks for keeping our Micro[/]Systems [Journal] alive.

Mike Swaine
Editor Dr. Dobb's Journal

P.S.: Yes, I think you're crazy.

Dear Sol:

I enjoyed ''The CP/M Bus'' in the May/June issue. Reading Bruce Ratoff's choices of favorite public domain programs started me thinking what were my most useful CP/M programs. I came up with at least a half a dozen in a few minutes.

My favorite is a directory program that has been named ''D''. It produces an alphabetical listing in a vertical format with file size and disk space included. The horizontal version, SDIR, is much harder to read and I do not use it. Next on the list is my file manipulation program DISK 7.6, that someone named ''T''. I really like the CRC verification feature on the copy command.

For handling library files I have gotten used to NULU10 and will get NULU11 off the board shortly. I figure this is really a matter of which library utility program you use. For handling the squeezed files off of bulletin boards, I find that NSWP207 is great. The view function for looking at squeezed files is very useful and eliminates the need to unsqueeze before you can read the file. It also has unsqueeze and the usual file manipulation features.

For viewing ASCII files, the similar programs BISHOW-2 and BISHOW31 replace the CP/M TYPE command. BISHOW31 will even unsqueeze squeezed files. Theyboth allow the forward and backward viewing of files. A gem with limited application but very useful is SORTDIR3 for alphabetizing the disk directory. It is useful for programs like BASIC where the FILES command does not sort the file names but rather just displays them like the CP/M DIR command.

There is one utility I have not been able to locate, a screen dump utility. I know there is one specifically for the KayPro, DUMP24.

Bob Phelps, Westerville OH

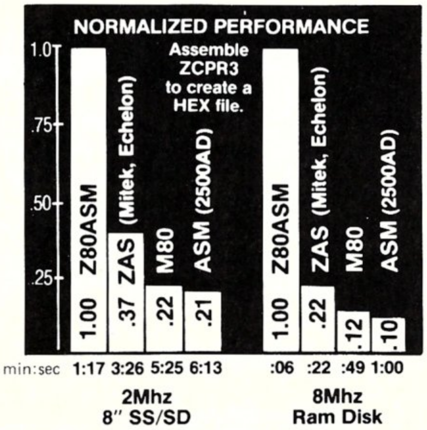
Dear Sol:

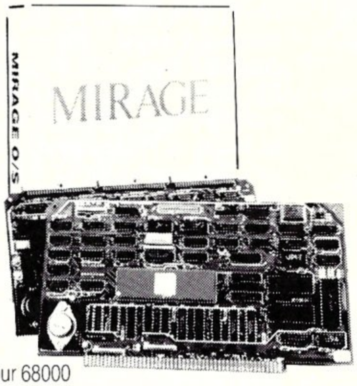
I enjoyed reading Steven Bosak's "Scientific & Technical Word Processors" (Part 1) so much that I took the liberty of calling him about PS, my *secret WP program*. The **PS Technical Word Processor** was written by Murray Sargent, author of *IBM PC From the Inside Out* (Addison-Wesley). Murray is a Professor of physics at the University of Arizona. He developed the PS for his papers but the project (like most computer projects) expanded over time.

PS is based on Phoenix's PMATE programmer's editor, but Murray has turned it into a powerful, menu-driven, proportionately-spaced, what-you-see-is-what-you-get word processor complete with mail-merge. Unlike most other technical editors, PS is most at home on Diablo-compatible daisywheel printers from the 1620 on up - and the proportionately-spaced text appears on the screen as well! Although I don't do scientific WP, I use PS for everything from general correspondence to our 500-page RUN/C manual. I don't think there is a more powerful, more flexible, faster word processing package available on any system.

Mr. Bosak agreed to take a look at PS and he should have it by now. I hope some day it will be possible to let your readers know about PS so they can compare it themselves to the other scientific packages on the market.

Todd Katz, Age of Reason Co., NY NY





Turbo Pascal Corner

by David W. Carroll

This column features tips and techniques for using Turbo Pascal productively on MS/PC-DOS and CP/M microcomputer systems. It discusses typical problems and their solutions. Reader suggestions, comments, and questions are encouraged. Address them to: Turbo Pascal Corner, Box 699, Pine Grove CA 95665.

In this issue we will discuss some advanced machine level interface techniques with Turbo Pascal, primarily DOS calls, command line parameters, and wild card file names. This article will address MSDOS operations and the next issue will address the same processes in CP/M Turbo Pascal systems.

DOS CALLS

Calls to MS‘PC-DOS, in Turbo Pascal, are performed using either the INTR procedure or the MSDOS procedure. Both provide a software interrupt and pass a register record variable to the system, presetting the CPU registers and returning any parameters upon termination of the interrupt. MSDOS(regs) is actually a *shorthand* notation for the INTR($21,regs) procedure, and the two can be used interchangably. The MSDOS call can only be used to access DOS functions, while the INTR call can access any software interrupt, including the BIOS routines.

The register record variable is set up like this:

```
type
 regtype = record
   (ax,bx,cx,dx,bp,si,di,ds,es,flags:
       integer);
  end;
var
 regs : regtype;
```

or a free union variant record can be used to allow accessing and setting the registers either as word or byte values. The variant part of the record is automatically selected by the field identifier used.

```
type
 regpack = record
   case integer of
     1: (ax,bx,cx,dx,bp,si,di,ds,es,
         flags: integer);
     2: (al,ah,bl,bh,cl,ch,dl,dh : byte)
   end;
var
 regs : regpack;
```

Advanced Machine Level Interface Techniques

COMMAND LINE

Up to 127 bytes of command line parameters, if any, are located at ABSOLUTE CSEG:$80 when an MS-DOS Turbo Pascal program begins. They must be copied to a program variable as the first action of the program or all but 32 bytes will be lost.

```
type
 comstr = string[127];

var
 buffer  : comstr;
 comline : comstr absolute cseg:$80;

begin
 buffer := comline;{This must be first}
       {statement in main program block.}
```

As with an user entered data, the command line parameters should be parsed and validated in your program.

WILD CARDS

The use of wild cards for file names in MS-DOS Turbo Pascal programs is not discussed in the *Turbo Reference Manual* or in the *Turbo Tutor* tutorial from Borland, so it is an often requested topic. There are two main considerations when using wild card file names - first, to properly parse and translate the entered wild card filename to produce an acceptable *search mask* or File Control Block (FCB) for DOS; and second, to *find* all the filenames that match the requested search mask.

MS-DOS allows two wild card characters (*?* and *) in filenames, but some internal DOS functions only recognize the *?* wild card character. DOS provides a built-in parsing function ($29) can be used to generate a *correct* search mask from a filename string with or without wild cards.

Version 2.0, and later, of DOS provide two sets of functions for accessing directory information, one using FCBs much like CP/M and the other using DOS file handles. To be compatible with earlier DOS 1.X versions, the FCB function calls will be used here. Refer to Section 5 of the PC-DOS 2.1 Technical Manual for more information.

The basic technique is the same as is used in a directory list program. A DTA (disk transfer area) is set up for the disk operations using the *set DTA* DOS function call ($1A). An unopened FCB is set up (containing the search mask) and passed to the *get first entry* DOS function ($11). If found, the name is returned in the current DTA. Then subsequent calls (using the same FCB) are made to the *get next directory entry* DOS function ($12) until no match is found. In this application, instead of listing the filenames to the screen, each is used as a parameter specifiying the file for processing by the program.

However, the DTA area used by the directory calls *getfirst* and *getnext* MUST be preserved if other disk accesses are done between directory calls. This is accomplished by using two DTA variables and switching between them. Then any required disk operations may be done by the routines which process each file found without destroying the directory search data.

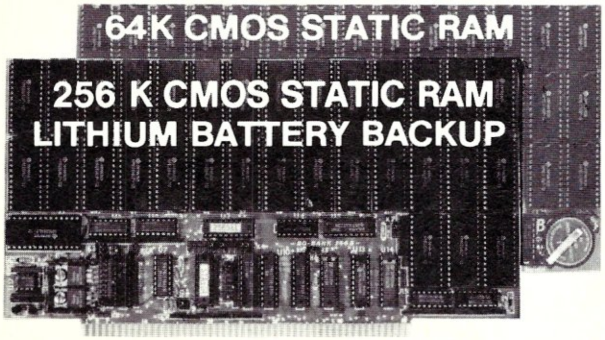
The following program uses all of the above concepts to provide a general form for wild card file processing. The process used here is a simple page format list to printer routine for listing program source files. To use this design for other file processing tasks, simply replace the *LISTPROC* function with your own (and change the call in procedure FIND), using a similar file open technique and returning an error code of zero if no problems were encountered, non-zero if an error occurred.

This program and some 300 other Turbo Pascal and public domain programs are available 24 hours a day for free downloading on the "High Sierra RBBS" system at 209/296-3534.

David W. Carroll is a freelance writer and computer consultant living in the Sierra Nevada foothills near Sacramento, California. He is the author of "Telecommunications with the IBM PCjr" co-published by Microtext/Prentice Hall in March 1985 and "Programming with Turbo Pascal" co-published by Microtext and McGraw Hill.

```
program listwild;

{This program accesses files using command line wild-cards.}
{It works with MS-DOS (or PC-DOS) versions 1 and 2. }

{Copyright 1985 by David W. Carroll}
{All commercial rights reserved.}

type
  regpack = record
        case integer of
          1: (ax,bx,cx,dx,bp,si,di,ds,es,flags: integer);
          2: (al,ah,bl,bh,cl,ch,dl,dh         : byte)
     end;

  fcbarray =    array[0..36] of char;
  strtype  =    string [14];
  comstr   =    string[127];

const
   getdta =       $1a;
   getlstdir =    $11;
   getnextdir =   $12;
   parsename =    $29;

var
  buffer  : comstr;
  comline : comstr absolute cseg:$80;
  inch    : char;
  filestr,
  filename: strtype;
  dfcb,
  dta,
  dta2    : fcbarray;
  user_input : boolean;

function listproc(fname:strtype) : byte;

const
  lines_per_page = 66;
  chars_per_line = 79;
  bottom_margin = 8;
var
  infile       :  text;
  timel,
  datel        :  string[8];
  infname      :  string[20];
  max_lines    :  integer;
  goodfile     :  boolean;

procedure open_file;
const
  bell = 07;
begin
    infname := fname;
    assign(infile,infname);
    {$I-} reset(infile) {$I+};
    goodfile := (IOresult = 0);

     if not goodfile then
     begin
       write (chr(bell));
       writeln ('FILE ',infname,' NOT FOUND');
       delay(2000)
     end;
end;

procedure list;
var
  p,
  line  : integer;
  txtline,
  printline  : string[255];

procedure print_heading(page:integer);
const
  space = ' ';
begin
  if page <> 1 then writeln(lst,chr(12));
  writeln(lst);
  writeln(lst);
end;

begin     {list}
  p := 0;
  while not eof(infile) do
  begin
    p := p + 1;
    print_heading(p);
    line := 4;
    while (not eof(infile)) and (line < max_lines) do
    begin
      readln(infile,txtline);
      writeln(lst,txtline);
      line := line + 1;
    end;
  end;
```

```
    writeln(lst,chr(12));     {form feed}
  end;     {list}

  begin     {listproc}
    max_lines  := lines_per_page - bottom_margin;
    open_file;
    if goodfile then
    begin
      list;
      close(infile);
      listproc := 0;
      writeln;
      writeln(' - listing done -');
    end
    else
      listproc := 1;
  end;    {listproc}

  procedure setDTA(num:byte);  {set Disk Transfer Address}
  var
    regs:        regpack;

  begin
    with regs do begin

     ah := getdta;
     case num of
     1:  begin
           ds := seg(dta);
           dx := ofs(dta);
         end;
     2:  begin
           ds := seg(dta2);
           dx := ofs(dta2);
         end;
     end;
     MSDOS(regs)
   end
end; {setDTA}

procedure calldir(calltype : byte; var errflag : byte);
var
  regs:        regpack;

begin
  with regs do begin
    ah := calltype;
    cx := 0;
    ds := seg(dfcb);
    dx := ofs(dfcb);
    MSDOS(regs);
    errflag:= al
  end
end; {calldir}

procedure parse(var errflag:byte);
var
  regs : regpack;
begin
  with regs do begin
    ah := parsename;
    ds := seg(buffer[1]);
    si := ofs(buffer[1]);
    es := seg(dfcb);
    di := ofs(dfcb);
    al := $0F;
    MSDOS(regs);
    errflag := al;
  end;
end;  {parse}

procedure find;
const
  space  = ' ';
  period = '.';
var
   i,
   err:    byte;

begin
  for i := 0 to 36 do dfcb[i] := chr(0);
  if not user_input then
    writeln('Search mask: ',buffer:15);
  writeln;
  parse(err);
  setDTA(1);              {set 1st DTA for get func.}
  calldir(getlstdir, err);{get first entry matching mask }
  while err = 0 do
  begin
    filename:= '';
    for i:= 1 to 11 do
    begin
      if dta[i] <> space then
        filename := filename + dta[i];
      if i = 8 then filename := filename + period;
    end;
```

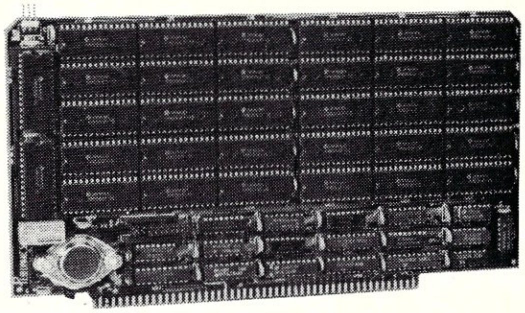


```
    writeln(filename);
    setDTA(2);             {set 2nd DTA for file processing}
    err := listproc(filename);{process file}
    if err = 0 then
    begin
      setDTA(1);
      calldir(getnextdir, err);          { get next entry }
    end;
  end;
  writeln;
end; {find}

begin  {listwild}
  buffer := comline;
  user_input := false;
  writeln('Wild card program lister');
  writeln('This program formats and lists all specified');
  writeln('files on default drive to system printer.');
  writeln;
  if length(buffer) < 1 then
  begin
    write('Enter search mask: ');
    readln(buffer);
    user_input := true;
  end;
  if length(buffer) > 0 then
    find
  else
    writeln('Program Terminated');
end.   {listwild}
```

YOU CAN'T BUY HIGHER QUALITY, MORE RELIABLE S-100 *STATIC MEMORY BOARDS* AT ANY PRICE!

256K BYTES
S-RAM 256™

$699* Qty. One



128K BYTES
S-RAM 128™

$449* Qty. One

STANDARD FEATURES:

- Operates in excess of 12 MHz
- Certified system ready (industrial grade dynamic burn in)
- Supports 8 and 16 bit data transfers
- IEEE 696/S-100 compatible
- 24 bit addressing
- Address strappable to any 128K block within the 16 meg address range
- Extremely low power consumption
- Single +5 volt operation
- One year warranty

OPTIONS

- Battery back-up capability
- Military MIL-STD-883B type burn in
- Faster ram chips available

***SAVE AN ADDITIONAL $100!** Buy one 256K board at $699 and receive a $100 certificate towards the purchase of your next 256K board. Buy one 128K board at $449 and receive a $50 certificate towards the purchase of either board.

NO HIDDEN CHARGES! Performics Inc., prices include dynamic burn in, insurance and UPS ground delivery within the continental U.S.

TERMS: Certified check, COD, Visa/Mastercard   Volume discounts available.

Performics continues to deliver high performance products and excellent service at realistic prices.

S-RAM 128 and S-RAM 256 are trademarks of PERFORMICS INC.

MasterCard VISA

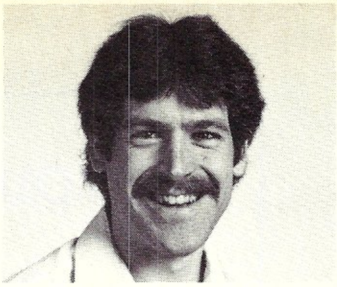
PERFORMICS INC.

ORDER NOW!
(603) 881-8334

11 Morning Dove Rd. • Kingston, New Hampshire 03848

The C Forum

by Don Libes



Context - independent macros

Macros provide a means of textual substitions in the source code at compile time. Most people get along fine with an intuitive idea of how they work, but occasionally stumble over anomolies.

These anomolies come from the C preprocessor's (cpp) limited understanding (less than ours) of C. For example, while cpp knows not to perform substitutions inside of quoted strings, it doesn't prevent changes in operator bindings, as the following example shows:

```
/* weight of a battery components */
#define ACID_WEIGHT       5
#define COVER_WEIGHT      1
#define BATTERY_WEIGHT  ACID_WEIGHT + COVER_WEIGHT  /* WRONG! */

        /* compute total weight of 10 batteries */
        total_weight = 10 * BATTERY_WEIGHT;
```

In this example, the assignment turns into a computation of 10 * 5 + 1. This is evaluated as (10*5)+1, not 10*(5+1) as the programmer wanted. The problem is that an operator (+) inside the macro was of lower precedence than one (*) outside the macro. Rather then learning all the rules of operator precedence, it is easier to simply remember to always enclose your entire macro with parentheses.

It is rare that this ability of cpp to change bindings is what is desired, although it is used occasionally (and typically produces obfuscated code). See Vol 1, #4 of M/SJ for some classic examples!

Unfortunately, cpp can complicate things even further because it has the same effect on statements that it has on expressions. This is a much more troublesome problem, because the indentation that we use is completely ignored by the C compiler. While we understand the control of the program at a glance because of the indenting of the code, it can be misleading when the program has a completely different structure in reality.

For example, suppose we define the macro #assert. #assert is typically used to state a predicate that is always true. It generates actual code that checks that its argument is true. What is the point of that?

Think of it as being able to execute your comments; to check that the code really does what the comments say it does. If the assertion is ever false, a serious error has occurred - your program no longer matches your specification. For example, if you are writing a program and at a certain statement, you are confident that a pointer will always be valid (non-zero), you can place the statement:

```
        assert(pointer != NULL,"pointer is null!");
```

This will generate code to verify the expression when the program is run. If the expression evaluates to 0 (false), a message will print stating that ''assertion failed: pointer is null!''. It is possible to set the macro up so that when you are ready to create a final program, the assertions are effectively ignored. (I will show this later.) Another possible use for this would be array bounds checking. During program development you might want this, but when you are convinced your program works, you can compile it without this feature.

A simple macro to do the assertion is:

```
#define assert(expr,msg) \
        if (!(expr)) printf("assertion failed: %s\n",msg)
```

Look at this macro very carefully. You should notice two things. The first is that we have put parentheses around expr. You should be able come up with an example where #assert would not perform correctly if we left the parentheses out. Also, take note that there is no terminating semicolon at the end of the if statement. This allows us to use the macro as if it were a function call, that is, to terminate it with a semicolon at the point of usage.

Now suppose that we have the following piece of code:

```
        if (a > b) assert(a+1 > b, "eh?");
        else c = 5;
```

What happens after the macro is evaluated, is:

```
if (a > b)
        if (a+1 > b) printf("assertion failed: %s\n","eh?");
        else c = 5;
```

The *else* has bound with the wrong *if*! Not only will this function behave incorrectly, but the C compiler will not complain about it. It will compile just fine. These kinds of errors can be very hard to find, because the block and control bindings of the program don't match the indentation level. Humans believe very strongly in white space, while the C compiler ignores it entirely.

The problem has occurred because our macro looks like a function call but doesn't behave like one. Because the macro is only a textual substitution, it is free to rebind itself with any part of the program that the control structure syntactically warrants. If #assert had been a function, this would never had happened (and sometimes this is your only recourse).

In this case, an if statement bound with an else to become an if-else statement. One way to stop this rebinding is to make the if statement into its complete cousin. For example:

```
#define assert(expr,msg) \
        if (!(expr)) printf("assertion failed: %s\n",msg); \
        else
```

Notice, the else just hanging off the end? This prevents another else from binding with the if. Remember that we are assuming that the user will terminate the macro use with a semicolon so that the else will be followed immediately by a semicolon (or more precisely, with a null statement).

Unfortunately, if you forget the semicolon, you will have more trouble since the following statement will be sucked up as the statement to be executed in the else clause (again without errors or warnings of any kind)!

Well, then, why don't we just put the whole thing in a pair of braces? This seems like a good idea, because it makes the macro into a complete C statement. You can also declare local variables and have other control structures such as loops in the macro.

Unfortunately, there is still a problem with this. (Think about it before reading on.) The problem appears in the following piece of code:

```
if (expr) assert(expr2,"...");
else ....
```

which turns into:

```
if (expr) {
        if (expr) printf .....
};
else ....         /* syntax error! */
```

What happened is that the assert is replaced by a block, but the block is followed by a semicolon. This semicolon came from the user adding it to the end of the macro. Unfortunately, it now terminates the if prematurely.

Is there anything that will do the job? The answer is yes, but its not exactly pretty. (You won't hit yourself on the head and say "Why didn't I think of that"!) The following macro does the job nicely:

```
#define assert(e,m)      do {\
                                if (!(e)) printf(m);\
                         } while (0)
```

This is very similar to the previous case where we enclosed the macro in a block. The difference is that this one requires a semicolon to make it into a syntactically correct and complete statement. The block is sealed from misinterpretation through rebindings, and it will be executed once and only once. Although it may seem like an unusual way to do it, rest assured that every (reasonable) compiler will optimize the statement so that the extra test is not performed.

I particularly like this macro because if the user leaves off the semicolon, by mistake, the C compiler will complain.

On the other hand, the compiler won't complain with a macro like:

```
#define assert(e,m)      if (1) {\
                                if (!(e)) printf(...); \
                         }
```

For people who absolutely have to have the shortest macro, (no matter much readability they sacrifice,) here it is:

```
#define assert(e,m)      if (e) ; else printf(...)
```

It is particularly ugly, because 1) the test is reversed which can be a little confusing, and 2) there is a null "then" clause due to the semicolon immediately following the expression, which is easy to miss.

I'm not encouraging you to use these last two macros (or impress you with my knowledge). Rather I'm pointing out that you may see things like this when reading C source, and you should be able to understand it when you see it.

Note that in each macro above you can replace the printf by a block of statements.

It is sometimes useful to have an if statement embedded in an expression (as opposed to a statement). This can be done with the following macro:

```
#define EXPR_ASSERT(expr,msg)     (expr || printf(....))
```

although this is not that useful, since you can't have local variables and looping control structures. Its occasionally called for, though.

What really makes #assert useful is that once you have debugged your program, you can turn off the assertion checking in the final version without doing any editing of your source.

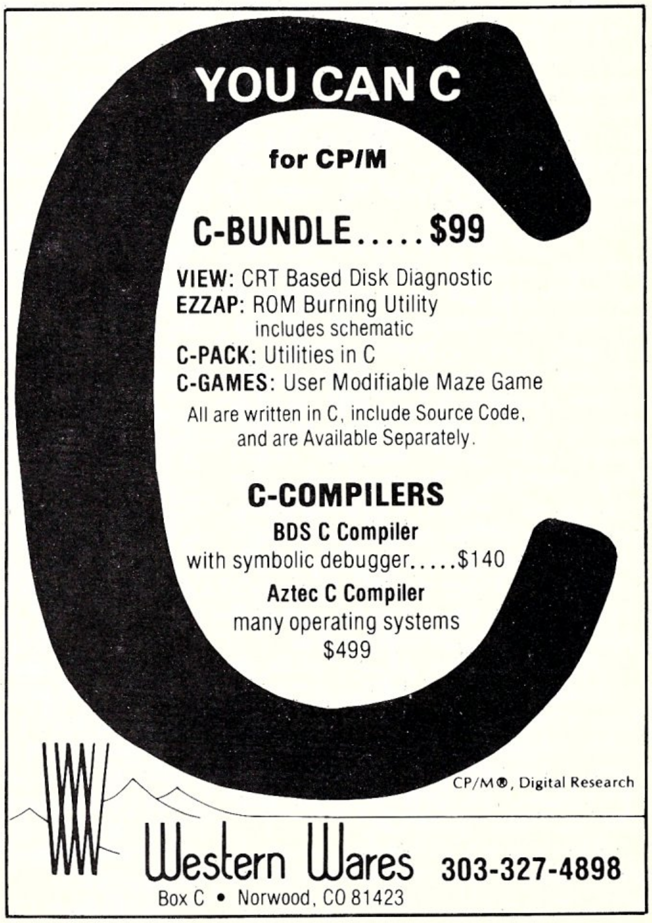
Simply store #assert and any other debugging macros in a file called debug.h as follows:

```
/* debug.h */
#ifdef DEBUG
#define assert (expr,msg)          if (1) {\
                                   if (!(expr)) printf(msg);\
                                   }
#elseif
#define assert(expr,msg)           ;
#endif DEBUG
```

Now if you define DEBUG when compiling, the assertions will generate real code, otherwise they are turned into null statements. (Convince yourself that the null statement doesn't mess anything else up!)

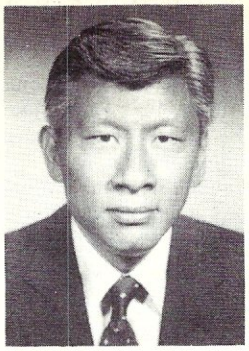
I encourage readers to write to me about topics or problems that you want to know about. I want this column to be reader driven. Write to me care of M/SJ, Box 1192, Mountainside, NJ 07092.

Don Libes is a computer scientist working in the Washington, DC area. He works on artificial intelligence in robot control systems. He is also the son of Lennie and Sol Libes.



PC/Blue Report

by Hank Kee



Editor's Note: Hank Kee is the librarian for the PC/Blue public domain software library. He is the person who collects, assembles, and checks all the software issued by PC/Blue and then compiles and edits them into the released volumes.

Build Your Own Program Library Using Public Domain Software

In the past Sol Libes and I, at different times, discussed, in this publication, and in the old Microsystems, assembling an IBM PC clone. The task has come a long way. Availability of components has increased and costs have dropped. Almost anyone who can use a screwdriver can put together a system. Not only have hardware costs dropped dramatically but so have software costs dropped, if one makes full use of public domain software, such as exists in the PC/Blue Library.

The main difference between commercial software and software in the public domain is mainly the depth of functionality. For example, Lotus 1-2-3 is currently the most popular commercial spreadsheet program. I would venture to guess that more than half the 1-2-3 users don't take advantage of its complex functions. This is also true for other programs like word processing and database management systems. There are however exceptions to the rule. I feel that public domain communication programs are far more creative and functional than their commercial counterpart systems.

We can divide public domain software into three basic catagories - spreadsheets, database management systems, and word processors. Another catagory can be best described as esoteric. We have in the public domain library systems like XLISP, graphic display and bulletin board systems which really don't command a very large market. The third category consists of utilities. These programs tend to be small in program size. They include screen blanking, color graphics selector and many others. There is a multitude of them on the various bulletin boards across the country.

For this issue I will comment on building your own software library from what is available in the PC/Blue public domain library. Our base library should contain a word processor, spreadsheet, database manager, and communication programs.

For a word processor, there is ***PC-Write*** (Vol. 130) and ***WPK*** (Vol. 95). PC-Write reminds me of Volkswriter and Wordstar. It is a very functional program. It is memory mapped for high speed like Volkswriter. Everything is constructed in memory. Alas, it is similar to Wordstar in that the functional controls are accessed via control characters. Once the function keys are mastered, however, it is an excellent word processor. For the young ones in the home, WPK (Word Processing for Kids) is very easy to use and learn. Its very intelligent use of graphics makes it a joy to use.

PC-Calc (Vol. 115) and ***FreeCalc*** (Vol. 50) function in a manner very similar to the classic VisiCalc. Simple and functional and yet not encumbered by the extended features like regression routines, present values, or macro generation. For the home, we don't need something as complex as Lotus 1-2-3. On the contrary, PC-Calc is as simple to use as 1, 2, and 3.

PC-File (Vol. 46) started off in the public domain or user supported sector. It has since graduated from the ranks and gone commercial. In its place, there are a number of candidates one can choose from. ***PC-DBMS*** (Vol. 131) and ***3BY5*** are pretty nifty programs. A clear cut leader from a bunch of good database managers will only come with their continued distribution and acceptance in the public domain library network. Database management systems will vary in functionality. They may perform the same tasks but the presentation can make two similar functions packages look like night and day. Pick the one that best meets your requirements.

QModem (Vol. 141) has fast become a very popular communications package. The author is constantly updating it. Sometimes a little too often. But we should never complain about a good thing. QModem is an excellent program that has incorporated graphics, windows, directories, and split screen operation. There are now many anciliary programs available for QModem such as background dialers. A challenger to QModem is ***Termulator*** (Vol. 143). Termulator does not have the cosmetics of QModem, but it has the Kermit and XModem protocols. It can also emulate a large number of terminals. Least of all, let's not forget ***PC-Talk III*** (Vol. 143). I think it has been surpassed by QModem, but for a very long time it set the benchmark standards for a public domain or user supportd communications program.

A favorite of mine in the PC-Blue library is PC Picture Graphics (Vol. 44). IBM has since included this in their class C software product catalog. The updated version is very modestly priced. But the original version works almost as well at a very modest copying cost.

I will comment on esoteric systems and utilities to be added to a base library consisting of public domain or user supported software in following issues.

During the past two months, a number of new volumes have been added to the PC/Blue public-domain software library. They are the following.

Vols. 138 & 139
- Remote Bulletin Board System v12.5A
- Capital PC Users Group - vol 1 & 2

Vol. 140
- PIBTERM version 2.0 terminal emulator and host communications facility
- JUDY version 1.01 background desktop utility

Vol. 141
QModem version 1.07 modem communications
PC-CALC - electronic spreadsheet
DIALER - background dialer
RM1200B - software power on reset

Vol. 142 Miscellaneous Utilities -
archive utility/overwrite boot sector
examine all RAM (1MB)/file encryption
list utilities/file managers
Buerg's library utilities
squeeze/unsqueeze
disk cataloging/display environment
sort large files/TREE directory

Vol. 143
PC/Talk III v2.6 - Modem Communications
Termulator v1.0 - Kermit/XModem Compatible
Minitel v3.2 - Modem Communications
Butler v1.01 - Automatic Modem Communications

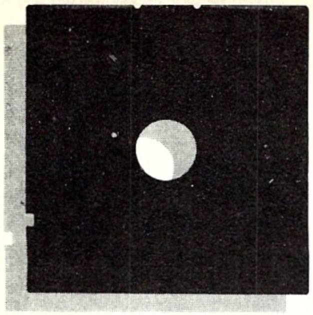
Vol. 144 Miscellaneous Utilities -
area code utility/create .BAT files
Bly's utilities/simple sci calculator
auto CAPS LOCK/file encryption
DES module/memory map display
disk and file utilities

Vol. 145 Hints, Tips, & Notes -
1-2-3 <--> dBase III
dBase III fixes
Macro Assembler tutorial
Copy B <--> C
extending DoubleDOS 1.0
640KB on XT systemboard
Multimate fixes
Turbo Pascal tutorial
Wordstar patches

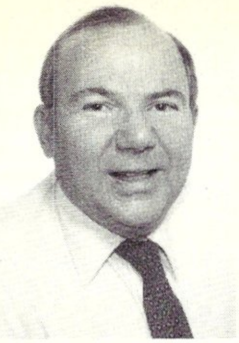
Vol. 146 XLISP version 1.5

Vol. 147 Fun & Games -
alfred / bumblebee / eliza / face / flys / funphone / furelise / happy / hopper / insults / joycal / kaleidoscope / keyclick / kirk / luv / merry / piano / polymaze / qt2 / romzap / santa / sc / sharks / simpmaze / soundcar / spock / thatsall / typerite / vertigo / werzwold

PC/Blue disks are available from the New York Amateur Computer Club, Inc., Box 106, Church Street Station, NY NY 10008. Price is $7 per volume which includes media, postage, and handling. On foreign orders, please add $2 per order. A cross referenced printed catalog may also be purchased from the above address at $5.



In the SIG/M Public Domain



by Stephen M. Leon

Steve Leon is the SIG/M Disk Editor. In other words, he is the person who assembles, compiles and edits all the of the SIG/M public domain software disks. Thus, he speaks with the greatest authority as to what is going on in the SIG/M public domain software area.

THE PROTOCOL WAR

I rented a video tape a few weeks ago...a film called 'Protocol' with Goldie Hawn. It was so bad we just never bothered to see more than a few minutes of it. What we did, however, was turn on the computer, phone Compuserve and tune in to the "Protocol War".

It was the hottest thing around, in recent weeks, on the CP/M SIG (GO PCS-47). Always an interesting place, in recent times CP/M SIG has been bubbling with sarcastic wit, attack, retreat and sundry forms of mayhem. The subject...revising Ward Christensen's XMODEM protocol to handle 1024 byte blocks.

The present XMODEM error checking is based upon transmission of a 128 byte data block. After the receiver gets the block and checks it, it advises the sender all is well - continue, or - send the block again. With 2400 baud modems (and even 1200 baud) the experts say transmission speed can be significantly improved by increasing the block size to 1024 bytes and changing the method of retransmitting errors.

Ward had put a message on the SIG that he was doing an article for BYTE on the protocol and proposed adoption of the YMODEM protocol to bring XMODEM to 1K blocks. It seemed a logical way of handling the matter. YMODEM has been extensively tested and in wide use in the 16-bit world. Paul Homchick then went ahead and did the code. Ron Fowler put the code into MEX and we released SIG/M Volume 241 with the new XMODEM (XMODM110) and the new MEX (MEX114).

One would expect the problem has been solved. However, this was just the beginning. It seems there is a movement, led by Irv Hoff, to have *two* standards for 1024 byte blocks. Out in California they never discovered that if you take your shoes off one can count to 16. As a result, the *revision an hour* crew, led by Irv, has been putting out *new* program versions with non-sanctioned protocols. Therefore, watch out for version numbers greater than XMODM110. In the meantime, Irv has promised to send SIG/M his new IMP - (Irv's Modem Program). More on the protocol wars next issue.

FOREIGN SIG/M CONTRIBUTORS

CP/M is very popular in Australia and every month or so SIG/M gets a contribution from Bill Bolton of Software Tools of Australia (PO Box 357, Kenmore, QLD, 4069) with their latest public domain software releases. Australia must be modem heaven. The vast distances must make for a telephone way of life and modem and BBS systems are high on the frequency list of releases. Their library is also heavy on programs for limited distribution computers.

Derek Fordred of the United Kingdom CP/M Users Group (72 Mill Road, Hawley, Dartford, Kent, England DA2 7RZ) is another frequent contributor. Unfortunately, the UK library features quite a few non-standard BASIC and PASCAL programs, which means that only a few of their releases make it into the SIG/M library for distribution in the U.S.

Readers interested in Australian and U.K. public domain software libraies can contact Bill and Derek directly.

NEW SIG/M RELEASES

Our new releases include an updated KERMIT for CP/M-80. KERMIT is widely distributed as *the* mainframe-to-micro modem program. Many IBM PC commercial packages include a version of it for communications. It has a number of limitations, but as its distribution keeps expanding it is becoming a *must have* package.

The Disk-Jockey Controller is in reasonably wide use and Robert Davies of Wellington, New Zealand has written a CP/M+ BIOS for it (SIG/M Vol 236). Also in that volume is a fine article by Bob on CP/M+ together with an updated LISP in PASCAL/Z.

Interest in transmission of computer data by radio is growing. The SIG/M library contains a number of programs of interest to packet radio buffs. The latest releases include two volumes in this area. Ed Elizondo has implemented all three levels of the CCITT X.25 Packet Standard for the Big Board (Vol 238). It contains full documentation and should be readily adaptable to other machines. Full documentation is also available for BASICODE radio transmission (Vol 235). Unfortunately, the documentation is mostly in Dutch. We are working on a translation. In the meantime, true radio fans should find it of interest. Additional packet radio material is also available on SIG/M Volume 143.

We really can't do justice to the ten volumes covered by SIG/M 232 through 241 in the space remaining, so we will just list them and perhaps cover them in more detail in the next issue. One last word, however, with respect to the fine contribution we received from Rick Surwild. He has written an excellent full screen Z80 debugger with full (90 pages) documentation (Vol 239). This is a major work, although as you can see from the list, there are a number of major innovations on these ten volumes.

SIG/M Volumes are available on 8" SSSD Disks for $6.00 each ($9.00 foreign) directly from SIG/M, Box 97, Iselin, NJ 08030. Printed catalogs are $3.00 each ($4.00 foreign). Disks in a variety of formats may be obtained through the worldwide SIG/M distribution network (which was published in the last issue of M/SJ). The distributor list is included with the printed catalog. A disk version of the catalog (Volume 00) is available for $6.00. Many bulletin boards have the software for downloading and most new releases are available on the Compuserve CP/M SIG.

Vol 232 - Kermit 3.9 for CP-M-80, Z80 translator
released July 19, 1985

BBCBIOS .MQC	Modified Ferguson Big Board 1 BIOS
KERMIT .VER	KERMIT communications version list
KRMSRC .LBR	KERMIT version 3.9 for CP/M-80
M7BBS+ .AQM	Modem 7 for Ferguson Big Board 1
XLATE216.LBR	Updated XLATE 8080>Z80 source translator

Vol 233 - Miscellaneous CP-M-80 Utilities
released July 19, 1985

BOY-MOR .LBR	Z80 Boyer-More string search function
CLEANUPC.LBR	Cleans up C source, makes cross listing
DISKNAME.LBR	Gives 45 char volume name on CP/M disk
MODEM7B .LBR	MODEM7A modified for LST device output
PMATCH .LBR	Checks print command pairing for Word*
SDDIRENT.LBR	Updated SD77, more information displayed
UDM1200 .DQC	Using Modem Tech UDM1200 at 1200/75 bps
UDMAUTOD.BAS	BASIC auto dialer for Modem Tech UDM1200
VDUPRNT .LBR	Screen dump Microbee to Tandy DMP 200
XPRINT .CQ	Extended print utility in C

Vol 234 - CP/M+ & Other Utilities, Curve Fitting
released July 19, 1985

AUTO .LBR	Auto execute program on cold boot
CPM3UTIL.LBR	Disk & other utilities for CP/M+
CUBE .LBR	Solves 5 x 5 cube
CURVFIT .LBR	Curve fitting in MBASIC
DISKSTAT.LBR	Display disk info and file allocation
DUMP .LBR	Improved file dump utility
SBCOPY .LBR	Fast copy utility for SuperBrain
SPZ .LBR	Full screen disk utility SuperBrain version
SYM .LBR	Make SID symbol table from MACRO-80 listing

Vol 235 - BASICODE Radio Communications (in Dutch)
Netherlands Enhancements to FIG FORTH
released July 19, 1985

ABSTRACT.235	Abstract of disk contents (in Dutch)
BASICODE.DOC	Documentation on BASICODE (in Dutch)
BASICODE.ENG	Brief intro to BASICODE (in English)
BC2 .BQS	"Standard" BASICODE-2 for MBASIC
BC2 .OSB	"Osborne" BASICODE-1 for Osborne 1
BC2 .OSI	"Ohio Scientific" BASICODE-2
BCREAD .AQM	Read BASICODE-2 from parallel port
BCTEST .AQM	Test program - BASICODE-2 port transfer
BCWRIT .AQM	Write out BASICODE-2 program, std
BCWRITE .AQM	Write out BASICODE pgm, (BASICODE-1?)
BOKA&EI .BQS	BASICODE-2 Tic Tac Toe, in Dutch
CONVERT .BQS	CBASIC source - BASICODE-2/MBASIC xlate
CONVERT .COM	Converts BASICODE-2> Microsoft BASIC
DELERS .BQS	BASICODE game of some sort, in Dutch
DIGIKLOK.BQS	BASICODE-2 on screen digital clock
FEESTEN .BQS	BASICODE-2 show public holidays-any year
FORTH-NL.LBR	FIG FORTH 1.1 with enhancements for CP/M
FUNCTION.DAT	Used with CONVERT.COM
KALFEEST.BQS	BASICODE-2 calendar-any month & year
KEYWORDS.DAT	Used with CONVERT.COM
MASSA .BQS	BASICODE-2 demo program, in Dutch
SORT .BQS	BASICODE-2 sort program, in Dutch
STELSELS.BAS	BASICODE-2 solves 3 order quadratics?
TANKIE .BQS	BASICODE-2 game, simple ASCII graphics
TESTBC2 .BQS	BASICODE-2 test/excerciser program
TIMER555.BQS	BASICODE-2 designs 555 timer circuits
UITLEG1 .BQS	Introduction to BASICODE? (in Dutch)
UITLEG2 .BQS	See UITLEG1.BAS

Vol 236 - CP/M+ BIOS for Disk-Jockey Controller
Updated Lisp in Pascal/Z
released July 19, 1985

CPM3BIOS.DQC	CP/M+ BIOS for Morrow Disk Jockey-Doc
SCB .ASM	BIOS routines
MOVE .MAC	" "
CHARIO .MQC	" "
DISKDEFN.ASM	" "
DISKHNDL.MQC	" "
SEMIHNDL.MAC	" "
BOOT .MAC	" "
GENCPM .DAT	Sample data for GENCPM
DDLDR .ASM	Loader BIOS routines
LDRBIOS .MAC	" "
LOADSYS .MAC	BIOS routine loader
REBOOT .MAC	To cold start
SEESCB .RAT	Print the SCB
ACCSCB .MAC	" "
SEESCB .COM	" "
BARB2 .MAC	Memory check
IF .COM	Conditional execution
IF .MAC	" "
OVLMNGR .MAC	Overlay manager
PRINT .COM	Spooling printer
PRINT .MQC	" "
CPMPLUS .TQX	Article on CP/M+
LISP .COM	Lisp in Pascal/Z
LISP .DOC	from Vol 27 Pascal/Z
LISP .PAS	Users Group
INITLISP.	/
INITLISP.STB	/

Vol 237 - Pilot in Pascal/Z
Solving Deductive Reasoning Puzzles
released July 19, 1985

PILOT-P .LBR	Pilot in Pascal/Z
PUZZLE .LBR	Helps solve deductive reasoning puzzles

Vol 238 - CCITT X.25 Packet Standard (all 3 levels)
released August 16, 1985

BUFFERS .AQM	Full implementation of CCITT X.25
FILES .AQM	packet standard (all 3 levels) under
LEVEL1 .AQM	CP/M-80. Implements a dumb terminal
LEVEL2 .AQM	with bi-directional file transfer
LEVEL3 .AQM	capability. Configured for Digital
PLOG .AQM	Research Big Board (uses SIO on BB
X25 .AQM	to handle HDLC bit stuffing and
X25 .CQM	polynomial check). Extensive diag-
X25 .MSG	nostic facilities built in. Full
X25SYS .DQC	User manual.
X25UM .DQC	/
XUTIL .AQM	/

Vol 239 - Z8E Z80 Debug Monitor
released August 16, 1985

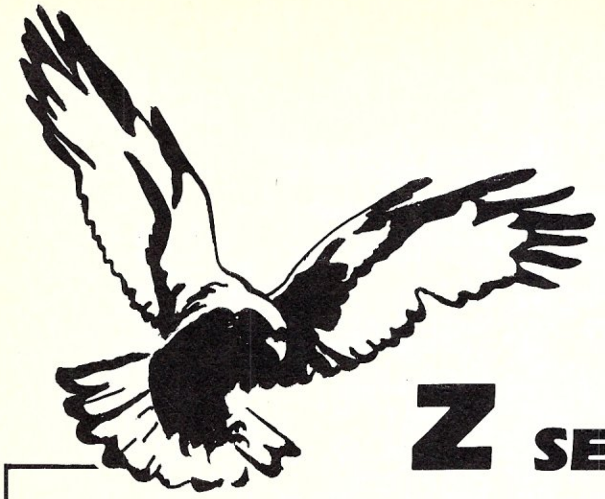
Z8E .CQM	Symbolic Z80 debugger. Full screen
Z8E .DQC	animated display of program under
Z8E .MQC	test, built in assembler, 16 break-
Z8E .SYM	points, 20 + commands. Full doc file

Vol 240 - Extracts from UK CPMUG - Volumes 21 & 23
BASIC (Z80) Bulletin Board
released August 16, 1985

ASM65 .DOC	converts ASM.COM to 6502
ASM65 .HEX	cross assembler
BACKUP .COM	Winchester backup program (Z80)
BACKUP .DOC	/
BACKUP .MAC	/
ECCE .DOC	Edinburgh Compatible Context Editor
ECCE48K .COM	line oriented editor with macro
ECCE56K .COM	capability and conditional commands
ECCE60K .COM	/
ACK .DOC	/
ZBBS .LBR	BASIC BBS (Z80) system

Vol 241 - Updated XMODEM, MEX with Revised XMODEM
Protocol; Turbo Pascal Bulletin Board
released August 16, 1985

MEX114 .LBR	Update to XMODEM & MEX using new
XMODM110.LBR	1024 protocol for faster transfer
TURBOBBS.LBR	Turbo Pascal BBS system



Z SETS YOU FREE!

Free to create computer environments right for you . . . free to automate repetitive tasks . . . free to increase your productivity. Z-System, the high-performance 8-bit operating system that flies! Optimized assembly language code — full software development system with linkable libraries of often needed subroutines —relocating (ROM and RAM) macro assembler, linker, librarian, cross-reference table generator, debuggers, translators, disassembler — ready to free you!

TERM III New generation communications package provides levels of flexibility, functionality, performance not available until now. Replaces BYE and XMODEM . . . master/server local area network capability . . . public or private bulletin board and electronic message handling are integral features . . . auto-dial/answer, menu install . . . XMODEM (CRC/Checksum), MODEM7 Batch, Kermit, CIS, and XON/XOFF protocols . . . 100-page manual **$99.00**

Z-MSG Rolls Royce of message handling systems . . . mates with TERM III or BYE for most advanced overall electronic mail/file transfer capabilities . . . menu installed . . . extreme configurability . . . many levels of access and security . . . word, phrase editor, field search . . . complete message manipulation and database maintenance **$99.95**

DISCAT Elegant, menu and command-line driven file and disk catalog manager. Generates and controls multiple master catalogs, working catalog used for update quickness. Nine flexible modules easily altered by user for custom requirements. Works with Z shells (VMENU, VFILER, MENU), aliases, and multiple commands per line **$39.99**

ZCPR3: The Manual Bound, 350 pages, typeset book describes features of ZCPR3 command processor, how it works, how to install, and detailed command usage. Bible to understand Z-System **$19.95**

ZCPR3 and I/OPS Loose-leaf book, 50 pages, 8-1/2" by 11", describes ins-and-outs of input/output processing using Z-System. Shows how to modify your BIOS to include I/O redirection . . . complements **The Manual** **$9.95**

More missing links found — Z Application Progams! Fly with eagles! Our programs promote high performance through flexibility! Productivity results from dynamically changeable work environments, matching operator to tasks and machines.

Above programs require 48K-byte memory, ZCPR3, Z-Com, or Z-System, and Z80/NSC800/HD64180-based computer. Shipping from stock. State desired disk format, plus two acceptable alternatives. As payment, we accept Visa, Mastercard, personal checks, money orders, and purchase orders from established companies. We also ship UPS COD.

Call or write to place order or to obtain literature.

Echelon, Inc. 101 First Street • Suite 427 • Los Altos, CA 94022 • 415/948-3820

Tools That Make Your Job Easier

For PCDOS/MSDOS (2.0 and above/128K) • IBM PC/Compatibles, PC Jr., Tandy 1000/1200/2000, & others
For CPM80 2.2/3.0 (Z80 required/64K) • 8″ SSSD, Kaypro 2/4, Osborne I SD/DD, Apple II, & others

MIX EDITOR

Programmable, Full/Split Screen Text Processor

Introductory Offer 29^{95}

Great For All Languages

A general purpose text processor, the MIX Editor is packed with features that make it useful with any language. It has auto indent for structured languages like Pascal or C. It has automatic line numbering for BASIC (255 character lines). It even has fill and justify for English.

Split Screen

You can split the screen horizontally or vertically and edit two files simultaneously.

Custom Key Layouts

Commands are mapped to keys just like WordStar. If you don't like the WordStar layout, it's easy to change it. Any key can be mapped to any command. You can also define a key to generate a string of characters, great for entering keywords.

Macro Commands

The MIX Editor allows a sequence of commands to be executed with a single keystroke. You can define a complete editing operation and perform it at the touch of a key.

Custom Setup Files

Custom keyboard layouts and macro commands can be saved in setup files. You can create a different setup file for each language you use.

MSDOS Features

Execute any DOS command or run another program from inside the editor. You can even enter DOS and then return to the editor by typing exit.

MIX C COMPILER

Full K&R Standard C Language
Unix Compatible Function Library

Introductory Offer 39^{95}

Complete & Standard

MIX C is a complete and standard implementation of C as defined by Kernighan and Ritchie. Coupled with a Unix compatible function library, it greatly enhances your ability to write portable programs.

The Best C Manual

MIX C is complemented by a 400 page manual that includes a tutorial. It explains all the various features of the C language. You may find it more helpful than many of the books written about C.

Fast Development

MIX C includes a fast single pass compiler and an equally fast linker. Both are executed with a simple one line command. Together they make program development a quick and easy process.

Fast Execution

The programs developed with MIX C are fast. For example, the often quoted prime number benchmark executes in a very respectable 17 seconds on a standard IBM PC.

Standard Functions

In addition to the functions described by K&R, MIX C includes the more exotic functions like *setjmp* and *longjmp*. Source code is also included.

Special Functions

MIX C provides access to your machine's specific features through BDOS and BIOS functions. The CHAIN function lets you chain from one program to another. The MSDOS version even has one function that executes any DOS command string while another executes programs and returns.

Language Features

- Data Types: char, short, int, unsigned, long, float, double (MSDOS version performs BCD arithmetic on float and double-no roundoff errors)
- Data Classes: auto, static, extern, register
- Struct, Union, Bit Fields (struct assignment supported)
- Typedef, Initialization
- All operators and macro commands are supported

30 DAY MONEY BACK GUARANTEE

Orders Only: **Call Toll Free 1-800-523-9520,** (Texas only **1-800-622-4070)**

MIX Editor ____ $29.95 + shipping ($5 USA/$10 Foreign) Texas residents add 6% sales tax
MIX C ____ $39.95 + shipping ($5 USA/$25 Foreign) Texas residents add 6% sales tax

Visa ____ MasterCard ____ Card # ______________________ Exp. Date ________

COD ____ Check ____ Money Order ____ Disk Format ______________________

Computer ______________________ Operating System: MSDOS ____ PCDOS ____ CPM80 ____

Name ______________________

Street ______________________

City/State/Zip ______________________

Country ______________________

Phone ______________________

MIX software
2116 E. Arapaho
Suite 363
Richardson, Tx 75081

Dealer Inquiries Welcome
Call (214) 783-6001

MSDOS is a trademark of Microsoft PCDOS is a trademark of IBM CPM80 is a trademark of Digital Research WordStar is a trademark of MicroPro

M

COMPETITIVE EDGE

P.O. Box 556 — Plymouth, MI 48170 — (313) 451-0665

HIGH PERFORMANCE PRODUCTS RUNNING UNDER

TURBODOS

TELETEK ENTERPRISES S-100 MASTER/SLAVE BOARDS

Teletek
Systemaster
$495

Systemaster II
$795

8MHZ SBC 86
512K - $963

5 MHZ SBC 86
512K - $795

6MHZ SBC-1
128K - $375

HDC
$375

4MHZ Z80A MASTER SYSTEMASTER® I, 2 SERIAL, 2 PAR, FOC, CLOCK	$495
8MHZ Z80H 128K MASTER, SYSTEMASTER II®, 2 SER, 2 PAR, FDC CLK	795
8MHZ 8086 512K SLAVE SBC 86 RUNS WITH EITHER MASTER ABOVE	963
5MHZ 8086 512K SLAVE SBC 86 RUNS WITH EITHER MASTER ABOVE	795
8MHZ 8087 OPTION FOR 8MHZ 8086 SBC 86	175
5MHZ 8087 OPTION FOR 5MHZ 8086 SBC 86	109
SBC 1 6MHZ Z80B 128K SLAVE RUNS WITH EITHER MASTER ABOVE	375
HDC HARD DISK CONTROLLER — ST506 INTERFACE RUNS WITH ABOVE	375
SBC II DUAL SLAVE BOARD (TWO Z80'S PORTS ETC.)	689

COMPETITIVE EDGE INTEGRATED SYSTEMS

4MHZ Z80A 64K, CPM 2.2, 2-5" FLOPPYS, 4 SLOT CABINET	1295
8MHZ Z80H 128K, SINGLE USER TURBODOS WITH SPOOLING, 2-5" FLOPPYS, 4 SLOT CABINET READY TO RUN	1695
8MHZ Z80H, 128K MASTER, 4-6MHZ 128K SLAVES, 1-5" 20MB HD 10 SLOT 1-5" FLOPPY READY TO RUN WITH TURBODOS	$4795
8MHZ Z80H, 128K MASTER, 4-8MHZ 512K SLAVES, 1-5" 20MB HD 10 SLOT 1-5" FLOPPY READY TO RUN WITH TURBODOS	$7495
4MHZ Z80A, 64K MASTER, 4-6MHZ 128K Z80B SLAVES. 1-5" 20 MB HD, 10 SLOT 1-5" FLOPPY READY TO RUN WITH TURBODOS	$4395
8MHZ Z80H 128K MASTER, 6-8MHZ 8086 SLAVES WITH 8087, 6-6MHZ Z80B 128K SLAVES, FAST 40 MB HARD DISK, 15 SLOT, 1-5" FLOPPY READY TO RUN WITH TURBODOS	$10,695

SYSTEMS AVAILABLE WITH UP TO 16 SLAVES AND 2-112 MD HD (224 MB TOTAL)

EARTH COMPUTER TURBO SLAVE I 8MHZ Z80H-128K SLAVE 2 SERIAL TURBO SLAVE I RUNS WITH ALL TURBODOS S-100 SYSTEMS	395

All Prices Subject to Change — Shipping Extra, Visa M/C Accepted

Systemaster I & II, HDC, SBC I, SBC II are either registered trademarks or trademarks of trademarks of Teletek Enterprises. CPM is a registered trademark of Digital Research Inc., Turbodos is a trademark of Software 2000 Inc.

COMPETITIVE EDGE

P.O. Box 556 — Plymouth, MI 48170 — (313) 451-0665

Competitive Edge brings you Lomas Data Products Inc.
HI SPEED 16 BIT COMPONENTS AND SYSTEMS

S-100 CIRCUIT BOARDS

Thunder 186, 256K, 2 Serial, 1 Parallel, Floppy Disk Controller, Clock, All on One Board with Concurrent DOS	$895

8MHZ Lighting 286 CPU	$821	6MHZ Lightning 286 CPU	$756
10MHZ Lighting 8086 CPU	520	8MHZ Lightning 8086	420
Control It All Floppy & HD Controller	487	LDP Floppy Contr.	220
Hazitall 2 Serial, 2 Par, Clock	260	Color Magic PC Compat.	476
256K Dram Megaram	358	512K Dram Megagram	448
1024K (1 Megabyte) Megaram	599	2048K Megaram	821
Ram 67 128K Static 100ns Chips	596	Ram 67 128K W/Bat B/U	671
Octaport 8 Serial to 38.4K	320	Octaport 4 Serial	200
NV Disk 512K, Memory Drive	371	NV Disk 2048K Wow!!	746
Concurrent DOS Single User	280	Concurrent DOS Multi-user	360
MSDOS for 86, 186, or 286	200	LDP-COM Modem Program	55
CPM22EM 8080 Emulator	55	PC DOS Drivers	35

COMPETITIVE EDGE INTEGRATED SYSTEMS

THUNDER 186, 4 SLOT PC STYLE CABINET, 2-5" FLOPPYS, CDOS	$1595
THUNDER 186, 4 SLOT, 20 MB HARD DISK, 1-5" FLOPPY, CDOS	2595
THUNDER 186, 4 SLOT, 20 MB HD, 512K, 1-5" FLOPPY 4 USER	3195
THUNDER 186, 4 SLOT, 2-5" FLOPPY, COLOR MAGIC, KEYBOARD	2250
6MHZ 286, 1-5" FLOPPY, 1024K, 20MB HD, 10 SERIAL, 15 SLOT, 7-8 USERS	4695
6MHZ 286, 1-5" FLOPPY, 1024K, 40MB HD, 10 SERIAL, 15 SLOT, 7-8 USERS	5295
8MHZ 286, 1-5" FLOPPY, 1024K STATIC, 40 MB HD, 10 SERIAL, 7-8 USERS	7550
8MHZ 286, 1-5" FLOPPY, 512K STATIC, 40 MB HD, 2 SERIAL & 80287	6495
8MHZ 8086/8087, 1-5" FLOPPY, 512K DRAM, 20 MB HD, 2 SERIAL CDOS or MSDOS	3695
S-100 AT, 512K, COLOR MAGIC PC COMPATABLE COLOR ADAPTER, 1-5", FAST 20MB HD, MSDOS, KEYBOARD, HI-RES COLOR MONITOR, READY TO RUN, YOU COULD BOOT PC DOS 3.1!	5295

A COMPLETE LINE OF TERMINALS, PRINTERS and SOFTWARE FOR YOUR SYSTEM NEEDS

All Prices Subject to Change — Shipping Extra

IBM is a registered trademark of International Business Machines, CP/M, Concurrent DOS are registered trademarks of Digital Research Inc. MSDOS is a trademark of Microsoft.

COMPETITIVE EDGE

P.O. Box 556 — Plymouth, MI 48170 — (313) 451-0665

FEATURING CompuPro® FROM VIASYN™

S-100 CompuPro® CIRCUIT BOARDS

Disk 1A™ Floppy Disk Controller	$371	Disk 3™ Hard Disk Controller	$417
8 MHZ 286 CPU A&T	671	10MHZ 8088, 85/88 CPU A&T	263
Ram 22™ 256K Static Ram A&T	446	Ram 23™ 128K Static Ram A&T	263
Ram 23™ 64K Static Ram A&T	188	System Support 1™ A&T	263
Interfacer 3™ 8 Serial Ports	371	Interfacer 4™ 3 Serial 1 Par	263
SPU Z™ 8MHZ Z80, 256K Multiuser	296	CPU Z™ 8MHZ Z80	193
MDRIVE-H™ 512K Memory Drive	371	MDRIVE-H 2048K, 2MB	821
PC Video Board for IBM® Compat	396	Keytronics 5150 for PC Video	169
Network 100™ Network Board	316	Passive Hub for Network 100	65
80287 Option for 286 CPU	199	HX-12 Hi-Res Color Monitor PC	459
Concurrent DOS™ 8-16™	300	CP/M® 8-16™	188
CP/M® 2.2 8" or 5"	125	Active Terminator	55

Your Choice $263

SYSTEM SUPPORT 1, 85/88-10MHZ, INTERFACER 4, RAM 23-128K

HARD DISK SUB-SYSTEMS

20 MEGABYTE HARD DISK, CABINET, DISK 3, SUB-SYSTEM	$1295
40 MEGABYTE HARD DISK, CABINET, DISK 3, SUB-SYSTEM	1895
20 MEGABYTE HARD DISK, CABINET, DISK 3, 5" FLOPPY	1435
LOMAS 24 MB TAPE BACKUP SUB-SYSTEM	1495
40 MEGABYTE HARD DISK, CABINET, DISK 3, 5" FLOPPY	2095
8" & 5" FLOPPYS, 40 MB HARD DISK SUB-SYSTEM	2395
2-40 MB HARD DISK DRIVES IN CABINET, DISK 3, SUB-SYSTEM	3095

SYSTEMS INTEGRATED BY COMPETITIVE EDGE WITH COMPUPRO® COMPONENTS

8MHZ 286 SYSTEM $4795

8MHZ 286 & 512K STATIC RAM
9 SERIAL PORTS, SS1 & I/O 3
ONE 5" FLOPPY & 20MB HARD DISK
DISK 1A & DISK 3 CONTROLLERS
CONCURRENT DOS 8-16 MULTI-USER
15 SLOT CABINET, 30 AMPS +8
EM8080 SOFTWARE 8 BIT EMULATOR
OPTIONAL 8MHZ Z80H SLAVES
ALL COMPONENTS ASSEMBLED & TESTED TOGETHER, ALL INTERNAL CABLES SUPPLIED

10MHZ 8088-85/88 SYSTEM $3395

85/88 CPU & 128K STATIC
4 SERIAL PORTS & 1 PAR
SYSTEM SUPPORT & I/O 4
ONE 5" FLOPPY & 20 MB HD
DISK 1A & DISK 3 CONTROLLERS
CP/M-8-16 8 & 16 BIT
15 SLOT CABINET
5" FLOPPY WILL READ IBM
SYSTEM UPGRADEABLE TO MULTI-USER & Z80 SLAVES

8MHZ 286 SYSTEM $5395

AS ABOVE BUT 40 MB HD INSTEAD OF 20 MB HD

10MHZ 8088 SYSTEM $3995

SAME AS ABOVE BUT 40 MB HD INSTEAD OF 20

ALL PRICES SUBJECT TO CHANGE — SHIPPING EXTRA

All above circuit board names are either registered trademarks or Trademarks of Viasys Corporation. CompuPro is a registered trademark of Viasyn Corporation, CP/M and concurrent DOS are either trademarks or registered trademarks of Digital Research Inc. IBM is a registered trademark of International Business Machines.

The Data Base Forum

by Nelson T. Dinerstein

This column will be a regular feature in M/SJ. It will discuss problems associated with the application of database managers such as dBase-II/III, Framework, and Rbase 5000. Suggestions, comments and questions are welcomed.

dBase-II Speed Techniques -Part I

THE NATURE OF THE PROBLEM

When dBase-II programs slow down, the cause is almost always the indices. As the file size grows, programs that used to run quickly may now run very slowly. It is not the size of the data file that makes the difference, but the size and number of index files. The techniques described herein will help to overcome this problem and can be incorporated in written as well as to future programs. These speed techniques often reduce execution time on indexed files containing as few as 100 records. The larger the files, the greater the resulting reduction in run time.

Since it is the updating of the index files that makes dBASE II programs slow, we need to avoid the use of indices as much as possible. But, since indices are valuable, we must continue to use them. The trick is to use them in the most efficient manner.

RULE 1 : If an operation is slow, use it as seldom as possible.

There are two major operations that use indices and change files, i.e. adding a new record to a file and modifying an existing record. Let's examine each of these operations.

ADDING A RECORD TO A FILE

Since the number of indices plays an important role in the amount of time needed to APPEND a record to a file, keep the number of indices as small as possible.

RULE 2: Do not create an index on the chance that you might eventually need it. Create an index only when you are sure you will need it. This will reduce the total number of indices on each file.

RULE 3: If an index will be used infrequently, add records to the file without using the index. Then, when the index is required, create the index just before it is used.

The time required to add new records to a file can also be controlled through the use of a combination of interactive and batch techniques. *Interactive* is used here to mean that all of the indices are in use when the record is added to the file. Thus, when a record is added, all of the index files are updated immediately. *Batch* is used here to mean that no index is in use when the record is added to the file. This means that records can be added to the data file very rapidly. When all of the desired records have been added, the files can be re-indexed. Consider examples 1 and 2.

EXAMPLE 1: INTERACTIVE ADDITION OF RECORDS

```
* IN THIS EXAMPLE, INDEXES ARE UPDATED IMMEDIATELY
USE X INDEX A,B,C
DO WHILE T
  STORE " " TO ANSWER
  ERASE
  @ 5,1 SAY "ENTER ANOTHER RECORD (Y/N)" GET ANSWER
  READ
  IF ANSWER <> "Y"
       RETURN
  ENDIF
  * Enter record here, in normal manner
ENDDO
```

EXAMPLE 2: BATCH ADDITION OF RECORDS

```
* IN THIS EXAMPLE, INDEXES ARE UPDATED AFTER
* ADDITIONS HAVE BEEN COMPLETED
USE X INDEX
DO WHILE T
  STORE " " TO ANSWER
  ERASE
  @ 5,1 SAY "ENTER ANOTHER RECORD (Y/N)" GET ANSWER
  READ
  IF ANSWER <> "Y"
       RETURN
  ENDIF
  * Enter record here, in normal manner
ENDDO
INDEX ON <field name> TO A
INDEX ON <field name> TO B
INDEX ON <field name> TO C
```

In some cases, you may wish to verify the record you are about to enter is not already in the file. To do this, first use the appropriate index and the FIND statement to verify that the record is not already in the file, then remove (deactivate) the index and add the record.

EXAMPLE 3: BATCH ADDITION OF RECORDS

```
* IN THIS EXAMPLE, INDEXES ARE UPDATED AFTER
* ADDITIONS HAVE BEEN COMPLETED, BUT CHECK IS MADE
* TO VERIFY THAT RECORD TO BE ENTERED IS NOT
* ALREADY IN FILE.
USE X
DO WHILE T
  STORE " " TO ANSWER
  ERASE
  @ 5,1 SAY "ENTER ANOTHER RECORD (Y/N)" GET ANSWER
  READ
  IF ANSWER <> "Y"
       RETURN
  ENDIF
  STORE "      " TO MEMKEY
  @ 7,1 SAY "ENTER THE RECORD KEY" GET MEMKEY
  READ
  SET INDEX TO A
  FIND &MEMKEY
  IF # <> 0
    @ 9,1 SAY "RECORD ALREADY EXISTS WITH KEY="+MEMKEY
    @ 11,1 SAY "TYPE ANY KEY TO CONTINUE"
    WAIT
    LOOP
  ENDIF
  SET INDEX TO
  * Enter remainder of record here,
  * in normal manner.
ENDDO
INDEX ON <field name> TO A
INDEX ON <field name> TO B
INDEX ON <field name> TO C
```

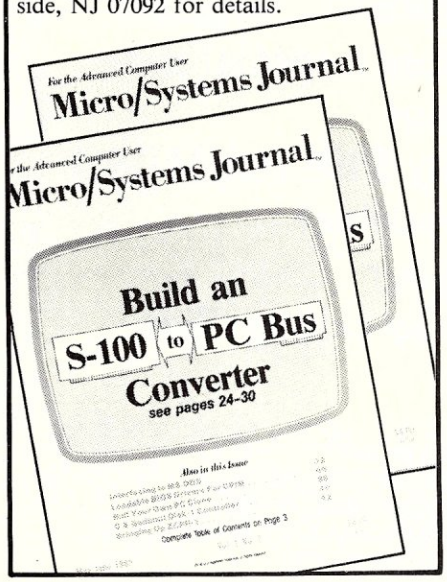
In Example 3, SET INDEX TO A activates the index A causing it to become the primary index. The FIND which follows will then work correctly, assuming that MEMKEY contains an appropriate value for the index A. After the FIND statement has been executed, # will have the value, 0, if there is no record in file X with the value contained in MEMKEY. Otherwise, the value of # will be greater than 0. To add a new record to the file, first remove (deactivate) the index with the SET INDEX TO statement and then add the new record.

I recommend that you have both types of functions (interactive and batch) in any system that you build. This will allow users to decide for themselves which to use. If, for example, a data file and its associated index files have grown so large that it takes 2 minutes to enter a single record, and 10 minutes to reindex all of the files, the interactive function should be used for entering five or fewer records and the batch function for entering five or more records. In this case, five records is the break-even point. As the file sizes grow or shrink, the break-even point will probably change.

RULE 4: Allow the users to choose interactive or batch entry of new records based upon the number of records to be entered.

In the next issue of M/SJ I will conclude my discussion of dBase speed-up techniques. In the meantime, if you have questions or topics that you would like to see discussed in this column drop me a line.

Nelson Dinerstein is an Associate Professor of Computer Science at Utah State University and an active consultant in database applications on micros. He has a Ph.D. in mathematics from the University of Utah. And, he is the author of 8 books on dBase-II/III, Framework, and Rbase 5000 applications.



EARTH COMPUTERS PRESENTS:

THE BEST OF BOTH WORLDS

TURBO**ACCEL**/286

Now you can increase the performance of your IBM PC or PC/XT up to 500% with TURBOACCEL/286, the first 80286 processor card under $1000.

We started with an 8 MHz Intel 80286. Then packed 514K of 16-bit high speed RAM on board. Made it expandable. And left room for an 80287 math chip. But there's more.

The latest VSLI technology makes TURBOACCEL/286 far superior to 8086/186 accelerator boards. Our custom CMOS chip allows the 80286 to emulate the 8088, insuring software transparency and compatiblity.

So you just plug it in, turn it on and go. No special software. No programming. No communications problems. Just pure speed: TURBOACCEL/286.

All for only $995. To order call (714) 964-5784.

TURBO**SLAVE**/PC

Run over 2000 CP/M programs faster than on any other computer with TURBOSLAVE/PC, the first Z-80H coprocessor for your IBM PC.

We use Zilog's 8MHz Z-80H. Include a CP/M emulator. Add on 128K RAM with parity, two RS-232 serial ports and full documentation to bring you the fastest, most versatile and reliable board of its kind.

Then we took it a step further. Under TurboDOS (optional) up to 16 TURBOSLAVE/PC users can be linked to a single PC for true multiuser capabilities. And our open architecture and utilities make TURBOSLAVE/PC ideal for those who need a special purpose processor.

All for only $495 (OEM/ISO discounts available).

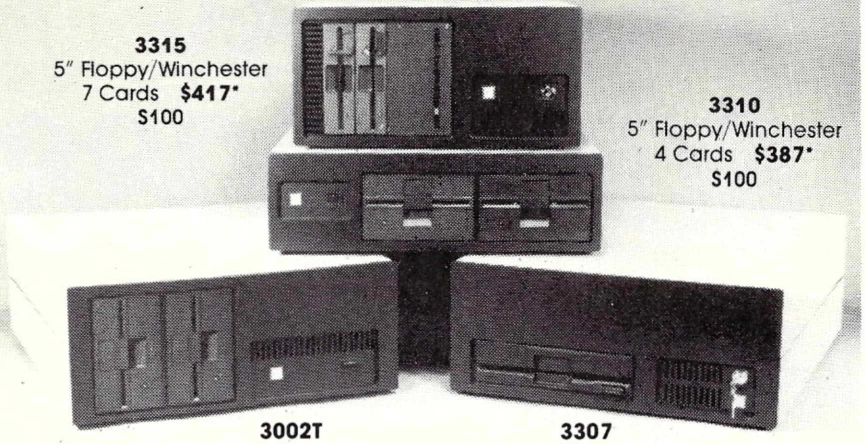
So order your TURBOSLAVE/PC by calling (714) 964-5784.

EARTH COMPUTERS

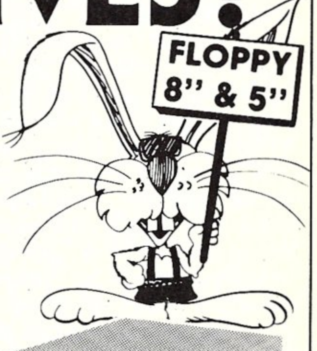
"Building Blocks For The Super Micro"

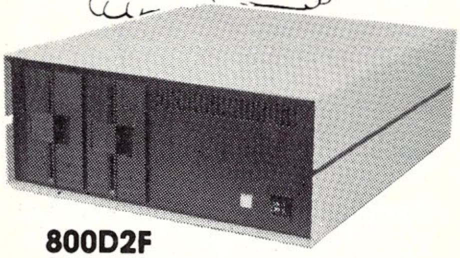
Dept. MP
P.O. Box 8067, Fountain Valley, CA 92728
TELEX: 910 997 6120 EARTH FV
FOR MORE INFORMATION AND QUANTITY DISCOUNTS
CALL (714) 964-5784

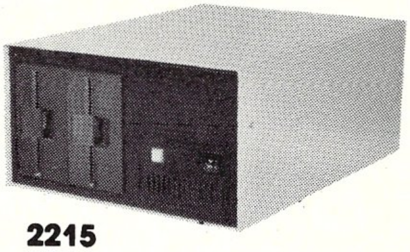
REGISTERED TRADEMARKS Z 80H ZILOG INC. TURBODOS SOFTWARE 2000, INC.

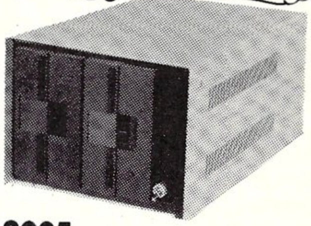


WINCHESTER 8" & 5"


FLOPPY 8" & 5"







32 Page
Free Fakt
Pakt Catalog

TurboCharge Your 8086/8088 Computer

by Stephen Davis

NEC recently announced a new family of processors- the V-Series. It is probable that all members of this family will eventually interest the computing world, especially the 32-bit supermicro V60. But as yet, only two members of the family c urrently exist in silicon, the V20 and V30. They should be of keen interest to owners of 8086- and 8088-based personal computers.

The V20 and V30 carry the more formal part numbers uPD70108 and uPD70116, respectively (apparently, all NEC microprocessors carry the designation, uPD). These chips are pin-compatible replacements for the Intel 8088 and 8086. Far from being mere second sources of the same processor, these CMOS chips offer substantial improvements, both in terms of speed and instruction set. In addition, these processors can run 8080 code if desired! In this two part series, we will examine these improvements, citing benchmarks when appropriate. We will also look into how these improvements can be incorporated into existing 8086 or 8088 systems, including, but not limited to, the IBM PC and compatibles.

We will examine closely the 8-bit bussed V20 and compare it to the 8088. Likewise this comparison can be extended to the 16 bit bussed V30 and 8086. Differences between the two pairs of chips, when they exist, will be pointed out. In this discussion keep in mind that both the 8086 and 8088 processors are available from several second sources besides Intel; however, for our purposes differences between the various 8086/88 versions are minimal.

THE V20 vs THE 8088

The V20 has the same exact pin out as the Intel 8088. NEC claims that there are no external electrical differences between the two chips. However, the chips differ internally in some significant ways.

The 8088 is known as an 8/16-bit processor, having a 16-bit word size, but an 8-bit external bus. The 8088 was designed to allow fir interfacing to 8-bit peripherals originally designed for the 8080 and Z80 microprocessors. (The 8086 is 16-bit both internally and externally.) The 8088 has 20 bits of address allowing for direct addressing of 1 megabyte of memory. Internally the 8088 is a microcoded machine, having a microcode word size of 21 bits. In a microcoded processor machine instructions are decoded and carried out by an internal program, known collectively as the microcode. It generally requires multiple microcode instructions to perform even the simplest machine instruction. Since each microcode instruction takes one clock cycle to perform, individual instructions may take many clock cycles to execute. As an extreme example, the MULtiply instruction takes more than 100 clock cycles to perform in the Intel chips.

To route data internally to the chip the 8088 has one 16-bit wide bus. Often extra clock cycles must be used in the 8088 to avoid collision of data on this internal bus. For example, in a compare (CMP) instruction, two 16-bit values must be passed to the ALU (arithmetic/logical unit) for comparison. These two values must be passed in 2 subsequent operations since only one value may be on the bus at one time, even if both are available simultaneously.

The V20 maintains two internal data busses. The second data bus is used to advantage in logical comparison and arithmetic instructions to fetch the second argument while the first argument is being fetched on the primary data bus. This results in a savings of one clock cycle, a 33% improvement. In addition, subtraction and comparison instruction times are reduced.

The 8088 allows several addressing modes to be used with its instructions. This makes the chip easier to program, but resolving some of its addressing modes into an effective address can be complicated as some modes require up to four additions to come up with a physical address. The 8088 uses microcoded subroutines to resolve these addresses. Although invisible to the user, these microroutines add from 5 to 12 clock cycles (depending on the addressing mode used) to the number of cycles needed to execute a single assembler instruction.

The V20 does not use microcoded subroutines to calculate effective addresses. Rather the chip contains a special circuit called the Effective Address Generator (EAG) which can calculate any effective address in a constant 2 clock cycles. (Note: The addition of an EAG circuit is one of the important improvements in the later members of the 8086 family.)

In order to improve chip performance, the 8088 has a 4-byte prefetch queue (6 byte in the 8086). The Bus Interface Unit (BIU) section of the chip uses clock cycles during which the bus would otherwise be idle to prefetch the next instruction to be executed into a faster memory within the chip (this technique is known as pipelining and is practiced on most modern microprocessors). The BIU in the 8088 allows the Instruction Pointer

(IP) to "get ahead" of the Execution Unit, the part of the microprocessor which actually carries out the individual instructions. Thus, although we programmers think of the IP as pointing to the currently executing instruction, in fact the IP points to the next instruction to be prefetched, which may be several bytes ahead.

This works fine for instructions which don't use the instruction pointer, but jumps and loops are coded relative to the IP. For example, a jump instruction at address 1F0 codes a 'JMP 200' as 'jump forward 10 bytes'..Thus, whenever the execution unit encounters such an instruction, it must first back the IP up to the currently executing instruction (the jump, etc.) before it can calculate the target address.

The effect of this is easily seen in the conditional jump and loop instructions. The conditional jump instruction takes a modest 4 cycles if the jump is not taken. Here the instruction pointer is not backed up since no jump is performed. The same instruction takes a relatively long 16 cycles to perform if the jump is taken and the program counter must be recalculated.

(Notice that every instruction which changes the instruction pointer causes the contents of the prefetch queue to be flushed. This forces the next instruction to be fetched from external memory, slowing down its execution by several clock cycles. This effect does account for a part of the time difference in the conditional jump timings.)

In far CALLs and INTerrupts the target address is not coded relative to the IP, but instead absolutely. A far CALL to 100:100 would encode the address 100:100 literally, independent of its own address. In these cases, however, the IP must still be backed up so that it may be pushed onto the stack as the return address. Therefore, the effect in these cases is the same.

The V20 has a Prefetch Pointer separate from the instruction pointer for maintenance of the prefetch queue. The IP actually does point immediately after the instruction being executed just like we had always thought, while the prefetch pointer scouts out ahead for the next instructions. Thus, when a IP relative instruction is executed there is no need to back the IP up and that time is saved. For example, the V20 requires 50 clock cycles to perform an INT instruction, compared to the 8088's 63.

A further internal improvement of the V20 is the addition of two internal registers, allowing more efficient algorithms in the microcode for many instructions. The V20's string instructions are roughly twice as fast, while its multiply and divide are up to three times as fast. Of all of the time improvements in this chip, the improvements in the multiply and divide have the potential for being the most significant, depending on the application. Since these are the longest instructions in the 8088's instruction set, in terms of time to execute, a 50% improvement may amount to many clock cycles. (It is not always obvious that your program is performing multiplies -- many high level language resolve matrix subscripts with MUL instructions.)

In addition to these timing improvements in the 8088 instructions, the V20 adds three sets of extra instructions to the 8088 instruction set. The first of these extensions are the 80186 extensions, which are also the only extensions available to the 80286 when left in real mode (the 80186 and 80286 are the next members of the Intel iAPX 86 family).

These include a PUSH/POP all (pushes/pops all registers in one instruction), an expanded class of rotate and shift instructions, a string OUT and IN (used with the repeat prefix REP) for faster CPU controlled I/O and a set of stack frame instructions useful primarily for higher level languages (see Table 1).

The PUSH/POP all instructions are quite handy for interrupt routines which must save off any registers they intend to use upon entry into the interrupt code and restore them upon exit. In the 8088 rotate and shift instructions may only specify the CL register or a 1 as the number of bit positions to rotate; no other constant is allowed. The extended V20 shifts allow any constant (or the CL register). The string IN and OUTs allow driving I/O devices at the bandwidth of the bus under CPU control (as opposed to DMA). This is a respectable 2MB per second on an 8 MHz V30.

The 8088 has no instructions to manipulate any field smaller than a byte. The V20 corrects this limitation in the second set of extensions, which NEC calls the unique instructions. These include nibble (4-bit) and bit instructions (test, set, clear and complement) as well as a pair of new repeat prefixes (repeat while and until carry) and a second floating point prefix (see Table 2). This second prefix allows NEC to offer an extended set of floating point instructions with their own numerical coprocessor, due out the beginning of next year.

The final members of the instruction set are truly unique, not being present on any other common microprocessor. They are the "enter emulation" and "exit emulation" instructions. When, in what NEC calls emulation mode, a different set of microcode is executed by the V20/30. This new set of microcode allows the NEC chips to execute 8080 code!

In emulation mode the V20 continues to look externally like an 8088, but it no longer understands 8088 instructions. Internally it now becomes an 8080 8-bit microprocessor. With the proper interface code, any computer using this chip can run either 8086 code (MS-DOS, CPM86, etc.) or 8080 code (primarily CPM80). We will leave the detailed discussion of how this is done (along with examples of a BDOS to do this) to the second part of our two part series.

These internal improvements to the

TABLE 1
EXTENDED INSTRUCTIONS

(USING INTEL 80186/286 MNEMONICS)

INSTRUCTION	FUNCTION
PUSH immed16	push an immediate value onto the stack
PUSHA	push all eight general registers
POPA	pop " " " "
IMUL reg16,mem8,immed8 ,mem16,immed16	multiply memory with immediate value and place result into reg16
SHL reg8 or 16,immed	similar to normal shift left except that any value
SHR	is allowed (not limited to CL or 1)
SAL	
SAR	
ROL	
ROR	
RCL	
RCR	
INSB,INSW	string input used with REP prefix (byte or word)
OUTSB,OUTSW	string output used with REP prefix (")
ENTER immed16,immed8	sets up a stack frame for Pascal or ADA
LEAVE	tears down a stack frame
BOUND reg16,mem block	check an index against a memory block for range

V20 have forced the internal microcode size to swell from the 8088's 21 bits to some 29 bits and the transistor count, a measure of the complexity of a chip, to almost double.

ADDING THE V20/30 TO YOUR 8088/86 COMPUTER

Were it not for the external similarity of the V20 with the 8088 (and the V30 with the 8086), the discussion so far might be merely interesting but of no immediate use. However, since the NEC chips are pin compatible with the corresponding Intel part, anyone who owns a computer containing either of the Intel chips can have access to these improvements by merely replacing a microprocessor chip.

The V20 and V30 should work with Intel's 8087 Numerical Processor (NP). In this mode of course the new floating point instructions are not available, but the standard Intel set works the same. I did not have sufficient access to the Intel numerical processor to test this completely.

Note also that the NEC parts will cause problems in the TI Professional Computer and the Victor 9000. The TI uses some rather critical timing loops in its self test BIOS code. Due to the higher speed of the V30, the TI erroneously times out its hardware and will not complete boot up. The Victor will not format a diskette properly, again due to the processor's higher speed. The IBM PC and XT, Compaq Portable and Deskpro, Hyperion, Eagle PC, and Columbia are among those computers known to work properly with the NEC chip. Computers not on either list will probably work without problems.

When selecting the proper NEC chip for your computer you should first note whether your computer uses the 8088 or the 8086. Your documentation should make this clear in the technical specifications. You should next note the clock speed of your system. It will probably be either 4.77 MHz or around 8 MHz (the 8088-2 is the 8 MHz model). The NEC processors come in 5, 8, and 10 MHz versions. If your machine runs at 4.77 MHz, then the 5 MHz model is the one for you; if your system clock is between 5 and 8 MHz or if you have a switchable clock who's higher speed is in this range, then the 8 MHz model is the correct processor. If your processor runs faster then 8 MHz, you are safest ordering the 10MHz part.

NEC only guarantees a chip's rated speed if your computer's clock maintains a 50% duty cycle, the optimum for CMOS processors. If your computer runs with a 33% duty cycle clock, the optimum for NMOS chips, the V20/30 will probably work at its specified rating, but you might want to order the next faster chip just to be safe.

TABLE 2
NEC'S UNIQUE INSTRUCTIONS

INSTRUCTION	FUNCTION
INS reg8,reg8 reg8,immed4	transfers number of bits in second operand from least significant bits of AL into DS:[DI] with bit address in lower 4 bits of first operand; both DI and first operand are updated to point to next bit field
EXT " , "	inverse operation to that above
ADD4S	adds BCD number pointed at by SI to the BCD number pointed at by DI; length of two BCD numbers is in CL (max no. of digits is 254); sets flags
SUB4S	subtracts similar to add above
CMP4S	set flags as subtract above without saving results
ROL4 reg8	rotates left by 4 bits 8 bit register or memory location specified with lower 4 bits of AL register
ROR4 "	complimentary operation
TEST1 CL or immed3,reg8 CL or immed4,reg16 CL or immed3,mem8 CL or immed4,mem16	set Z flag if bit specified by first argument in byte or word pointed at by second argument is clear.
NOT1 " , "	complement bit specified as above
CLR1 " , "	clear bit specified as above
SET1 " , "	set bit specified as above
REPC	repeat while carry following string instruction
REPNC	repeat until carry " " "
FPO2	prompts NEC numerical processor for unique NEC floating point op.s
BRKEM immed8	execute 8080 code pointed at by interrupt vector immed8 (not included in EXTND.INC - will be covered in part 2)

With this information in hand, you are now ready to order your V20 or V30. As of this writing, few computer retail outlets have these parts in stock. I did find two retailers handling the parts: Lolir Electronics, 13933 N. Central Expressway, Suite 212, Dallas, Tx 75243 (214-234-8032) and GFI Electronics, 1800 Avalon St., Olathe, KS 66062 (913-829-0157). The price for the 5 MHz V20 was $24.95. Wherever you purchase yours, cost should be in the $25 to $35 range, depending on speed and whether ordering the V20 or V30.

To install the new chip, you will of course need to locate and remove your existing processor. Locating the chip is fairly straightforward: look for a 40 pin dual in-line package (DIP) with the designation 8088 or 8086 on it. In the case of IBM PC's and their clones, the 8088 is at the far rear of the board, immediatly in front of the keyboard connector. Before removing the old chip, note in which direction the half circle groove on top of the chip points. Removing the chip is best done with a chip puller tool, but can be done by sliding a screwdriver up under the chip and carefully prying the chip up, being especially careful not to bend the chip legs.

To insert the new chip, set it down on top of the socket with each pin sitting in the slot into which it will be going. If one or more pins do not match the hole(s), bend them until all do. Be sure the chip is oriented with its half-circle groove facing the same as the old chip. Now carefully, but firmly, press the chip into the socket (be careful to support the other side of the board, if possible, to keep from bending it any more than is necessary).

USING THE NEC V20/30

Once you have installed your V20 or V30 in your system, you should notice some increase in speed. This increase will depend on what type of work you are doing and upon the configuration of your hardware. Programs which perform a lot of floppy disk I/O will not improve significantly. Programs which are not I/O bound and perform lots of calculations (multiplies and divides) will improve dramatically.

TABLE 3

		8088 (4.77MHz)			8086 (7.14 MHz)		
BENCHMARK	REPS	INTEL	NEC	%DIFF	INTEL	NEC	%DIFF
Sieve of Erastosthenes	10	3.9	3.7	5%	1.9	1.7	12%
Bench A +	50,000	2.0	1.8	11%	1.0	0.8	25%
Bench B	5,000	7.2	6.8	6%	3.4	3.0	13%
Bench E	20,000	10.6	7.9	34%	6.3	4.6	37%
Bench F	20,000	10.9	10.3	6%	5.1	4.5	13%
			(6.3)*	(73%)*		(3.0)*	(70%)*
Bench H	10,000	7.1	6.8	4%	3.5	3.1	13%
Bench I	100	11.1	9.8	13%	6.0	4.6	30%
Bench K	10,000	29.9	26.0	15%	15.8	12.8	23%
			(18.0)*	(66%)*		(8.1)*	(95%)*
Filter ++	20,000	22.6	13.5	67%	13.0	6.7	94%

8088 times measured on an unmodified IBM PC; 8086 times on a Compaq DeskPro in "fast" mode

* = time when algorithm recoded to take advantage of NEC unique instructions

+ = benchmarks A thru K were selected in EDN Sept. '81 from a 1979 Carnegie Mellon study for comparing 8086, 68000 and 32016 microprocessors; these benchmarks were selected as being most applicable to microprocessors.
Individual benchmarks measured:
Bench A - interrupt response time
Bench B - interrupt response time
Bench E - string instruction performance
Bench F - bit manipulation
Bench H - linked list manipulation
Bench I - quick sort
Bench K - bit manipulation

++= filter benchmark selected from IEEE Micro 2/81 as being particularly advantageous for their microprocessors

To test the increase in speed I ran several benchmarks on an IBM PC (5MHz), and on a Compaq Deskpro (8MHz 8086), first with their standard Intel processors and then again with the NEC chips. The results of these timings appear in table 3. It should be noted that I had no problem in replacing the processors on either computer and neither system showed any signs of rejecting the foreign chip.

When interpreting the ''percentage improvement'' column of Table 3, keep in mind that this is the improvement of the NEC chip over the Intel chip. Thus a figure of 100% means that the NEC part performed the benchmark in one half the time, ie. twice as fast.

The first benchmark, the Sieve of Eratosthenes, is something of a standard, appearing in almost every discussion of compiler or processor speed. The sieve is small and easily reproduced on different processors, but does not test very many aspects of a chip. Nevertheless, the assembler source code for the sieve appears as program listing 1.

A further set of benchmarks was reproduced from an article appearing in EDN magazine, which compared the performances of several popular microprocessors. This bank of benchmarks tests various aspects of processor performance. In every case, the code for each benchmark was taken directly from the article. In some cases, however, the benchmark could have been coded more efficiently using the NEC chip's instruction extensions. In these cases, the benchmarks were recoded using the more efficient instructions and rerun. These times appear after the times using only the ''standard'' 8086 instructions.

A final digital filter benchmark was selected from the IEEE's Micro magazine. This benchmark contains a high percentage of multiply and divide instructions to highlight the improvements in the V20 and V30. As expected the relative increase in performance was largest for this benchmark.

The improvement on the 8088 equipped PC was not large, in some cases hardly amounting to more than 5%. The improvement in the 8 MHz 8086 was much more substantial, averaging more in the 25 to 35%. This is primarily because in programs with long strings of simple to execute instructions, such as those in the sieve, the 8088 runs at roughly 90% bus utilization. It is very difficult for the V20 to get data in and out through this 8-bit bottleneck very much faster than the 8088 chip and thus the meager throughput improvement. The relatively greater improvement in the V30 is due to the wider bus being much less of a performance limiting factor.

However, for those who tend to run programs which perform large amounts of math, such as spreadsheets without 8087 support, the improvements of even the V20 are substantial. This is equally true for those who code their own programs and are willing to make use of the V20/30 extensions.

To easily quantify your throughput increase, a benchmark program called CPU was adapted from a public domain program of the same name (see Program 2). Run this program before installing the NEC chip to get a reference for your system. The program compares the speed of your system against and IBM PC. You will get a time to execute (10 seconds on the PC) and an effective clock speed - that is, your PC runs as fast as an IBM PC with this speed clock. Rerunning the benchmark with the NEC part should result in shorter times and a faster effective clock speed. Notice that your clock speed has not changed, only the effective clock speed -- the difference is due to the NEC using fewer cycles to perform the same instructions.

Picking the instruction mix to use in the timing loop of CPU is a difficult thing. The increase in throughput using NEC CPUs can vary anywhere from near 0%, if only simple instructions are used, to more than 100% if multiplies and divides are present. I have tried to pick a mix which accurately reflects the types of instructions I code. You may want to change the instruction mix to fit the types of programs you run.

To do this, with the Intel part still in place, change the instruction mix in the timing loop to whatever you prefer. Adjust the number of iterations of the timing loop until you get the same value for clock speed that you got when you ran my mix (4.77 for the IBM PC). When you now install the NEC chip rerun the program to get an accurate measure of how much improvement you should expect. Including lots of string instructions or multiplies in your instruction mix will favor the NEC; lots of simple instruction will minimize the improvement of the NEC.

USING THE NEC INSTRUCTIONS

Programs which you buy or download from bulletin boards will continue to use only 8086/88 instructions and so, unless you write your own assembler programs, you cannot expect to see any improvement from the extended instruction set. However, if you do write some of your own programs, you may want to use some the improved instructions.

If you own a later copy of Microsoft's MASM assembler, simply specifying the .186 or the .286C pseudo-op at the top of your program will allow you access to the extended instructions (but not the unique instructions). The Microsoft assembler manual, as well as Intel publications, describe the proper format for these instructions, so I will not describe them in detail here. A list of these instructions appears in Table 1. Programs using the extended instructions will also execute properly on 80186 and 80286 equipped computers.

The unique instructions defined by NEC fall into several related groups. The first group is comprised of the variable length bit field operations, INS (insert) and EXT (extract). The insert instruction transfers the least significant bits from the AX register to the address pointed at by DS:DI. The starting bit position in the memory location is specified by the first argument, which must be a register, and the number of bits transfered by the second argument, which may be either a register or an immediate value. The bit fields may cross from one byte or word to the next. The first argument and the DI register are updated after each transfer. The extract instruction performs the same function in the opposite direction. This instruction pair allows for efficient packing and unpacking from a structure of fields which are not an even 8 or 16-bits.

Also provided are the single bit instructions, TEST1 (test), NOT1 (complement), CLR1 (clear bit), and SET1 (set bit). The test bit instruction sets the Z flag if the specified bit is a 0 and clears the Z flag if not. All four of the above instructions specify a bit position in a byte or word field. The bit position is in the first argument and is either an immediate value or else the CL register. The word or byte in the second argument can be any valid reference: register, variable, or indexed location.

Three operations are included in the packed BCD group: ADD4S, SUB4S and CMP4S. Each of these operate on the BCD number pointed at by the DI register with the BCD number pointed at by the SI register. The number of BCD digits is specified by the CL register. These operations set the overflow, carry and zero flags, but if the number of digits is odd, the zero flag is not set correctly. An extra set of BCD rotate instructions. ROL4 and ROR4, rotates the 8-bit register or memory location specified by the single argument through the least significant 4 bits of the AX register. This instruction group allows efficient BCD algorithms to be implemented.

Two new prefixes to string instructions are added with the unique instructions: REPC (repeat as long as carry set) and REPNC (repeat until carry set). An extra floating point instruction is included for eventual use with the NEC numerical processor to be introduced later.

The NEC's unique instructions are summarized in Table 2. If you would like to use these unique instructions and are using the popular Microsoft MacroAssembler, including the macro definition file EXTND.INC (Program 3) at the beginning of your program will define the unique instructions for you. (This is the same macroassembler sold and endorsed by the MS-DOS personal computers' manufacturers, including IBM.) Assembling a program which uses unique instructions can be done on either an Intel or NEC microprocessor, but of course, the resulting program will only run properly on the NEC chip.

The reader will probably be struck by the complexity of these macro definitions; this is primarily due to the severely limited macro capabilities of the Microsoft assembler. An example subroutine has been provided in program 4 showing usage of the unique instruction set.

Notice that the assembler limitations resulted in two concessions. As with all memory references, NEC allows segment overrides to be added to memory references in unique instructions. The macros provided do not allow adding a segment override directly to a memory reference. Generally a segment override can be avoided, but you can specify a segment with the override macros provided. Unlike normal segment overrides, these macros precede the instruction. Thus, instead of coding 'TEST1 CL, CS:memory_loc', you would need to code 'CS_TEST1 CL, memory_loc'. If you prefer, you can use these segment

NEC VERSUS INTEL

The similarities between NEC's uPD70108/116 and Intel's 8086/88 microprocessors did not go unnoticed by either company. Before introducing the 108 and 116, NEC asked the U.S. District Court in December of 1984 to rule that the chips' microcode did not infringe upon Intel's copyrighted 8086/88 microcode (past rulings had made it clear that the underlying hardware was neither copyrightable nor patentable). Intel responded by suing NEC in February of 85 for copyright violation. Intel's suit must have been anticipated by NEC's engineers for some time, since NEC had already gone to some pains to build a defense for their chip.

First, NEC introduced these chips as members of the newly announced V-Series of microprocessors and refer to them as the V20 and V30. Other members of the V-Series do not directly resemble any existing microprocessors. Secondly, in its instruction set descriptions, NEC is careful to call its 80186 compatible instructions the "extended instructions", while designating the bit instructions (not present in any of Intel's chips) as the "unique instructions". This NEC does without ever mentioning the 80186.

Perhaps even more damning, is NEC's access to internal documentation on the 8086. As a contracted second source for the Intel processors, NEC would have had detailed drawings and perhaps even microcode listings at its disposal when designing the V20 and V30.

NEC does have some real defenses built right into the chip. NEC added a set of instructions to their processors not present on any Intel processor, including an emulation mode not present on any other common microprocessor. NEC improved the hardware by introducing a dual rail internal bus, hardware calculation of effective address (which the Intel 80186 has but not the 8086) and an extra prefetch pointer to reduce the penalty of JUMPs and CALLs. These improvements result in increased CPU throughput by reducing the number of clock cycles required to perform most instructions. Further, the internal word size of the V20 and V30 processors was increased to 29 bits vs. the 8086 and 8088's 21 bits.

NEC may or may not win its case in district court, but no matter what the decision, the case will undoubtedly end up in the Supreme Court. By the time a decision is eventually handed down, it might well be a moot as the 8086/88 processor is already in the autumn years of its existence. In late April, Intel raised the stakes by asking for a summary judgement on the issue of whether microcode is copyrightable (a recent decision in the case of Apple vs Franklin established that software is copyrightable, but this decision probably does not apply to microcode). With that move the case took on an importance far more important than the original question, one which will have long reaching effects on the entire semiconductor industry.

override macros for all instructions, not only the unique opcodes.

Secondly, I was forced to code the bit instructions with the arguments in reverse order from that specified by NEC. Thus, where NEC specifies TEST1 AX,CL, I was forced to use TEST1 CL,AX.

Since incorrectly coding a unique instruction might lead to completely misleading error messages in the macro expansion, the macros check all arguments to the instructions for proper type. If a type is wrong, the offending argument is displayed on the listing along with a comment explaining the problem and an *Improper syntax* error message.

A special macro has also been included for Intel compatibility. This instruction macro, called J_NOT_NEC, checks the processor upon which it is being run to determine whether it is an Intel or an NEC part. If it is the NEC part, the jump is not taken; if it is an Intel processor, the jump is performed. This allows the programmer to code sections both with and without the NEC extensions. Such programs use the faster, more convenient NEC instructions when possible, but continue to work, albeit more slowly, when executed on Intel processors.

The emulation instructions are not defined in EXTND.INC as they cannot be executed like normal instructions without going into 8080 mode. This entire subject will be dealt with in Part 2 of this series.

A second macro definition file has also been provided for Intel's assembler as Program Listing 4. This assembler is superior to Microsoft's but, due to its exorbitant price plus Intel's marketing policy, is almost unknown. These macro definitions are much more straight forward due to that assembler's extensive macro capabilities. These macros perform more or less identically to those above except for the absence of the peculiarities noted in EXTND.INC.

When disassembling a program with MS-DOS's DEBUG, the debugger will not disassemble the extended and unique instructions correctly. Attempting to disassemble them with the U command will result in some garbaged instructions at, and immediately after, the NEC instruction. This does not mean that the instruction is encoded incorrectly, so this is a big nuisance when debugging. (The debugger provided with MS-DOS 3.0 and later does handle the extended instructions properly.)

Several users groups are beginning to spring up around the country to offer support for the V20 and V30. This effort is being encouraged and supported by NEC. Already a version of DEBUG which understands and supports the NEC unique (and extended instructions) is available in the public domain. If you would like a copy of this debugger, send your name and address plus $10 to cover postage, disks, mailers, etc. to: Stephen R. Davis, Route 5 Box 107K, Greenville, Tx 75401. This disk also includes source and executable code for all benchmarks mentioned in the article, source files for all listings appearing in both Parts1 and 2, sample 8080 programs which run on the V20/30, etc. I will also include the numbers and addresses of any support groups.

CONCLUSION

The NEC V20/30 microprocessor provides a means of turbo charging your conventional 8086/8088 personal computer both in terms of speed and in terms of instruction set. Although the improvement may not come close to the improvements claimed for some of the add-on turbo cards, neither is the rather modest $25 cost close to their price.

PROGRAM 1 LISTING

```
;       Sieve of Erasthosthenes
;
;       Stephen R. Davis
;
NTIMES  EQU     8190                    ;STANDARD SIZE
NLOOPS  EQU     10                      ;TEN ITERATIONS STANDARD

CSEG    SEGMENT PUBLIC PARA
        ASSUME CS:CSEG,DS:CSEG,ES:CSEG

        ORG     100H
START:
        CLI                             ;DON'T CONFUSE WITH INTERRUPTS
        MOV     LOOP_CNT,NLOOPS ;INIT LOOP COUNTER
OUTER_LOOP:
        XOR     AX,AX                   ;FIRST ZERO OUT FLAG ARRAY
        MOV     DI,OFFSET ARRAY
        MOV     CX,NTIMES/2
        CLD
        REP     STOSW
                                        ;NOW COUNT THE PRIMES
        XOR     DX,DX                   ;KEEP THE COUNT IN DX
        XOR     BX,BX                   ;START AT SUBSCRIPT 0
        MOV     CX,NTIMES               ;GET THE LOOP COUNT
        MOV     DI,CX
INNER_LOOP:
        CMP     BYTE PTR ARRAY[BX],0
        JNZ     SKIP_AROUND
        MOV     AX,BX
        ADD     AX,BX
        ADD     AX,3
        MOV     PRIME,AX
        MOV     SI,BX                   ;NOW MARK OUT MULTIPLES
        ADD     SI,AX
MARK_OUT:
        MOV     BYTE PTR ARRAY[SI],0FFH ;MARK THIS MULTIPLE OUT
        ADD     SI,AX
        CMP     SI,DI                   ;IS THAT IT?
        JNA     MARK_OUT
        INC     DX                      ;INCREMENT PRIME NUMBER COUNT
SKIP_AROUND:
        INC     BX
        LOOP    INNER_LOOP
        MOV     COUNT,DX                ;NOW COUNT IS IN DX REGISTER
        DEC     LOOP_CNT                ;DO IT 10 TIMES
        JNZ     OUTER_LOOP
        INT     20H                     ;OK — STOP NOW
;DEFINE OUR VARIABLES DOWN HERE
LOOP_CNT        DW 0
PRIME           DW 0
COUNT           DW 0
ARRAY           DB NTIMES DUP (0)
CSEG    ENDS
        END     START
```

PROGRAM 2 LISTING

```
TITLE   CPU

;CPU WAS TAKEN FROM PUBLIC DOMAIN AND MODIFIED TO PROVIDE
;A REPRESENTATIVE INSTRUCTION MIX FOR QUANTIFYING SPEED
;OF ONE'S 8086/8088/80186/80286 COMPUTER IN TERMS OF STANDARD
;IBM PC.  RESULT IS IN EFFECTIVE CLOCK RATE (I.E. YOUR COMPUTER
;RUNS AS FAST AS AN IBM PC WITH FOLLOWING CLOCK RATE)

LF      EQU     0AH
CR      EQU     0DH
HUNDRED EQU     64H
THOUSAND EQU    3E8H
TEN     EQU     0AH

CODE    SEGMENT
        ASSUME CS:CODE,DS:CODE,SS:CODE,ES:NOTHING
        ORG     100H

;FIRST OUTPUT A 'WAIT A MINUTE' MESSAGE & THEN GET TIME

START:
        MOV     AH,9
        MOV     DX,OFFSET INIT_MSG
        INT     21H
        MOV     AH,2CH
        INT     21H
        MOV     TIME1,DX
        STI                             ;BE SURE CLOCK IS RUNNING
;HERE IS THE BIG TIMING LOOP
        XOR     AX,AX
        MOV     DX,TEN
OUTERLOOP:
        MOV     CX,03B84H               ;THIS NUMBER MUST BE AD-
                                        ;JUSTED FOR DIFFERENT MIXES
TIMERLOOP:
;MY MIX OF REPRESENTATIVE, USEFUL INSTRUCTIONS —
;USER MAY PROVIDE HIS OWN AND REASSEMBLE, BEING CAREFUL
;TO ADJUST CONSTANT ABOVE FOR TOTAL NUMBER
;OF CLOCK CYLES IN LOOP
        MOV     AX,BX                   ;REGISTER TO REGISTER MOVE
        MOV     AX,BX                   ;(FAR AND AWAY MOST COMMON)
        MOV     AX,BX
        MOV     AX,BX
        MOV     AX,BX
        MOV     AX,BX
        MOV     AX,10[BX]               ;COMPLEX ADDRESSING
        MOV     AX,10[BX]               ;(PRETTY COMMON)
        MOV     AX,10[BX]
        MOV     AX,10[BX]
        ADD     AX,AX                   ;REG - REG ARITHMETIC
        ADD     AX,10                   ;REG - IMMED ARITHMETIC
        PUSH    CX                      ;PUSH (POP FOLLOWS) ARE COMMON
        MOV     CL,5
        SHL     AX,CL
        POP     CX
        CALL    DUMMY_ROUTINE ;THESE CALLS/JUMPS SHOULD ALSO
                              ; FAVOR NEC
        JMP     CONTINUE
DUMMY_ROUTINE PROC NEAR
        RET
DUMMY ROUTINE ENDP
CONTINUE:
        LOOP    TIMERLOOP
        DEC     DX
        JNZ     OUTERLOOP
;NOW GET TIME AGAIN (AFTER LOOP)
        MOV     AH,2CH
        INT     21H
;OK — CONVERT THIS INTO CLOCK SPEED RELATIVE TO 8088
;AND OUTPUT IN ASCII FORM
        MOV     AX,HUNDRED
        IMUL    DH
        MOV     DH,0
        ADD     AX,DX
        MOV     BX,AX
        MOV     DX,TIME1
        MOV     AX,HUNDRED
        IMUL    DH
        MOV     DH,0
        ADD     AX,DX
        SUB     BX,AX
        JNB     SKIP
        ADD     BX,1770H
SKIP:
        MOV     AX,BX
        MOV     TIME2,BX
        MOV     BP,OFFSET VALUE1
        MOV     BX,THOUSAND
        CWD
        IDIV    BX
        CALL    CONVERT2NUM             ;CONVERT FIRST VALUE & STORE
        MOV     BX,HUNDRED              ;HUNDREDS DIGIT
        MOV     AX,DX
        CWD
        IDIV    BX
        CALL    CONVERT2NUM
        INC     BP
        MOV     BX,TEN                          ;TENS DIGIT
        MOV     AX,DX
        CWD
        IDIV    BX
        CALL    CONVERT2NUM
        MOV     AX,DX                           ;ONES DIGIT
        CALL    CONVERT2NUM
        MOV     AH,9
        MOV     DX,OFFSET RES_MSG
        INT     21H

        MOV     AX,12A2H/TEN            ;CALCULATE EFFECTIVE CLOCK RATE
        CWD
        MOV     BX,TIME2
        DIV     BX
        MOV     BP,OFFSET VALUE2
        CALL    CONVERT2NUM                     ;TENS DIGIT
        MOV     AX,DX
        MOV     CX,TEN
        MUL     CX
        DIV     BX
        CALL    CONVERT2NUM                     ;ONES DIGIT
        INC     BP
        MOV     AX,DX
        MUL     CX
```

ATTENTION: S100 BUS USERS!

LOMAS DATA PRODUCTS PRESENTS:

IBM-PC COMPATIBILITY FOR ONLY

$1749*

**User supplied keyboard and monitor required.*

LDP is offering a compatibility upgrade package to convert your current S100 BUS system quickly and easily to IBM-PC compatibility. This upgrade includes the high performance THUNDER 186 and COLOR MAGIC, IBM-PC compatible color graphics board. You get 10 Mhz 8086 performance, 256K bytes of no wait state RAM, two serial ports and a parallel printer port. No other S100 bus boards are required. Included with this set is the advanced CONCURRENT DOS operating system bringing PC-DOS compatibility and multitasking to the system. This board out performs the IBM-PC by a three to one margin. Gain IBM-PC compatibility without sacrificing S100 bus compatibility.

THUNDER186/COLOR MAGIC SET $1749.00

IBM COMPATIBLE S100 BUS COLOR GRAPHICS

LOMAS DATA PRODUCTS presents COLOR MAGIC, the most complete compatibility solution for S100 bus computer products. COLOR MAGIC includes three major hardware subfunctions which allow it to emulate the IBM–PC: An entirely compatible video function, an IBM–PC keyboard interface and an IBM–PC compatible timer interface including IBM–PC sound compatibility. HOW COMPATIBLE IS IT? Currently we are running MICROSOFT's FLIGHT SIMULATOR recognized as one of the severest tests of compatibility. We can also directly boot PC–DOS for the IBM–PC with no alterations. Other programs which have been tested and function without problems are: LOTUS 1-2-3, DBASE III, WORDSTAR and VOLKSWRITER.

COLOR MAGIC (16K byte version) .. $595.00

■**THUNDER 186** single board computer provides a high performance 16 bit computer all on one board. It is an ideal companion to the COLOR MAGIC to provide a low cost, high performance IBM–PC compatible system. The 8Mhz 80186 offers 10Mhz 8086 performance. THUNDER 186 provides all the components necessary to form a complete system including: 256K bytes of no wait-state RAM, 2 serial ports, a parallel printer port, high performance floppy disk controller controls both $5^{1}/_{4}$″ and 8″ drives simultaneously, full IEEE 696 (S100) bus for system expansion. The COLOR MAGIC and THUNDER 186 combine to provide performance you won't find in other compatibles. The price includes the powerful Concurrent DOS operating system.

THUNDER 186 .... $1195.00

■**NV–DISK** is a solid state memory with software to emulate a disk drive under MS–DOS, Concurrent DOS, and CP/M-86. NV–DISK is entirely COMPUPRO software compatible allowing COMPUPRO users to take advantage of the lower cost and battery protection support offered by NV–DISK. It offers the advantage of high speed access and no moving parts. It can be battery protected to allow data to stay even while powered down. The board is available in either 512K or 2 Megabyte configurations and multiple boards may be used to create disk drives with up to 16 Megabytes of storage.

NV–DISK 512K, **$595.00** 2 MBYTE, **$1695.00**

NEW LOWER PRICING

■**MEGARAM** is a high density, high performance dynamic RAM board with up to two Megabytes of storage. Megaram offers no wait state performance in 8086 systems with up to 10 Mhz processors at a fraction of the cost of comparable performance static RAM.

MEGARAM .... 1/4 MBYTE, **$595.00**
1/2 MBYTE, **$749.00** .... 1 MBYTE, **$1095.00**
2 MBYTE, **$1795.00**

IBM–PC COMPATIBLE S100–BUS SYSTEM

LOMAS DATA PRODUCTS offers IBM–PC compatible systems with performance far exceeding that available from IBM. You can purchase systems offering performance of an eight Mhz 8086 or up the performance of an 8Mhz 80286. Each system is capable of supporting 8Mhz math coprocessors. Our 8Mhz 80286 system offers IBM–PC compatibility while offering up to 2 times the performance of the IBM–PC–AT. For applications where PC compatibility is desirable but higher performance is a benefit or requirement LDP offers the only viable solution.

WHO IS LOMAS DATA PRODUCTS...

LOMAS DATA PRODUCTS has been shipping 16 bit microprocessor S100 bus products for five years. We have earned a strong reputation for reliability and performance over these five years. We were running MS–DOS (SCP–DOS/PC–DOS) before IBM knew the operating system existed. We offer a wide range of 16 bit operating systems including MS–DOS, CP/M-86, CONCURRENT CP/M–86 and CONCURRENT DOS. All our products are backed by a one year guarantee. We offer no 8 bit products and concentrate entirely on high performance 16 bit systems. *If you are looking for the highest performance possible on the S100 bus, you can be sure LOMAS DATA PRODUCTS offers it.*

Dealer inquiries invited.

MS-DOS, trademark of Microsoft
PC-DOS, trademark of IBM
Concurrent CP/M-86, concurrent DOS, trademark of Digital Research

LOMAS DATA PRODUCTS, INC. LDP

182 CEDAR HILL ROAD, MARLBORO, MASSACHUSETTS 01752 □ TELEPHONE: (617) 460-0333 □ TELEX: 4996272

For orders outside the U.S., contact our exclusive dealers: □ **Australia** – LAMRON PTY. LTD., (02) 808-3666 □ **Malaysia** – EXA COMPUTER (m) SENDIRIAN BERHAD, 795284 □ **England** – RATIONAL SYSTEMS, 0908-613209 or 0908-611349; SHARPBORN LTD., 018764559.

```
         DIV     BX
         CALL    CONVERT2NUM                 ;TENTHS DIGIT
         MOV     AX,DX
         MUL     CX
         DIV     BX
         CALL    CONVERT2NUM                 ;HUNDRETHS DIGIT
         MOV     AH,9
         MOV     DX,OFFSET RES_MSG2
         INT     21H
         MOV     AH,4CH           ;TERMINATE
         INT     21H
CONVERT2NUM:
         OR      AL,30H  ;'0'
         MOV     [BP+0],AL
         INC     BP
         RET
TIME1    DW      0000
TIME2    DW      0000
INIT_MSG         EQU     $
         DB      'CPU SPEED CHECKER ',2CH,' please w'
         DB      'ait...',CR,LF,CR,LF
         DB      '$'
RES_MSG DB       'Execution time should be 10.00 secs on a
         DB      '4.77 Mhz 8088',CR,LF
         DB      'with no WAITs on RAM access',CR,LF
         DB      'Actual execution time here was '
VALUE1   DB      '00.00'
         DB      ' seconds',CR,LF
         DB      '$'
RES_MSG2         EQU     $
         DB      'Effective 8088 clock speed = '
VALUE2   DB      '00.00'
         DB      ' Mhz',CR,LF
         DB      '$'
CODE     ENDS
END      START
```

PROGRAM 3 LISTING

```
;
;  EXTND.INC  --
         .SALL               ;;DONT BORE POEPLE WITH EXPANSION
         IF1                 ;;WE NEED ONLY INVOLVE PASS 1
;        WRITTEN FOR MICROSOFT ASSEMBLER TO DEFINE
;        UNIQUE INSTRUCTIONS IN NEC'S PD70108/PD70116
;        MICROPROCESSOR.  INCLUDE THIS FILE IN YOUR
;        PROGRAM WITHIN ANY SEGMENT DEFINITION BEFORE
;        USING NEC INSTRUCTIONS; E.G.
;
;CSEG    SEGMENT PUBLIC PARA
;        ASSUME CS:CSEG,DS:CSEG,SS:CSEG
;
;        INCLUDE EXTND.INC
;
;        ORG     100H
;START:
;        <ETC.>
;        WE NEED TO DEFINE SOME LABELS TO HELP DIFEREN-
;        TIATE BETWEEN 8-BIT REGISTER ARG.S & 16-BIT
;        REGISTER ARG.S TO OUR MACROS & TO GIVE
;        REGISTERS VALUES TO BUILD INSTRUCTIONS
;
         .VAL    = 0
         IRPC    Z,ACDB  ;; DEFINE FIRST 4 16-BIT REGGIES
@&Z&X    = .VAL
         .VAL = .VAL + 1
         ENDM

         .VAL    = 0
         IRPC    Z,ACDB     ;; DEFINE ALL OF 8-BIT REGISTERS
@&Z&L    = .VAL
@&Z&H    = .VAL + 4
         .VAL = .VAL + 1
         ENDM
;
;SET_TYPE -
;        GIVEN AN ARGUMENT, SETS VARIABLE 'TYPE OF_ARG' TO
;                0   IF IT IS AN IMMEDIATE VALUE
;                1            A  BYTE VARIABLE REFERENCE
;                2               WORD VARIABLE
;               ...          ETC
;               101              BYTE REGISTER
;               102              WORD REGISTER
;               201              BYTE PTR TYPE REFERENCE
;               202              WORD PTR TYPE REFERENCE
;
SET_TYPE MACRO ARG
    IFIDN <ARG>,<BYTE>
         TYPE_OF_ARG = 201H
    ELSE
         IFIDN <ARG>,<WORD>
                TYPE_OF_ARG = 202H
         ELSE
             TYPE OF ARG = TYPE ARG
             IFE TYPE OF ARG
                 IFDEF .&ARG
                     TYPE OF ARG = TYPE .&ARG
                     IF TYPE OF ARG
                         TYPE_OF_ARG = TYPE_OF_ARG + 100H
                     ENDIF
                 ENDIF
             ENDIF
         ENDIF
     ENDIF
     ENDM
```

```
;OK WE CAN NOW BEGIN DEFINING NEC UNIQUE INSTRUCTIONS
;THE BCD INSTRUCTIONS-
;        THESE HAVE NO ARGUMENTS, SO JUST PUT OPCODE
ADD4S    MACRO
                  DB       0FH,20H
         ENDM

SUB4S    MACRO
                  DB       0FH,22H
         ENDM

CMP4S    MACRO
                  DB       0FH,26H
         ENDM

;THE BIT FIELD INSTRUCTIONS-
;     THESE ALLOW EITHER 2 1-BYTE REGISTERS OR ELSE
;     A BYTE REGISTER & AN IMMEDIATE BYTE VALUE.
;     HERE WE CAN DO ALMOST COMPLETE ERROR
;     CHECKING BECAUSE OF SIMPLE ARGUMENT TYPES.
;     DEFINE A TEMPLATE TO DO THAT.

BIT_FIELD MACRO OC,ARG1,ARG2

    .OK = 1
    SET TYPE ARG1          ;;CHECK ARGUMENT 1 FOR VALIDITY (REG8)
    IF TYPE_OF_ARG - 101H
               ARG1     ; ARGUMENT MUST BE BYTE REGISTER
               .OK = 0
    ENDIF

    OPCODE = OC
    IF .OK
         SET TYPE ARG2  ;;NOW CHECK ARGUMENT 2 (REG8 OR IMM8)
         IFE TYPE OF ARG
               OPCODE = OPCODE + 8       ;;IMMEDIATE ARG
         ELSE
               IF TYPE OF ARG-101H       ;;CHECK FOR REG8
                    ARG2 ARGUMENT MUST BE BYTE REG OR IMMED
                    .OK = 0
               ENDIF
         ENDIF
    ENDIF

    IF .OK                 ;;IF ARGUMENTS OK, BUILD INSTR.
         DB     0FH
         DB     OPCODE
         IFE TYPE OF ARG
               DB 0C0H+@&ARG1  ;;IMMED8 TYPE SECOND ARG
               DB ARG2
         ELSE
               DB 0C0H+(@&ARG2 SHL 3)+@&ARG1 ;;REG8 TYPE 2ND ARG
         ENDIF
    ENDIF
    ENDM

;INS-

INS MACRO     ARG1,ARG2
              BIT_FIELD 31H,ARG1,ARG2
     ENDM
;EXT-

EXT MACRO     ARG1,ARG2
              BIT_FIELD 33H,ARG1,ARG2
     ENDM

;NOW BIT INSTRUCTIONS
;BIT INSTRUCTIONS HAVE A COMPLICATED STRUCTURE, ALLOWING
;EITHER CL OR ELSE AN IMMED VALUE FOLLOWED BY ANY REG. OR
;MEMORY ;REFERENCE (IT WAS NECESSARY TO REVERSE ARGUMENTS
;FROM NEC'S ;DEFINITION BECAUSE OF COMPLESITY OF ADDRESS-
;ING MODES ALLOWED.  THIS TEMPLATE BUILDS USES 'INC' IN-
;STRUCTION TO BUILD MORE COMPLICATED ADDRESSING MODES FOR
;IT (THE OPCODE OF INC INSTRUCTION IS OVERWRITTEN WITH
;PROPER CODE AFTER ASSEMBLER GENERATES IT)

BIT      MACRO   OC,ARG1,ARG2,ARG3,ARG4
         LOCAL   OPC,CONT
         OPCODE = OC
         .OK = 1
         IFDIF <ARG1>,<CL>      ;;1ST ARG MUST BE CL OR IMMED
             SET TYPE ARG1
             IF TYPE OF ARG
                 ARG1 ;ARGUMENT MUST BE CL OR IMMEDIATE VALUE
                 .OK = 0
             ELSE
                 OPCODE=OPCODE+8;;IF NOT CL, SET CL BIT IN OPCODE
                 ENDIF
         ENDIF
         IF .OK
                 SET TYPE ARG2
                 IFE TYPE OF ARG ;;2ND ARG ANYTHING BUT IMMED
                         ARG2    ;ARGUMENT MAY NOT BE IMMED. VALUE
                         .OK = 0
                 ENDIF
                 IFE (TYPE_OF_ARG-2H) MOD 100H ;;IF MEM16 OR REG16
                                               ;;SET W BIT IN OPCODE
                         OPCODE = OPCODE + 1
                 ENDIF
         ENDIF
```

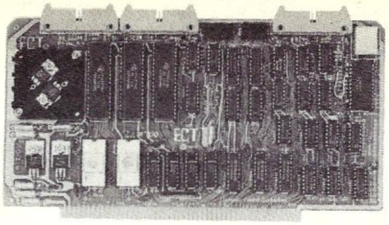
```
        IF .OK
           DB      0FH
           IFE (TYPE OF ARG-102H);;FOR REG16 WE MUST BUILD INSTR
                   DB OPCODE
                   DB 0C0H+@&ARG2
           ELSE
OPC:               INC ARG2 ARG3 ARG4 ;;LET 'INC' BUILD ARGUMENT
CONT:              ORG OPC
                   DB OPCODE        ;;NOW OVERWRITE OPCODE W/ OURS
                   ORG CONT
           ENDIF
           IFDIF <ARG1>,<CL>
                   DB ARG1
           ENDIF
        ENDIF
        ENDM

;TEST1 -

TEST1   MACRO   ARG1,ARG2,ARG3,ARG4
                BIT     10H,ARG1,ARG2,ARG3,ARG4
        ENDM

;NOT1 -

NOT1    MACRO   ARG1,ARG2,ARG3,ARG4
                BIT     16H,ARG1,ARG2,ARG3,ARG4
        ENDM

;CLR1 -

CLR1    MACRO   ARG1,ARG2,ARG3,ARG4
                BIT     12H,ARG1,ARG2,ARG3,ARG4
        ENDM

;SET1 -

SET1    MACRO   ARG1,ARG2,ARG3,ARG4
                BIT     14H,ARG1,ARG2,ARG3,ARG4
        ENDM

;NEW STRING PREFIXES
;       NO ARGUMENTS ALLOWED HERE

REPC    MACRO
                DB      65H
        ENDM

REPNC   MACRO
                DB      64H
        ENDM

;BCD ROTATES
;ONLY ONE ARGUMENT BEING ANY VALID 1-BYTE REFERENCE.
;AGAIN WE USE 'INC' INSTRUCTION TO BUILD PROPER
;ADDRESSING BYTES & OVERWRITE OPCODE WITH OUR OWN

BCD_ROT MACRO   OPCODE,ARG1,ARG2,ARG3
        LOCAL   OPC,CONT

        SET TYPE ARG1
        IF (TYPE OF ARG - 01H) MOD 100H
                ARG1    ARG2 ARG3; ARGUMENT MUST BE
                                 ; A BYTE REFERENCE
        ELSE
                DB      0FH
OPC:            INC     ARG1 ARG2 ARG3
CONT:           ORG     OPC
                DB      OPCODE
                ORG     CONT
        ENDIF
        ENDM

;ROL4-

ROL4    MACRO ARG1,ARG2,ARG3,ARG4
                BCD_ROT 28H,ARG1,ARG2,ARG3
        ENDM

;ROR4-

ROR4    MACRO ARG1,ARG2,ARG3,ARG4
                BCD_ROT 2AH,ARG1,ARG2,ARG3
        ENDM

;
;SEGMENT OVERRIDE
;PROVIDE METHOD FOR INCLUDING SEGMENT OVERRIDES
;
CS_     MACRO ARG1,ARG2,ARG3,ARG4,ARG5
        DB      2EH
        ARG1    ARG2,ARG3,ARG4,ARG5
        ENDM

ES_     MACRO ARG1,ARG2,ARG3,ARG4,ARG5
        DB      26H
        ARG1    ARG2,ARG3,ARG4,ARG5
        ENDM

SS_     MACRO ARG1,ARG2,ARG3,ARG4,ARG5
        DB      36H
        ARG1    ARG2,ARG3,ARG4,ARG5
        ENDM
;J_NOT NEC <LABEL>
;  NOT STRICTLY A NEC UNIQUE INSTRUCTION, THIS MACRO
;  TESTS CPU TO DETERMINE IF IT IS AN INTEL 8086/88
;  OR NEC V20/V30; IF 86/88 JUMPS TO LABEL ARGUMENT,
;  ELSE CONTINUES.  ALL REGS ARE RETAINED EXCEPT CX.

J_NOT_NEC MACRO ARG
        LOCAL   V_SERIES

        PUSH    CS          ;;PUSH CS IN CASE IT'S INTEL
        XOR     CX,CX
SET1    0,CX                ;;INTEL WILL POP CS,
                            ;;NEC WILL SET BIT IN CX
        OR      CX,CX       ;;IS CX STILL 0?
        JNZ     V_SERIES
        JMP     ARG         ;;YES — AHA, ITS AN INTEL CHIP
V_SERIES:
        POP     CX          ;;NO—ITS NEC, BUT READJUST STACK
        ENDM

        ENDIF               ;;END OF MACRO INCLUDE FILE
```

PROGRAM 4 LISTING

```
$NoList

;       Repeat Prefixes

CodeMacro       REPC    Prefx
        DB      065H
EndM

CodeMacro       REPNC   Prefx
        DB      064H
EndM

;       Bit Field Instructions

CodeMacro       INS     dst : Rb, src : Rb
        DW      0310FH
        ModRM   src, dst
EndM

CodeMacro       INS     dst : Rb, count : D(0,15)
        DW      0390FH
        ModRM   0, dst
        DB      count
EndM

CodeMacro       EXT     dst : Rb, src : Rb
        DW      0330FH
        ModRM   src, dst
EndM

CodeMacro       EXT     dst : Rb, count : D(0,15)
        DW      03B0FH
        ModRM   0, dst
        DB      count
EndM

;       Bit Manipulation Instructions

CodeMacro       TEST1   dst : Eb, off : Rb(CL)
        Segfix  dst
        DW      100FH
        ModRM   0, dst
EndM

CodeMacro       TEST1   dst : Ew, off : Rb(CL)
        Segfix  dst
        DW      110FH
        ModRM   0, dst
EndM

CodeMacro       TEST1   dst : Eb, off : D(0,7)
        Segfix  dst
        DW      180FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       TEST1   dst : Ew, off : D(0,15)
        Segfix  dst
        DW      190FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       SET1    dst : Eb, off : Rb(CL)
        Segfix  dst
        DW      140FH
        ModRM   0, dst
EndM

CodeMacro       SET1    dst : Ew, off : Rb(CL)
        Segfix  dst
        DW      150FH

        ModRM   0, dst
EndM

CodeMacro       SET1    dst : Eb, off : D(0,7)
        Segfix  dst
        DW      1C0FH
        ModRM   0, dst
        DB      off
EndM
```

CUSTOM PRODUCTS

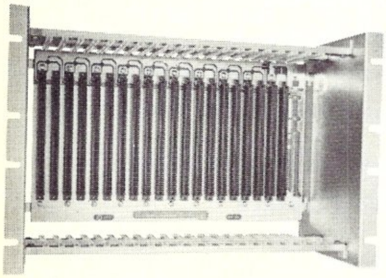
DESIGN • LAYOUT MANUFACTURING



R ²I/O

ROM/RAM & I/O

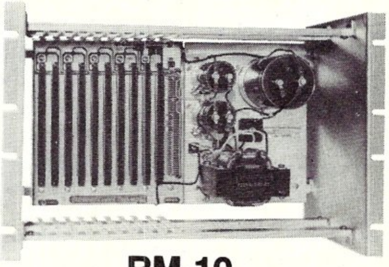
S-100 PRODUCTS



ECT-100-F

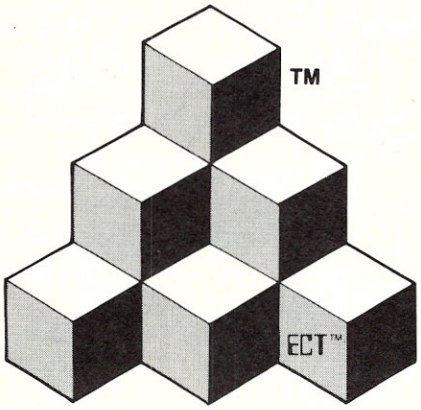
RACKMOUNT CARD CAGES

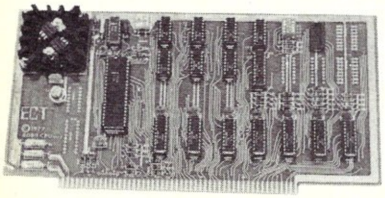
ECT™



RM-10

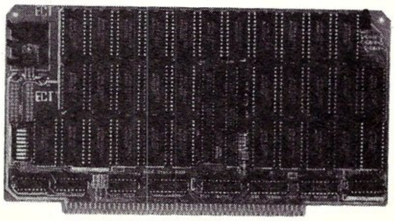
CARD CAGE & POWER SUPPLY





8080 CPU

CENTRAL PROCESSING UNITS

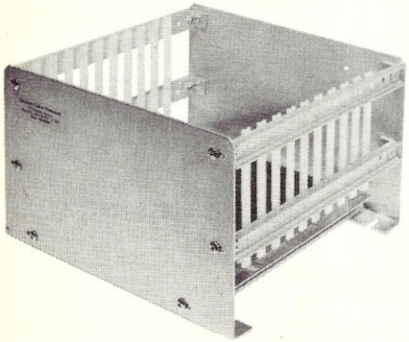


64K RAM

FULLY STATIC MEMORY

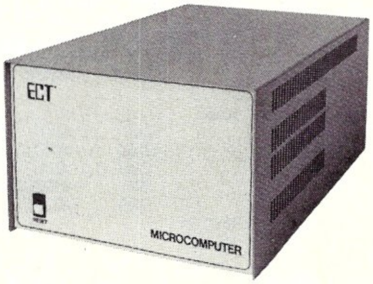
BUILDING BLOCKS FOR MICROCOMPUTER SYSTEMS, DEDICATED CONTROLLERS AND TEST EQUIPMENT

CARD CAGES, POWER SUPPLIES MAINFRAMES, CPU'S, MEMORY I/O, OEM VARIATIONS



CCMB-10-F MIN

6,10 OR 20 SLOT CARD CAGES



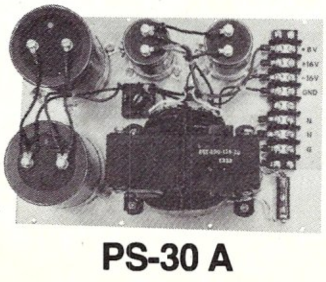
TT-10

TABLE TOP MAINFRAMES

ELECTRONIC CONTROL TECHNOLOGY, INC.

10 Cottage St., Berkeley Heights, NJ 07922 (201) 464-8086

SPECIALIZING IN QUALITY MICRO COMPUTER HARDWARE



PS-30 A

POWER SUPPLIES

MULTIBUS® PRODUCTS

MULTIBUS IS A TRADEMARK OF INTEL CORP.

```
CodeMacro       SET1    dst : Ew, off : D(0,15)
        Segfix  dst
        DW      1D0FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       NOT1    dst : Eb, off : Rb(CL)
        Segfix  dst
        DW      160FH
        ModRM   0, dst
EndM

CodeMacro       NOT1    dst : Ew, off : Rb(CL)
        Segfix  dst
        DW      170FH
        ModRM   0, dst
EndM

CodeMacro       NOT1    dst : Eb, off : D(0,7)
        Segfix  dst
        DW      1E0FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       NOT1    dst : Ew, off : D(0,15)
        Segfix  dst
        DW      1F0FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       CLR1    dst : Eb, off : Rb(CL)
        Segfix  dst
        DW      120FH
        ModRM   0, dst
EndM

CodeMacro       CLR1    dst : Ew, off : Rb(CL)
        Segfix  dst
        DW      130FH
        ModRM   0, dst
EndM

CodeMacro       CLR1    dst : Eb, off : D(0,7)
        Segfix  dst
        DW      1A0FH
        ModRM   0, dst
        DB      off
EndM

CodeMacro       CLR1    dst : Ew, off : D(0,15)
        Segfix  dst
        DW      1B0FH
        ModRM   0, dst
        DB      off
EndM

;       Emulation Mode Instructions

CodeMacro       BRKEM   vector : Db
        DW      0FF0FH

        DB      vector
EndM

;       BCD String Arithmetic

CodeMacro       ADD4S
        DW      200FH
EndM

CodeMacro       ADD4S   dst:M, src:M
        NoSegfix        ES, dst
        Segfix          src
        DW      200FH
EndM

CodeMacro       SUB4S
        DW      220FH
EndM

CodeMacro       SUB4S   dst:M, src:M
        NoSegfix        ES, dst
        Segfix          src
        DW      220FH
EndM

CodeMacro       CMP4S
        DW      260FH
EndM

CodeMacro       CMP4S   dst:M, src:M
        NoSegfix        ES, dst
        Segfix          src
        DW      260FH
EndM

;       BCD Rotates

CodeMacro       ROL4    dst:Eb
        Segfix  dst
        DW      280FH
        ModRM   0, dst
EndM

CodeMacro       ROR4    dst:Eb
        Segfix  dst
        DW      2A0FH
        ModRM   0, dst
EndM

$List
```

16 USERS ON YOUR I.B.M. P.C.

TURBOSLAVE - P.C.

- 8 MHZ Z-80H
- 128K RAM
- 2 RS-232 PORTS
- MULTIPROCESSOR ARCHITECTURE
- P.C. COMPATIBLE
- FASTEST CPM COPROCESSOR
- 16 TURBOSLAVES PER P.C.
- CPM-80 COMPATIBLE
- TRUE MULTI-USER
- LOW COST - $495. RETAIL

TURBOSLAVE P.C. TURNS YOUR I.B.M. INTO A MULTI-USER MULTI-PROCESSOR SUPER MICRO. ONE TURBOSLAVE CAN BE USED AS A SIMPLE CPM COPROCESSOR. BY ADDING THE TURBODOS OPERATING SYSTEM, UP TO 16 TURBOSLAVE P.C.'S CAN BE ADDED, EACH SUPPORTING ONE USER ON A STANDARD TERMINAL. ALL CPM 2.2 PROGRAMS AND MOST MULTI-USER MPM APPLICATIONS CAN BE RUN UNDER TURBODOS. TRUE RECORD LOCKING AND PRINT SPOOLING ARE FULLY SUPPORTED.

EARTH COMPUTERS

"Building Blocks For The Super Micro"

TELEX: 9109976120 EARTH FV
P.O. BOX 8067 • DEPT. D1 • FOUNTAIN VALLEY, CA 92728
CALL: (714) 964-5784

★ S-100 VERSION AVAILABLE ★

REGISTERED TRADEMARKS: Z-80H, ZILOG INC.; TURBODOS, SOFTWARE 2000, INC.

TERMINAL EMULATION $49

With XTERM your PC can emulate the DEC VT100/102, VT52, IBM3101, HP2624, ADDS, and ADM3A. Site licenses available for all products.

BASICA WINDOWS $49

Enhance your basica programs with a dozen new window commands for both interpreter or compiler. Add $399 for complete "ASM" source code.

DES ENCRYPTION $59

Government approved data encryption standard with compression and telecommunications formatting. Add $399 for complete "C" source code.

REMOTE OPERATIONS $69

Operate your PC/DOS remotely via telephone using CONSOLX. Great for LAN or dBase II applications. Add $399 for complete "ASM" source code.

HAWKEYE GRAFIX
23914 Mobile Street
Canoga Park, CA 91307
(818) 716-5220

RP/M2™ creates

CP/M©2.2 compatible

IBM PC

1. Remove the 8088
2. Install the NEC μPD70108
3. Boot PC RP/M2

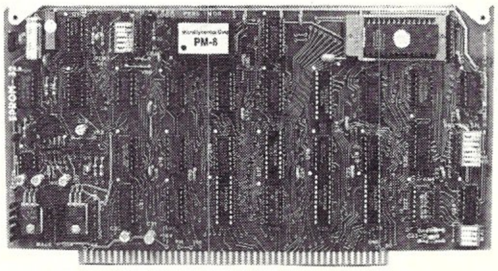
MSDOS and CP/M 2.2 capability coexist in your IBM PC. The NEC μPD70108 CPU chip is a fast 8088 that can also execute 8080 machine code. PC RP/M2 is standard 2.2 compatible RP/M2 with our CBIOS for the IBM PC. System disk with manual and NEC μPD70108 $129. Shipping $5 ($10 nonUS)

Micro Methods, Inc.

118 SW First St. - Box G
Warrenton, OR 97146
(503)861-1765

S-100 EPROM PROGRAMMER

EPROM-32



- Field-proven board meets IEEE-696 standard.
- Programs 1K through 32K (byte) EPROMs.
- Textool zero-insertion-force programming socket.
- EPROM is programmed through I/O ports and can be verified through I/O ports *or* located in memory space for verification.
- Programming voltage generated on-board.
- Personality Modules adapt board to EPROMs:

PM-1—2508, 2758 2516, 2716	PM-3—2732, 2732A	PM-6—68764
	PM-4—2564	PM-8—27128
PM-2—2532	PM-5—2764	PM-9—27256

- Feature-packed CP/M-compatible control software includes **fast programming algorithm.**
- One year warranty.

$269.95*
(A & T)

MicroDynamics Corporation

P.O. Box 17577 • Memphis, TN 38187
(901)-755-0619

* Price includes EPROM-32, documentation and two Personality Modules(specify). Additional Modules—$17.95. Control software on 8" SSSD diskette—$29.95. UPS ground—$2.00, UPS air—$4.00, COD—$2.00, foreign add $15.00, VISA & MASTERCARD welcome.

See Dec. 1983 **Microsystems** for a review of the EPROM-32.

COMMERCIAL SYSTEMS CORPORATION

P.O. Box 2307, Fairfax, VA 22031 (703) 425-5221 / Telex:380002/Easylink:62730560

MULTI-USER MICROPROCESSORS & COMPONENTS Featuring Intercontinental Micro Systems

8-BIT MASTER PROCESSOR

CPZ-48004	$746	Z80A 4MZ 64K Master Processor
CPZ-48006	$784	Z80B 6MZ 64K Master Processor

8-BIT SLAVE PROCESSORS

CPS-6A	$412	Z80B 6MZ 64K Slave-Memory Mapped
CPS-B6A	$562	Z80B 6MZ 128K Slave-Memory Mapped
CPS-B6D	$487	Z80B 6MZ 64K Depopulated CPS-B6A

ARCNET CONTROLLERS

LANS100	$371	ARCNET Controller
LANPC	$371	ARCNET Controller IBM-PC BUS
LANPC-256	$521	LANPC with 256K RAM

ARCNET HUBS

HUB-2A	$146	Repeater-ARCNET Active Hub-2 Port
HUB-4P-ENC	$41	Passive Hub-4 Port
HUB-8A	$319	Active Hub-8 Port
HUB-8A-ENC	$469	HUB-8A with Enclosure & Power Supply

HARD DISK CONTROLLERS-Includes cables, TurboDOS Software Drivers

MD-1013	$521	Monitor Dynamics HD Controller
MD-1016	$521	Monitor Dynamics HD Controller
OMTI-5100	$521	HD Controller-SASI Interface
OMTI-5300	$596	HD & TAPE Cartridge Controller-SASI Interface

CARTRIDGE TAPE DRIVES

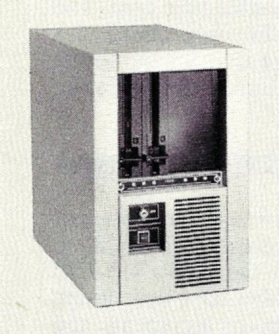
T-540	$975	Cipher, 90 IPS, 45/60 MB
T-525	$520	Cipher, Floppy Tape, 32MB

HARD DISKS

XT1085	$2239	MAXTOR, 30ms, 85MB
XT1140	$3495	MAXTOR, 30ms, 140MB
3051/E	$1395	ATASI, 30ms, 52MB

CABINETS

C2810	$849	Para Dynamics Mini-Pronto, 10 slots.
C3810	$1595	Custom modified Para Dynamics Pronto-all peripherals mounted horizontally in brackets. Holds 2-8″ half heigh floppy drives, 1-5 ¼″ cartridge tape drive, and 2-full height 5 ¼″ Winchester HD. 20 slots. Custom front bezel. All mounting brackets removed by loosening (not removing!) 4 screws. No more cut or scraped knuckles or standing on your head to mount peripherals!



16-BIT MASTER PROCESSORS

CPZ-186	$1121	80186 8MZ 256K Master Processor
CPZ-186-512	$1271	80186 8MZ 512K Master Processor
CPZ-186-1MB	$1496	80186 8MZ 1MB Master Processor

16-BIT SLAVE PROCESSORS

CPS-16	$746	8086 8MZ 256K Slave-Memory Mapped
CPS-16-512	$971	8086 8MZ 512K Slave-Memory Mapped
CPS-16-1MB	$1196	8086 8MZ 1MB Slave-Memory Mapped
CPS-186	$971	80186 10MZ 256K Slave-Memory Mapped
CPS-16-512	$1121	80186 10MZ 512K Slave-Memory Mapped
CPS-186-1MB	$1346	80186 10MZ 1MB Slave-Memory Mapped
CPS-286	AVAILABLE-3rd Quarter, 1985	

ARCNET WORKSTATIONS

WS80-6A	$796	Z80B 6MZ 128K Workstation-Busless
WS80-6A-ENC	$896	WS80-6A with Enclosure & Power Supply

VIDEO DISPLAY DEVICES

ESP-6310	$556	Esprit Systems, Inc.

SHEET FEEDERS

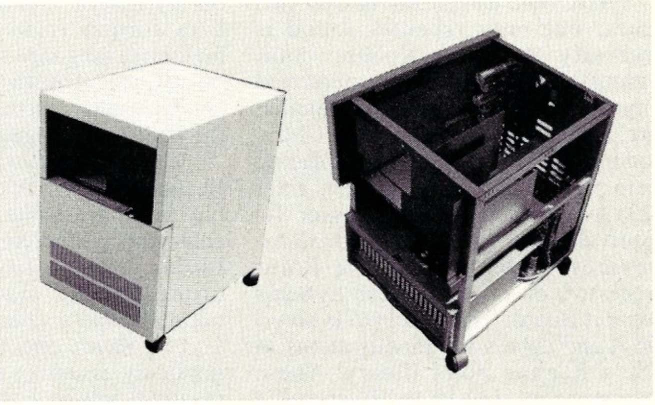
Z-300	$1995	Ziyad Electronic, 2 bins & envelope feed
Z-200	$1596	Ziyad Electronic, 2 bins
Z-100	$495	Ziyad Mechanical, 1 bin

FLOPPY DRIVES

8-242	$375	QUME, 8″ DSDD, 1.2MB

SPECIAL SERVICES

Custom Hardware Configurations
TurboDOS System Generation
Tape Backup Sub-Systems
CALL FOR PRICING



TurboDOS COMPATIBLE MULTI-USER SOFTWARE

DATABASE SYSTEMS

Informa-20	$720	Unlimited Processing, 8-bit
Informa-20	$720	Unlimited Processing, 16-bit
The Language	$995	O'Hanlon Systems

WORD PROCESSING

Word Processor	$250	Palantir, 8 bit, includes Mailout
Word Processor	$250	Palantir, 16 bit, **AVAILABLE IN SEPTEMBER, 1985**
The Filer	$145	Palantir, 8 bit
The Speller	$145	Palantir, 8 bit

OFFICE AUTOMATION

T-Office	$995	Office automation, includes multinode ELECTRONIC MAIL and REMOTE MAIL FORWARDING, File Cabinet, Calendar & Scheduling and terminal communications.

APPLICATION SOFTWARE

BACS	CALL	Business Accounting Control System, includes, GL, PR, AP, AR, OE/IC and POS by American Business Systems, Inc.
MCS-3	CALL	Manufacturing System, includes IC, MRP, BOM, OE, Shop Floor Control, and Capacity Planning by Micro Manufacturing Systems, Inc.

All prices, specifications & availability subject to change and stock on hand. System House and Dealer inquires invited. TurboDOS is a trademark of Software 2000, Inc. T-Office is a trademark of Commercial Systems Corp.

Faster Floating Point Math

by N. T. Carnevale

Speed Up A C Compiler's Math Operations Using A Math Coprocessor IC

The Software Toolworks' C80 compiler has received high marks in several reviews for being a nearly complete implementation of C that compiles in a reasonable time to produce compact, fast integer code. It even has an optional float/long library that extends its arithmetic capabilities beyond the basic byte and integer data types.

However, the speed of floating point operations in most languages for 8-bit micros leaves something to be desired, and C80 is no exception. In a list of benchmark timings based on Bill Savage's program (Dr. Dobb's Journal #94 (Aug.1984), pp.110-114), C80 clocked in at 238 seconds on a 6mHz 8085, but BASIC 80 took only 175 seconds on a 5mHz 8085!

I have been using C80 for over two years, but only recently found it necessary to use its floating point arithmetic. Appalled that an interpreted language could outperform a compiled one, I decided to speed up C80's floating point operations by replacing them with routines to run the Intel 8231A floating point processor IC (equivalent to the Advanced Micro Devices Am9511 Floating Point Processor) on my CompuPro Systems Support Board. This article tells how I did it and details my modifications to C80's floating point library. These routines were written for Microsoft's M80, but they can be altered easily for use with Digital Research's RMAC or the assembler provided with C80. Throughout this article I will also try to give helpful suggestions for others who might want to add hardware floating point support to other compilers and languages.

FLOATING POINT FORMATS

Unfortunately, there is no FPP for 8-bit systems that monitors the bus for data and commands destined for itself, responding automatically without specific intervention by the CPU. Instead, the CPU must feed it the data and instructions and retrieve the results by ported or memory-mapped I/O. Any speed advantage that an FPP may have over software arithmetic can be wiped out by delays required to convert the data format from that used by a high-level language to its own internal format, feed it the data and commands, get the results, and then convert back to the high-level language's format.

C80 and the 8231A both represent floating point variables with four bytes, but there are some differences (see comments just before label C2AMD: in Listing 3). Both have a 24-bit two's complement mantissa that is normalized to a value between 0.5 and 1. C80 saves one bit of storage by implicitly assuming the high bit of the mantissa will be 1, so the 24th bit can be used to represent mantissa sign. The 8231A retains the high bit of the mantissa, and puts the mantissa sign bit in the exponent byte. This reduces the number of bits available for the exponent to seven, so the range of numbers that the 8231A can represent is 2^-64 to 2^+63. The more efficient storage scheme used by C80 and many other languages lets them represent numbers from 2^-128 to 2^127 with no loss of precision in the mantissa.

Despite this difference of range, the fundamental similarity of the formats meant that reasonably efficient routines could be written to convert between them (see C2AMD: and AMD2C: in Listing 3). I first wrote these routines as functions, tested them for proper operation with a program that printed out the bit patterns for various floating point values before and after conversion. Listing 1 gives this program for the convenience of others who wish to adapt it to investigate the internal representation of floats in other versions of C, some of which do not provide such information in their documentation. Single precision floating point variables in Microsoft's FORTRAN 80, and single floats in Alcor's C, also occupy 32-bits. There is a hardware floating point library to use the 8231A with FORTRAN 80, and it might be possible to develop one for single floats in Alcor C. The 8232 (Am9512), which does both four and eight byte floating point arithmetic, might be a better choice for Alcor C, which also has a double float type. The 8232 would be the only choice for ECO-C, which uses an internal eight byte double float format in all floating point calculations.

PASSING ARGUMENTS & RETURNING RESULTS

The next step was to write a set of routines to pass arguments from C80 to the FPP and return the result. The C80 manual gives a rather terse description of how floating point arguments and results are passed, and suggests that interested users examine the code that is produced to figure it out for themselves.

Which I did. The intrinsic float operations (add, subtract, multiply and divide) and the *extrinsic* functions (e.g. sin(), sqrt(), or user-defined functions) all return their result in registers BCDE, but there are important differences in how they receive arguments. The intrinsic operations are called with the first argument on the stack and the second in registers

BCDE. The library routines for these elementary operations remove the first argument from the stack and return the result in BCDE.

The extrinsic functions retrieve all arguments from the stack without disturbing the stack pointer. The stack pointer is still unchanged upon return, so the calling routine must remove the arguments from the stack to prevent stack overflow. Code to preserve and fix the stack is generated automatically when the C80 compiler processes C source code. It is up to the programmer to keep track of the stack pointer and restore it as needed when writing functions in assembly language.

Although the LGFLTLIB.ASM and FLOATLIB.ASM sources provided by the Software Toolworks are uncommented, they were invaluable for tracing what happened to arguments and function results. Few other compiled languages provide the source code for their libraries, but many produce assembly code that can give useful clues to internal data formats, the names or locations of entry points for important library routines, and what source code constructions produce the most efficient object code. Such access to assembly code also offers the possibility of hand-optimization of critical sections to improve speed and reduce storage requirements.

At this point, I was able to estimate the time required just to convert formats and pass data back and forth to the FPP. This process took more than 167 microseconds (usec), slow enough that it didn't seem worth the effort to replace the floating-point addition and subtraction routines. However, multiplication, division, and the other functions promised to be much faster with the FPP.

My next step was to write routines to use the FPP to duplicate but not replace C80's floating point operations. I wrote these for assembly with M80, following the general format produced by C80. Then I tested them with a program that stepped through a series of arguments, fed them to C80's original floating point library and to my special FPP routines, and compared the results to be sure they gave the same answers as their software counterparts.

DISSECTING THE LIBRARY

Satisfied that no errors had crept in, I now had to decide how to replace the multiplication and division routines, and where to put the functions that were previously part of MATHLIB.C. This is when I became very grateful for two facts about C80: the relocatable floating point library was in Microsoft's REL format and could be examined by LIB; source code for the floating point routines was provided, even though it was really intended for use with the Software Toolworks assembler.

Processing FLIBRARY.REL with LIB (/L option) identified its modules and their entry points. This revealed that the module that contained the floating point multiplication and division routines was called FLTLIB.

FLOATLIB.ASM includes the source for several modules. The source for FLTLIB ends with a RET just before the label f_stak:. I isolated what looked like the FLTLIB code from FLOATLIB, added a list of ENTRYs and EXTRNs as indicated by LIB's listing of FLIBRARY.REL, and inserted DSEG and CSEG to preserve the 15 byte data area revealed by LIB. I named this NUFLTLIB.MAC and assembled it with M80 using the /C option to produce NUFLTLIB.REL and NUFLTLIB.CRF.

The cross-referenced NUFLTLIB.PRN file generated from the CRF file by CREF80 showed that the floating-point multiplication section could be excised in toto - the only entry point that was referenced by other routines was at the label fmul:. However, the division routine had an entry point (div10.:) that might be called by something outside FLTLIB, so I decided not to tamper with the short section between div10.: and fdiv:.

The task of modifying the floating point library would have been more difficult if the source code were not available. Some versions of C (e.g. Alcor) have a runtime package that contains all of the math routines and is linked as a complete unit into every compiled program. Alcor says they would provide information such as locations and lengths of the software math functions to help interested users implement hardware math routines.

FLTLB - THE HARDWARE FLOATING POINT ROUTINES

I gave the hardware math module a different name than the original FLTLIB. This was accomplished with the TITLE pseudo-op (see Listing 2). The DSEG and CSEG pseudo-ops preserve data areas exactly as they were in the original, just in case they are referenced by some other part of the C80 library. The list of EXTRNs and ENTRYs was derived from LIB's output, as described above.

C80 uses a dispatch table to jump to the floating point addition, subtraction, multiplication and division routines. I replaced this with a simple JMP for multiplication and division, and commented out the two unused entries in the dispatch table located at ftab: (Listing 3). Since the dispatcher now only has to decide between addition and subtraction, further simplification is possible, but this would produce only a small improvement of speed and storage.

The actual floating point routines replace two major blocks of code. The first block contained the software multiplication routines starting with label fmult3: and ending just before label pophrt:. I decided to put everything but the hardware floating point division routine in this area. The division routine replaces the original section that started with fdiv: and ended just before mul10.:. Listing 4 gives all the new code, plus the small portion between pophrt: and the old fdiv: that remains unchanged. I have not included the absolute value or polynomial functions (fabs() and F_poly()) in this listing, although they are part of the MATHLIB. If you prefer to include these in your version of FLTLB, use C80 to generate the appropriate assembler code from the C source and patch the assembly code into your FLTLB before assembly.

I don't use the long data type much, but tests showed that it would be beneficial to replace the software long multiplication and division routines with hardware operations. C80 and the 8231A use the same four-byte format for longs, so this should be easy.

The new code includes routines to convert data formats, transfer data to and from the FPP, initiate the various operations, test when results are ready, and detect and handle errors. Two types of errors may occur. A software error results if C80 attempts to pass an argument that is too big or too small for the 8231A (< 2^-64 or > 2^63). A hardware error is generated by the FPP itself in case of arithmetic overflow, underflow, divide by zero, taking the log or square root of a negative number, or passing too large an argument to the exponential or inverse sine or cosine functions. If any one of these errors happens, the present code prints a message and exits to the operating system.

There are times when some hardware errors, such as underflow, could be handled more sensibly by returning a zero or some other value instead of aborting the program. Therefore I included a function fpmask() that can be used to set the error detection mask under program control. The default condition is to test for, and crash in case of, any hardware error. If any of the FPP error tests is disabled, it is up to the programmer to test for and handle such errors in a meaningful way. This might be done by writing a routine that would check the

Super assemblers plus the world's largest selection of cross assemblers!

Z-80 Macroassembler

Power for larger programs! This 2500AD macroassembler includes:

- Zilog Z-80 Macroassembler (with the same powerful features as all our assemblers)
- powerful linker that will link up to 128 files. Com files may start at any address
- Intel 8080 to Zilog Z-80 Source Code Converter (to convert all your Intel source to Zilog Syntax in one simple step)
- COM to Hex Converter (to convert your object files to Hex for PROM creation, etc.)
- 52 page User Manual

8086/88 Assembler with Translator

Available for MSDOS, PCDOS, or CPM/86! This fully relocatable macroassembler will assemble and link code for MSDOS (PCDOS) AND CPM/86 on either a CPM/86 or MSDOS machine. This package also includes:

- An 8080 to 8086 source code translator (no limit on program size to translate)
- A Z-80 to 8086 translator
- 64 page user manual
- 4 linkers included:
  - –MSDOS produces .EXE file
  - –CPM/86 produces .CMD file
  - –Pure object code generation
  - –Object code and address information only

Linker features:

- Links up to 128 files
- Submit mode invocation
- Code, Data Stack and extra segments
- Handles complex overlays
- Written in assembly language for fast assemblies.

Z-8000 Cross Development Package

Instant Z-8000 Software! This package allows development and conversion of software for the Z8001, 8002, 8003 and 8004 based machines on a Z-80, Z-8000 or 8086 machine. This powerful package includes:

- a Z-80/8080 to Z-8000 Assembly Language Source Code Translator
- Z-8000 Macro Cross Assembler and Linker

The Translators provide Z-8000 source code from Intel 8080 or Zilog Z-80 source code. The Z-8000 source code used by these packages are the unique 2500AD syntax using Zilog mnemonics, designed to make the transition from Z-80 code writing to Z-8000 easy.

All 2500 AD Assemblers and Cross Assemblers support the following features:

Relocatable Code — the packages include a versatile Linker that will link up to 128 files together, or just be used for external reference resolution. Supports separate Code and Data space. The Linker allows Submit Mode or Command Invocation.

Large File Handling Capacity —the Assembler will process files as large as the disk storage device. All buffers including the symbol table buffer overflow to disk.

Powerful Macro Section— handles string comparisons during parameter substitutions. Recursion and nesting limited only by the amount of disk storage available.

Conditional Assembly—allows up to 248 levels of nesting.

Assembly Time Calculator— will perform calculations with up to 16 pending operands, using 16 or 32 Bit arithmetic (32 Bit only for 16 Bit products). The algebraic hierarchy may be changed through the use of parentheses.

Include files supported—

Listing Control—allows listing of sections on the program with convenient assembly error detection overrides, along with assembly run time commands that may be used to dynamically change the listing mode during assembly.

Hex File Converter, included —for those who have special requirements, and need to generate object code in this format.

Cross reference table generated—

Plain English Error Messages—

System requirements for all programs: Z-80 CP/M 2.2 System with 54k TPA and at least a 96 column printer is recommended. Or 8086/88 256k CP/M-86 or MSDOS (PCDOS).

Cross Assembler Special Features

Z-8—User defined registers names, standard Zilog *and* Z-80 style support. Tec Hex output option.

8748—standard Intel *and* Z-80 style syntax supported.

8051—512 User defined register or addressable bit names.

6800 Family—absolute or relocatable modes, all addressing modes supported. Motorola syntax compatible. Intel Hex or S-Record format output.

6502—Standard syntax or Z-80 type syntax supported, all addressing modes supported.

	Z80 CP/M®	ZILOG SYSTEM 8000 UNIX	IBM PC MSDOS	IBM PC CP/M 86	OLIVETTI M-20 PCOS
Z8000™	$299.50	$750.00	$299.50	$299.50	$299.50
Z80	99.50	500.00	199.50	199.50	199.50
Z8	199.50	500.00	199.50	199.50	199.50
8086/88	199.50	750.00	99.50	99.50	199.50
80186	199.50	750.00	199.50	199.50	199.50
8748	199.50	500.00	199.50	199.50	199.50
8044/51	199.50	500.00	199.50	199.50	199.50
8080	199.50	500.00	199.50	199.50	199.50
8085	199.50	500.00	199.50	199.50	199.50
8096	199.50	500.00	199.50	199.50	199.50
68020	399.50	750.00	399.50	399.50	399.50
68000,08,10	299.50	750.00	299.50	299.50	299.50
6800,02,08	199.50	500.00	199.50	199.50	199.50
6801,03	199.50	500.00	199.50	199.50	199.50
6804	199.50	500.00	199.50	199.50	199.50
6805	199.50	500.00	199.50	199.50	199.50
6809	199.50	500.00	199.50	199.50	199.50
32000	399.50	750.00	399.50	399.50	399.50
COPS400	199.50	500.00	199.50	199.50	199.50
NSC800	199.50	500.00	199.50	199.50	199.50
6301	199.50	500.00	199.50	199.50	199.50
6501/11	199.50	500.00	199.50	199.50	199.50
6502	199.50	500.00	199.50	199.50	199.50
65C02	199.50	500.00	199.50	199.50	199.50
1802	199.50	500.00	199.50	199.50	199.50
F8/3870	199.50	500.00	199.50	199.50	199.50
NEC7500	199.50	500.00	199.50	199.50	199.50
NCR/32	399.50	750.00	399.50	399.50	399.50
Subtotal	$ ________	$ ________	$ ________	$ ________	$ ________

Name ______________________________
Company ___________________________
Address ____________________________
City ____________ State ________ Zip ________
Phone ______________________________
Make and model of computer
system _____________________________
□ COD (2500AD pays COD charges)
□ VISA or MasterCard
Number ____________________________
Expiration Date ______________________

TO ORDER. **Simply circle the product or products you want in the price columns, and add up your order.**

Total $ ____________

Check one:
□ 8" Single
□ Osborne
□ IBM PC
□ Cartridge tape
□ Apple (Softcard)
□ Kaypro DSDD
other formats available, please call!

Shipping UPS Blue Label no charge, $15.00 International per unit $ ____________

Total Order $ ____________

Signature ______________________________

220

2500AD SOFTWARE INC.

17200 E. Ohio Drive, Aurora, CO 80017 (303) 369-5001 status register immediately after any suspect operation, branching to an error-handling routine if an error occurs.

RESULTS

These routines illustrate two major bottlenecks in the use of any hardware math device that is not a coprocessor. Both are related to data transfer between the CPU and the FPP. The first is format conversion, which takes a major toll: about 46usec per argument passed to the 8231, and 20usec to convert the result back to C80's format. The second bottleneck is the fact that the C80 compiler maintains intermediate results in the CPU's address space rather than leaving them in the FPP's stack. Aside from the time needed to pass and retrieve these results, this imposes extra overhead for needless format conversions. Short of rewriting the C80 compiler, the only way around this would be to modify FLTLB and manually revise critical portions of the 8080 code produced by C80 so that frequently used intermediate results stay in the FPP's stack until all floating point operations are complete.

Despite these shortcomings, the new floating point library is a definite improvement over the original. Savage's benchmark took only 16 seconds on a 4mHz Z80 with a 2mHz 8231A, producing a final result of 2327 (rounded off to the nearest integer - error 2E+2). This is faster than any other 8-bit machine running at 4mHz, and about 15 times faster than C80's software floating point library. The result is slightly more precise than the software arithmetic, and just as (im)precise as any other Am9511 or Intel 8231A FPP in the list--roughly comparable to single precision FORTRAN. This is not my first choice for processing ill-conditioned matrix equations, but it is more than adequate for less demanding applications, such as averaging and display transformations of data following A/D conversion.

Ted Carnevale, an assistant professor of neurology at SUNY Stony Brook NY, is involved in research on information processing in neurons. His primary use of computers is in the implementation of the following algorithm:

```
while not (broke or fired) {
  do_experiments(time_and_money--);
  write_papers(time_and_money--);
  submit_research_proposals_to_get_\
    more_money(time_and_money--);
}
```

LISTING 1

```
/* fpc.c--tests conversion to/from amd9511 fpp format */

#include fprintf.h
#include c80def.h

#define MESSAGE  "\nFloating-point format conversion program")
#define DASHES "\n\n-----------------------------------------")

#define MAX 6   /* how many different values to feed
                   to the format conversion routines */

/* replace the next two functions with
     your own format conversion routines as needed */
extern long c2amd();    /* link .rel file containing these */
extern float amd2c();   /*   to fpc.rel */

main()
{
        int i;
        static int sx=10;
        float x;
        static float dx=1.0;

        printf(MESSAGE);
        printf(DASHES);

        /* a geometric series of positive values */
        for (i = 0, x = 100.0; i<=MAX; x /= sx, i++) showbits(x);
        printf(DASHES);

        /* a geometric series of negative values */
        for (i = 0, x = -100.0; i<=MAX; x /= sx, i++) showbits(x);
        printf(DASHES);

        /* a linear series of positive and negative values */
        for (i = -MAX, x=(float)(-MAX); i<=MAX; x += dx, i++) showbits(x);
}


/* show bit patterns used by C80 and AMD FPP floating point formats
     to represent the float n */
showbits(n)
float n;
{
        int i;
        union {
                float f;
                long l;
        } x,z;  /* unions are easier to use here than pointers */
        long y;

        x.f=n;

        /* show C80's preconversion bit pattern */
        printf("\n\nx = %e,\t\tBCDE = ",x.f);
        prntlong(x.l);

/* eliminate next few lines if you just want to check the format
     used by your version of C */
        /* show AMD formatted data */
        y=c2amd(x.f);
        printf("\n\t\t\t\t AMD = ");
        prntlong(y);

        /* convert back to C80's format */
        z.f=amd2c(y);
        printf("\nz = %e,\t\tBCDE = ",z.f);
        prntlong(z.l);

/* end of C80 -> AMD -> C80 format conversion test */
}


/* print bit pattern of a long, starting with the high-order bit
     of the most significant byte, working down from left to right */
prntlong(k)
long k;
{
        int i;
        union {
                long l;
                char b[4];
        } datum;

        datum.l=k;
        for (i=3; i>=0; i--) {
                prntbyt(datum.b[i]);
                printf(" ");
        }
}


/* print the bit pattern for a byte from left to right,
     high order bit first (does the dirty work for prntlong) */
prntbyte(i)
int i;
{
        int j;
        char bit;

        for (j=0x80; j>0; j=j>>1) {
                if (i & j) bit='1';
                else bit='0';
                printf("%c",bit);
        }
}
```

LISTING 2

```
        TITLE FLTLB
        PAGE 64
; Floating point library
; C/80 3.0 (7/7/83) - (c) 1983 Walter Bilofsky
;MODIFIED 8/84 to use amd9511 for floating point multiply/divide
;and MATHLIB functions as well.
;Replaces modules FLTLIB and MATHLIB
; -- N.T.Carnevale
;
;
;these were gleaned from LIB's listing of FLIBRARY.REL:
;
ENTRY Bf.Bl,Bf.Hc,Bf.Hi,Bf.Hu,Bl.Bf
ENTRY cf.eq,cf.ge,cf.gt,cf.le,cf.lt,cf.ne
ENTRY div10.,mul10.
ENTRY dum
ENTRY errcod
ENTRY F.add,F.div,F.mul,F.neg,F.not,F.sub
ENTRY facl_,facl_1,facl_2,fac_,fac_1
ENTRY fadd.,fadda.,fcomp.,fdiv_a,fdiv_b,fdiv_c,fdiv_g
ENTRY flneg.,float.,flt.0,flt_pk
ENTRY fmlt_1,fmlt_2
ENTRY Hc.Bf,Hi.Bf,Hu.Bf
ENTRY inxhr.
ENTRY movfm.,movfr.,movmf.,movrf.
ENTRY pushf.,qint.,save_,save_1,sign.,zero.
;
EXTRN c.neg,c.sxt,eq.4,g.,Hc.Bl,Hi.Bl,Hu.Bl,L.add,L.neg
EXTRN llong.,movrm.,neq.4,slong.
;
;this preserves the 15 byte data area revealed by LIB:
        DSEG
;
facl_:  DB      0
facl_1: DB      0
facl_2: DB      0
fac_:   DB      0
fac_1:  DB      0
save_:  DB      0
fmlt_1: DB      0
fmlt_2: DB      0
dum_:   DB      0
save_1: DB      0
errcod: DB      0
fdiv_a: DB      0
fdiv_b: DB      0
fdiv_c: DB      0
fdiv_g: DB      0
;
;       CSEG
;
flt_pk: DS      0
F.add:  XRA     A
        JMP     Dual
F.sub:  MVI     A,1
        JMP     Dual
F.mul:  JMP     fpmul           ;jump to new routines
;       MVI     A,2
;       JMP     Dual
F.div:  JMP     fpdiv
;       MVI     A,3
Dual:   CALL    movfr.
```

LISTING 3

```
Ftab:   DW      fadd.
        DW      fsub            ;next two addresses not used
;       DW      fpmul
;       DW      fpdiv
```

LISTING 4

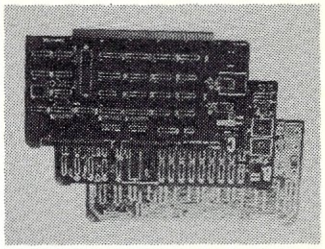
```
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;
;
;What follows replaces the original code in the FLTLIB section of
;FLOATLIB.ASM that started with fmult3: and ended just before pophrt:.
        ;
        ;
CR      EQU 0DH
LF      EQU 0AH
BDOS    EQU 5
BOOT    EQU 0
PSTRNG  EQU 9
        ;
        ;
        ;send message to console
PRMSG:
        PUSH PSW
        PUSH B
        PUSH D
        PUSH H
        MVI C,PSTRNG
        CALL BDOS
        POP H
        POP D
        POP B
        POP PSW
        RET
        ;
;****************************************************************
        ;
        ;
        ;Port addresses
        ;
BASE    EQU 050H        ;base address of Compupro SS1 board
CREG    EQU BASE+9      ;location of 9511's control & data ports
DREG    EQU BASE+8
        ;
        ;
        ;FPP error codes
        ;
ERRBITS EQU 1EH         ;the status register's error bits
OVRFLO  EQU 2           ;overflow
UNRFLO  EQU 4           ;underflow
NEGARG  EQU 8           ;negative argument to sqrt or log function
DIVZER  EQU 10H         ;divide by zero
TOOBIG  EQU 18H         ;arg of asin, acos, or exp too big
        ;
        ;
        ;FPP command codes
        ;
FMUL    EQU 12H
FDIV    EQU 13H
XCHF    EQU 19H         ;swap top two locations in fpp's stack
FSQRT   EQU 1
FSIN    EQU 2
FCOS    EQU 3
FTAN    EQU 4
FASIN   EQU 5
FACOS   EQU 6
FATAN   EQU 7
FLOG    EQU 8           ;log base 10
FLN     EQU 9           ;natural logarithm
FEXP    EQU 0AH         ;e^x
FPWR    EQU 0BH         ;x^y
        ;

;****************************************************************
;****************************************************************
;
;
;
;note:  timing indicated for some of the following
;
;This block converts c80 float in BD to FPP format, then loads it into FPP.
;If out of range for FPP, aborts with warning.
;
;Data format conversion routine C2AMD--
;converts c80's float to amd's fp format.
;Based on suggestions by J.Shook, Electronics Lab, Dept.of Chemistry,
;SUNY Stony Brook
;
```

```
;Floating point formats
;----------------------
;C80 stores floats as:
;         mantissa sign in C7
;         mantissa  =  24 bit two's complement in CDE,
;                          with bit 23 assumed = 1.
;         exponent is added to 128 (80H) and stored in B.
;                          If the number is 0, B=0.
;
;FPP stores floats as:
;         mantissa sign in B7
;         mantissa  =  24 bit two's complement in CDE,
;                          with bit 23 = 1.  However,
;                          the value 0 is represented by BD=0.
;         exponent sign in B6
;         exponent in B5-B0 (only 6 bits).
;
;
        ;Call with value to convert in BD.
C2AMD:
        MOV A,B           ; 4
        ORA A             ; 4
        JZ FPPZERO        ;10--B=0 implies x=0
        ;take care of register B
        SUI 80H           ; 7--corrects exponent
        ;does exponent lie in FPP's range?
        CPI 64            ; 7
        JP EXPHI          ;10--exceeds 2^63
        CPI -64           ; 7
        JM EXPLO          ;10--smaller than 2^-64
        ;exponent ok, proceed
        ANI 7FH           ; 7--mask out hi bit of B
        MOV B,A           ; 4-- and save til later
        MOV A,C           ; 4
        ANI 80H           ; 7--isolate mantissa sign bit
        ORA B             ; 4--fix mantissa sign bit for FPP
        MOV B,A           ; 4--done with B
        ;take care of C
        MVI A,80H         ; 7
        ORA C             ; 4--set hi bit of C
        MOV C,A           ; 4
        JMP LOADFPP       ;10--put data in FPP
        ;114 t cycles @ 4mHz = 28.5usec
        ;
 PPZERO:                  ;x=0
        XRA A
        MOV C,A
        MOV D,A
        MOV E,A
        ;a short cut, time not calculated
        ;fall through to next routine
        ;
        ;
        ;Data transfer routine
        ;LOADFPP puts BD into fpp

LOADFPP:
        MOV A,E           ; 4
        OUT DREG          ;11
        MOV A,D           ; 4
        OUT DREG          ;11
        MOV A,C           ; 4
        OUT DREG          ;11
        MOV A,B           ; 4
        OUT DREG          ;11
        RET               ;10
        ;70 t cycles = 17.5usec
        ;
        ;TOTAL TIME for C2AMD & LOADFPP = 46usec
        ;
;*************************************************************
        ;
        ;
        ;C2AMD found float to be out of range for FPP--warn & abort
EXPHI:
        LXI D,HIMSG
        JMP ERREXIT
EXPLO:
        LXI D,LOMSG
ERREXIT:
        CALL PRMSG
        JMP BOOT          ;bail out!
```

```
        ;
        ;
HIMSG:  DB CR,LF,'float --> FPP exponent overflow ( > 2^63)$'
LOMSG:  DB CR,LF,'float --> FPP exponent underflow ( < 2^-64)$'
        ;
;*************************************************************
;*************************************************************
        ;
        ;
        ;
        ;FLD1 gets 1 float from the stack, puts it into fpp,
        ;and DOES NOT restore stack.  Called by "intrinsic"
        ;float arithmetic operations, e.g. fpmul and fpdiv.
FLD1:
        POP H             ;10--return addr for this block
        SHLD RETADR       ;16
        POP H             ;10--return addr for calling function
        POP D             ;10
        POP B             ;10--BD = argument
        CALL C2AMD        ;17--convert format & load FPP
        ;
        PUSH H            ;11--restore return address of calling function
        LHLD RETADR       ;16--return to calling function
        PCHL              ; 4--  without disturbing stack
        ;104 t cycles = 26usec, + 46 for C2AMD = 72usec
        ;
;*************************************************************
        ;
        ;
        ;FLOAD1 gets 1 float from the stack, puts it into fpp,
        ;and restores stack for c80.  Called by "user-defined"
        ;functions, not by fpmul or fpdiv.
FLOAD1:
        POP H             ;10--return addr for this block
        SHLD RETADR       ;16
        POP H             ;10--return addr for calling function
        POP D             ;10
        POP B             ;10--BD = argument
        CALL C2AMD        ;17--convert format & load FPP
        ;fix stack for c80
        PUSH H            ;11
        PUSH H            ;11
        PUSH H            ;11--restore return address of calling function
        LHLD RETADR       ;16--return to calling function
        PCHL              ; 4--  without disturbing stack
        ;126 t cycles = 31.5usec, + 46 for C2AMD = 77.5usec
        ;

;*************************************************************
        ;
        ;
        ;FLOAD2 gets 2 floats from the stack, puts them into fpp,
        ;  and restores the stack for c80
FLOAD2:
        POP H             ;10--return addr for this block
        SHLD RETADR       ;16
        POP H             ;10--return addr for calling function
        POP D             ;10
        POP B             ;10--BD = second argument
        CALL C2AMD        ;17--convert format & load FPP
        POP D             ;10
        POP B             ;10--BD = first argument
        CALL C2AMD        ;17
        ;fix stack for c80
        PUSH H            ;11 x 5 = 55
        PUSH H
        PUSH H
        PUSH H
        PUSH H            ;restore return address of calling function
        LHLD RETADR       ;16--return to calling function
        PCHL              ; 4--  without disturbing stack
        ;185 t cycles = 46.25usec, + 2 * 46 (for C2AMD) = 138.25usec
        ;
RETADR: DS 2              ;used by all FPP load routines
        ;
;*************************************************************
;*************************************************************
        ;
        ;
        ;
        ;fpmask() allows the user to specify which error bits
        ;are tested for error detection.  The argument to fpmask
        ;becomes the ERRMASK that is ANDed with the status word
        ;to detect a hardware (FPP) error.

        PUBLIC fpmask
fpmask:
        POP H           ;return address
        POP B           ;argument
        PUSH B          ;fix stack
        PUSH H
        MOV A,C         ;get the user-specified error mask
        ANI ERRBITS     ; and mask out all but the genuine error bits
        STA ERRTEST+1
        RET
        ;
;***************************************************************
        ;
        ;
        ;FPP Error handling routine
        ;print message according to what sort of error occurred
FPERR:
        LXI D,EMSG0
        CALL PRMSG      ;print general message
        MOV C,A         ;save error code
        MVI B,MSGPTR-ETYPES     ;# of codes to check
        LXI H,ETYPES    ;M=first code in the list
        LXI D,MSGPTR    ;(DE) = address that contains
                        ;  location of message for first error code
ERCHEK:
        ANA M
        JMP ERMATCH
        MOV A,C         ;restore error code
        INX H           ;advance to next code and message
        INX D
        INX D
        DCR B
        JNZ ERCHEK      ;more codes to test
        ;should never fall through, but if it does, will print 'huh?'
        ;

ERMATCH:
        XCHG            ;(HL) holds address of error message
        MOV E,M
        INX H
        MOV D,M
        JMP ERREXIT
        ;
        ;
EMSG0:  DB CR,LF,,LF,'FPP error:  $'
EMSG1:  DB 'overflow$'
EMSG2:  DB 'underflow$'
EMSG3:  DB 'sqrt or log of negative number$'
EMSG4:  DB 'divide by 0$'
EMSG5:  DB 'argument of asin, acos, or exp too large$'
EMSGX:  DB 'huh?$'      ;should never occur!
        ;
        ;
ETYPES: DB OVRFLO,UNRFLO,TOOBIG,NEGARG,DIVZER
MSGPTR: DW EMSG1,EMSG2,EMSG5,EMSG3,EMSG4,EMSGX
        ;
;***************************************************************
;***************************************************************
        ;
        ;
        ;Next come the hardware floating point routines themselves.
        ;The multiplcation routine contains most of what the
        ;other routines use, so it is listed first.
        PUBLIC  fpmul
fpmul:
        ;Note:  for fp multiply and divide,
        ;c80 passes the first arg in the stack, the second in BD.
        ;For the intrinsic multiply and divide operations,
        ;the calling program does NOT have to remove args
        ;from the stack upon return!  This is quite different from
        ;the situation with user-defined functions.
        ;
        ;take 2nd arg from BD, convert to amd format, and pass to fpp
        CALL C2AMD      ;17
        ;take 1st arg from stack, convert format, and pass to fpp
        CALL FLD1       ;17
        ;send command, test for error, return with result in BD
        MVI A,FMUL      ; 7
        ;fall through to DOIT-ERRTEST-GETIT-AMD2C
```

```
        ;
        ;
        ;DOIT-GETIT starts an FPP operation, tests for error,
        ;gets the result & converts to c80 format
        ;
        ;send command in A to FPP
DOIT:   OUT CREG        ;11--send command
WAIT:   IN CREG         ;10
        ORA A           ; 4
        JM WAIT         ;10--until done
        ;11 + n*24 t cycles = 2.75 + 6n usec (X)
        ;
        ;A contains error code
ERRTEST:
        ANI ERRBITS     ; 7--default = test all error bits
        JNZ FPERR       ;10--if error found, else fall through to GETIT
        ;
        ;get top of fpp's stack into BD
GETIT:
        IN DREG         ;10
        MOV B,A         ; 4
        IN DREG         ;10
        MOV C,A         ; 4
        IN DREG         ;10
        MOV D,A         ; 4
        IN DREG         ;10
        MOV E,A         ; 4
        ;
        ;Fall through to AMD2C, which converts FPP format in BD
        ;to c80 float

        ;AMD2C is based on suggestions by J.Shook, Electronics Lab,
        ;Dept.of Chemistry, SUNY Stony Brook
        ;
AMD2C:
        MOV A,C         ; 4
        ORA A           ; 4
        JP C80ZERO      ;10--bit 7=0 implies value is 0
        ;take care of mantissa sign
        XRA B           ; 4--complements bit 7 of B, among other things
        ANI 80H         ; 7--mask out all the garbage
        XRA C           ; 4--hi bit of C was 1 in AMD format, so this
        MOV C,A         ; 4--  sets the hi bit of C to mantissa sign
        ;take care of exponent sign
        MVI A,40H       ; 7
        ANA B           ; 4--is exponent negative?
        MOV A,B         ; 4
        JZ POSEXP       ;10--no, set hi bit to 1 (add 128)
        ANI 7FH         ; 7--  yes, mask out hi bit
        MOV B,A         ; 4
        RET             ;10
        ;
POSEXP:
        ORI 80H         ;set hi bit for positive exponent
        MOV B,A
        RET
        ;format conversion delay identical whether exponent is + or -
        ;
C80ZERO:
        XRA A
        MOV B,A
        RET
        ;short cut--no time calculated
        ;
        ;(DOIT-ERRTEST-GETIT-AMD2C takes 156 t cycles
        ;  = 39usec, + X (for operation))
        ;
        ;41 t cycles = 10.25usec, + 46 (C2AMD) + 72 (FLD1) + 39
        ;  + X (DOIT) = 167.25 + X usec


;
;float sqrt(x) float x;
        PUBLIC  sqrt
sqrt:
        CALL FLOAD1
        MVI A,FSQRT
        JMP DOIT

;float sin(x) float x;
        PUBLIC  sin
```

```
sin:
        CALL FLOAD1
        MVI A,FSIN
        JMP DOIT

;float cos(x) float x;
        PUBLIC  cos
cos:
        CALL FLOAD1
        MVI A,FCOS
        JMP DOIT

;float tan(x) float x;
        PUBLIC  tan
tan:
        CALL FLOAD1
        MVI A,FTAN
        JMP DOIT

;float asin(x) float x;    /* arc sin in radians */
        PUBLIC  asin
asin:
        CALL FLOAD1
        MVI A,FASIN
        JMP DOIT

;float acos(x) float x;
        PUBLIC  acos
acos:
        CALL FLOAD1
        MVI A,FACOS
        JMP DOIT

;float atan(x) float x;
        PUBLIC  atan
atan:
        CALL FLOAD1
        MVI A,FATAN
        JMP DOIT

;float log(x) float x;  /* log10 */
        PUBLIC  log
log:
        CALL FLOAD1
        MVI A,FLOG
        JMP DOIT

;float ln(x) float x;    /* log e */
        PUBLIC  ln
ln:
        CALL FLOAD1
        MVI A,FLN
        JMP DOIT

;float exp(x) float x;
        PUBLIC  exp
exp:
        CALL FLOAD1
        MVI A,FEXP
        JMP DOIT

;float pwr(x,y) float x,y;       /* x to the yth power */
        PUBLIC  pwr
pwr:
        CALL FLOAD2
        ;
        MVI A,XCHF
        OUT CREG            ;11--send command
PWAIT:  IN CREG             ;10
        ORA A               ; 4
        JM PWAIT            ;10--until done
        ;
        MVI A,FPWR
        JMP DOIT
;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;briefly back to the original source for fltlib--
;
;
pophrt: POP     H
        RET
```



```
;
;since div10.: is called by an external routine, I left it alone.
div10.: CALL    pushf.
        LXI     B,8420H
        LXI     D,0000H
        CALL    movfr.
fdivt:  POP     B
        POP     D


;
;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;the fdiv: routine is replaced by fpdiv:, and muldiv: is not needed

        PUBLIC  fpdiv
fpdiv:
        ;take 2nd arg from BD, convert to amd format, and pass to fpp
        CALL C2AMD          ;17
        ;take 1st arg from stack, convert format, and pass to fpp
        CALL FLD1           ;17
        ;swap fpp's A and B registers
        MVI A,XCHF          ; 7
        OUT CREG            ;11--send command
DWAIT:  IN CREG             ;10
        ORA A               ; 4
        JM DWAIT            ;10--until done
        ;send command, test for error, return with result in BD
        MVI A,FDIV          ; 7
        JMP DOIT            ;10
        ;takes 169.75usec + X plus delay for register swap

;END OF MODIFICATIONS TO FLTLIB.MAC<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
;back to the original code again at mul10.:
mul10.: CALL    movrf.
```

MEX-PC — Full-featured modem software for your IBM-PC (Versions also available for Tandy 2000 and most CP/M machines). Now can ı be menu driven ● Fully programmable function keys ● Simple built-in programming language for custom applications development ● Choice of 8 background/foreground colors ● All popular protocols (KERMIT available Dec. 85) ● Automatic & repeat dialing ● Runs DOS commands and programs while on-line ● "Clones" customized versions.

$59.95

REO — REmote Operation module with password security. This inexpensive add-on module adds remote access capability to your MEX program. It allows you to call up your office computer from home — or your home computer from your office — and swap data or text files, even programs. Download a spreadsheet, rework it at home, then send it back, updated, to your office. Password security keeps unauthorized persons from gaining access to sensitive files.

$29.95

TEM — Intelligent terminal emulation add-on module for MEX-PC. As shipped, MEX emulates an ADM-3A equivalent terminal. **TEM** expands MEX to emulate either a Televideo 925 or DEC VT52/VT100.

TEM is full-function emulation, allowing you to use remote text processors and spreadsheet programs running on mainframes that require a specific terminal interface. Simple installation. Fully documented.

$29.95

NightOwl Connection — A multi-user on-line service that provides you with round-the-clock access to MEX support, plus all the latest and best in public domain and user-supported software. The **Connection** features 36 megabytes of free software — some of it exclusive to **Connection** subscribers — including 18 megabytes worth of programs for the IBM-PC and compatibles. MEX-PC owners can download all updates to their communications program free of charge!

$100/yr.
($50/yr. renewal)

All MEX products have a 30-day, money-back guarantee.

MEX 'n' Match

FEATURE	MEX-PC®	Cross-talk®	PC-Talk®	Symphony®
"CLONE" customized versions	YES	NO	NO	NO
Run DOS commands & programs	YES	YES	NO	NO
Repeat dialing	YES	YES	NO	NO
List dialing	YES	NO	NO	NO
Maximum baud rate	57,600	9,600	1,200	9,600
"Interactive" script files	YES	YES	NO	YES
Programmable keys	40	10	10	***
Modem-7 batch transfers	YES	NO	NO	NO
CompuServe transfer protocol	YES	NO	NO	NO
Price (suggested retail)	$60	$195	$35	$695

*** Symphony supports keyboard "macros."

MEX and MEX-PC are trademarks of NightOwl Software, Inc. ● CROSSTALK is a registered trademark of Micro Stuf ● SYMPHONY is a trademark of Lotus Corporation ● PC-TALK is a trademark of Headlands Press, Inc.

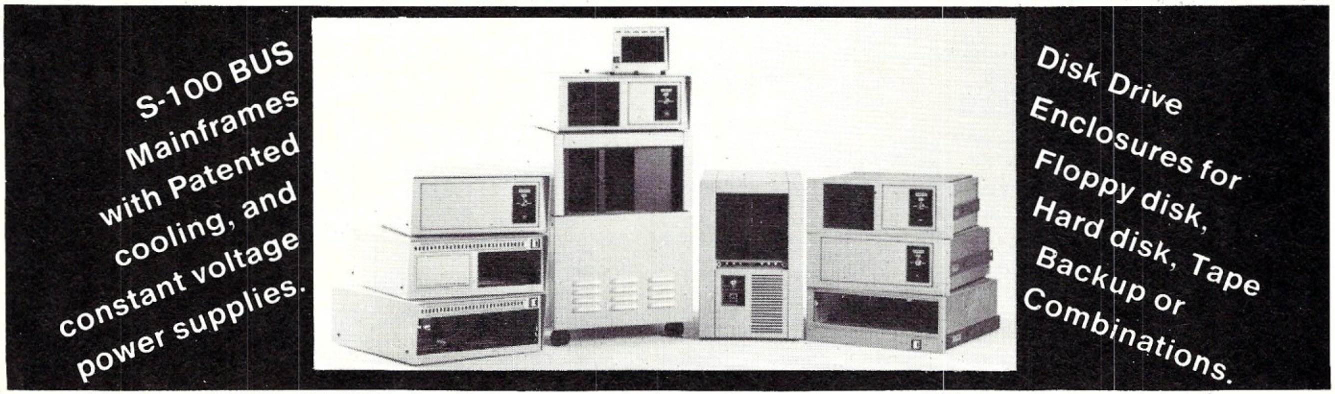
NightOwl Software Inc., Rt. 1, Box 7, Ft. Atkinson, WI., 53538

MEX PACK....................$99.95

As a special introduction to our remote operating module and terminal emulation system, we're offering an all-in-one package for a limited time only. **MEX-PACK** is a complete communications package that includes: **MEX-PC** (version 1.50); **REO** remote operating module; **TEM** intelligent terminal emulation module; and a six-month subscription to the **NightOwl Connection**. Purchased individually, the items in this package would cost you $144.85 — a savings of almost $45!

Give us a toll-free call at
1-800-NITEOWL
In Wisconsin, 1-414-563-4013

NightOwl
software, inc.


S-100 BUS Mainframes with Patented cooling, and constant voltage power supplies.
Disk Drive Enclosures for Floppy disk, Hard disk, Tape Backup or Combinations.

PARA DYNAMICS builds a variety of modern, efficient, trouble-free expandable housing systems for most S-100 BUS configurations. Whether a rack mount, desk top, or stand alone, our patented super-efficient heat dissipation system can end your board-level failures due to high temperatures. Please call today for full details. **(602) 991-1600**

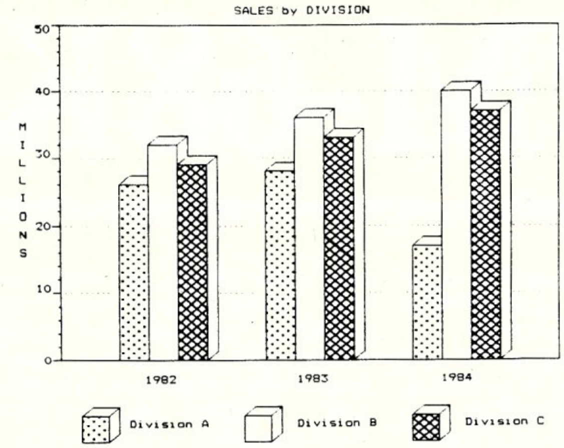
PARA DYNAMICS CORPORATION ● 7895 EAST ACOMA ● SCOTTSDALE, AZ 85260

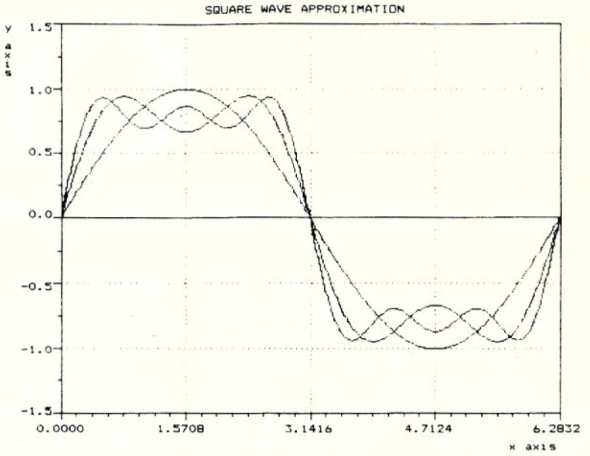
Introducing GRAF 3.0

the complete **BUSINESS** and **SCIENTIFIC** printer graphics program

$49^95 CP/M-80

$69^95 MS-DOS / PC-DOS



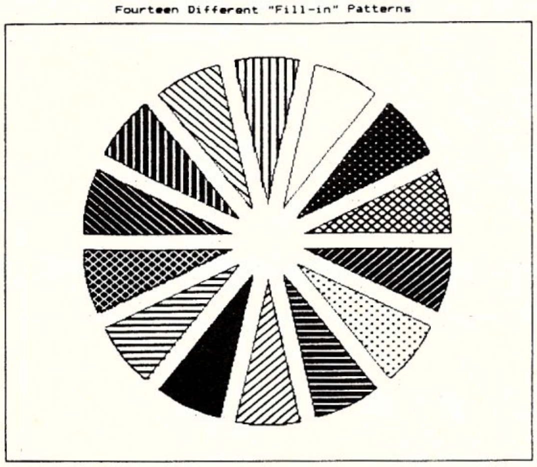


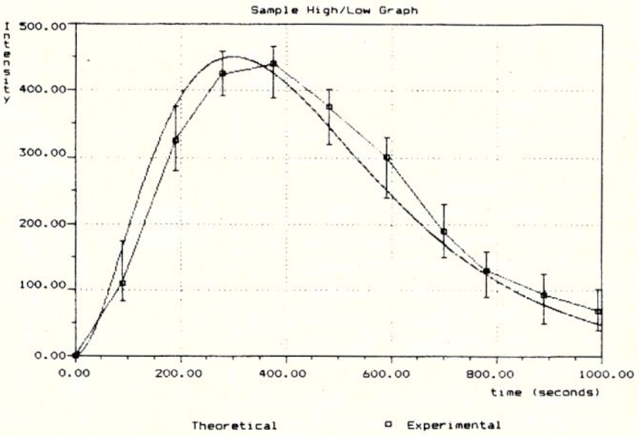
BUSINESS APPLICATIONS

- display floating point data directly from spreadsheets, data bases, and word processors (or the keyboard) in a wide variety of bar, pie, line, and scatter plots
- plot and group up to 6 different variables on a single graph, distinguished by up to 14 different "fill-in" patterns and 8 different point-plotting symbols
- menu driven operation supporting automatic graph scaling, labeling, and legend creation
- program default values may be set once and for all

SCIENTIFIC APPLICATIONS

- simple interface allows plotting floating-point data obtained from all popular programming languages
- plot any number of curves (e.g. experimental data vs. theoretical values) on the same graph, choosing from 8 different plotting symbols.
- automatically created legends distinquish variables
- add up to 5 different-density grid lines, and choose from a wide variety of numerical labeling options
- high/low graphs are supported directly





TERMS: We ship via first class mail. The above prices include $5.00 for s/h. (Orders outside USA require additional $5.00 for postage. NJ residents add 6% tax.) When ordering you MUST state your computer and printer make and model. We support MS-DOS (PC-DOS) version 2.0 or later on computers with at least 192k RAM, and CP/M-80 version 2.2 or later on Z80 computers (other than Apple) supporting a TPA of at least 54k (requires 64k of RAM). Most soft-sector disk formats are available. (If you can read several formats, please send us a list.) GRAF 3.0 works with any printer fully compatible with one of the following: Epson FX, RX, LX, MX (with GRAFTRAX), or LQ-1500; C. Itoh Prowriter; NEC 8023A, Star Micronics Gemini 10X, 15X, SG-10, SG-15, IBM Graphics Printer, Okidata 192, and earlier Okidata models equipped with the "IBM Plug 'n' Play" chips. (If you have an Okidata printer, other than the 192, the Plug 'n' Play chips are required!)

GRAF 2.0 Update Policy: Returning your original GRAF 2.0 disk to MSC entitles you to $20.00 off the above prices.

MSC Microcomputer Systems Consultants

27 Forest Avenue Port Jefferson Station New York 11776-1820

CP/M DRI MS-DOS MicroSoft

Bringing Up CP/M-68K

by J.L. Calaway & Ben Hill, Jr.

A few months ago, we were developing an idea for a new product. As the investigation proceeded, it became apparent that the real-time processing requirements would exceed the capabilities of an 8-bit processor by a wide margin. It was time to look into the world of 16 bits.

We currently manufacture equipment based on the S-100 bus and it was decided not to change buses in midstream. We knew we needed speed and most of the 16-bit processors seemed to have plenty. Programming ease and flexibility loomed large in our decision, as well as hardware multiply and divide capability along with plenty of room for expansion. These basic requirements can be met by several 16-bit processors, but the Motorola 68000 was given the edge for its extended, linear addressing facility, along with its programming similarity to the DEC PDP-11, a minicomputer with which we have had past experience.

With the choice of processor settled, the next order of business was to find some compatible hardware and software. Since CompuPro markets an S-100 68000 processor card and we have used a number of their boards, we felt this would solve the hardware problem nicely. For software, we decided to go with Digital Research's CP/M-68K.

Anxious to press on after the delivery of the hardware and software, we now devoted our attention toward finding a Z80/68000 cross-assembler to work with our existing Z80 computer. Avocet Systems and Quelo have such software. We opted for the Quelo version, but the Avocet cross-assembler should work as well.

GETTING THE SYSTEM TOGETHER

A swift appraisal of the Digital Research documentation disclosed that the BIOS listing required the use of a Tarbell double-density disk-controller! Oh, no time like the present to replace our old, single-density model. A panic call was placed to our dealer (pusher?), Patio Computer, in Los Angeles. Fortunately for us, they rounded up a controller in record time. Yea Patio! Now we could begin work in earnest.

A thorough reading of all the documentation was the next item of business. The new CP/M-68K manuals are several light years ahead of the old, turgid, CP/M-80 manuals. Its a good thing, too, because it turned out that we would get no additional help from Digital Research. We tried to contact them several times during the painful (and expensive) start-up process. Each time, the customer service desk promised to have someone call back on the problem, but nobody ever did.

We used the sample assembly language, BIOS listing in the the CP/M-68K manual as a guide. Unfortunately, while the mnemonics and the logic in the listings are correct, in our copy of the manual, some of the assembled op-codes were incorrect. Things like this are not especially helpful when implementing something new, complex and unfamiliar. However, things could be worse - and they quickly got that way. The operating manual provided for the 68000 CPU was typical CompuPro documentation - exasperatingly brief. At one point, in response to our call for technical assistance, we were informed by CompuPro that they would not give us any assistance. We would only be able to get the information secondhand, from the distributor! The distributor didn't know the answer, of course. In contrast, we are happy to report, the people at Quelo not only

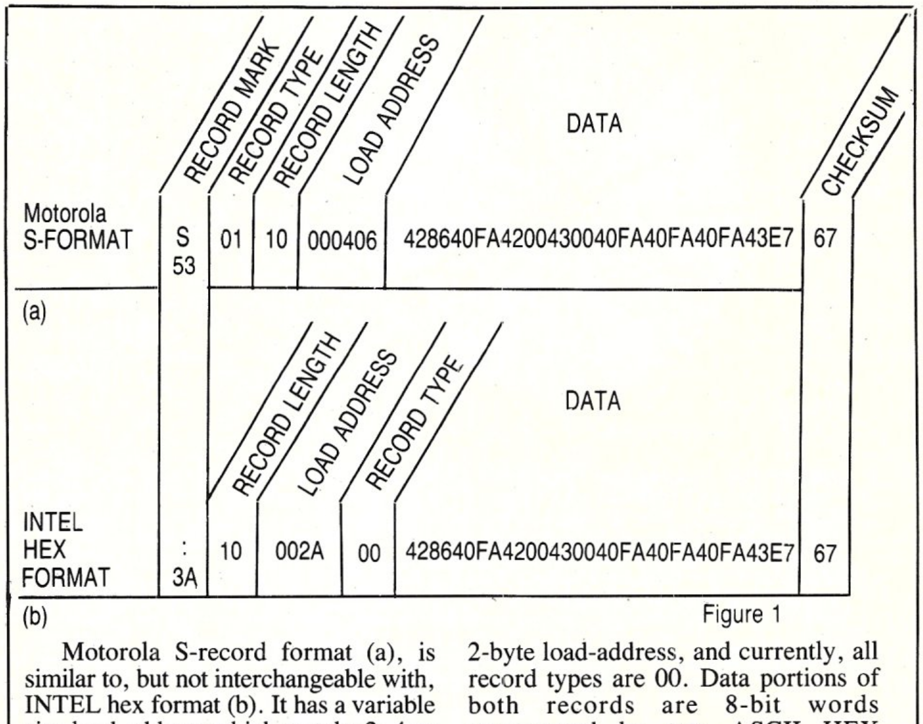


Figure 1

Motorola S-record format (a), is similar to, but not interchangeable with, INTEL hex format (b). It has a variable size load-address which may be 2, 4 or 6 bytes depending on data type. The type is a hex number (0 - 9), depending on whether record is for file housekeeping, actual program data or an end-of-file record. INTEL format has a fixed, 2-byte load-address, and currently, all record types are 00. Data portions of both records are 8-bit words represented by two ASCII HEX characters. The checksum is negative sum of all the bytes after the record mark. Thus, ignoring carries, the sum of all the bytes plus the checksum equals zero.

answered several questions about their cross-assembler's operation by phone, but have twice updated the assembler, sending along a new disk at no cost, each time. Yea Quelo!

Further reading of the CP/M documentation disclosed the fact that CP/M-68K is configured to run with the Motorola EXORmacs development system for the 68000. Since we didn't have one of these systems, another approach had to be used. CP/M for the 68000 is similar in concept to the old 8080 CP/M. You have the CCP (Command Control Processor) that handles commands from the system terminal, the BDOS (Basic Disk Operating System) that processes all the disk activity, and the BIOS (Basic Input/Output System) that handles all the interfacing with the printer, terminal, disks and other peripherals.

MODIFYING THE SOFTWARE

The CP/M68K system file (SR400.SYS) contains the CCP and the BDOS. This file is a Motorola S-record format file and contains no BIOS. Since the BIOS is hardware dependent, it must be written to fit the system in use. A listing of a sample BIOS is provided by Digital Research in an assembly language file called ERGBIOS.S. This listing can be used as a starting point for creating your own. The Motorola S-file (figure 1a), is an all ASCII file. Everything in the file is in hex code. It serves the same purpose as the familiar INTEL hex format (figure 1b), and allowed us to read the file data.

It was obvious at this point that we would have to do a bit of work on the software to accomplish the following:

- Modify the CP/M BIOS to work with the hardware and cross- assembler.
- Work out a method of downloading the HEX S-files from the 8'' standard CP/M release disk into the 68000's system.
- Provide a method of displaying and changing the 68000's memory.
- Provide a method of jumping to and executing code stored in the 68000 memory to check out the BIOS modifications and start the system running.

Modifying the BIOS was not too difficult. We had a cross-assembler and a good bit of experience with CP/M BIOS requirements from earlier systems, so it was a matter of copying the CP/M supplied sample into our Z-80 system in order to modify some of the lines of code to work with our 68000 hardware. We did find that some of the listed code in the sample BIOS wouldn't assemble with the Quelo cross-assembler, but this was simply a syntax problem and a minor wording change here and there readily converted the code so it would assemble properly.

A few Input/Output addresses required changing in order to resolve the difference between the way the CP/M-68K BIOS and the CompuPro board sees I/O addresses. The Godbout CPU uses the top 64K of memory for I/O addresses. There are no IN or OUT instructions in the 68000. Instead, it sees everything as being memory-mapped. For example, address of FF0001H is converted to port 1 by the CPU board. The top 64K address of 'FF' identifies it as a peripheral access address/instruction. The sample BIOS, on the other hand, expected to see an address of FFFF01H for the same port, so it was necessary to change the listed BIOS address to FF0001H. All the disk-access ports were changed as well as the console data and status ports.

Another BIOS change necessary was an adjustment in the step-rate value loaded into the disk controller board. The disk I/O board uses the bus clock for timing. If this clock is too fast, the disk stepping motor won't be able to keep up. The step-rate is a software-loaded variable that may be changed to get proper stepping-motor action. The Tarbell disk-controller manual lists the byte that has to be changed in the BIOS to adjust the rate.

The final change was to insert the address of the Command Control Processor (_ccp), into the BIOS. This is the address location the system will jump to on a warm-boot command. The BIOS label is 'WBOOT:', and the required _ccp address is supplied with the CP/M-68K release notes. After the BIOS changes were thought to be correct, it was assembled to give us a hex S-file ready to be down-loaded into the 68000 system.

DEVELOPING A 68000 SYSTEM MONITOR

We needed a simple monitor that we could burn into a prom that would allow us to download, display, change and execute programs and data. We looked about for just such a program, but couldn't find it, so we decided to do our own. The results are shown in listing 1. This simple tool is an invaluable aid when struggling with programming at the machine level in an unfamiliar system. The listing is heavily commented and should prove interesting to those of you who haven't had a chance to see much 68000 assembly language software. In our development system, we had the convenience of two terminals. One for the Z-80 and the other dedicated to the 68000. If you haven't the luxury of two terminals, the alternate, shown in figure 2, works as well.

There are three commands in the monitor, S (Substitute), L (Load), G (Go). While there is virtually no error checking in the program, it does the job. Once the monitor has been burned in a PROM and inserted into the 68000 system it is ready for use.

To change the data at memory location 1000H (hex), for example, type:

```
S1000 <RETURN>
```

The display will show:

```
00001000 XX where XX is hex data at that location.
```

Enter the desired data change in hex, followed by a <RETURN>. The new data will be entered at the displayed memory location. For no change, just hit <RETURN>. To return to the monitor command mode, type a PERIOD (.). The monitor prompt, (also a PERIOD), will re-appear.

To start a program at location 1000H type:

```
G1000 <RETURN>
```

The program will jump to that location and execute. In essence, the 'G' command acts as a jump-to-subroutine. If the code at the jump location ends with a return instruction, control returns to the monitor and the prompt reappears. The 'G' command is great for checking things like altered BIOS routines. To download an S-format hex file, enter the 'L' command. This starts the 68000 monitor program running. The Z-80 BIOS is altered slightly so that PIP can call PUNCH (or any unused IOBYTE label) to output characters to the RS-232 port connected to the 68000 (figure 2). The monitor program loops while waiting for data to appear on this line. When it does, the monitor strips off the non-data portion of the incoming S-file, converts it to binary machine-code and loads it into consecutive memory locations. The command sequence is:

```
L <RETURN> on 68000 system.
PIP PUN:=ERGBIOS.HEX <RETURN> on Z-80 system.
```

This command sequence downloads the file ERGBIOS.HEX byte-by-byte into the 68000. At the conclusion of the loading process, control returns to the monitor as indicated by the monitor prompt.

BIOS TEST PROGRAM

With our monitor working we tested our altered BIOS. To make that little job as simple as possible, we wrote a short program called BTEST (see listing 2) to give us some test data for the BIOS. This program allows the loading of data in registers D0, D1 or D2, jumping to a segment of code in the BIOS and returning with the data stored in a memory location labelled 'RESULT' (00201CH). The process allows section-by-section testing and lets you know if the program is working correctly. This approach is better than trying to run a whole new BIOS and wondering where the problem is if it doesn't run.

To test the console keyboard input-routine, for example, load the altered BIOS and the BTEST program into the 68000 system. The CP/M read-console-character routine is Function 3, so using the SUBSTITUTE command of our monitor, load 03H into register D0, (location 002000H), and enter G2000 <RETURN>. The system starts running at address 002000 and loops, waiting for a typed character. When a key is pressed, the monitor prompt reappears as the system exits the loop. Using the SUBSTITUTE command again examine the contents of location RESULT (202CH). If all is well, the hex value of the pressed key should be found there. If it isn't, something in the CONIN section of the BIOS isn't quite right. In a similar manner, all the functions — at least the ones that return a value — can be tested. In our case, we were fortunate enough to get all the right answers with no further changes.

IN CONCLUSION

Now that we had the BIOS checked out and working, things got easier. Using our monitor, we downloaded the SR400.SYS file along with the BIOS hex file. One final patch remained to be made at this point. Since this was the first time everything has been put together, the BDOS section of the system did not know the address of the new BIOS. Using the SUBSTITUTE command of our monitor, we inserted the BIOS INIT (initialization) address, into the required location in the BDOS. The BDOS address for this entry was provided with the CP/M-68K documentation. In our case the address was 004F97H. After completing the patch, at long last, everything was ready. Since the start address of the SR400.SYS is 0400H, we typed in G400 and hit <RETURN>. The reward for all our effort was synthesized in a two-character message displayed on the terminal screen: A>, the CP/M prompt.

Testing from this point on was merely a matter of making certain all the normal CP/M functions work. PIP, DDT, ED, DIR, etc., all the standard functions should be operational. Most users will want to continue on to develop an auto-loading, boot-up CP/M-68K system. The CP/M documentation supplied is fairly explicit about the additional steps necessary to accomplish this, but if there is sufficient interest perhaps we'll do another article on the procedure.

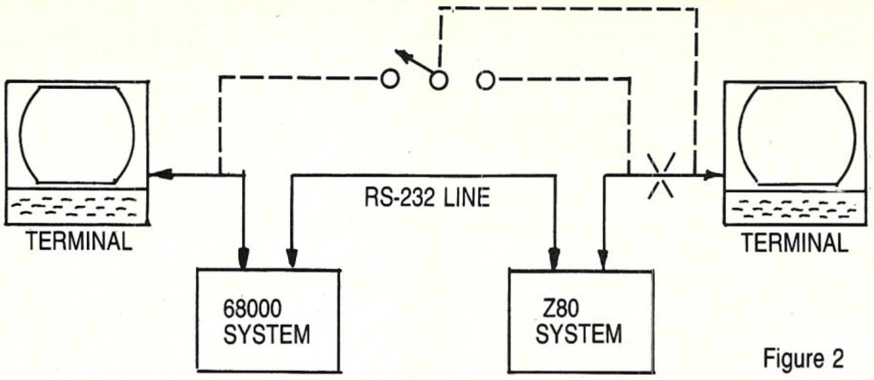


Figure 2

Z-80 to 68000 development system. each system has its own terminal and the two are connected through one RS-232 line operating at any convenient baud rate. Data and program output from Z-80 will be transmitted to 68000 via this line. A simple, PROM-based monitor running in 68000 system controls loading and handling of incoming data. An alternative connection (dashed lines), allows single terminal to do same job by using a 3-pole, double-throw switch across RS-232 line to connect terminal to either system.

Although this project was somewhat trying, don't let it discourage you. The information in this article should do much to ease the pain if you decide to follow in our footsteps. It is actually more difficult to describe than to do. We feel that the 68000 chip is going become a defacto standard in the world of 16/32 bit microprocessors and as such, will have a long and productive life. As more companies come on-line with 68000 related products and software, system implementation will get easier.

Jack Calaway is an engineering consultant specializing in hardware and software systems for the television broadcast and post-production industry. He has extensive experience with many types of computers, peripherals and videotape editing systems. His special interest in software is assembly languages and C, but he has an avid interest in other computer languages as well. When his schedule permits, his hobby is flying. His equipment ranges extensively from a PDP/11 to an IBM/PC with a 4 disk-drive S-100 system in between.

Ben Hill is president of Teleshows, Inc, a company that provides television production services to the broadcast industry. He has a special interest in computer business applications and uses Cobol extensively. His equipment is a three-disk, S-100 system. As a hobby, he is an avid photographer.

LISTING 2

```
************************************************************
*                                                          *
*                       BTEST                              *
*     A PROGRAM TO TEST THE BIOS CHANGES IN 68000 SYSTEM.  *
*            * WRITTEN BY J. L. CALAWAY, 03/17/85          *
*                                                          *
************************************************************

*
* ENTER THIS TEST PROGRAM WITH A CALL AFTER FIRST PATCHING THE
* REQUIRED VALUES IN REGISTERS D0, D1, AND D2 WITH THE MONITOR.
*

        ORG.L   $002000

        MOVE.L  #$00000000,D0           * MOVE PATCHED DATA
        MOVE.L  #$00000000,D1           * TO REGISTERS
        MOVE.L  #$00000000,D2

        TRAP    #3                      * READ CONSOLE CHAR IN

        MOVE.L  D0,RESULT               * RETURN FROM TRAP WITH
        RTS                             * DATA IN 'RESULT' REGISTER
                                        * D0; SAVE IN MEMORY LOCATION
RESULT: DC.L    $0

        END
```

LISTING 1

```
**************************************************************
*                                                            *
*   A MONITOR FOR THE MOTOROLA 68000 MICROPROCESSOR TO ENABLE *
* LOADING PROGRAMS, CHANGING DATA VALUES, AND BEGIN EXECUTION *
*   AT ANY MEMORY LOCATION.    68000 ASSEMBLY LANGUAGE LISTING *
*                                                            *
* Copyright (C) 1985, Jack L. Calaway.  All rights reserved.  *
*                                                            *
**************************************************************
*

* EQUATES
*
DATA    EQU.L   $0FF0010          * CONSOLE UART DATA
STATUS  EQU.L   $0FF0011          * CONSOLE UART STATUS
SDATA   EQU.L   $0FF0020          * DOWN LOAD UART DATA
SSTAT   EQU.L   $0FF0021          * DOWN LOAD UART STATUS
STACK   EQU.L   $0000300          * STACK BELOW CP/M-68K

        ORG     $0FD0000

*
* START UP CODE FOR GODBOUT CPU
*   ON HARDWARE RESET, PROMS ON CPU BOARD ARE GLOBAL
*

        DC.L    $8                * FIRST INSTRUCTION ADDRESS
        DC.L    STACK             * INITAL STACK VALUE
        JSR.L   INIT              * SWITCH PROMS TO REAL ADDRESS

*
* DO INITIALIZING TASKS
*

INIT:
        MOVEA.L #STACK,A7         * SET STACK POINTER

        BSR     CRLF              * OUTPUT A CR/LF PAIR
        MOVEQ   #$04D,D0          * THEN LOAD AND OUTPUT
        BSR     OUTCH             * THE MONITOR SIGN-ON
        MOVEQ   #$036,D0          * MESSAGE: M68K
        BSR     OUTCH
        MOVEQ   #$038,D0
        BSR     OUTCH
        MOVEQ   #$04B,D0
        BSR     OUTCH

*
* BEGIN PROCESSING COMMANDS.  THE PROGRAM
* RETURNS TO THIS POINT BETWEEN COMMANDS
*

START:
        MOVEA.L #STACK,A7         * RESET STACK POINTER
        BSR     CRLF              * NEXT LINE
        MOVEQ   #$02E,D0          * OUTPUT THE MONITOR
        BSR     OUTCH             * PROMPT CHARACTER:
        BSR     INCH              * GET A COMMAND
        CMPI.B  #$0D,D0
        BEQ.S   START             * IGNORE CARRIAGE RETURN
        BSR     OUTCH             * ECHO THE CHARACTER
        CMPI.B  #$053,D0          * 'S' = SUBSTITUTE
        BNE.S   START1            * NOT 'S', TRY NEXT COMMAND
        BSR     SUBST             * ON 'S' GO TO SUBSTITUTE
        BSR     CRLF              * ROUTINE.
        BRA.S   START             * BACK TO START

START1:
        CMPI.B  #$047,D0          * IF COMMAND IS A 'G',
        BNE.S   START2            * JUMP TO GOTO AND BEGIN
        BSR     GOTO              * EXECUTION
        BSR     CRLF
        BRA.S   START

START2:
        CMPI.B  #$04C,D0          * ON 'L' COMMAND, JUMP
        BNE.S   START             * TO THE LOAD ROUTINE
        BSR     LOAD
        BSR     CRLF
        BRA.S   START             * NONE OF THE ABOVE
**************************************************************
*                                                            *
*                           GOTO                             *
* THIS ROUTINE WILL JUMP TO INPUT ADDRESS AND BEGIN EXECUTION *
*                                                            *
**************************************************************

GOTO:
        BSR     INADR                        * GET THE ADDRESS
        JMP     (A0)                         * JUMP TO IT
```

```
**************************************************************
*                                                            *
*                            LOAD                            *
* THIS ROUTINE WILL DOWN LOAD AN S-FILE INTO THE 68000 MEMORY *
*                                                            *
**************************************************************

LOAD:
        BSR     CRLF                        * WE ARE HERE

*
* FIND THE S-RECORD HEADER
*
LOAD1:
        CLR.B   D4        * D4=LENGTH FLAG
        BSR     GCHR      * GET CHARACTER
        CMPI.B  #$053,D0  * IS IT AN 'S'?
        BNE.S   LOAD1     * IF NOT, TRY AGAIN
        BSR     GCHR
        CMPI.B  #$030,D0  * CHARACTER EQUALS ZERO
        BEQ.S   LOAD1     * TRY AGAIN IF SO
        CMPI.B  #$039,D0  * CHECK FOR FILE END, (9)
        BEQ.S   LOADX     * DONE IF SO
        CMPI.B  #$032,D0  * 4, OR 6 BYTES OF ADDRESS
        BEQ.S   LOAD3
        ADDQ.B  #$001,D4  * PUT 1 IN LENGTH FLAG

*
* GET BYTE COUNT
*
LOAD3:
        BSR     GHEX      * GET RECORD LENGTH
        MOVE.B  D0,D2     * SAVE IN COUNTER
        SUBI.B  #$03,D2   * ADJUST COUNT, AND TEST
        TST.B   D4        * AND TEST FOR LONG OR SHORT ADDRESS
        BNE.S   LOAD4     * SHORT
        SUBQ.B  #$001,D2  * LONG, SUBTRACT ONE MORE
*
* GET LOAD ADDRESS OF RECORD
*
LOAD4:
        BSR     GADR

*
* NOW GET FILE DATA
*
LOAD2:
        BSR     GHEX      * GET BYTE OF DATA
        MOVE.B  D0,(A0)+  * STORE IN MEMORY
        SUBQ.B  #$001,D2  * DECREMENT COUNT
        BNE.S   LOAD2     * CONTINUE
        BRA.S   LOAD1     * SKIP CHECKSUM

LOADX:
        RTS               * RETURN FOR NEW COMMAND

*
* GET TWO ASCII CHARACTERS FROM SERIAL PORT
* AND CONVERT TO ONE BINARY BYTE
* RETURNS THE DATA IN THE D0 REGISTER
* D1 IS CLOBBERED
*

GHEX:
        CLR.B   D1          * CLEAR TEMPORARY
        JSR     CAHEX       * GET HIGH NIBBLE
        ASL.B   #$4,D1      * SHIFT TEMPORARY LEFT
        OR.B    D0,D1       * 'OR' WITH D1
        JSR     CAHEX       * GET LOW NIBBLE
        ASL.B   #$4,D1      * POSITION HIGH NIBBLE
        OR      D1,D0       * COMBINE LOW AND HIGH
        ANDI.L  #$0000FF,D0
        RTS

*
* CONVERT ASCII TO HEX
*

CAHEX:
        BSR     GCHR        * GET CHARACTER
        SUBI.B  #$030,D0    * REMOVE ASCII
        CMPI.B  #$00A,D0    * A-F?
        BLT.S   CAHEX1      * NO
        SUBI.B  #$007,D0    * YES, SUBTRACT 7
CAHEX1:
        RTS

*
* GET LOAD ADDRESS FROM SERIAL PORT
* RETURNS WITH THE ADDRESS IN REGISTER A0
* WORKS JUST LIKE GHEX, BUT DOES 2 OR 3 BYTES
* RATHER THAN JUST ONE
*
```

```
GADR:
        CLR.L   D3                      * TEMPORARY=0
        BSR     GHEX
        ASL.L   #$08,D3                 * MAKE ROOM FOR NIBBLE
        OR.L    D0,D3
        BSR     GHEX
        ASL.L   #$08,D3
        OR.L    D0,D3
        TST.B   D4                      * LONG OR SHORT ADDRESS?
        BNE.S   GADRX                   * SHORT
        BSR     GHEX
        ASL.L   #$08,D3                 * LAST BYTE
        OR.L    D0,D3

GADRX:
        MOVE.L  D3,A0
        RTS

*
* GET SERIAL CHAR
*

GCHR:
        ANDI.B  #$02,SSTAT              * ANY WAITING?
        BEQ.S   GCHR                    * NO
        MOVE.B  SDATA,D0                * GET IT
        ANDI.B  #$07F,D0                * STRIP HIGH BIT
        RTS

**********************************************************
*                                                        *
*                      SUBSTITUTE                        *
* THIS ROUTINE WILL SUBSTITUTE DATA AT ANY MEMORY LOCATION *
*                                                        *
**********************************************************
SUBST:
        BSR     INADR                   * GET ADDRESS
        BSR     CRLF                    * CLEAN UP

SUBST1:
        BSR     CRLF                    * FOR NEW LINE
        BSR     PADR                    * PRINT ADDRESS
        MOVEQ   #$020,D0                * SPACE
        BSR     OUTCH
        MOVE.B  (A0),D0
        BSR     PHEX                    * PRINT CURRENT DATA
        MOVEQ   #$020,D0
        BSR     OUTCH                   * SPACE
        BSR     INCH                    * GET KEYBOARD CHAR
        CMPI.B  #$0D,D0                 * NEXT ADDRESS?
        BEQ.S   SUBST4                  * YES
        BSR     OUTCH                   * ECHO
        CMPI.B  #$02E,D0                * PERIOD EQUAL DONE?
        BEQ.S   SUBSTX                  * YES, EXIT

*
* INPUT NEW BYTE OF DATA
*
        CLR.B   D1                      * CLEAR REGISTER

*
* CONVERT 1 OR 2 ASCII CHARACTER
* TO BINARY BYTE
*

SUBST2:
        SUBI.B  #$030,D0                * STRIP ASCII
        CMPI.B  #$0A,D0
        BLT.S   SUBST3
        SUBI.B  #$07,D0
SUBST3:
        ANDI.L  #$0000000F,D0
        ASL.L   #$04,D1
        OR.L    D0,D1
        BSR     INCH                    * NEXT
        CMPI.B  #$0D,D0                 * DONE?
        BEQ.S   SUBST5                  * YES
        BSR     OUTCH                   * NO
        BRA.S   SUBST2
SUBST5:
        MOVE.B  D1,(A0)                 * STORE NEW DATA
SUBST4:
        ADDQ.L  #$001,A0                * NEXT MEMORY ADDRESS
        BRA.S   SUBST1
SUBSTX:
        RTS                             * FINI

*
* INPUT ADDRESS FROM CONSOLE
* EXIT WITH A0=ADDRESS
*
```

```
INADR:
        CLR.L   D1              * TEMPORAY = 0
INADR1:
        BSR.S   INCH            * GET ASCII CHARACTER
        CMPI.B  #$0D,D0         * CR=END?
        BEQ.S   INADRX          * YES
        BSR     OUTCH           * SHOW
        SUBI.B  #$030,D0        * STRIP ASCII
        CMPI.B  #$0A,D0         * 0-9
        BLT.S   INADR2
        SUBI.B  #$07,D0         * FOR A-F
INADR2:
        ANDI.L  #$0000000F,D0 * CLEAR
        ASL.L   #$04,D1         * POSITION HIGH NIBBLE
        OR.L    D0,D1           * MAKE ONE BYTE FROM TWO NIBBLES
        BRA.S   INADR1
INADRX:
        MOVE.L  D1,A0           * ADDRESS TO ADDRESS REGISTER
        RTS

*
* PRINT ADDRESS
* (A0)=DATA
*

PADR:
        MOVEQ   #$04,D2         * PRINT FOUR BYTES
        MOVE.L  A0,D1           * COPY ADDRESS
PADR1:
        ROL.L   #$08,D1         * SHIFT BYTES
        MOVE.B  D1,D0
        ANDI.B  #$0FF,D0
        BSR.S   PHEX            * PRINT BYTE
        SUBQ.B  #$001,D2        * COUNTER -1
        BNE.S   PADR1           * MORE TO PRINT
        RTS

*
* PRINT HEX
* D0=DATA
PHEX:
        MOVE.B  D0,-(SP)        * SAVE LOW NIBBLE
        ROR.B   #$04,D0         * PRINT HIGH NIBBLE FIRST
        BSR.S   PHEX1
        MOVE.B  (SP)+,D0        * GET LOW NIBBLE BACK
PHEX1:
        ANDI.B  #$0F,D0         * CLEAR LOW NIBBLE
        CMP.B   #$0A,D0         * A-F?
        BLT.S   PHEX2           * NO
        ADD.B   #$07,D0         * YES - CORRECT
PHEX2:
        ADD.B   #$030,D0        * MAKE ASCII
        BSR.S   OUTCH
        RTS

*
* CRLF TO CONSOLE
*

CRLF:
        MOVEQ   #$0D,D0         * CR
        BSR.S   OUTCH
        MOVEQ   #$0A,D0         * LF

*
* PRINT CHAR IN REG D0 TO CONSOLE
*

OUTCH:
        ANDI.B  #$01,STATUS     * BUSY?
        BEQ.S   OUTCH           * YES
        MOVE.B  D0,DATA         * SEND
        RTS

*
* INPUT CHAR FROM CONSOLE
*

INCH:
        ANDI.B  #$02,STATUS     * WAITING
        BEQ.S   INCH            * NO
        MOVE.B  DATA,D0         * GET IT
        ANDI.B  #$07F,D0        * CLEAR HIGH BIT
        RTS

***********************************************************
*                                                         *
*          THAT'S ALL FOLKS                               *
*                                                         *
***********************************************************

        END
```

Quality S-100 Products

Now Available! FULCRUM'S New...

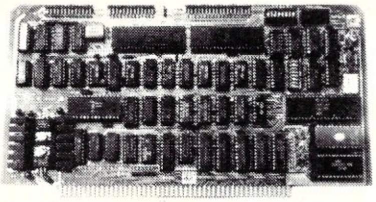
256K STATIC RAM

- 8/16 Bit Transfers
- 24 Bit Extended Addressing
- Battery Back-Up

$495

Designed For Speed & Flexibility.

MPUZ CPU

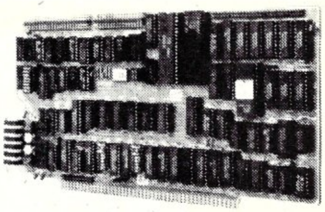


This NEW MPUZ CPU utilizes the Z-80 8MHz uPas a basis for its 8MHz CPU for S-100 systems, and has been carefully designed to meet the requirements of the IEEE - 696 standard. The quality and performance this CPU provides is rarely found in S-100 products, and you can see why... ***only $299***

- 4 or 8MHz clock rate
- Two RS-232 serial ports
- Centronics printer ports
- Real time clock with battery back-up
- Vectored interrups to any block location in memory
- Programmable timer
- ROM monitor
- Power on Jump
- On board wait states
- 2K of RAM space
- 24-bit extended addressing
- Latched Status
- Front panel compatibility
- MPM support

Best Value In Disk Controllers.

OMNIDISK



Now the FULCRUM OMNIDISK offers S-100 systems users a unique marriage of component compatability and technological innovation. These together produce features not found in any conventional disk controllers made today. See for yourself what tomorrow looks like... ***only $299***

- Simultaneous support of both 5¼" and 8" floppy disks and hard disks
- Complete 24 bit DMA
- Power on boot for 5¼" and 8" floppy and hard disks
- Power on boot PROM
- On board de-blocking to save RAM space over BIOS
- Interfaces with the WD 1001*hard disk controller
- Supports 13 devices simulatneously
- Full track buffer allows controller to recall entire track
- DMA'S at 10 MHz
- Supports MS DOS
- 10K on board buffer saves two K of TPA

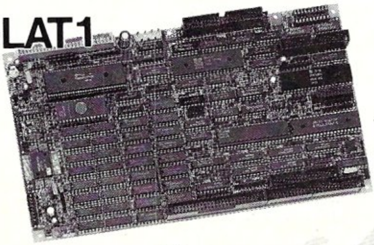
So before you buy another S-100 component, call or write for our *FREE* catalog. And see how your system can benefit from the FULCRUM difference.

*CP/M *2.2 configured for OMNIDISK $60. *Trade mark of Digital Research. FREE U.P.S. ground shipping on prepaid orders. Shipping is added to VISA, M/C, and C.O.D. orders. CA residents, please add sales tax.

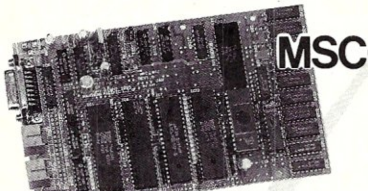
FULCRUM
COMPUTER PRODUCTS
707/433-0202 459 Allan Court, Healdsburg, CA 95448.

Also in FULCRUM'S Family: OMNIRAM 64K memory board. Serial I/O 2-2 & Video I/O Interface boards, Relay board, I-8080, 8015 and 8035 main frames with 21 slot mother boards, CPA front panel and A/D board.

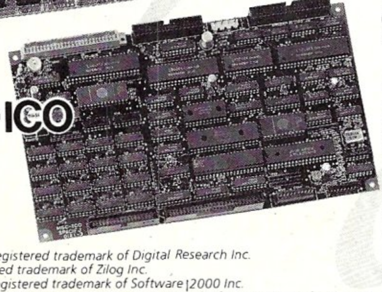
MSC·LAT1 **$649**



MSC·MTC



MSC·ICO **$499**



The Peak 68K8-CP

by Dave Hardy & Ken Jackson

Peak Electronics' 68K8-CP Coprocessor card offers the S-100 user a fast and painless way to upgrade to the world of high-speed, high-powered multi-processing. Using the MC68008 (8-bit bus version of the 68000) processor at 8 or 10MHz, the IEEE-696 compatible 68K8-CP can be installed in just about ANY S-100 system in less than five minutes, and will coexist with that system's current processor.

Software provided with the 68K8-CP allows the user to switch between the system's original processor and the 68000 with just a simple command, and full source for the board's firmware is available for those users who prefer to "roll their own" operating systems.

THE HARDWARE

The 68K8-CP is a standard single-height S-100 board, well made, solder-masked, silk-screened, and fully socketed. A spare DIP socket is available for those who wish to perform any circuit modifications. All options are either switch or jumper selectable, so no soldering is necessary for user configuration.

Naturally, the 68K8-CP is fully IEEE-696 compatible (after all, Peak's president, Don Pannell, is one of the authors of the IEEE-696 standard), including the ability to do full bus arbitration and act as an IEEE-696 temporary bus master.

Communication between the master processor and the 68K8-CP is performed via two S-100 ports that provide the ability for the two processors to exchange commands and status information. This is really the key to the operation of the 68K8-CP. Each processor can talk to the other via this simple I/O interface for commands and status, but the 68K8-CP actually takes over the S-100 bus (via its TMA ability) to perform data transfers, or use other S-100 bus resources. Because the IEEE-696 standard allows for up to 16 processors to exist on a single S-100 bus in this manner, it is actually possible for up to 16 68K8-CP's to live in a single S-100 frame.

An easy-to-install 68000 S-100 Co-processor card

Our 68K8-CP came "out of the box" with 8K of ROM and 128K of RAM on board, and with an 8MHz clock, although other configurations are available, as mentioned above. The documentation provided gives information on configuring the board for several different types of EPROMS and RAMS, and even gives information for reconfiguring the addressing PLA. Interestingly, in spite of the 8 or 10MHz clock speed, no wait states are needed for any of the on-board RAM or EPROM.

Features:

Works as a temporary bus master with full bus arbitration and TMA ability

MC68008 8 or 10MHz CPU

Fully IEEE-696 compatible

128K on-board RAM, expandable to 512K

8K on-board EPROM, expandable to 128K

Two on-board serial ports (requires daughter boards)

On-board parallel printer port and sense switch

Polled or interrupt-driven port communication with master processor (used only to S-100 I/O ports)

Eleven on-board interrupts

On-board timer

Easy to configure for most S-100 systems

The serial I/O, timer, and on-board interrupt handler are all contained in the same versatile IC, an MC 68681 DUART, which requires only the addition of RS-232 or current-loop driver (daughter) boards, if serial I/O from the 68K8-CP is desired.

The parallel printer port is actually just a simple latched output port with strobe and acknowledge lines. The input port at the same address serves as the sense switch input that is used at power-up to read configuration information into the 68K8-CP.

THE SOFTWARE

The 68K8-CP is currently available with software to allow the user to boot directly from CP/M 2.2 into CP/M 68K, although a proficient 68000 programmer could program the board to do most anything else, since it is basically an S-100 peripheral device. In fact, the manual gives several ideas for alternate uses of the 68K8-CP, including a numeric processor, and a high-speed data buffer. Although they were not ready at the time of this writing, 68K8-CP firmware is also available for use as a printer buffer and a RAM disk.

We were impressed with the implementation of CP/M-68K, especially due to the fact that the CP/M-68K system could be loaded and executed from CP/M 2.2 with just a single command. Eight single-density 8" standard CP/M format disks are provided with the 68K8-CP, along with a demo disk that contains some simple "C" programs and a benchmark program. (Because the 68K8-CP uses the host system's disk drivers after it boots up, it can read any disk that the host system can read, so transferring data files from CP/M-80 to CP/M 68K requires no additional work at all.)

Disk #1 is the CP/M 68K boot disk,

which contains all of the files necessary to run CP/M 68K, along with the files "CPM68K.COM" and "RETURN.68K," which are used to boot CP/M 68K from CP/M-80 and to return to CP/M-80 from CP/M 68K. All of the usual CP/M utilities are present with CP/M 68K, like STAT, DDT, ASM, etc., along with a few new ones. Not surprisingly, the BIOS is written in "C."

DOCUMENTATION

The 68K8-CP comes in a ring-binder containing four separate parts: the Introduction, the Hardware manual, the Software manual, and the Firmware manual.

The introduction section comes with two very important pieces of information. First, it lists the factory-shipped configuration and serial number of the 68K8-CP. Second, and most importantly, it contains a one-page mini-manual called "How to bring up your new 68K8-CP card in 5 minutes." We tried it, and it works, except that it actually takes only about 3 minutes if you have already removed the cover from your S-100 machine.

The Hardware manual is a complete little book about the 68K8-CP, including an overview of the entire board, precise explanations about how the board works and how to configure it, appendices, an index, and complete schematics. Also included are instructions on how to set up the 68K8-CP for use in older non-IEEE-696 S-100 machines.

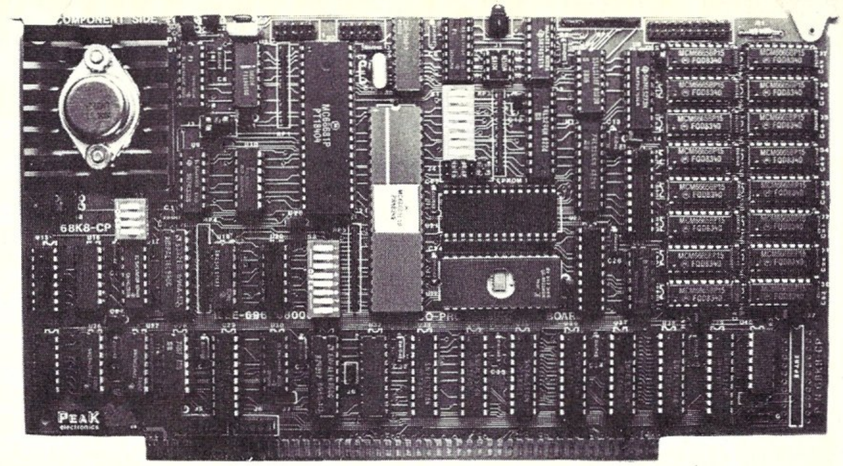
The Software manual is only about 10 pages long (any source listings provided are on the floppy disks), and briefly covers how to use the CPM68K.COM program and how to set up a system configuration file.

The Firmware manual was not yet available at the time of this review.

INSTALLATION

In most cases, installation of the 68K8-CP should take only a few minutes. We installed the board in more than a dozen different S-100 machines with virtually no trouble at all. As shipped from the factory, the board is set up to use S-100 I/O ports 0C0H and 0C1H for data and status exchange, but it is easily reconfigurable to any other port addresses by just changing a few switch settings and editing the system configuration file to indicate the new port addresses.

Our technical staff took great delight in plugging the 68K8-CP into every S-100 machine it could find, including everything from dinosaur IMSAI's to multi-processor TurboDos-based frames, and never had ANY problems at all. The 68K8-CP contains nine user configurable jumpers and three user configurable switches, although it can usually be installed using factory settings. The user configurable jumpers are:



J1 - Wait States for RAM, EPROM or I/O

J2 - Type of parallel I/O IC used (74LS374 or 74LS273)

J3 - Off-board RAM paging size

J4 - EPROM type (2764, 27128, 27256, 27512)

J5 & J6 - S-100 Vectored Interrupt number (VI0-VI7, NMI*, INT*)

J7 - Ground/CPU DSB for use with older front-panel machines

J8 - pSTVAL* transfer disable (for old non-IEEE-696 machines)

J9 - sM1 disable (for old non-IEEE-696 machines)

The user configurable switches are:

S1 - 8-bit sense switch (used to determine serial I/O speed at boot-up)

S2 - 4-Bit TMA priority number (needed for bus arbitration)

S3 - S-100 I/O address

TESTS

Our usual benchmark tests were of little use with the 68K8-CP, since they are meant primarily for testing other processors in stand-alone configurations. However, we did duplicate the manufacturer's tests (using the Sieve of Erotosthenes "C" language benchmark provided), and the results were:

68K8-CP @ 8MHz: 32 seconds
Z80 @ 4MHz: 141 seconds

These results are largely meaningless, except they do demonstrate the 68000's increased efficiency. The 68K8-CP could yield some very attractive time savings' in a system that could off-load number-crunching from a Z80 master processor.

CONCLUSION

The 68K8-CP is a fast and easy way to expand an existing S-100 system into a multiprocessing machine, or at least a painless way to bring up a powerful 68000-based system. Although the 68K8-CP is currently only available with CP/M 68K, other operating systems (like UNIX) are almost certainly going to be available in the future.

The price of the 68K8-CP with an 8MHz CPU and 128K of RAM is $995. The 10MHz version with 512K of RAM costs $1545, and various other configurations are available. CP/M 68K is available for $350, and source code for the 68K8-CP programs is available for an additional $50 with the purchase of a board.

For more information, contact:

Peak Electronics
Box 700112
San Jose, CA 95170

EXTRA!... VALUE and PERFORMANCE with Relocatable Z-80 Macro Assembler

FROM MITEK

It's a real bargain! Here's why:

- Only $49.95 plus shipping
- 8080 to Z-80 Source Code Converter
- Generates Microsoft compatible REL files or INTEL compatible hex files
- Compatible with Digital Research macro assemblers MAC & RMAC
- Generates Ditigal Research compatible SYM files
- Full Zilog mnemonics
- INCLUDE and MACLIB files
- Conditional assembly
- Phase/dephase
- Separate data, program, common and absolute program spaces
- Customize the Macro Assembler to your requirements with installation program
- Cross-reference generation
- Supports Hitachi HD64180 additional instructions
- Z-80 Linker and Library Manager for Microsoft compatible REL files available as a total package with Macro Assembler for only $95.00 plus shipping; manual only is $15

NEW for Turbo Pascal Users

Mitek's Relocatable Z-80 Macro Assembler will also:

- Generate Turbo Pascal in-line machine code include files
- Include files provide Turbo Pascal compatible machine code with assembly language mnemonics as comments

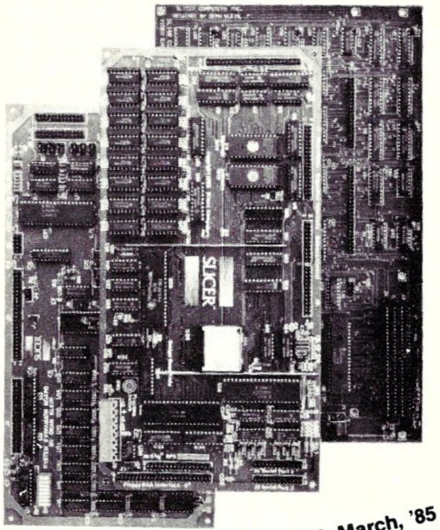
TO ORDER, CALL TOLL FREE: 1-800-367-5134, ext. 804
For information or technical assistance: 1-808-623-6361

Specify desired 5¼" or 8" format. Personal check, cashier's check, money order, VISA, MC, or COD welcomed. Include $5 for postage and handling.

Z-80 is a trademark of Zilog, Inc. MAC, RMAC, and ZSID are trademarks of Digital Research, Inc. Turbo Pascal is a trademark of Borland International, Inc.

MITEK
P.O. Box 2151
Honolulu, HI 96805

SLICER—THE SYSTEM THAT GROWS TO FIT YOUR NEEDS



see BYTE, March, '85 page 193

THE SLICER
Real 16 Bit Power on a Single Board—Featuring the Intel 80186 (C) Step CPU

- Complete 8 MHz 16-bit microprocessor on a 6″ × 12″ board
- 256K RAM, plus up to 64K EPROM
- SASI port for hard disk controller
- Two full function RS232C serial ports with individually programmed transmission rates—50 to 38.4K baud
- Software compatibility with the 8086 and 8088.
- 8K of EPROM contains drivers for peripherals, commands for hardware checkout and software testing
- Software supports most types and sizes of disk drives
- Source for monitor included on disk
- Bios supports Xebec 1410 and Western Digital WD 1002 SHD controller for hard disks

Fully assembled and tested only $895

THE SLICER 1-MByte EXPANSION BOARD NEW
For expanded memory, additional ports, and real time clock

- 1-MByte additional dynamic RAM
- 2 RS232C asychronous ports with baud rates to 38.4K for serial communication
- 2 additional serial ports for asynch (RS 232) or synch (Zilog 8530 SCC) communication
- Real Time Clock with battery backup
- Centronics type parallel printer port

Fully assembled and tested only $750

SLICER/1MByte EXPANSION COMBO
The Slicer (without RAM or RAM controllers) with Full 1-MByte Memory Expansion NEW
Kit form only $1295

THE SLICER PC EXPANSION BOARD
Gives your Slicer high performance video capability

- IBM compatible monochrome video
- Video memory provides 8 pages of text or special graphics capability
- 2 IBM type card slots for color video, I/O expansion, etc.
- IBM type keyboard port

Fully assembled and tested only $600

All boards available in kit forms

The SLICER Bulletin Board (300/1200 baud) 612/788-5909

Runs MS DOS generic software; PC DOS program operation not guaranteed

Also available: The µSlicer 188 $700; 8087 Math Co-Processor Bd. (call); 10 MB Hard Disk $700; W.D. 1002-SHD H.D.C. Bd. $200; Enclosures, Power Supply, and Support Hardware.

CP/M 86 $85, CCP/M $250 (Digital Research, Inc.); MS DOS $175 (Microsoft Corp.)

MasterCard, Visa, Check, Money Order, or C.O.D. Allow four weeks for delivery. Prices subject to change without notice.

QUALIFIED DEALER INQUIRIES INVITED

SLICER™

Slicer Computers Inc.
2543 Marshall St. N.E.
Minneapolis, MN 55418
612/788-9481
Telex 501357
SLICER UD

Program Interfacing To MS-DOS

by William G. Wong

Part IV - Basic File Access

UNIX-style file functions were first included in DOS V2 along with the hierarchical directory structure. Although these tools provide a more powerful and flexible environment the functions are actually easier to use than the CP/M-style file functions. For this reason, new applications tend to use the UNIX-style file functions exclusively.

I will first discuss what a subdirectory is for those unfamiliar with their use, then deal with file names and handles since they differ significantly from the CP/M-style file function parameters. The file functions themselves are covered after this.

Another unique feature of the UNIX-style file functions is the ability to redirect the data transfer of a particular file to another file including built-in files such as the console. This process will be covered in the section on file redirection.

Finally, a simple file copy program example will be presented to show how simple it is to use the UNIX-style file functions at the assembly language level.

SUBDIRECTORIES & DEVICES

One of the more advanced features found in UNIX is the hierarchical file system. It allows files to be organized in a multi-level tree structure. The basic hierarchical file structure has been incorporated into all versions of DOS except V1.

The top-level collection of files on a disk is called a directory. Files in the directory may be data files or a special file called a subdirectory. The subdirectory works and looks just like the main directory. Subdirectories may also contain data files or subdirectory files. Subdirectories always contain at least two special files named '.' and '..'. The first is the current subdirectory and the second is a reference to the parent directory. The latter allows access to files in higher level directories.

File names and subdirectory names still retain the CP/M-style syntax with an eight character file name and an optional three character file type. The file name and type are separated by a period. For example: NAME.TYP or A-NAME. A file name alone refers to a file in the current subdirectory. Like the current disk, the current subdirectory allows simple file names to be used without having to explicitly include drive or path information.

A path is a list of subdirectories. This is presented in text form as the file names of the subdirectories separated by a backslash, '\'. The last name in the list is the reference file. Some examples are:

```
\SOURCE\LIST.ASM
\SYSTEM\FORMAT.COM
\LEVEL.1\LEVEL.2\LEVEL.3\DATA
C:LEVEL.2\LEVEL.3\DATA
..\PARENT.DAT
\A-FILE
```

The path name can be viewed as directions on how a file is to be found and the current subdirectory information as preliminary search information. A current subdirectory is assocaited with each drive. The current subdirectory information is used only if the first character of the path is not a backslash, otherwise, the search is started from the main directory (on the current disk, if none is specified). The last example listed above (\A-FILE) shows how a file in the main directory is referenced while in a subdirectory.

Subdirectories could, in theory, be created ad infinitum. However, DOS does restrict a path name to 64 characters. This translates to 30 levels of subdirectories if single character subdirectory file names are used and 4 levels if full size file names are used at each level.

Special file names have been delegated to devices because the UNIX-style file functions allow devices to be treated as files. These names may be used in the file open function. The following table lists device names and descriptions.

Device	Description
CON	Console/Display
AUX	Same as CONM
COM1	First serial port
COM2	Second serial port
PRN	Same as LPT1
LPT1	First printer port
LPT2	Second printer port
NUL	Always end of file on input Throws away output

FILE NAMES & HANDLES

The CP/M style File Control Block (FCB) contained a file name which was parsed into a special format. Extending this mode of operation would be tedious at best. Instead, the UNIX-style file functions use zero terminated strings and a 16 bit value called a file handle. A program no longer needs to keep an FCB around for each open file.

A file name is stored as an ASCII string with the same syntax for a path name, as described in the previous section. It must not contain any spaces and is followed by (terminated) with a NULL character (a decimal 0). An assembly language constant would be defined as:

```
db '\LEVEL.1\LEVEL.2\DATA',0
```

A pointer to this type of string is normally passed in the DS:DX register pair with the UNIX-style file functions. The file open and create functions return a file handle as a result if a file exists or is created. This handle is then used in subsequent operations such as read, write, and close. The file name string area can changed after a file handle is acquired since it is not used by DOS unless, of course, the file name is used in a subsequent operation.

A set of predefined handles is provided by DOS. These files are open by default when a program starts and may be used with the read and write functions by using the handle value specified in the following table:

Handle	Description
0	Standard input device
1	Standard output device
2	Standard error device
3	Standard auxiliary device
4	Standard printer device

UNIX-STYLE FILE FUNCTIONS

The UNIX-style file functions return their result in the AX register. The carry bit is set if an error has occurred. Pointers to file names are passed as parameters in the DS:DX register pair. Additional parameters are passed the specified name. Otherwise, it truncates an existing file to a length of zero. Read and write access is always enabled for a newly created file.

```
MOV DX,OFFSET FNAME ; DS:DX := file name
MOV CX,0            ; CX := file attrib
MOV AH,3CH          ; AH := create opcode
INT 21H             ; create a file


FNAME: DB '\SUB-DIR\FNAME',0
```

The file open operation works in a similar fashion except the file attribute parameter is not required but the file access method is passed in the AL register. The file access method may be any of the following:

Access Method	Description
0	Read-only
1	Write-only
2	Read and write

Here is an example of the file open function.

```
MOV DX,OFFSET FNAME ; DS:DX := file name
MOV AX,3D02H        ; AH := open opcode
                    ; AL := r/w access
INT 21H             ; open a file
```

There are four operations which can be done using the file handle returned by the create or open functions. These are: closing a file, reading from the file, writing to the file, and changing the value of the file's read/write pointer.

The close operation takes an open file handle in the BX register and an opcode of 3E hex in the AH register. All buffers associated with the file are written to disk.

The parameters for read and write operations are:

Register	Description
AH	Opcode: 3FH = read 40H = write
BX	File handle
CX	Buffer size in bytes
DS:DX	Buffer address

The number of bytes transfered is returned in the AX register. This value will be less than or equal to the buffer size. A value less than the buffer size on a write indicates an error. On a read, it indicates the end of a file has been reached. On character devices it indicates a partial record. For example, a keyboard input file returns one line at a time.

All file data transfers occur from the current file position which is initially 0 for the start of the file. The position is adjusted by the number of bytes transfered by a read or write operation. Random access to data in a file is done using the file seek function. This function should be called before a read or write operation. The parameters and results for the seek function are:

Parameters

Register	Description
AH	42H, seek opcode
AL	Seek method 0 = from beginning of file 1 = from current position 2 = from end of file
BX	File handle
CX:DX	Offset in bytes (CX is the most significant part)

Results

Register	Description
DX:AX	New file position (DX is the most significant part)

The most commonly used operations are movement to the start or end of the file. It can also be used to move forward or backward from the current file position allowing access of the next or previous record. The function can also be used to determine the current file position by using method 1 with an offset of 0.

The remaining functions deal with maintenance of files contained within a directory. The first two functions are used to get the names of files within a directory. These functions used the Disk Transfer Area (DTA) used with the CP/M-style functions. The set DTA function must be used before the two UNIX-style file functions are used. The first function is used to find the initial file. The address of the file name string is passed in the DS:DX registers and the search file attribute is passed in the CX register. The latter is the same as the attribute search byte used in the corresponding CP/M-style file function. The file name string may contain wildcard characters (* and ?) in the file name portion. The DTA is filled with the following information if a matching file is found.

DTA Offset	Description
0-14H	Reserved by DOS
15H	File attribute
16H-17H	File time stamp
18H-1_H	File date stamp
1AH-1DH	File size (least significant byte first)
1EH-2BH	File name

The file name in the DTA is a zero terminated string containing the name of the first file which matches the path given as the parameter. It does not contain the path or drive names.

The subsequent file function is used to find additional file names which are in the directory selected by the initial search path. The opcode value is 4F hex and it requires no parameters. The results are the same as the initial function. The DTA used by the first function should be selected. Subsequent file names which match are placed into the DTA file name area. The file names should be used (copied) before doing subsequent search operations. Listing 1 shows a simple example of a directory search program which collects all the matching file names in the buffer FNBUF. The file names are zero terminated and the end of the list is marked by a zero length string.

Changing a file's attributes after it is created is done using the following code:

```
MOV DX,OFFSET FNAME ; DS:DX := string addr
MOV CX,0            ; CX    := attrib
MOV AL,1            ; AL    := set attrib
MOV AH,43H          ; AH    := set opcode
INT 21H             ; set file attrib
```

A file's current attribute can be returned in the CX register by changing the AL register to 0 instead of 1. The value in the CX register before the INT 21H is ignored. See the previous article for a description of the file attribute byte.

A read-only file cannot be deleted. Use the set file attribute function first to enable reading and writing to a file. Then use the following code to delete the file.

```
MOV DX,OFFSET FNAME ; DS:DX := string index
MOV AH,41H          ; AH := delete opcode
INT 21H             ; delete file
```

The file name may contain a drive and path name but not wildcard characters (* or ?). This function only works with data files. See the first part of this section to see how subdirectories are deleted.

Renaming a file is almost as easy as deleting one except that some additional rules must be followed. First, the strings containing the old name and the new name must have the same drive specification. Do not explicitly use drive A: in one name and drive B: in the other. Also, do not explicitly use drive A: in one name and specify no drive in the other when the current disk is not A:. Second, all subdirectories listed in the old and new file names must exist. No directory creation is performed.

Files may be moved from one subdirectory to another. This is the same as copying the file and deleting the source but it is much quicker. As mentioned before, files cannot be renamed across disks.

```
OLD: DB 'OLD.NAM', 0

NEW: DB 'NEW.NAM', 0


MOV DX,OFFSET OLD     ; DS:DX := old name
MOV AX,DS
MOV ES,AX
MOV DI,OFFSET NEW     ; ES:DI := new name
MOV AH,56H            ; AH := rename opcode
INT 21H               ; rename file
```

Finally, there is the matter of the file time and date stamps. DOS keeps track of these values and initializes them when the file is created. The values can be examined using the directory search functions or the values can be returned using the following code.

```
MOV DX,OFFSET FNAME;DS:DX := string index
MOV AH,57H          ;AH := stamp opcode
MOV AL,0            ;AL := get opcode
INT 21H             ;get time/date stamp
```

The time and date stamps are returned in the CX and DX registers respectively. These values are the same as described in the previous article except the bytes are reversed. That is, CH and DH contain the least significant portion of the stamp.

FILE REDIRECTION

One very useful operation which can be performed using the UNIX-style file functions,but not the CP/M-style file functions, is redirection. This allows a program to change where data is transferred even though a specific routine uses a predefined file. For example, the standard output device is normally attached to the console. Redirection allows this to be changed to the printer or a disk file.

Two functions are provided by DOS to support the redirection feature. One creates a duplicate file handle given a previously opened file handle. The other forces one file handle to refer to another.

The duplicate file handle function takes the open file handle in the BX register and returns a duplicate in the AX register assuming the additional file does not exceed the file open limit. Both handles may be closed independent of the other. However, the file read/write position is the same for both files. Moving the file position using one file changes it for the other file. The following is an example of the code used to make a duplicate file handle.

```
HANDLE: DW 0    ; place open handle here

MOV BX,HANDLE  ; BX := open file handle
MOV AH,45H     ; AH := duplicate opcode
INT 21H        ; AX := duplicate handle
```

Although this function is useful in creating a file which can be closed twice (once via each handle). It is most often used to keep a copy of a file so the duplicate handle can be used with the redirection facility.

The redirection file function requires two open file handles. The second file is closed and first file handle is duplicated such that the second file handle is the duplicate. The duplicate follows the same restrictions as duplicates created using the previous function. The following example shows how the standard output can be directed to the printer.

```
MOV BX,4        ; BX := printer handle
MOV AH,45H      ; AH := duplicate opcode
INT 21H         ; AX := duplicate handle
MOV BX,AX       ; BX := a printer handle
MOV CX,1        ; CX := standard output
MOV AH,46H      ; AH := redirect opcode
INT 21H         ; use printer for output
```

Redirection is not restricted to device files. Either file handle may refer to a device file or a disk file.

FILE COPY EXAMPLE

A simple file copy program is shown in listing 2. This uses constant file names and creates a new file with the default attribute settings. The error result is not specific, either the file copies or it does not. It uses the file create function so any existing destination file is removed.

More sophisticated error handling, programmable file names, and status output can be added using functions listed in this and previous articles.

SUMMARY

The UNIX-style file functions are an obvious improvement over the CP/M-style file functions in power and ease of use both from the programmers and users point of view. The UNIX-style file functions also provide redirection facilities which cannot be duplicated with the CP/M-style file functions.

Subdirectories can be used to organize a large disk into meaningful sections. The UNIX-style file handles and variable length data records also simplifies programming.

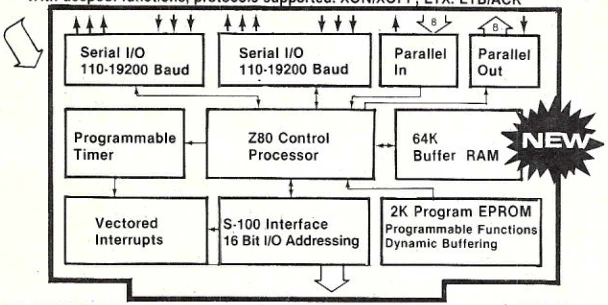
```
                LISTING 1: DIRECTORY SEARCH

SFNAME:  DB 'C:\SYSTEM\*.*', 0      ; search name

FNBUF:   DB 8096 DUP (?)            ; name buffer

; Setup loop parameters
; Assumes data is in the DS segment

     MOV DX,OFFSET SFNAME
     MOV AX,DS
     MOV ES,AX
     MOV DI,OFFSET FNBUF
     MOV CX,0

; DS:DX = search string address
; ES:DI = file name buffer
; CX    = search attribute

     PUSH ES           ; save file buffer index
     PUSH DI
      PUSH DS          ; save string index
      PUSH DX
       MOV AX,CS       ; set local DTA
       MOV DS,AX
       MOV DX,OFFSET SDTA
       MOV AH,1AH
       INT 21H
      POP DX
      POP DS           ; DS:DX := string index
      MOV AH,4EH       ; AH := search opcode
       INT 21H         ; search for first name
     POP DI
     POP ES            ; ES:DI := buffer index
LOOP:
     JC ENDLOOP        ; skip if no file found
     MOV SI,OFFSET SRESULT
CLOOP:
     LODSB             ; AL := character
     STOSB             ; add to string
     TEST AL,0         ; check for end of string
     JNZ CLOOP         ; loop until done
     PUSH ES
     PUSH DI           ; save buffer index
      MOV AH,4FH       ; AH := search next opcode
      INT 21H          ; search for next name
     POP DI
     POP ES            ; ES:DI := buffer address
     JMP LOOP          ; loop until all found

ENDLOOP:
     MOV AL,0          ; AL := terminator
     STOSB             ; add to buffer
     RET

SDTA:
     DB 21 DUP (0)   ; reserved area
SATTRIBUTE:

     DB 0            ; file attribute
STIME:
     DW 0            ; file time stamp
SDATE:
     DW 0            ; file date stamp
SSIZE:
     DW 0            ; file size
SSIZEL:
     DW 0            ; most significant part
SRESULT:
     DB 13 DUP (0)   ; file name
```

LISTING 2: FILE COPY PROGRAM

```
; Copy file program
; Open files

     MOV DX,OFFSET SOURCE       ; DS:DX := file name index
     MOV AH,3DH                 ; AH := open file opcode
     INT 21H                    ; AX := file handle
     MOV DX,OFFSET ESOURCE      ; DS:DX := error message
     JC PRINT                   ; skip if open failed
     PUSH AX                    ; save source handle
      MOV DX,OFFSET DEST        ; DS:DX := file name index
      MOV AH,3CH                ; AH := create file opcode
      INT 21H                   ; AX := file handle
     POP BX                     ; BX := source handle
     JC ECREATE                 ; skip if create failed
     MOV CX,AX                  ; CX := destination handle
LOOP:
     PUSH BX                    ; save handles
     PUSH CX
      MOV DX,OFFSET BUFFER      ; DS:DX := buffer address
      MOV CX,512                ; CX := buffer size
      MOV AH,3FH                ; AH := read opcode
      INT 21H                   ; read from source file
     POP CX                     ; restore handles
     POP BX
     JC BAD READ                ; skip if error
     TEST AX,0                  ; zero := end of file
     JZ DONE                    ; skip if end of file
     PUSH AX                    ; save buffer size
     PUSH BX                    ; save handles
     PUSH CX
      MOV BX,CX                 ; BX := destination handle
      MOV DX,OFFSET BUFFER      ; DS:DX := buffer address
      MOV CX,AX                 ; CX := bytes to write
      MOV AH,40H                ; AH := write opcode
      INT 21H                   ; write buffer to file
     POP CX                     ; restore handles
     POP BX
     POP DX                     ; DX := read size
     JC BAD WRITE               ; skip if error
     CMP AX,DX                  ; zero := write successful
     JNE BAD WRITE              ; skip if disk full
     JMP LOOP                   ; loop until file copied
; Close files and show transfer complete
DONE:
     CALL CLOSE                 ; close files
     MOV DX,OFFSET MDONE        ; DS:DX := done message
PRINT:                          ; DS:DX = string index
     MOV AH,9                   ; AH := print string opcode
     INT 21H                    ; print string
     RET
; Close source (BX) and destination (CX) files
CLOSE:
     PUSH CX                    ; save destination handle
      MOV AH,3EH                ; AH := close opcode
      INT 21H                   ; close source file
     POP BX                     ; BX := destination handle
     MOV AH,3EH                 ; AH := close opcode
     INT 21H                    ; close destination handle
     RET
; Handle destination create error
ECREATE:                        ; BX  = source handle
     MOV AH,3EH                 ; AH := close opcode
     INT 21H                    ; close source file
     MOV DX,EDEST               ; DS:DX := error message
     JMP PRINT                  ; print error message

; Handle error while reading source file

BAD_READ:
     CALL CLOSE                 ; close both files
     MOV DX,OFFSET EREAD        ; DS:DX := error message
     JMP PRINT                  ; print error and exit

; Handle error while writing source file

BAD_WRITE
     CALL CLOSE                 ; close both files
     MOV DX,OFFSET EWRITE       ; DS:DX := error message
     JMP PRINT                  ; print error and exit

; File names (In data segment)

SOURCE:    DB 'SOURCE',0
DEST:      DB 'DEST',0

; Data buffer for copy operation

BUFFER:    DB 512 DUP (?)

; Error and status messages

ESOURCE:   DB 'Cannot open source file.$'
EDEST:     DB 'Cannot create destination file.$'
EREAD:     DB 'Error reading source file.$'
EWRITE:    DB 'Error writing destination file.$'

MDONE:     DB 'Copy done.$'
```

BUFFERED I/O BOARD Introductory Price * $59.95

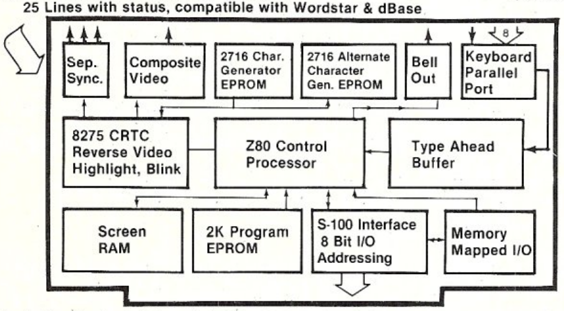
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK



8
8
Serial I/O 110-19200 Baud
Serial I/O 110-19200 Baud
Parallel In
Parallel Out
Programmable Timer
Z80 Control Processor
64K Buffer RAM
NEW
Vectored Interrupts
S-100 Interface 16 Bit I/O Addressing
2K Program EPROM Programmable Functions Dynamic Buffering

80 CHARACTER VIDEO BOARD * $49.50

25 Lines with status, compatible with Wordstar & dBase



8
Sep. Sync.
Composite Video
2716 Char. Generator EPROM
2716 Alternate Character Gen. EPROM
Bell Out
Keyboard Parallel Port
8275 CRTC Reverse Video Highlight, Blink
Z80 Control Processor
Type Ahead Buffer
Screen RAM
2K Program EPROM
S-100 Interface 8 Bit I/O Addressing
Memory Mapped I/O

Includes Bareboard, Heatsink & Documentation Call or write for more information.

SPC

Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL 60195
(312) 359-7337

VISA

MasterCard

OEM dealer pricing available, $3.00 S/H, IL. Res. add 7% tax

dBase™ - of Ashton - Tate Corp. — Wordstar™ - of Micropro Int'l. Corp

NEW PRODUCTS

SERIAL ADAPTER FOR THE VDB-A2-

- Plugs into keyboard socket
- Direct connect to Keytronic (IBM) serial ASCII keyboard
- Internal speaker for BEEPER
- Tx port for non VDB-A2 use
- Hardware programmed Rx & Tx
- Dual baud-rate generator
- Bare-board OR assembled

DOUBLE-HEIGHT S-100 PROTOTYPING-

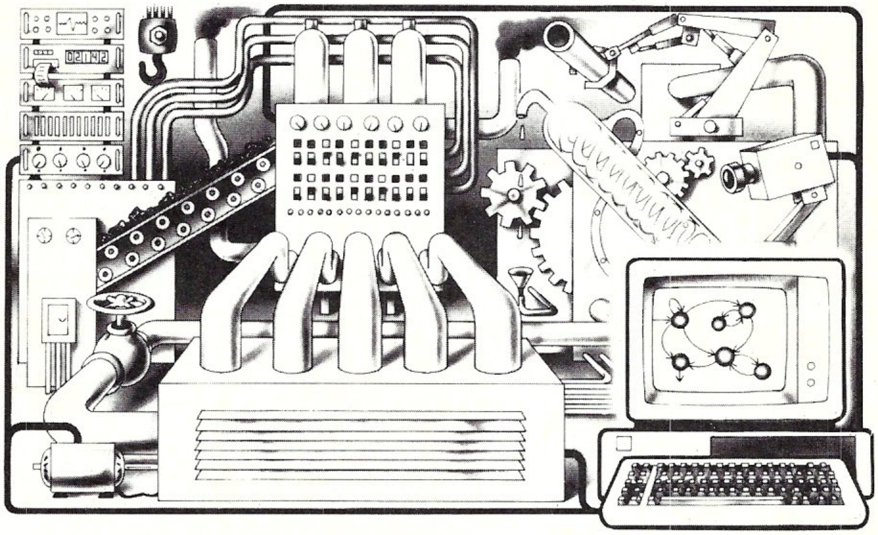
- Oversized heatsinks for HI-CURRENT reg's: +5, +/−12 V.
- Use with wire-wrap sockets OR direct solder connections
- GOLD plated edge connector
- NO PLATING CUTS required
- NO wasted address decoding

WATCH OUR ADS FOR OTHER PRODUCTS COMING SOON

- MODEM ADAPTER FOR THE BIO-1
- 256K RAM SOFTWARE FOR BIO-1
- HIGH FEATURE DISC CONTROLLER
- Z80 C.P.U. / MEMORY MANAGER
- 1 MEGABYTE DRAM BOARD

Specifications subject to change without notice

Csharp Realtime Toolkit



Realtime on MSDOS? Csharp can do it! Get the tools without operating system overhead. Cut development time with C source code for realtime data acquisition and control. **Csharp** includes: graphics, event handling, procedure scheduling, state system control, and interrupt handling. Processor, device, and operating system independent. **Csharp** runs standalone or with: MSDOS, PCDOS, or RT11. **Csharp** runs on: PDP-11 and IBM PC. **Csharp** includes drivers for Hercules and IBM graphics boards, Data Translation and Metrabyte IO boards, real time clock, and more. Inquire for Victor 9000, Unix, and other systems. Price: $600

SYSTEMS GUILD

Systems Guild, Inc., P.O. Box 1085, Cambridge, MA 02142
(617) 451-8479

SOMETIMES GOOD IS JUST NOT GOOD ENOUGH.

Frankly, there's a lot of good development software on the market today. But when your programming project is on the line, good isn't good enough. You need the best.

Because we're an independent publisher and distributor, we can pick and choose. We've been selling to microcomputer programmers since 1976, so we know quality when we see it. Lifeboat offers each of the following popular software development tools and more... When we offer a product, you know it is the very best in its class.

MARK WILLIAMS C PROGRAMMING SYSTEM™ New!

When programming, it takes an incredible amount of time just to get the code written. Then comes debugging, which can take even longer. *Mark Williams C Programming System* combines a C compiler with an easy-to-use debugger to make C programming simpler.

One-step compilation produces an executable file with a single command.

Options include object file in MC- or MS-DOS formats or ROMable code.

Full UNIX capabilities, with large library of more than 100 routines.

Debugging facilities allow setting tracepoints at source level; windowing enables source to be viewed while debugging; optional assembly listing.

MWC compilers are available for 8088/8086, 80186, 80286, 68000, Z8000, and PDP-II. Vax cross compilers are available.

RUN/C LITE™ New!

Learn C the natural way with the *RUN/C Lite* introductory package. Over 100 sample programs are included on the disk to coincide with the thorough and professionally written C reference manual. *RUN/C Lite*'s user interface is similar to that of the BASIC interpreter with easy familiar commands such as LOAD, LIST, SAVE, EDIT, and RUN. Debug with TRON, TRACE, and PRON. Try your hand at graphics with easy-to-learn functions like PUT, GET, CIRCLE, LINE, SCREEN, and DRAW. Lighten up on the time it takes to learn C with *RUN/C Lite.*

RUN/C PROFESSIONAL™ New!

All of RUN/C Lite's features, plus powerful enhancements for program development. LOAD and RUN, your favorite library functions (GSS Drivers®, Multi-Halo™, Panel™, Lattice Window,™ etc.) from within your RUN/C™ interpreted code. Polish program modules before compiling. New source code debugging facilities allow you to set multiple breakpoints, single step through a program, and view and change variables during execution.

BetterBASIC™ New!

BetterBASIC is just what it sounds like. Better than the BASIC you got with your PC. What's better about it? Just about everything.

Fast. Programs run fast. Each line of code is compiled as it is entered into memory.
Big. Break the 64K barrier. Use the full memory of your PC to write big programs.
Powerful. Windows and graphics, DOS and BIOS calls, chaining, overlays, local and global variables, recursions and more.
Interactive. Responds to user commands in immediate mode.
Structured. Write structured programs using functions and procedures.
Modular. Create re-usable "library" modules for more efficient programming. Easy interface to assembly language.

BetterBASIC Runtime System,™ 8087/80287 Math Module,™ and Btrieve Interface™ also available.

Lattice® C Compiler

The best selling C compiler!

- *Lattice C* produces fast and compact code.
- The widest selection of C support tools is available to the *Lattice C* user.
- Choose from four memory models for optimum efficiency.
- Full compatibility with the C implementation established by Kernighan and Ritchie assures portability.
- Superbly written documentation augments the lengthy list of reasons why software developers continue to choose *Lattice C,* the industry standard.

GSS® PRODUCTS New!

Industry leading graphics software based on a commitment to standardization. Device independence and source code portability are assured with these powerful software tools.

GSS DRIVERS®—Provide a standard VDI interface between your PC and a wide variety of peripherals.

GSS® Bindings—The language binding provides an interface between the language source and the VDI. Bindings to Lattice C and the following MicroSoft languages are available: Fortran, Assembler, Pascal, Basic Compiler.

GSS-TOOLKITS®:

Kernel System—Subroutine library provides quality graphics for computer aided design and engineering. This toolkit adheres to standard described in Level 2b of the ANSI Graphical Kernel System (GKS) specifications.

Plotting System—Device independent subroutine library of two dimensional plotting functions for creating programmer-specified charts efficiently and easily.

Metafile Interpreter—Reads any standard metafile and converts it to graphic output commands.

Lifeboat, The Source for Software with Full Support, and Lifeboat logo are trademarks of Lifeboat Associates.
BetterBasic, Runtime System, 8087/80287 Math Module and Btrieve Interface are trademarks of Summit Software Technology, Inc.
RUN/C Lite, RUN/C Professional and RUN/C–The C Interpreter are trademarks of Age of Reason Co.
C Programming System is a trademark of Mark Williams Co.
Lattice is a registered trademark of Lattice, Inc.
GSS, GSS-TOOLKIT, and GSS DRIVERS are registered trademarks of Graphic Software Systems, Inc.

For immediate delivery or more information,
call: 1-800-847-7078 In NY: 212-860-0300

or write: **LIFEBOAT™ ASSOCIATES**
1651 Third Avenue
New York, NY 10128

LIFEBOAT™

The Source for Software with Full Support™

The Scientific Computer User

by A.G.W.Cameron

Editor's Note

We expect "The Scientific User" to be a regular column in Micro/Systems Journal appearing in every other issue. Initial feedback from reader study we are conducting indicates to us that a very large percentage of our readers are scientists and engineers using computer systems in their work. Hence, we feel there will be a great deal of interest in this column. We would appreciate hearing reader reactions to the column. And, we aresure the author would also like to hear from readers.

A Look At Scientific Workstations

In discussions of computer workstations, the computer-related press has made the most of the business-oriented aspects of the uses of IBM PCs and similar machines. Spreadsheets and data bases and word processors. Bar graphs and pie charts. Links to mainframes. This is not the world of the scientist and the engineer. Nevertheless, the scientist and engineer have much to be thankful for in these developments. They have made the PC workstation affordable for other uses.

Technical people have been working with mainframes and minicomputers for quite a long time now, time-sharing number crunching and data analysis and whatnot. Recently very powerful resources have been put at the disposal of individuals, particularly for such applications as CAD-CAM. Some aspect of system sharing has usually remained, but names such as Apollo and Sun and Masscomp denote very capable workstations, costing typically $10,000 to $20,000 for each such station. I shall not be dealing with these, other than as prototypes toward which the IBM PC style of workstation can aspire.

I am a theoretical astrophysicist, and I have been interested for some time in transferring all of my scientific activities to workstations at a lower level of sophistication, the $2,000 to $5,000 range represented by IBM PCs and their clones and emulators. The performance of hardware and software that is particularly useful to me in this endeavor has been rapidly improving, and will continue to do so, although most of it is not nearly as well publicised as business-oriented items. This seems to be a useful time to take stock of where we are and where we are going.

PROCESSING SPEED

Most technical people want to crunch numbers in some form. A fairly commmon phenomenon recently has been the University departmental DEC VAX 11/780, which I will henceforth simply call a VAX. It is a time-shared resource. At one time at the Center for Astrophysics, when we had only one VAX to handle all our computing, we held the world's record for the number of terminals tied to ports on this VAX, well over 100. I hardly need award you a prize for guessing what response times were like in the middle of the afternoon. Now there are two VAXes tied together, which helps a little, and soon there will be lots of DEC Microvax II's around (a small machine of similar performance costing in the range $20,000 to $40,000). So I will use the CPU speed of the VAX with floating point accelerator as a basis for comparison with microcomputer floating point computation speeds.

Some benchmark speed comparisons which have been been made at the Center for Astrophysics by those with IBM PCs have shown that the PC runs close to ten percent of the speed of the VAX, using the 8087 and Microsoft Fortran. This was a highly cost effective number in favor of the PC until recently, but DEC has managed to scramble back onto this new cost-performance curve with the Microvax II. I had hoped for a large improvement in computation speed with the advent of the IBM AT. Indeed, the AT proved capable of moving bytes around (a typical business use) at about three times the rate of the PC. But it is a computational disgrace. Benchmarks carried out by Avram Tetewsky of Draper Labs have shown that for number-crunching intensive jobs the IBM XT and AT compute at the same speed! The reasons appear to be the following. The XT uses the Intel 8088 and 8087 running at 4.77 megahertz with an 8-bit bus. The AT uses the Intel 286 with a 16-bit bus, allowing it to move bytes around twice as fast per clock cycle; it uses a faster clock, and it performs CPU functions in fewer clock cycles than the 8088 or 8086. The problem lies with the 287 numeric coprocessor. This is divided into two parts, a communications part running at the same 6 megahertz clock speed as the 286, and a 4 megahertz computation part that is not enhanced over the 8087. So what advantage the AT gains in communication speed is lost in a slower computation speed.

The lesson for the technical person interested in fast computation is clear. For the near term, it is better to enhance the speed of the PC or look-alike than to invest in an AT. Following this logic, a few months ago I invested in a Number Smasher board from Microway. This runs at 9.54 megahertz (twice the PC clock rate), and it uses on-board 8086 and 8087 chips that have been designed for 8 megahertz and tested at the slightly higher speed. It has an on-board 16-bit bus into which up to 640 kilobytes of memory can be plugged. This board plugs into the 8088 and 8087 sockets on the PC mother board and interacts with all other hardware services through the regular 8-bit bus. With this board I find that I can crunch numbers at 25 percent of the VAX speed.

For certain specialized problems, the PC technical person can do still better by using array processor insert boards. These achieve a faster computational throughput by using pipeline techniques and a faster numeric processor. However, so far these are very limited by 32-bit floating point arithmetic, and I have not been personally interested in them. For more general-purpose computations we must await the (presumably soon-to-be announced) 64-bit array processsors using very fast Weitek numeric coprocessor chips.

However, it is already apparent that the Motorola 68020 CPU teamed with the Weitek coprocessor can strongly outperform the VAX. I am indebted to Steve Ward of the Center for Astrophysics engineering staff for giving me the following preliminary benchmark numbers. For a double precision floating multiply, the VAX with floating point accelerator takes 3.40 microseconds. The 68881 (the Motorola numeric coprocessor for the 68020) takes 4.00 microseconds at 16.67 megahertz and 5.33 microseconds at 12.5 megahertz, so it is slower than the VAX. The National Semiconductor 32081 (coprocessor for the 32016) takes 6.20 microseconds at 10 megahertz and 7.75 microseconds at 8 megahertz. According to trade journal reports, the Weitek 1164+1165 numeric coprocessor chip set takes 0.60 microseconds! This chip set clearly has a very promising future.

Is this the road for future upgrades by the PC workstation user? Does this mean a totally new personal computer architecture or will we get these Weitek chips as parts of insert boards for PCs? In that case what will the software situation be? Or will the IBM commitment to the Intel line mean that we should expect to use the Intel 386 and 387 chips and to migrate our present software base? The answers to these questions may not be clear but the existence of the questions suggests that the individual desktop workstation will continue a rapid evolution in capability.

STORAGE

There are two types of storage of vital interest to the workstation user, RAM and permanent storage.

The business PC user has been demanding larger and larger spreadsheets and data bases, and this has spurred several important technical developments. The 640 kilobyte limitation of DOS, which once seemed much more than adequate, has lately been seen as a serious straightjacket. For a while the solution seemed to be to go to the 286 chip, which can address 16 megabytes, but not when it is running DOS. If this were to be the solution, then we would all migrate to XENIX as an operating system. But the large installed base of PCs has brought forward an alternative solution, and since the processing enhancements described above at least temporarily make the PC superior to the AT as a hardware base, it is possible that the new memory developments will maintain DOS as a viable alternative for some considerable time.

The recent announcement of a new proposed hardware and software standard for bank switching of memory by Lotus jointly with Intel is of considerable interest to the scientific workstation user, because many of the Fortran programs that he may have been using on mainframes exceed the 640 kilobyte limit. The Intel AboveBoard will have many clones and variants which will bring multimegabyte memory resources to our disposal, once our language compilers and linkers have been brought up-to-date to use them. However, I am aware of other technical developments which are in progress, and it may be premature to conclude that the AboveBoard implementation of memory management will turn out to be the best for the scientific workstation user.

The 360 kilobyte floppy disk remains a useful medium of backup storage and program and data exchange, but it is seriously cramping the style of the workstation user who develops big programs. For example, Microsoft Fortran 3.2 creates intermediate compiler files that are serious memory hogs, leading to a lot of floppy shuffling for those with floppy-only systems. Fortunately, the price of 10 megabyte hard disks has come down very dramatically in the last year, and the new 20 megabyte disks tend to cost only an additional $300. Many of my colleagues say they have passed the point of no return; having invested in a hard disk, they will be both unable and unwilling to go back to a floppy system, and I feel that way myself. A couple of years ago ten megabytes seemed more than adequate, but with some of the memory demands that I will discuss below, I would ordinarily counsel the investment in 20 megabytes.

Of course, there are various hard disk technologies from which to choose. Hard disks tend to come with random access times close to either 30 or 80 milliseconds, and the slower ones are a lot cheaper than the faster ones. I tend to think that for the scientific user larger capacity is more important than a faster access time, and one's investment strategy should be governed accordingly, but there will obviously be situations where the reverse is true. But fast access time can often be achieved with the use of RAM disks, particularly if one uses a specialized RAM disk board such as the JRAM-2 from Tall Tree Systems, which can hold up to 2 megabytes for this purpose.

An entirely different set of options becomes available when one thinks of hard disks with removable media. Unfortunately these still tend to be rather expensive. In a class by itself is Iomega's Bernoulli Box, a unique technology which is not only safer to use but is also an excellent way to store off-line data bases or other specialized materials in a form that can easily be loaded into the computer. It is also an excellent form of backup medium, more reliable and faster than magnetic tape. There are also more conventional types of hard disks which are removable or have a removable platter.

Coming up are the laser disks, initially write-once read-many-times, but eventually read and write at will. These will represent another large step upward in storage capacity, and once again we will have the pressure of the business market to thank for driving the development. I think these will be extremely useful to the scientific workstation user. Not only will they enable specialized scientific data bases to be distributed for his use, but it seems likely that one of the outcomes of artificial intelligence research will be knowledge-based expert systems with a convenient prompting system that the user can assemble in modules as needed. This is the kind of development that is likely to be available for workstations compatible with business systems much sooner than for the more expensive and specialized types of workstations mentioned at the beginning of this article.

LANGUAGES

Fortran remains the workhorse language for most of the scientific and engineering community. Fortran has had its vociferous critics in the computer sciences community, and just about all the nasty things they have said about the language are true. Despite that, it remains my favorite computational language. Why is that so? Partly from familiarity, but mainly because of the large software base associated with it.

Until quite recently, the version of Fortran generally available was the 1966 ANSI standard, usually with some extensions. This lacked the structure of the better modern languages and resulted in what is often called "spaghetti code", a tangled mess of GO TO's. Now on IBM PC's and nearly everywhere else the Fortran available is the 1977 ANSI standard. This still lacks most of the usual control structures, but it does have IF...THEN...ELSE, and you can do nearly everything you want with that. A few years ago I started using a Fortran preprocessor called RATFOR, which introduced C-like control structures into the way one writes code, and which is compiled into standard Fortran. The extra compilation procedure was certainly a nuisance, so I learned to write structured Fortran by mimicking the results of the translation procedures used by RATFOR. Now I simply use IF...THEN...ELSE.

At last count there were some seven Fortran packages being offered for the IBM PC. Avram Tetewsky of Draper Labs recently ran some benchmark comparisons of the three Fortrans that seemed to be most capable. In decreasing order of execution speed these were the Fortrans prepared by Ryan McFarland (IBM Professional Fortran), Microsoft, and Digital Research, Inc., all of which have nearly transparent procedures for utilizing the full amount of memory available under MSDOS. Under ordinary conditions IBM Professional Fortran would therefore be the preferred choice for most number crunching tasks. However, Microsoft Fortran has been out longer, and there are many libraries of auxiliary routines for various purposes that are designed to link to it, but so far very few of these have released versions that link to Professional Fortran. Thus in many practical cases the Fortran of choice becomes that from Microsoft.

Probably the most useful of the Fortran add-on packages are those that support computational functions that are not included with the standard Fortran. There are in general much more extensive packages that have been produced for Fortran than for other languages.

First (to my knowledge) in the field was a package called MicroSUB:MATH sold by Foehn Consulting. This has a useful set of procedures and routines, but one often wants to change the Fortran source code, and the source code for this is rather expensive.

Then recently the FORTRAN Scientific Subroutine Library, a package from the Professional Software division of John Wiley, the book publisher, appeared. This was a more extensive package which did not provide source code.

On mainframes the most extensive package of subroutines and procedures is from IMSL, who have just brought out subsets of these FORTRAN libraries for the IBM PC. The source code for this is a closely guarded secret, and this limits the usefulness of the package. I have heard that the source code for the IMSL package is quietly circulating "underground" among some IBM PC users.

Probably the most useful package is one that has not yet appeared, a book entitled "Numerical Recipes" (by W.H. Press, B.P. Flannery, S.Teukolsky, and W.T. Vetterling) to be published by Cambridge University Press this fall, and to be accompanied by source code disks which may be obtained in either Fortran or Pascal. In order to include the source code and examples of its use on two disks, the authors have stripped out all the commented statements, so that this is a package to be used hand-in-hand with the textbook. The authors have provided a separate summary in Pascal code because of its extensive use in teaching programming, but all the illustrations in the main part of the text are in Fortran. They hope to provide C source code later.

In the last few years I have transported a fair amount of Fortran code from mainframes to minicomputers and from there to IBM PC compatibles. This procedure has turned out to be relatively painless, with most of the effort involved in translating the system calls. It is here that the limitations of MSDOS cause the most grief. For example, it is frequently painful not to have a Fortran procedure for opening a file for appending. There remains a clear need for more extensive Fortran utilities for dealing with the operating system.

A fair number of scientific and technical people are now using C in laboratory environments for interfacing with equipment, for which it is well adapted and Fortran is not. Not much use of other languages has come to my attention. Some of my colleagues have used Pascal a little bit. Opinions on it are divided. On the positive side, some people like the strong typing in the language, because "it won't let you make a mistake." Others dislike the strong typing in the language because "it is a royal pain." Such people would

rather use C. Perhaps there will be a drift away from Fortran in coming years as other languages build up a base of mathematical and physical supporting software, but I suspect that any such motion will resemble that of a glacier. It is also likely that Fortran itself will undergo further evolution.

PLOTTING

There is an incredible number of software programs now available for plotting with IBM PCs. Most of them have been developed for business use. These tend to be screen oriented and pen-plotter oriented, and usually offer a set of utilities for providing bar graphs and pie diagrams and that sort of thing. Very few provide a facility for doing the kinds of drawings that most scientific people want, with lines and curves and isolated points, perhaps showing least squares fits or spline interpolations. Fewer still provide a decent facility for using a dot matrix printer for plotting (most just do screen dumps with very poor resolution).

For quite a long time I have been looking for a plotting package that will provide a library of Fortran plotting routines and which will use the full resolution of the FX-80 printer. The closest I have yet found is Plot88 from Plotworks. This provides a good set of Fortran plotting routines and supports several Hewlett-Packard pen plotters, the IBM color graphics monitor, the Tektronix 4025, and Epson dot matrix printers. Unfortunately the FX-80 is only supported for a vertical resolution of 72 dots per inch, one-third of its capability, but it does support up to 240 dots per inch horizontally. That means that you must still use a pen plotter to prepare a reasonably good publishable graph.

There is no fundamental reason why we cannot get plotting programs which make better use of rasterized devices. The need for these will accelerate when cheap laser printers become available and are desired also for plotting uses.

EDITING AND TEXT PROCESSING

Everyone seems to have a favorite word processor, and arguments about the superior qualities of some other word processor fall on deaf ears. Scientists are no different. However, at least some scientists prefer an editor with extremely powerful macro command capabilities, and that preference is unlikely to be shared by the average person. Many of my colleagues and I share this preference, and our editor of choice is PMATE, distributed by Phoenix Software Associates. Some of my colleagues are interested in little more than the direct entry of text, and they prefer a wide variety of other word processors.

Those word processors that try to integrate editing with spelling checking and hyphenation and text formatting tend not to do any or all of those things as well as other programs do individually. Personally I have for a long time preferred to do such operations separately. Some years ago I started using a hardware text formatter called a Retroscroller, which consisted of a Z80-based board that was inserted into a Diablo 1620 daisy wheel printer. This could be fed text prepared with any editor and it produced better scientific output than any software available at the time. This has evolved into a medley of programs for the IBM PC called the PS Technical Word Processor, and sold by Scroll Systems, Inc., which includes PMATE. In fact, PMATE is merged into the package through the use of about 9 kilobytes of permanent macro commands which give it menus and make it considerably more user-friendly. Until very recently, this combination was unsurpassed for technical text output by any combination short of a minicomputer-driven phototypesetter. Even with this set of programs I still preferred to use The Word Plus from Oasis Systems for spelling checking and also for hyphenation. Although PS does hyphenation, I consider the algorithm used by The Word Plus to be superior.

The situation has very recently been changed by the advent of TEX on the IBM PC. There are two vendors for this: Addison Wesley (MicroTEX) and Personal TEX (PCTEX). TEX is a very elaborate program designed for mathematical typesetting; it has developed a devoted following and is now available on a considerable variety of mainframes and minicomputers, it has been adopted by the American Mathematical Society, and it has its own users' group. The PC implementation does not come cheaply; you can get a minimal working version with 512 kilobytes of RAM, but you are much better off with 640 kilobytes. Right now the principal output device supported by both versions of TEX for PCs is the Epson FX-80, and the storage of the necessary fonts will take 4 to 6 megabytes of hard disk space, depending on the version that you have. This is a very slow output device, and PC TEX users will certainly look forward to obtaining a supported laser printer for output which will also produce superior quality characters. My own expectation is that I will use PS for ordinary material and TEX for very technical material until such an inexpensive laser printer is available, and then I will switch entirely to TEX. However, I am likely to continue to use PMATE with PS macros to prepare text for TEX output.

OVERVIEW

In this discussion of the scientific workstation I have not attempted to discuss control of laboratory equipment or analysis of the data that results, since I have not had any experience in this area. Nevertheless, that is an important area of scientific workstation activity. I expect this to be even more important in the future. Large data bases of scientific and technical information have become available both within government agencies and on the on-line information resource networks, and PCs can be used for data retrieval from these. I expect laser disks to become available as cheap distribution media for specialized scientific and technical data bases, sort of the on-line technical encyclopedia, and these will be invaluable both for those workstations that are used for laboratory control and for those that are not.

From a general point of view what is important is probably not the specifics of the moment, but the sense of a field rapidly evolving both in the hardware and software senses. Much of the momentum for these changes comes from the business community, but there is also a growing constituency on the technical side. We would not have seven versions of Fortran to choose from if that were not the case. Since the scientific and technical community forms a substantial market for workstations in the $20,000 to $40,000 class, the market will become very large indeed when all of those functions can be provided for $5,000. The components of that workstation are rapidly becoming available.

VENDORS MENTIONED IN THIS ARTICLE

Addison-Wesley Publishing Co. Inc., Jacobway, Reading MA 01867 (617)944-3700-X2677

Cambridge University Press, 32 East 57th St, NY NY 10022

Digital Research, Inc., Box 579, Pacific Grove CA 93950 (408)649-3896

Foehn Consulting, Box 5123, Klamath Falls OR 97601 (503)884-3023

IBM Corp., Personal Computer, Box 1328-C, Boca Raton FL 33432

IMSL Inc., NBC Bldg, 7500 Bellaire Blvd, Houston TX 77036 (800)222-IMSL

Intel Corp., 3065 Bowers Ave., Santa Clara CA 95051

Iomega Corp., 1821 West 4000 South, Roy UT 84067 (800)556-1234-X215

Lotus Development Corp., 55 Wheeler St, Cambridge MA 02138 (617)492-7171

Microsoft Corp., 10700 Northup Way, Bellevue WA 98004 (800)426-9400

MicroWay, Inc., Box 79, Kingston MA 02364 (617)746-7341

Oasis Systems, 2765 Reynard Way, San Diego CA 92103

Personal TEX, Inc., 20 Sunnyside, Suite H, Mill Valley CA 94941 (415)388-8853

Plotworks, Inc., Box 12385, La Jolla CA 92037 (619)457-5090

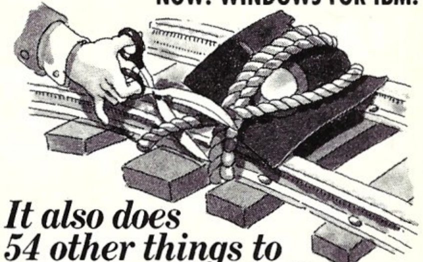
Phoenix Software Associates, 1416 Providence Highway, Suite 220, Norwood MA 02062 (800)344-7200

Scroll Systems, Inc., 6930 East Acoma Pl., Tucson AZ 85715 (602)885-1633

Tall Tree Systems, 1032 Elwell Ct., Suite 124, Palo Alto CA 94303 (415)964-1980

Wiley Professional Software, 605 Third Ave., NY, NY 10158 (212)850-6788

A.G.W. Cameron is Professor of Astronomy at Harvard University, Center for Astrophysics, 60 Garden St. Cambridge MA 02138.



C Programmers:
File System Utility Libraries

Source Code Included, No Royalties, Powerful & Portable.

BTree Library $75.00

- High speed **random** and **sequential** access.
- **Multiple keys** per data file.
- Up to **16 million** records per file.
- Full documentation and example programs included.

ISAM Driver $40.00

- Works with the BTree Library.
- Greatly speeds application development.
- Combines ease of use of **database manager** with flexbility of **programming language.**
- Supports **multi key files** and **dynamic index definition.**
- Very **easy to use; fully documented; example programs included.**

Both products

Are written entirely in **K&R C.**
Come with complete **source code.**
Are **free of any royalty charges.**

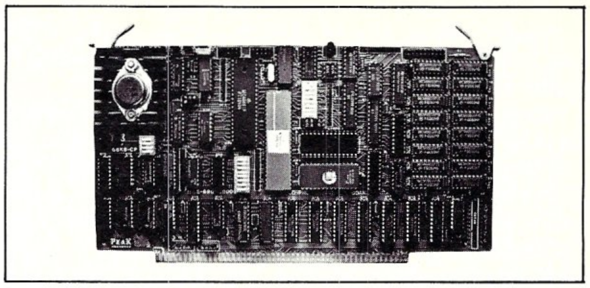
+ $3.00 Shipping &
Handling Charge.

For more information call:

softfocus

1277 Pallatine Drive
Oakville, Ontario, Canada
L6H 1Z1
(416) 844-2610

Credit cards accepted. *Dealer inquiries invited.*



68K8-CP

The World's First 68000 Coprocessor

Peak Electronics' 68K8-CP is a high performance processor card designed to function as one of several CPU elements within a multiprocessor S-100 system. This card features the MC68008™ (8-bit object code compatible version of the industry renowned 68000), up to 512K bytes of RAM and 128K bytes of EPROM, two serial ports and a parallel printer port.

The true power of the 68K8-CP card is shown with its ability to be plugged into any existing S-100 system running CP/M®-2.2 and to be running CP/M-68K within minutes without any change in existing hardware or software. This card does not replace your current processor. All of the original system's devices (RAM, disks, and other peripherals) are immediately available to the user of CP/M-68K, files can be accessed by whichever operating system is currently active.

Features:

- Does not replace your current CPU card
- IEEE-696-1983, S-100 Compatible
- MC68008 8 or 10 MHz CPU
- 128K bytes of RAM expandable to 512K
- 8K bytes of EPROM expandable to 128K
- No wait state access to on board RAM and EPROM
- Two high speed serial ports (up to 38.4K BAUD)
- 8-bit parallel printer port
- Supports CP/M-68K and Concurrent-68K
- Onboard 16-bit counter-timer
- Includes printer buffer and RAM disk firmware

$995.00, includes CP/M-68K

PEAK
electronics
P.O. Box 700112
San Jose, CA 95170-0112
(408) 253-5108

EUREKA!®

In Greek it means "I found it". In CP/M it means you'll never lose it again! **Eureka!** is the premier CP/M disk cataloger. It puts the critical information about your library of disks and files in one place and allows you to quickly and easily retrieve any of the information. Never again will you have to wonder what disk that letter SMITH37.LTR. is on - or what it is for...**Eureka!** can tell you in a snap. Lawyers, writers, programmers and many others save time with **Eureka!**. Haven't you suffered long enough without it ???

> *"We finally chose EUREKA! ... largely because it has the ability to read comments directly from a file ... EUREKA! is easy to learn and use, has more access and report choices, finds files by many different ways, and has an attractive price."*
> T. Bove & C. Rhodes, USER'S GUIDE No. 11

EUREKA!, the popular CP/M® disk cataloger

Unprotected version	**$75**
Protected version	$50
Additional Licenses	$15

MENDOCINO SOFTWARE COMPANY, INC.

Dept. M -5
P.O. Box 1564
Willits, CA 95490
(707) 459-9130

add $2.50 shipping
Calif. residents add 6% sales tax

VISA & MasterCard
accepted

A **EUREKA!** package is designed to run on only a single system.

EUREKA! is a trademark of Mendocino Software Company, Inc.
CP/M is a registered trademark of Digital Research, Inc.

Now available with 8087 Support!

MTBASIC

Basic Compiler

Features:

Multi-line functions	Multitasking
No runtime fee	Windowing
Handles interrupts	Interactive
Fast native code	Compiles in seconds

MTBASIC is easy to use since you can write programs in an interactive environment and then compile them using only one command. MTBASIC has many advanced features like multitasking, random file access, formatted I/O, assembly language calls, and ROMable code.

The MTBASIC package includes all the necessary software to run in interpreter or compiler mode, an installation program (so *any* system can use windows), demonstration programs, and a comprehensive manual.

Ordering

MTBASIC is available for CP/M, MS-DOS, and PC-DOS systems for $49.95. MTBASIC with 8087 support is available for MS-DOS for $79.95. Shipping is $3.50 ($10.00 overseas). MD residents add 5% sales tax. MC, Visa, checks and COD accepted.

SOFTAID, Inc.

P.O. Box 2412 Columbia, MD 21045-1412
301/792-8096

PUBLIC DOMAIN SOFTWARE

Complete SIG/M and CP/M-UG Libraries Available in 5-1/4" Formats—

The complete SIG/M and CP/M User Group libraries are now available from Dusch Computer Services in nearly all 5-1/4" disk formats. Disks are $7.00 per volume including UPS Ground or First Class Mail. Add $3.00 per order for UPS Blue Label. Foreign shipments: $10 per volume, including Air Mail. A $1.00 copying charge will be paid to SIG/M-ACGNJ for each disk copied. **Specify computer and format.**

PC-Blue Library

The complete PC-Blue Library is now available. Disks are $7.00 per volume including UPS Ground or First Class Mail. Add $3.00 per order for UPS Blue Label. Foreign shipments: $10 per volume including Air Mail. A $1.00 copying charge will be paid to PC-Blue/NYACC for each disk copied. **Specify computer and format.**

Maximum 2 Day Shipping Time from Receipt of Order

Order From:
Dusch computer services
405 East 6th Avenue
Roselle, NJ 07203
201-241-0719

UNIX POWER

100,000 software developers can't be wrong.*

UNIX is the chosen operating system for more than 100,000 software developers because it has the power they need. But developers aren't the only people who need computing power. Any business that wants multi-users to access the same files at the same time or wants to simultaneously run multi-task operations ... needs UNIX. At Dynacomp, we offer UniPlus + ® System V by UniSoft Corp. For $1495. U.S. dollars you can run UNIX on the CompuPro® 816/E™ ... a powerful 68K S-100 bus computer system that maximizes its memory for multi-user/multi-task operations.

UniPlus + includes all the standard UNIX System V features PLUS performance enhancements found only in UniPlus +. These features increase the portability, flexibility, and performance of UNIX, allowing an affordable operating system for program development, text preparation, and general office use.

If it's time for you to upgrade to UNIX, call your local Full Service CompuPro System Center in the United States or call Dynacomp in Canada for complete details.

FROM

210 W. Broadway
Vancouver, B.C.
V5Y 3W2
(604) 872-7737

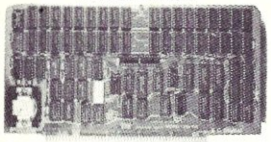
DYNACOMP
COMPUTER SYSTEMS LTD.

46-6535 Mill Creek Dr.
Mississauga, Ont.
L5N 2M2
(416) 826-8002

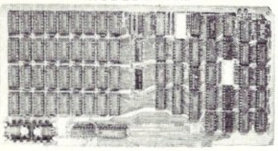
*AT&T estimates that there are more than 100,000 people currently developing software under UNIX. Dynacomp serves all of Canada and parts of Asia and the Pacific Rim. Call us for details and information on our full product line including Plexus. • UNIX is a trademark of Bell Laboratories, Inc. CompuPro is a registered trademark and System 816/E is a trademark of Viasyn Corp. UniPlus + is a registered trademark of Unisoft Corp. AT&T is a registered trademark of AT&T Information Systems.

For S100 bus by S. C. Digital, Inc.

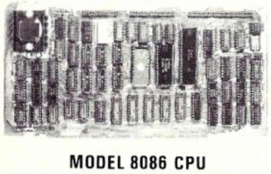
MODEL 256KM



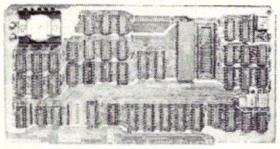
MODEL 256KB



MODEL FDC-1



MODEL 8086 CPU



256 K/1M DYNAMIC RAM Board Model 256KM-512 $375
• 256K/1M bytes using 64K or 256K DRAMs • 8/16b data • 24b Address • Parity per byte • 175nsec access time • will run Z80/Z8000 to 6mhz, 8086, 80186, 68000 to 8mhz without wait states • transparent refresh, unlimited DMA • with 512KB using 256K drams.

256K/1M DYNAMIC RAM Board Model 256KB-256 $325
• 256K/1M bytes using 64K or 256K DRAMs • 8b data • 16 or 24b address • parity per byte • Memory mapping in 16K blocks • 175 nsec access time • Addressable in 128k, 192k, or 256k boundaries, compatible with Z-100* systems • with 256KB using 256K drams, expandable to 1 mega byte, less memory mapping • add $20 for memory mapping.

FLOPPY DISK CONTROLLER Board Model FDC-1 $325
• Single or double sensity, sides, in any combination of up to four 8" or 5¼" drives • Digital phase-locked loop • DMA data transfer with cross 64K bountaries, 24b address, DMA arbitration • built in monitor/boot EPROM that accomodates two different processors • serial port to 19.2Kbaud • uses 765A/8272 • with CPM bios programs

8086 CPU Board Model 8086 CPU $325
• 8/4 (or 10/5) mhz SW selectable • 8087 interface • provision to fun two processors on a bus such as our Z80 CPU • convertable to 10, 12mhz clock • optimized for DRAM boards • specify clock speed.

80286 CPU Board Model 80286 from $350
• 8/4 mhz switchable • 80827 interface • provision to run two processors on a bus • convertable to 8mhz • separate built in colck for 80287 • optimized for DRAM boards.

Z80 CPU Model Z8CPU $249
• 6 or 8mhz, specify • memory mapping in 16K blocks, addressable four M bytes.

SUPPORT Board Model Support-1 $325
• 4 serial, full handshakes, two with software programmable baud rates • Centronics • SASI interface • Real/interval times • Calendar-clock with battery backup • expandable interrupt controllers for 8086 or 8080/Z80 • CPU switching to run 2 processors on a bus such as our 8086 or 80286 and Z80 CPU boards.

I/O Interface board Model 3SPC-N $229
• 3 serial RS-232C with switch selectable baud rates, 110 to 19.2kbaud. • 1 parallel.

128K STATIC RAM/EPROM Model 128KS $349
NO RAM $169
• 128K bytes four 6264 Ram or 2764 Eproms, can be mixed • 8/16 b data • 16/24 b address • bank select, RAM shadowing, etc. • ROM simulation in RAM, battery back-up provision.

S-100 COMPUTER 'System 16' $3200
8086 based at 8mhz, with 512kb, 5 serial ports, 1 centronics, 1 SASI, battery backed calendar clock, real time clock, interrupt driven, 10 slot card cage, two 5.25" floppy drives with 500 kb transfer rates and 1.2 mbyte storage each, with CPM 86 operating system (Concurrent Dos available soon). Cabinet has room for full size 5¼" hard disk.

Operating Systems available: CPM 2.2, CPM 3.0, CPM 86, MSDOS.

*CPM is registered trade mark of Digital Research Inc. Z-100 is registered trade mark of Zeith Corporation.

Please call for latest prices.
Prices subject to change without notice.

S.C. DIGITAL, INC.
1240 N. Highland Ave., Suite 4 • P.O. Box 906, Aurora, Illinois 60507
Phone: (312) 897-7749

The UNIX File

by Ian F. Darwin

This column discusses the UNIX operating system. If you have comments or questions about UNIX, or this column, please write to Ian Darwin, Box 603, Station F, Toronto, Ontario, Canada M4Y 2L8. If you have access to the uucp network, mail "ihnp4!darwin!ian". I can't always answer immediately, but I will get back to you; electronic mail gets answered first!

A Hacker's OS & Tuning Unix Programs

This UNIX File tells how to get the source for a hacker's operating system, shows how to tune programs, and tells about upcoming UNIX conferences.

HACKER'S OPERATING SYSTEM

Many hackers dream of getting the source to the UNIX system. The GNU system has been promised for some time, but to date GNU has only released an EMACS-like editor. Here's a UNIX-like system that you can get the complete source for. The system is XINU. Its author, Douglas Comer, wrote an excellent book *Operating System Design: The Xinu Approach* (Prentice-Hall, 1984, ISBN 0-13-637539-1; also available via M/SJ -see "The Book Mart"). The book describes how Xinu works, rather than enumerating ways an OS might work.

Xinu is not a complete UNIX-like system. As a teaching/learning tool it consists of a kernel with some UNIX functionality. One important function to be added before you would have working timesharing system is the ability to dynamically load new programs. Instead, your application is linked with Xinu, and the combined module loaded and run. Comer alludes that it would not be hard to add this facility. You'd still have to add a shell and utilities before you'd have full UNIX functionality.

The OS source is included, and a dedicated hacker could type it in. Tom Hartnett (electronic address ihnp4!nvuxd!th), did so and is willing to make the part he typed in available by UNIX electronic mail (or put you in touch with someone near you to whom he's sent it). Tom is setting up a *uucp* system so you can dial in and pick up his version of the source files; contact him for details. As important, Tom is running an electronic news letters for people playing with Xinu, getting it up on different machines, etc. The first issue summarizes the work needed to port Xinu to a new machine.

Alternately, you can get the complete source on nine-track magnetic tape from the book's publishers for under a hundred dollars. Information on how to get the source tape is not included in the book, so here it is. To order the tape, contact James F. Fegen, Jr., College Editorial, Prentice-Hall, Englewood Cliffs, NJ 07632, phone (201)592-3122. We got our tape with no problems, and have even received minor corrections made since the tape was written.

What it means is that there's an OS that you can get the source for, hack at, and (maybe) expand to a working timesharing system. The operating system, like UNIX, is written in C with an underpinning of assembly language. The *real machine* on which it's running is the ubiquitous LSI-11 micro. Tom Hartnett has it running on a 68000, and through his electronic newsletter you may find it running on other machines by now.

NEED TO OPTIMIZE?

How much execution time are you willing to trade off for programmer time? How much can you slow down a small system and still consider it usable? There are no hard answers for such questions, of course. But there are techniques for improving execution time of C programs.

Before you even think about optimizing, think about several things. First, the importance of *making it right before you make it faster*. There's no point solving an equation quickly if you get the wrong answer! This often overlooked aspect of program efficiency is emphasized in *The Elements of Programming Style* by Kernighan and Plauger, a book I recommend for all programmers.

Second is the degree of use the program gets. To shave one second off the run time of an accounting program that's used once daily, and at 2 AM at that, is to make an utter waste of your life's work. The improvement is so useless compared to a day's worth of CPU time that nobody will ever notice. But shave that same second off *grep* or *cat* or some frequently-used user program, and you will contribute some fraction of a percent of CPU time back to system throughput. Two UNIX tools that give numerical data on program performance are *Time* and *prof*. *Time* tells how long a program takes to execute for a given set of inputs. You just say, for example,

```
time cat /unix >/dev/null
```

to see how long the particular *cat* takes to run. *Time* reports three numbers, which are the wall clock time (elapsed time you would see with a stopwatch), the program CPU time, and the amount of CPU time the kernel spent processing requests on behalf of the program.

The sum of the user and system CPU times is what timesharing services conventionally use to bill for *CPU time*. The real or elapsed time must always be greater than (just possibly equal to) the sum of the user and system CPU times. Elapsed time varies with system load. If you had the system to yourself, the elapsed time should be close to the CPU time; if not, there may be time lost waiting for disk swapping, system-provided processes, etc.

Prof is used to print an execution profile of a running program. You compile the program with the *-p* option specified to the compiler for each module you want to examine, *and* for the final *cc* command that loads the modules together. This option causes inclusion of code in your compiled program to indicate where time is being spent, and to write the results to disk at exit for later analysis by *prof* itself (note that if your program is interrupted or dumps core, the counts do not get written).

For example, I have a Research-UNIX-like version of *cat* i.e., no options such as -s, -n, -vb, -vd (*cat* is a program to copy files to the standard output; its most common uses are printing files at the terminal and con*cat*enating files together). My version lives in two source files, *main* and *process*, each containing one function with the same name as the file. A testing sequence might look like this:

```
$ make
        cc -O -p -c main.c
        cc -O -p -c process.c
        cc -O -p main.o process.o -o cat
$
$ time ./cat /etc/group >/dev/null

real        2.3
user        0.0
sys         0.8
$ prof cat
_memccpy100.00%  0.02sec
_process  0.00%  0.00sec          1         0.00ms
    _main  0.00%  0.00sec          1         0.00ms
$
$ ./cat /unix >/dev/null
$ prof cat
_memccpy 78.63%  1.61sec
   _fgets  9.92%  0.20sec
   _fputs  6.11%  0.13sec
   _fstat  2.29%  0.05sec
_process  1.53%  0.03sec          1        31.25ms
  _filbuf  0.76%  0.02sec
  _profil  0.76%  0.02sec
    _main  0.00%  0.00sec          1         0.00ms
$
```

The command *make* rebuilds the program. Then I run my newly-made cat *against a short file, and run prof* to look at the profile. The columns are function name, percent of CPU time, CPU time, and for some modules the number of calls and msec/call. In this case, every measureable bit of CPU went into a routine called *memccpy* that is called from inside the *stdio* (standard I/O) library. As its name suggests, *memccpy* copies data in memory, presumably copying the input buffer to the output buffer.

If you don't know about *stdio* , suffice to say, that there are two sets of calls for doing I/O in UNIX; the system calls (open, read, write) and the 'standard IO' or *stdio* routines. The system calls pass control to the operating system to carry out some request, but system calls only read blocks of data (or read one byte at a time, which is very inefficient). For byte at a time processing, or line-at-a-time processing, you almost invariably should use the *stdio* routines (*fopen, fread, fwrite,* etc) since they take care of buffering, lines that are stored across block boundaries on disk, etc.

When I run the program on a larger file, although *memccpy* is still the largest component of time (at only 79 % of the total CPU), a number of other functions show up with measurable CPU time.

As a general principle of program tuning, you need to run a program like this with one small file, with several files, and with one or more large files. Often this will show different parts of the code dominating the CPU usage.

The first version of *process* was written using *stdio* routines *fgets* and *fputs*. Noticing that *stdio* routines and subordinate library routines dominate the CPU usage, I rewrote the *process* function to use the read and write system calls. Here's what happened:

```
$ make
        cc -O -p -c process.c
        cc -O -p main.o process.o -o cat
        time ./cat /etc/passwd >/dev/null

real        5.1
user        0.0
sys         0.9
        prof cat
_process  0.00%  0.00sec          1         0.00ms
    main  0.00%  0.00sec          1         0.00ms
$ time ./cat /unix >/dev/null

real        9.1
user        0.0
sys         2.8
$ prof ./cat
_process  0.00%  0.00sec          1         0.00ms
    main  0.00%  0.00sec          1         0.00ms
$ time ./cat /unix >/dev/null

real       11.5
user        0.0
sys         2.7
$ prof cat
_process100.00%  0.06sec          1        62.50ms
    _main  0.00%  0.00sec          1         0.00ms
$
```

The change was so dramatic I ran it several times to be sure. User CPU time was not measurable. Once on copying a large file (*/unix*), the system noticed 62 msec (0.06 sec) of CPU time in the program; on the other two runs, the system did not notice any user CPU time. I have since installed this version of *cat* in */bin* on my system.

Now this is not to argue that one should not use *stdio* in general. A good intermediate would have been to use the *fread*() and *fwrite*() library calls. But *cat* is a program that by design does not need to transform or even look at the data it is reading, so it's appropriate to use *read*() and *write*() here. Most programs would want to examine the data in some way; for such programs, *stdio* is definitely the way to go, despite the apparent CPU overhead of using *stdio*. The programmer overhead saved is worth it.

I've shown some quick examples of using *time* and *prof*. These are two main tuning tools that every UNIX/C programmer should know. There are many fancier tools on some versions of UNIX and some that are available as proprietary add-ons; I have limited the discussion to two tools that I can be sure every real UNIX system has. For an alternate method of studying the CPU program usage, see *Dynamic Instruction Counting* by Peter J. Weinberger, AT&T Bell Labs Technical Journal, October 1984, Vol 63, No 8, Part 2, p 1815.

I hope you find this discussion useful; please let me know if you are interested in a similar discussion of standard debuggers or other programming aids.

CONFERENCES

USENIX has scheduled its 1986 winter technical conference for January 15-17, in Denver, Colorado, three weeks before /usr/group's *UniForum* show February 3-7, in Anaheim, California. The two groups couldn't put together a joint conference this year, but continue to cooperate with each other and may do so again at some point in the future.

This *USENIX* will be different from previous *USENIXes*, consisting of three one-day technical workshop sessions with a new topic each day, namely *UNIX and Window Systems* , *UNIX on Big Iron* , and *UNIX and Ada* . This is obviously an experiment; if it works well *USENIX* will probably continue the 'workshop' organization; if not, they may try other approaches or revert to the standard multi-stream conference.

UniForum, in February, will be a conventional conference and commercial trade show, including a day of Tutorials (I will be presenting a one-day tutorial on *Effective UNIX Programming*) and a program of technical and commercial papers. *USENIX* will also hold a traditional Summer technical conference in Atlanta, Ga., June 9-13.

For more details, contact *USENIX Association,* Box 7, El Cerrito CA 94530, (415)528-UNIX (528-8649), and */usr/group,* 4655 Old Ironsides, #200, Santa Clara CA 95054, (408)986-8840. Hope to see you there!

For European readers, or those who like to go there, the European *UNIX User's Group* (EUUG) holds conferences periodically. The September 1985 conference was in Copenhagen in September. They planned talks on many topics including the long-awaited 4.3 BSD. I always seem to hear about EUUG gatherings just a little too late to tell you in this column; if you're planning to attend future EUUG conferences, you might write to EUUG, Helen Gibbons, Owles Hall, Buntingford, SG9 9PL, Great Britain, phone +44 763 73039.

That's all for this month. I welcome cards and letters and electronic mail on these and other topics. Cheers!

Eureka

by Robert Hazelwood

A CP/M-80 Disk Cataloging Program

Ever had a bad case of DIR:itis? When you forgot which disk a badly-needed file was on and you wear out your fingers inserting disk after disk and typing DIR: over and over again! Well, I may have found an answer for those of you out there who were like me. It's called *Eureka*, a program that lets you catalog and list all the necessary information about your floppy disk collection. My problem started out just as, I guess, everybody's does. I never tried to catalog my growing disk collection because I never thought it necessary, until I discovered it had grown to over a hundred disks with, at best, scanty comments on the labels. I first tried using index cards, then printing out each disk directory but these methods were too difficult to update and didn't help with duplicate file names.

What I needed was a series of comments for each file and a master alphabetical listing of all the disks. I then tried a few experiments with some of the public domain software for disk catalogs, but was never quite satisfied with the results. I could set up an index file on each disk and place comments in it, but still had to search each listing either manually or use the FIND program to search a cobbled-together, master catalog of all the disks. Then I saw an ad in Microsystems for a catalog system called Eureka by Mendocino Software Inc. I got a copy of Eureka and tried it.

DOCUMENTATION

The first thing that impressed me about the program was the manual. It was eighty pages of detailed instructions on installation, operation, and explanations about the program. I was so fascinated with the manual that I actually read it before I installed the program!

The manual was well thought out and had an index, table of contents, and several appendices, one of which listed all of the error messages, that were used by the program. With the listing of the error messages there is a set of explanations telling you when you would normally encounter these errors and what probably caused them. The manual is very user friendly!

INSTALLATION

I then followed the step-by-step installation instructions. You must use their program (M-COPY.COM) to install the program on your system as the normal copy programs, pip etc... will not properly copy the program. This made me curious, but I followed their instructions and used M-COPY to copy over EUREKA.COM.

When finished copying Eureka to the work-disk the installation program (M-INSTALL) is run. First pick a terminal from the selection which includes most terminals in use. If yours is not on the list, you will have to answer a few questions about your terminal, i.e. its screen width, height, and the code used to clear the screen.

Next are the questions about your printer, i.e. the width and number of lines-per-inch it prints. If your printer cannot handle the ASCII form feed code, you will have to set the printer parameters differently than if your printer used form feeds, but Eureka will work either way.

To try to prevent BDOS errors from dumping you out of Eureka, the program checks for valid drive names. So in the installation you will need to specify all of the valid drive names you have on your system.

When I installed my copy of Eureka, I had to decide which files I didn't want cataloged. As shipped, Eureka will not catalog any file with the attribute $$$, BAK, or HEX. To this you can add two more types or change any of the original three for a maximum of five types that can be excluded. All files that are declared to be system files are not cataloged and this cannot be changed from the install program. The remaining installation consists of selecting default toggles for the ACCESS menu. Most of these toggles can be changed from the ACCESS menu and they affect how the catalog information is presented on the screen and the printer. I left mine with the default settings. The instructions were very easy to understand and I did not have any problems with installation.

SETTING UP THE CATALOG

Next I had to design a cataloging method for my disk collection. It should be well thought out as you will have to live with it for quite a while. A file containing the contents of each disk's directory is given a name. This "volume label" is preceded by a hyphen and is in the same format as a file name under CP/M, except you are allowed one less character due to the hyphen. Eureka searches for this file when it catalogs a disk, so there can be only one file of this type on the disk at a time. Eureka uses this file as the name of the disk that is stored in the master catalog.

USING THE CATALOG

The program itself is very easy to use, as it is menu driven with the selections self-explanatory to the point that you should not really need the manual after you have used Eureka a few times. If you get stuck and cannot find a menu selection that helps, hit the escape key to return to the previous menu.

A key feature of Eureka is that you can add comments to your catalog listings in two ways. The first (recommended in the manual) is to add the comments to the files on your disk. Eureka searches the first 256 bytes of each file and looks for the four identifiers that are part of its cataloging system. These are Desc:, Date:, Vers:, and Auth:. If these are placed in the first 256 bytes of the file, Eureka will read them and insert them in the catalog file for that disk. Only the first two of these must be exactly as they are listed. The last two can be any four letters followed by a colon. They must be four letters in length and if the identifier is less than that, you must fill in with blanks to the colon. The first two identifiers can be changed using the install program.

Eureka looks only for its special identifiers which can be hidden inside of the file using whatever method will work with that language or program. Comments inserted in various types of programs look like this:

```
FORTRAN
C Desc:Fortran program to calculate
       distance
C Date:04/27/83
C Vers:3.34
C Auth:R.Hazelwood

Basic
10 REM Desc:Calculations program
       Date:04/27/83
       Vers:3.34
       Auth:R.Hazelwood

Pascal
(* Desc:Calculations program *)
(* Date:04/27/85 *)
(* Vers:3.34 *)
(* Auth:RHH *)
```

The manual shows several other methods for other types of programs.

The second method of adding comments to your program is to insert them into the disk name file that was created on the disk. If Eureka searches the disk files for comments and fails to find any, it checks the catalog file to see if there are any comments there. If it finds any, it uses these comments. If you have comments in both places, it uses only the comments inserted in the file itself. These comments are formatted in a slightly different way than the rest of the file so that you can insert comments into files you wouldn't ordinarily be able to comment, such as COM files. Each file name must be on a line by itself and preceded by an asterisk. Then on the following lines the comments are inserted just as before.

```
*FOXTAG.ASM
Desc:New improved game program
Date:04/27/85
Vers:3.99
Auth:R.H & C.K.
```

Eureka's nicest feature is available under the ACCESS menu. "T(ext)" allows you to selectively sort through your disk files, using logical operators which Eureka provides. They allow you to include and exclude various groups of characters from the search string. For instance, if you wanted to search for all the Fortran programs written by RHH and DKT, you would enter "Fortran+RHH+DKT" on the text line. Eureka then searches the comments, looking only for those programs that were Fortran and were written by RHH and DKT. In addition to the text options, you can narrow the search further by using wildcard references when you select the disk title at the ACCESS menu.

It is always nice to have a printed copy of your catalog. Eureka will do this for you in a variety of different forms. The first shows all of the catalog files alphabetically, both with or without comments. The second prints out a selected disk's files in alphabetical order by disk. The third lists all of the disk titles in alphabetical order. The last method lists disk informationis by disk title telling you the disk size, number of files cataloged, space used, and space free on each disk. It also shows the last date that the disk was updated. Each of these listings can be modified using the search parameters explained earlier. This allows you to print out just the listings on the disks that have been changed without printing all of the catalog.

COPY PROTECTION

One last area of concern is that Eureka uses a novel approach to software piracy. When the program is first installed, it reads certain items from your operating system and stores them in the program. If you try to use the program on another machine you will get the following warning message:

```
*** EUREKA! runs on a single system ***
```

When I took my copy of Eureka into work with me and tried to bring it up on the Apple II, all I got when I tried to run Eureka was the above warning message. This does not mean that you cannot make copies of Eureka. One of the selections on the main menu is "Put Eureka on a new disk". When selected, it copies Eureka to the new disk. The copy program works with your current catalog disk and does not require the distribution disk for copying.

My curiosity got the better of me however, so I called the company and was told that additional licenses were fifteen dollars each and included a new disk to install on the other system. I asked about possible problems with ZCPR3. They said that they didn't think there would be any problem as they had ZCPR2 installed on a home system and had not had any problems with Eureka. I asked him about bugs in Eureka and was told they had not received any bug reports in over eighteen months. If any are found they use the registration form to contact users.

There is one problem that I didn't encounter that is warned against in the manual. Certain versions of the BIOS used with CP/M cannot handle mixed-disk densities while you are in Eureka. In this case, you will have to exit Eureka, type control-C and return to Eureka to finish cataloging your disks. Sort your disks before you start cataloging, so as not to have do this more than once during your use of Eureka.

Eureka requires CP/M 2.2, 48K of RAM, at least two disk drives, and a terminal that can handle at least 50 columns.

I like the program very much. It is easy to use and searches through the catalog files very quickly. The manual is very good and the people who supply it are friendly and courteous. Eureka should be one of those programs that is automatically supplied with your system. That way you could start cataloging your disks before their number becomes too large to work with easily. If you don't already have a copy of Eureka, go buy one, it's $50 very well spent. It can be purchased directly from Mendocino Software Inc., Box 1564, Willits, Ca. 95450, (707) 459-9130.

Robert Hazelwood works as an Electronic Engineer for Atlantic Satellite Communications, designing and maintaining audio, video, and microwave communications systems at their satetllite uplink facility in Northvale, NJ. He attended the University of Missouri at Rolla where he majored in Electrical Engineering and Computer Science. He has been involved with microcomputers since the early seventies.

S–100 BARE BOARDS

8086/8087 CPU - plus
2764 or 27128, 8253, 8259

8088 Auxiliary Processor
I/O mapped, 4K EPROM,
4K RAM, prototype area

$45.00
Each

Call or write for brochure.

Terms: Check or money order only. CA residents add sales tax. Prices include UPS shipping.

Applied Innovations
3000 Scott Blvd. Suite 106
Santa Clara, CA 95054
(408) 748-1875

BOBCAT 3 DISK CATALOG

The most versatile and powerful catalog program available

- creates, adds, updates, and deletes a filename catalog
- seven report formats
- hard disks, multiple drives, and user numbers
- individual file titles
- wildcard searches for filenames and file titles
- CP/M or PC/MS DOS

8" CP/M SSSD or Popular CP/M or PC/MS DOS 5¼"

U.S. residents ....$49.95 U.S. $
Canadian residents ...$49.95 Cdn $ (Ont. residents add $3.50 pst)
Other countries ...... $54.95 U.S. $
plus $3.00 P & H

NAME, NUMBER
EFFECTIVE & EXPIRY DATES
(MC FOUR EXTRA DIGITS)

Bank drafts; certified checks; money orders; company checks

R&L MicroServices Inc.
Box 15955, Station F
Ottawa, Ont.
K2C 3S8 (613) 225-7904

R&L
THE HOME OF THE BOBCAT

C Users' Group

Over 60 volumes of public domain software including:

- compilers
- editors
- text formatters
- communications packages
- many UNIX-like tools

Write or call for more details

The C Users' Group
Post Office Box 97
McPherson, KS 67460
(316) 241-1065

The Public Domain Software Copying Company

est. 1983

brief Catalogue & Sample Disk (2 sided) $12

PC-DOS MS-DOS© SIG/M
also
Commodore$_r$ CPM/UG Apple II$_r$
~ Osborne Dealer ~

33 Gold St. L3M N.Y.C. 10038
212-732-2565
Overseas include Post.-N.Y. incl. tax
Prepaid Only
©copyrights acknowledged

NEW! AFFORDABLE ENGINEERING SOFTWARE

VISA **Free Catalog** **CP/M-80 PC/MSDOS TRSDOS** MasterCard

- **LOCIPRO** Root Locus — **$69.95**
- **ACTFIL** Active Filter Design/Analysis — **$69.95**
- **STAP** Static Thermal Analysis — **$69.95**
- **MATRIX MAGIC** Matrix Manipulation — **$69.95**
- **RIGHTWRITER** Proofreader & Writing Analyzer **$74.95**
- **ACNAP2** AC Circuit Analysis — **$69.95**
- **DCNAP** DC Circuit Analysis — **$69.95**
- **SPP** Signal/System Analysis — **$69.95**
- **PLOTPRO** Scientific Graph Printing — **$69.95**
- **PCPLOT2** High Resolution Graphics — **$69.95**

BV Engineering
Professional Software
2200 Business Way, Suite 207 • Riverside, CA 92501
(714) 781-0252

68000 BROWSE for CP/M-68K

Look at any CP/M-68K file

Display ANY CP/M-68K disk file in ASCII or HEX. Scroll UP, DOWN, LEFT, or RIGHT any amount via PFKEYS or commands. Online HELP plus an extensive manual. Very powerful and flexible FIND command searches the file and highlights the found string. Repeat FIND, forward and reverse searches, hex, ascii, etc. etc. PRINT will print a line, group of lines, the screen image or the complete file. HELP, TABS, LOCATE, KEYS, CAPS, COLS etc. Support for PFKEYS, PRINTER, ANSI X3.64 and other terminals is provided through a menu driven configuration program. Written completely in 68000 assembler. Shipped on 8" SSSD (IBM 3740) diskette.

WB SOFTWARE DEVELOPMENT
112 OAKHAMPTON PL. S.W.
CALGARY, ALBERTA CANADA
T2V 4B2 (403) 238-3216

$49.95US

MasterCard VISA

CP/M-68K trademark of DIGITAL RESEARCH.

C CODE FOR THE PC

source code, of course

Concurrent C	$45
LEX	$25
YACC & PREP	$25
Small-C compiler for 8086/88	$20
tiny-c interpreter & shell	$20
Xlisp 1.4 & tiny-Prolog	$20
C Tools	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409
(512) 258-0785

PC–PRO is PC–DOS for CompuPro/Viasyn

- PC DOS Ver 3.1
- PC Graphics Support
- All Viasyn Hardware Supported

for information write

Computer House, Inc.
P.O. Box 709 Woodacre, CA 94973
(415) 453-0865
Trademarks: Viasyn Corp.,
PC-PRO – Computer House.

FREE SOFTWARE
FROM THE PUBLIC DOMAIN

User Group Software isn't copyrighted, so no fees to pay! 1000's of CP/M and IBM software programs in .COM and source code to copy yourself! Games, business, utilities! All FREE!

	Rent	Buy
IBMPC-SIG 1-370 Disksides	400.	800.
IBMPC-BLUE 1-116 Disksides	145.	375.
SIG/M UG 1-222 Disksides	130.	625.
CP/M UG 1-92 Disksides	45.	250.
PICO NET 1-34 Disksides	25.	100.
KAYPRO UG 1-39 Disksides	45.	90.
EPSON UG 1-39 Disksides	45.	90.
Commodore CBM 1-28 Disksides	25.	65.

Public Domain User Group Catalog Disk $5 pp.
Send your check & specify computer format.
Rental is for 7 days after receipt, 3 days grace to return. Use credit card, no disk deposit.
Shipping, handling & insurance $9.50 per library.
(619) 941-0925 Orders & Technical (9 to 5)
(619) 727-1015 24 Hr. 3 Min. Info Recording
Have your credit card ready!

National Public Domain Software
1533 Avohill Dr.
Vista, CA 92083

BBB

AM EX

Complete your Heath/Zenith system with a subscription to *Sextant*!

SEXTANT

SEXTANT

The Independent Magazine for Users of Heath/Zenith Microcomputers

$2.95
July-August 1985
Issue No. 17

ZDS Introduces Five New Computers

Moving Up to MS-DOS 2
Add Automatic Key Repeat to Your '89 or '19
Build Your Z100 a Real-Time Clock
A Public Domain Enhancement for CP/M's Command Processor
An Exercise in Z100 Assembly Language

The Independent Magazine for Users of Heath/Zenith Microcomputers

Whether you use an **H/Z100, H/Z89, Z90, H88, or H8**—you'll find articles in every issue which apply to your system.

Explore **CP/M**, **HDOS**, **Z-DOS**, and **MS-DOS** capabilities in *Sextant*. Applications, programs, compatible hardware and software, and the latest developments in the Heath/Zenith community are among the topics covered in every issue of *Sextant*.

Regular *Sextant* features include:

- **"How-to" articles** to help you enhance your system. Read how other users have altered their systems to suit their needs—and how you can too.
- **Reviews of products** from Zenith and independent suppliers, which can help answer your questions about hardware and software purchases you're considering.
- Short program listings—including **utility programs and games**—which you can use immediately.
- **Advertising by independent suppliers** who support Heath/Zenith systems. In *Sextant* you'll find many products advertised that you just won't find anywhere else.
- Coverage of community affairs. You'll read about **major events and personalities** in the Heath/Zenith user community, and what effect they could have on you.

You can't afford to miss all the information packed into each issue of *Sextant*. **Your Heath/Zenith system just isn't complete without a *Sextant* subscription!**

Get the information you need to get the most from your system!

Start your subscription today!

Call Toll Free: 800/341–1522 (DATATEL 800™, for orders only, Monday-Friday 8 a.m.–9 p.m., and Saturday 9 a.m.–5 p.m., Eastern Time)

Or mail this coupon to:
Sextant, Dept. M3
716 E Street, S.E.
Washington, DC 20003
Please allow 6–8 weeks for delivery of your first issue.

Your satisfaction with *Sextant* is completely guaranteed. If at any time you're not completely satisfied, just let us know and your money will be refunded—even for the issues you've already received.

Send me:

□ **12 issues for $29.91 ($34.50 to Canada)**
□ **6 issues (1 year) for $14.97 ($17.25 to Canada; $21.00 overseas via surface mail, $35.00 via air mail)**
□ Payment enclosed (Checks must be in U.S. dollars payable on a U.S. bank.)
□ Bill me
□ Charge my: □ Visa □ MasterCard
Card # ______________________ Expires ________
Name ______________________
Address ______________________
____
____

Sextant, Dept. M3, 716 E Street S.E., Washington, DC 20003

THE SOFTWARE DIRECTORY

When contacting software publishers please mention that you read about them in Micro/Systems Journal.

Program Name: REVAS4
Requirements: CP/M, MP/M or Z-System
Description: Disassembles 8080, 8085, Z80 and HD64180 object code files as large as 64K. Can assign labels, insert comments and generate cross-reference listings.
Price: $90
Publisher: Echelon Inc.
101 First Street
Los Altos CA 94022
(415)948-3820

Program Names: ZAS and ZDM
Requirements: CP/M, MP/M or Z-System
Description: ZAS is a relocating macro assembler. Produces HEX and REL files. Compatible with Digital Research's ASM, MAC and RMAC, Microsoft's MACRO-80 and Xitan's TDL assemblers. Assembles Hitachi's HD64180 and Zilog Z80 instructions. Nestable conditionals and full expression handling, relocation by absolute, code, common, and data criteria, and complete macro expansion and library insert capabilities. Creates SYM tables used by DSD, SID and ZSID debuggers. ZDM is a debugger and monitor for HD64180 and Z80 code. Has 21 commands
Price: ZDM - $50; ZAS - $69.
Publisher: Echelon Inc.
101 First Street
Los Altos CA 94022
(415)948-3820

Program Name: MB+ Tools & MBPASLIB
Requirements: CP/M-80, CP/M-86 or MS-DOS
Description: MB+ Tools is a library of tested functions and procedures for use with Digital Research's Pascal MT+. Functions are coded in assembler to improve size and speed. All are supplied in 8080 or 8086 source code form and most are supplied in both. Program profiler is included to see where to optimize programs. MBPASLIB is PASLIB rewritten in assembler to optimize programs so that they are smaller and run faster.
Price: MB+ Tools $175, MBPASLIB $75, plus $5 shipping
Publisher:
Minnow Bear Computers
Box 2233 Sta. A
Champaign IL 61820-8233
(217)398-6883

Program Name: LOCIPRO
Requirements: PC/MS-DOS, CP/M-80 or TRSDOS system
Description: Provides simple means to quickly determine closed loop system stability from open loop transfer functions. Solves locus of roots for systems up to 26th order and ten loop elements. Output compatible with other BVE products adding transient analysis and high resolution graphics.
Price: $72.95
Publisher: BV Engineering
2200 Business Way #207
Riverside CA 92501

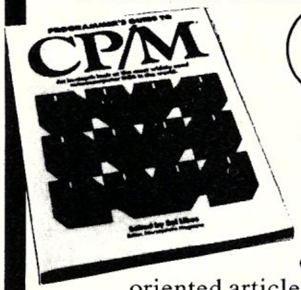
Program Name: SMARTS
Requirements: 256K S-100 RAM card(s)- GSR, QT, Great Salt Lake, California Digital or Computime.
Description: RAM disk emulator. Allows user to define drive as RAM disk. Automatically configures itself for number of boards present in system. Has asterisk option - specifies software will run but not reinitialize RAM memory so disk area will not be erased.
Price: $40
Publisher: GSR Computers
60-10 69th St
Maspeth NY 11378
(718)476-2091

Program Name: SNOBOL4+
Requirements: MS-DOS or CP/M-86 system, 8087 optional
Minimimum Memory: 128K
Description: Implementation of SNOBOL4 as used on mainframes, except Fortran output formats. Produces concise programs for string, algebraic, and list processing, and all forms of non-numerical computation. Based on intricate pattern matching supported by by an extensive collection of string manipulation primitives. Provides integer and double-precision real arithmetic, recursion (in patterns and functions), associative data structures, program-defined datatypes, automatic storage management, assembly language subroutines and self-modifying code. Extensions include additional string and real functions, SPITBOL compatible operators, binary and random access I/O, built-in sorting, include files, case-folding, and BREAK key control. Over 100 sample programs and functions provided.
Price: $95 + $3 shipping & handling
Publisher: Catspaw, Inc.
Box 1123
Salida CO 81201
(303)539-3884

Program Name: db VISTA Version 2.0
Requirements: C Compiler for MS-DOS (Lattice, Microsoft, Computer Innovations, DeSmet or Aztec C) or UNIX (Fortune 32:16, Altos 586, AT&T 3B2 and generic).
Description: Multi-user or single-user database management system in C source form. Provides shared file protection (record locking), transaction processing. Interactive access utility (menu driven/screen and command/batch oriented). Can import and export dBASE II/III and ASCII files. Can handle more than 16,000,000 records/file. Record sizes up to limit of memory.
Price:? $990 with source $495 without source. Single user version $495 with source and $195 without source. Demo version $50.
Publisher: Raima Corp.
11717 Rainier Ave South
Seattle WA 98178
(206)772-1515
(800)843-3313 then dial 700-992

Program Name: d/MULTI
Requirements: TurbDOS Z-80 based system
Description: Converts dBASE-II into a multi-user system with true file and record locking. Adds new dBASE-like commands. Allows interactive real time keyboard commands, making debugging of programs easier.
Price: $695
Publisher: Martian Technologies
8348 Center Dr. Ste. F
La Mesa CA 92041
(619)464-2924

Program Name: MATRIX 100
Requirements: PC or compatible
Language: Written in C & Assembler
Description: BASIC enhancement to perform complex matrix operations with simple BASIC statements. Has extended comands for estimating numerical stability of solutions, LU factorizations, etc. and on-line help. Also inlcudes an interactive linear programming package for small dense problems and Eigenvalue Solver for real symmetric matrices.
Price: $80; with 8087 support $125; with compiler/8087 support $250. Add $4 U.S/$7.50 foreign shipping
Publisher: Stanford Business Software Inc.
4151 Middlefield Rd, Suite 215
Palo Alto CA 94303
(415)424-9499



Coming in future issues

FEATURES

* Implementing PC-DOS On Non-IBM Compatible systems

* Build A Smart Keyboard

* Converting CP/M Programs To Run Under MS-DOS

* Use Debug with NMI

* Transferring Files Between CP/M & MS-DOS

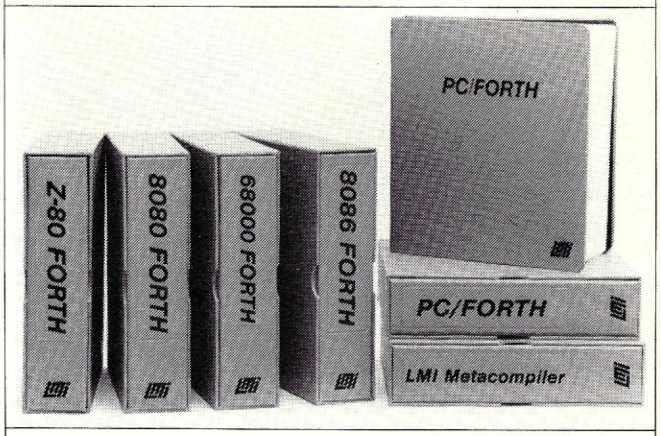
REVIEWS

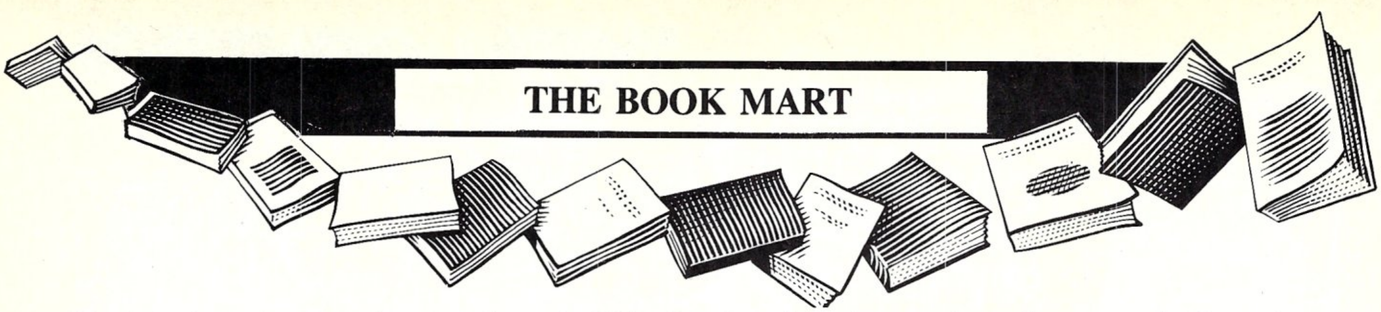
* TurboDos/PC

* Language-Oriented Editors

* CONIX

* C Interpreters & Debuggers





THE BOOK MART

The following selected books can be ordered thru Micro/Systems Journal.

Prices include shipping (NJ residents add sales tax) with prices shown for U.S., Canada and Mexico. Add $5 for other countries. Send check for full amount payable in U.S. funds by a U.S. bank. We cannot bill or take credit cards. Allow 15-20 days for order to be processed. Please do not give a P.O. Box address as we ship via UPS. Send orders to: Micro/Systems Journal, Box 1192, Mountainside NJ 07082.

NEW WORTHWHILE BOOKS

C: A Reference Manual by Harbison & Steele Jr. This is the book that Don Libes (C Forum) uses as his referenced book. $20.95

Operating Systems Design: The XINU Approach by Comer. How to write a Unix-like operating system with much of the source code included. $33.95

The Turbo Pascal Primer by David Carroll. Contains lots of programs. By the author of the *Turbo Pascal Corner* in this magazine. $35.95 (includes a companion disk).

Inside the IBM PC: Access to Advanced Features and Programming by Peter Norton. A detailed analysis of MS-DOS, the PC-system architecture, the ROM-BIOS service routines and much more......$20.95 - Companion disk.....$66.00

The following books are listed only by topic, title and author due to lack of space.

8086/8088

IAPX 86/88, 186/188 User's Manual Hardware Reference, Intel; $21.95

IAPX 86/88, 186/188 User's Manual Programmer's Reference, Intel; $19.95

From Basic To 8086/8088 Assembly Language, Temlepton/Wordware; $18.95

The 8086 Book, Rector/Alexy; $19.95

80286

80286 Programmer's Guide, Childs/Evanczuk; $15.95

80286 System Guide, Childs/Evanczuk; $15.95

IAPX 286 Programmer's Reference Manual, Numeric Supplement, Intel; 17.95

68000

Self-Guided Tour Through The 68000, Andrews; $15.95

68000 16/32 Bit Microprocessor, 4th Ed., Motorola; $19.95

68000 Assembly Language Programming, Kane/Hawkins/Leventhal; $19.95

SCIENTIFIC & ENGINEERING

Microcomputers For Engineers & Scientists, Gibson/Liu; $37.95

Basic Programming For Scientists & Engineers, Hubin; $21.95

IBM PC Basic For Scientists & Engineers, Weinman/Kurshan; $20.95

IBM PC/XT

Business Graphics For the IBM PC & XT, Glau; $18.95

Complete Guide To IBM PC Graphics, Hyman; $20.95

Assembly Language Routines For The IBM, Dorner; $20.95, companion disk $30

Assembly Language For The IBM PC/XT, Scanlon; $22.95, companion disk $30

Input/Output Assembler Programming For IBM-PC DOS, Germain; $25.95

Programming Tools For The IBM PC: Screen Design, Code Generator, and High Memory Access, Fugate; $20.95

CP/M

A Programmer's Notebook: Utilities For CP/M Plus, Cortesi; $18.95

CP/M Solutions: Improving CP/M, Barbier; $15.95

CP/M Assembly Language Programming, Barbier; $13.95

CP/M Techniques, Barbier; $20.95

The Programmer's CP/M handbook, Johnson-Laird; $22.95

CP/M-86 User's Guide, Sachs; $19.95

Inside CP/M, Cortesi; $27.50

Inside CP/M-Plus, Cortesi; $19.95

Inside CP/M-86, Cortesi; $18.95

Inside Concurrent CP/M, Cortesi; $18.95

C

C Programmer's Handbook, AT&T; $15.95

C Programming Language, Kernighan/Richie; $23.50

Learning To Program In C, Plum; $26.00

HARDWARE

Interfacing to S-100/IEEE-696 Microcomputers, Libes/Garetz; no longer published. We have only a limited supply; $19.95

Designer's Guide To Disk Drives, Teja/Gonnella; $20.95

RS-232 Made Easy, Seyer; $19.95

Guide To Local Area Networks, Byers; $16.95

WORDPROCESSING

Getting The Most From Wordstar & Mailmerge, Stone; $15.95

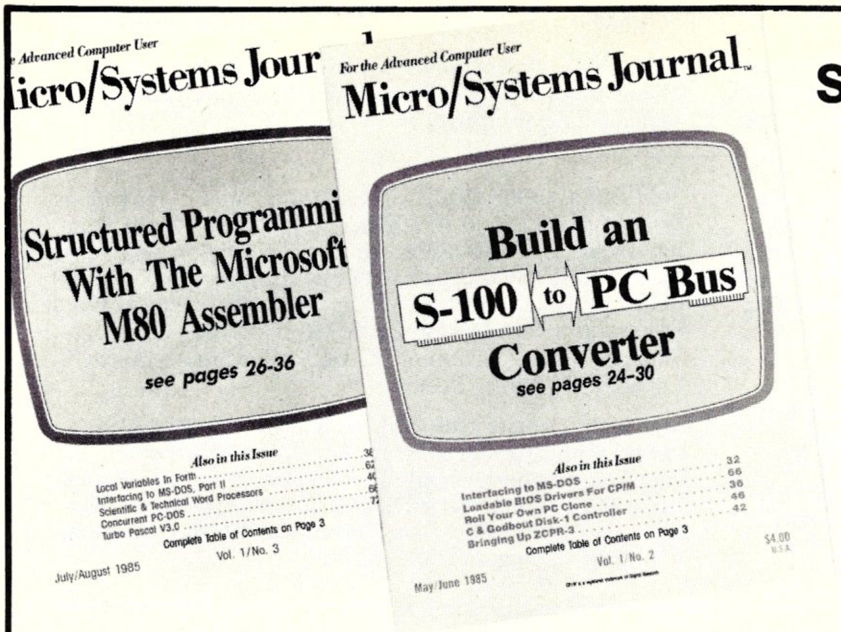
dBASE-II

The dBaseII Cash Manager: Cash Receipts/Cash Diskbusements for the Small Owner or Accountant; Heiser/Pickney; $15.95

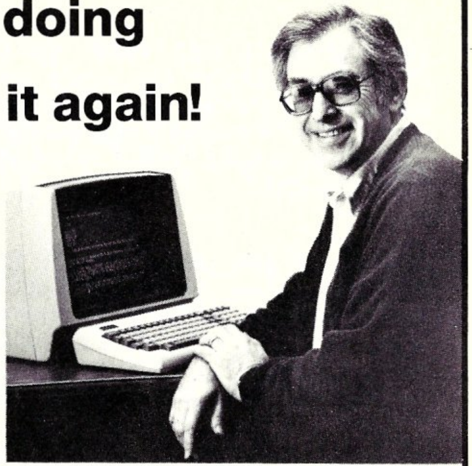
Advanced dBaseII User's Guide, Green; $30

dBase Guide For Small Business, Schademaid/Dickey; $25.95

Using dBase III, Jones; $18.95


For the Advanced Computer User
Micro/Systems Journal
Structured Programming With The Microsoft M80 Assembler
see pages 26-36
Also in this Issue
Local Variables In Forth
Interfacing to MS-DOS, Part II
Scientific & Technical Word Processors
Concurrent PC-DOS
Turbo Pascal V3.0
Complete Table of Contents on Page 3
Vol. 1/No. 3
July/August 1985
For the Advanced Computer User
Micro/Systems Journal
Build an S-100 to PC Bus Converter
see pages 24-30
Also in this Issue
Interfacing to MS-DOS 32
Loadable BIOS Drivers For CP/M 66
Roll Your Own PC Clone 36
C & Godbout Disk-1 Controller 46
Bringing Up ZCPR-3 42
Complete Table of Contents on Page 3
Vol. 1/No. 2
May/June 1985
$4.00 U.S.A.

Sol Libes is doing it again!



The only magazine by and for advanced micro users.

Micro/Systems Journal is the independent magazine for advanced software and hardware hackers who use MS/DOS, CP/M, MP/M, TurboDOS, Concurrent-DOS and Xenix. An important resource for programmers who use C, Turbo Pascal, Forth, Lisp and Assembler. It also covers hardware based on the S-100, PC and AT busses, as well as powerful single board and multi-user systems. Its name — MICRO/SYSTEMS JOURNAL!

Each issue is packed with practical articles...in depth tutorials, reviews, hints...the latest information on SIG/M, PC/Blue, C-User Group and PC-SIG Public Domain Software. Advanced topics such as 16-bit and 32-bit microprocessors, high performance data acquisition systems, interfacing to peripherals, patching application software packages for improved performance...and lots more! Strictly technical info...no fluff...stuff to keep every hacker up-to-date on the ever-changing micro technology.

Edited by Sol Libes, former editor of Microsystems, former columnist in Byte, PC Tech Journal, PC Week, Computers & Electronics and other magazines and author of 15 books. An authority in the field since the early seventies.

Here are examples of topics covered in recent issues.

- Build an S-100 to PC-Bus Interface
- Loadable Drivers for CP/M
- Structured Programming with Microsoft's M80 Assembler
- Review of Three 16-Bit Lisp & Prolog Packages
- Local Variables in Forth
- Installing ZCPR3
- Build a PC Clone for Under $1,000
- Interfacing to MS-DOS
- Assembly Language Extensions For MS-Basic
- Reviews of Scientific Word Processors, Concurrent DOS and Turbo Pascal Version 3.0

TAKE ADVANTAGE OF SPECIAL LIMITED TIME OFFER.

Offer expires December 31, 1985 so hurry and mail your subscription now.

[ ] Send a sample issue (enclose $4.00, $6.00 other foreign)	
[ ] YES! Start my subscription!	
[ ] 1 YEAR*	$20.00
[ ] 2 YEARS**	35.00
[ ] 1 YEAR (Canada & Mexico)*	28.00
[ ] 2 YEARS (Canada & Mexico)**	52.00
[ ] 1 YEAR (other foreign)*	35.00
[ ] 2 YEARS (other foreign)**	64.00

* 6 issues **12 issues

Checks must be payable in U.S. funds by a U.S. bank

Name: ______________________

Address: ______________________

____

City: ____________ State: ________ Zip: ________

make check payable to: MICRO/SYSTEMS JOURNAL
Box 1192
Mountainside, NJ 07092

Micro/Systems Journal™ Back Issues

The following back issues are available at $4.50 per copy ($6 foreign, cash, Postal Order or U.S. bank check) including shipping. If order 3-9 copies deduct 10%, 10 or more copies deduct 15%. Orders can be combined with back issues of Microsystems for discount. Send to: M/SJ, Box 1192, Mountainside, NJ 07092.

MARCH/APRIL 1985 (Vol 1., No. 1): Bringing Up CP/M Plus, Assembly Language Extensions for MS-Basic, New Tricks for CP/M2.2, Building an IBM/PC or XT Clone, Extended Single Density Storage, Variable Size Arrays in C;REVIEWS: dBase-III and 16-Bit Lisp & ProLog-Part I.

MAY/JUNE 1985 (Vol 1., No. 2): Build an S-100 to PC-Bus Converter, Interfacing to MS-DOS Part-1, Loadable Drivers for CP/M2.2, Roll Your Own PC-Clone, Bringing Up ZCPR-3, C & Godbout Disk-1 Controller, Writing Translation Programs in C and Turbo Pascal; REVIEWS: 16-Bit Lisp & ProLog-Part II.

July/August 1985 (Vol 1., No. 3): Structured Programming With Microsoft M80 Assembler, Local Variables In Forth, Interfacing to MS-DOS Part-II, Data Translation with Turbo Pascal, Implementing Sets with Bit Operations in C, A Unix Mail List System; REVIEWS: Scientific & Technical Word Processors - Part I, Macrotech MI-286 S-100 CPU Card, Slicer System, Concurrent PC-DOS, Concurrent PC-DOS, Coherent Operating System.

September/October 1985 (Vol. 1., No. 4): Who Prints on Printer With CCP/M, Interrupt Borrowing With Turbo Pascal, Upgrading CompuPro I/O Boards, Program Interfacing to MS-DOS Part-III, SIG/M & PC/Blue Public Domain Software Distribution Points, TurboDos INSTALL program; REVIEWS: CompuPro 10+, TEX.

Microsystems Back Issues

Some of the issues of the old Microsystems magazine are still available in very limited quantities. They are $4 per copy ($5.50 foreign, cash, Postal Order or U.S. Bank check) including shipping. If ordering 3-9 copies deduct 10%, 10 or more deduct 15%. Orders can be combined with M/SJ back issues for discount. Send to: M/SJ, Box 1192, Mountainside, NJ 07092.

1984

FEBRUARY: Using WordStar to Create Mailmerge/DBase-II files; Moving data files between CP/M software packages; Datestamp DBase-II; CP/M 2.2 Deblocking; Building S-100 diagnostic hardware; Enhance CP/M+ with RSX; REVIEWS: DBase-II, S-100 Mainframes, DRI Display Manager, AutoDex, Turbo Pascal.

JANUARY: Enhancing MP/M - Part 1; Installing MP/M; Add Concurrency to MP/M; Two Users on CP/M; Relocating Assemblers & Linkage Editors - Part 3; S-100 Wait States; REVIEWS: MP/M-8/16, ProComp-8, Paragraphics Game Board, ProLog.

1983

DECEMBER: CP/M Software Directory; A Debug Subroutine; Implement IOBYTE on North Star; Floppy Disk Problems; Improve Trig Functions in CBasic-80; Build Cheap S-100 Memory; Extended Memory Management; CP/M-86 BDOS Calls; REVIEWS: XLISP, LISP/80, TLC LISP, APC Basic, Microdynamics S-100 EProm Programmer, Ackerman S-100 Digital Synthetalker, Digital Research 16K & 32K S-100 Memory cards.

NOVEMBER: Intro to 80286, 68000, and 16032 Microprocessors; Intro to Local Area Networks - Part 2; Extended Memory Management for older S-100 Systems; Notes on Microsoft Fortran-80; Building S-100 Parallel Ports; REVIEWS: CompuPro CPU-68K, System 8/16, Xenith Z-100, Nevada & Ellis Computing Fortran.

OCTOBER: Intro to Local Area Networks, Part-1; Build Low-Cost LAN; Build S-100 Bubble Memory Card; Use Radio Shack Model 100 portable with a CP/M system; Write Menu-Driven Utility for Setting Printer Options; North Star Improvement; True Z-80 Random Number Function; Hide Code in Basic REM statements; Machine Code loader for MBasic; Increase Single-Density Disk Formatting; Relocating Assembler & Linkage Editors, Part-2; Run MX-80 with North Star; User Group Directory; CP/M-86 Versus CP/M-80; REVIEWS: CP/NET, QBAX, S-Basic.

AUGUST: XERA Program; Logging-On CP/M; WordStar Date/Time Patch; Find Location of Variable in NorthStar Basic; Prevent System Crashes During Warm Boot; Enhance Spreadsheet Print Files; Plotting Package-Part 3; Run WordStar under TP/M; 50-line Text Formatter; Using the LU Utility; User Areas under CP/M; REVIEWS: Stiff Upper Lisp, MuLisp-80, Supersoft Lisp, Cromenco C-10, Access Manager, Fancy Font, Computime SBC-880 S-100 card.

JULY: Using RCPMs; RCPM Directory; PIP Data Between Computers; Toward Smarter Modem Programs; Interface MX-80 via Parallel Interface; Digital Audio On CP/M System; Customize CP/M CBIOS; Plotting Package Part-2; REVIEWS: DRI PL/I-86 and PL/I-80, S-100 PMMI MM-VT1.

JUNE: Plotting Package Part 1; Drive HP Plotter; Laboratory Graphics Applications; Console Keypressed interrupts; Cutomize Wordprocessor Keyboard; WordStar Patch hor H-19/Z-19 Terminal; Relocatable Code; REVIEWS: Graftalk, JES S-100 Graphics Controller, ZCPR2.

1982

NOVEMBER/DECEMBER: CP/M Vs MS/DOS; CP/M-86 Vs MS-DOS; Intro to ADA Part 2; Virtual Disk for NorthStar; CP/M Program Auto-execute; Macros & Macro-Assemblers; REVIEWS: Janus, Aztec-C, C/80, Morrow S-100 M26 Hard Disk System, Teleram S-100 Bubble Memory Card, Jade S-100 Bus Probe.

Microsystems Back Issues *continued*

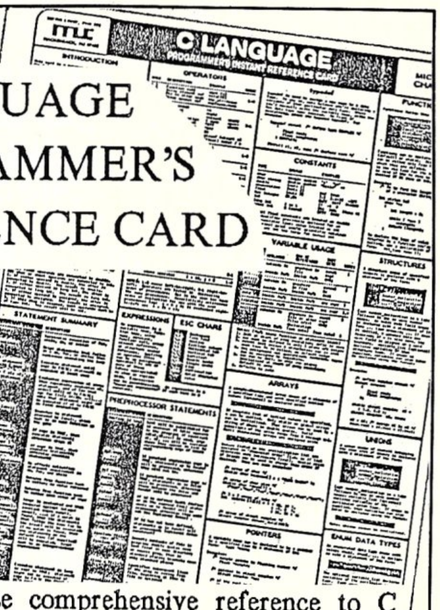
JULY/AUGUST: Hardware Random Byte Generator; Error Detection & Correction Codes; Getfile CP/M Utility Program; CP/M Patches; CP/M Application Notes; Run old NorthStar programs under new DOS; Cloning Disk Drives; Low Cost Floppy Disk Power Supply; Intro to Computer Graphics; Using Supersub Utility; REVIEWS: D80, RAID-8080, Three Macro-Assemblers, PDS, Cer-Tek S-100 UniProm Board; GrafPak.

MAY/JUNE: Intro to DBMS; Three ways to implement a mail list; Cursor Addressing; Structured Programming in Basic; Replacement for CP/M Submit; CP/M Disk Directory & Table Secrets; Mods for SDS VDB-8024; Run NorthStar Basic with CP/M; REVIEWS: DataStar, MDBS, TIM, Mince, ZDM.

1981

JULY/AUGUST: 16-Bit Disk Operating Systems; Input Queuing For NorthStar; Variable Speed Automatic Slow Step; Build S-100 Clock/Calendar Card; REVIEWS: TEC-86 System, Seattle Computer 8086 System, AlphaMicro, Godbout Dual Processor, CP/M-86, Televideo 920-C Terminal.

NOVEMBER/DECEMBER: Introduction to the C Language, Virtual Segment Procedures, Little-Ada Part-II, A Disk Alignment Routine, Northstar DOS/BIOS Directory & File Conversion using UCSD Pascal; REVIEWS: BDS-C, Small-C, Tiny-C and Whitesmiths-C; Tarbell Double-Density Disk Controller



CLASSIFIEDS

Micro/Systems Journal will accept Classified Ads. The charge is $6/line (3 lines minimum); 50 characters maximum per line. Three times Frequency $5/line; six times $4/line; non-profit clubs $2/line. Logos, special type, etc., are an extra charge. Check must accompany ad copy. Send to **Micro/Systems Journal**, POB 1192, Mountainside NJ 07092.

S-100 CARD SALE

S-100 ENTHUSIASTS! Write the **S-100 BOARD BANK** for unusual, scarce USED items. For example: 88-PCI 8-ch. relay/opto-isolator card, new. $95 DM6400 64K RAM card, used untested.........$95 Let's help keep S-100 bus alive and usable. Mail order only, write us about your needs. Subscribe to our goody list. Send $4.37 or more to **S-100 BOARD BANK**, Mailorder Dept., M1-Box 344, Olympia WA 98507.

$995 LAP-TOP CP/M MACHINE

NEC 8401A with serial and parallel interfaces, 300b modem built-in, battery powered, and 16 line x 80 col. display. **Interface Technology, Inc.**, Box 3040, Laurel MD 20708, (301)490-3608.

WANT S-100 CARDS!

BOARD BANK BUYS 'EM ALL! (Well, almost.) ROM- RAM- CPU- VIDEO-GRAPHICS- I/O- AD/DA- FDC-HDC- SPEECH- the uniqer the better! Convert your extras to CASH. Send FULL description, CHEAPEST asking PRICE to **S-100 BOARD BANK,** Mailorder Dept., M1-Box 344, Olympia WA 98507.

FOR SALE: Complete set of Microsystems magazines, January 1980 through November 1984. Will take best offer. Call Jerry Throckmorton (513) 426-8781 or write, 2706 Lillian Lane, Fairborn OH 45324.

WANTED: Issue March/April 1982 of Microsystems. Will pay $10 for a good copy. Don Pannell, 6428 Edgemoor Wy, San Jose CA 95129; (408)-253-5108.

JONTEL BUSINESS SYSTEM

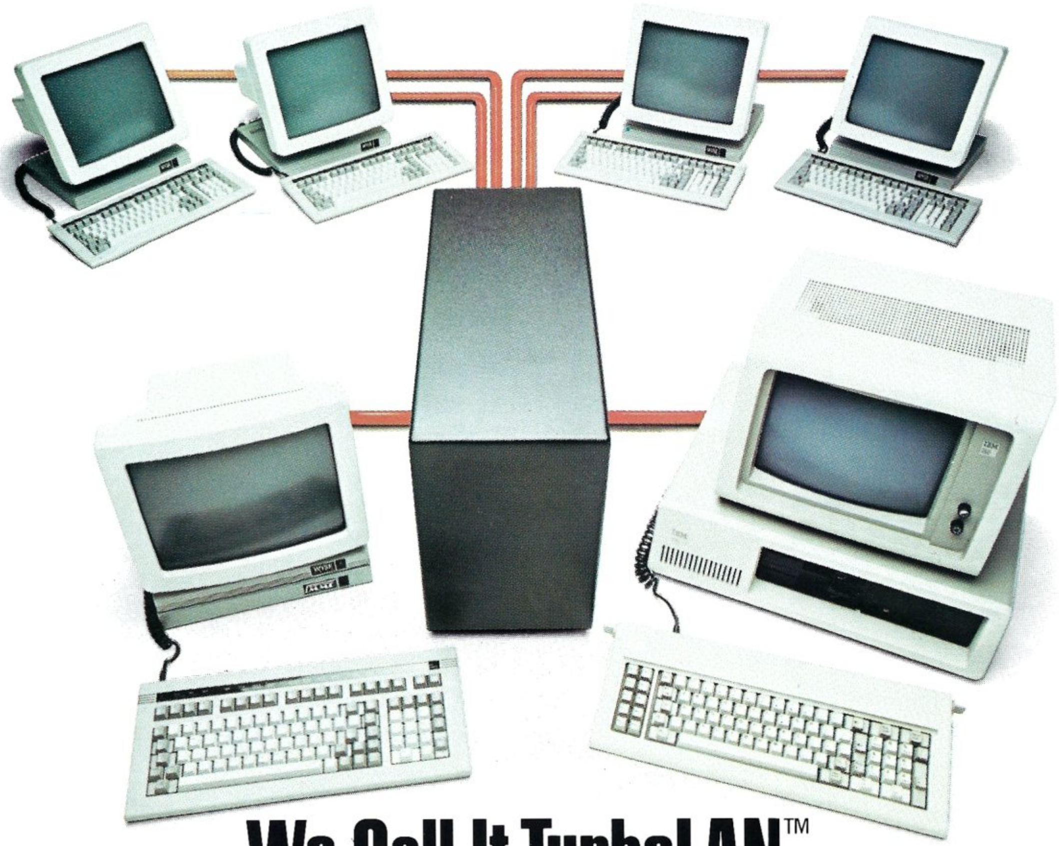
The Jontel Business System is a five journal accounting package based on a dual disk CP/M computer. Included with the seven disk package are mailing and inventory programs with 100 + pages of documentation on disk. 8 & 5 inch disk formats available. $69.95. **Moonlite Computer,** 707 Edge Hill Rd., New Bern, N.C. 28560 Source code available for custom installations, compiled CB80. (919) 638-6976.

DISK CONVERSION SERVICE

Over 300 CP/M and MS/Dos (PCDos) formats. Originals and copies returned in 48hrs. Reasonable rates. Personal service. For information and list of available formats, write: **RH Associates**, Box 7035, Silver Spring MD 20907, or call (301)587-6230.

Advertiser Index

They Called It Impossible.



We Call It TurboLAN.™

Imagine IBM PCs, intelligent workstations and S-100 BUS multiuser systems all on the same local area network.

Impossible?

Not with TurboLAN, Intercontinental Micro's networking solution.

The key is a unique hardware and software approach to local area networks.

You get the flexibility of 8-bit and 16-bit processors on the same network, the cost savings of shared peripherals, and expandability to over 4000 users.

You simply organize the network the way you want, mixing IBM PCs, S-100 BUS multi-user systems, S-100 BUS file servers, and workstations.

Intercontinental's sophisticated single board computers, 8-bit and 16-bit slaves, memory and interface boards make your job even easier. TurboLAN will work with almost any S-100 BUS system including NORTHSTAR, IMS, Teletek, and any other S-100 master featuring phantom deselection or extended addressing capability.

The ability to run PC-DOS, MS-DOS, CP/M, MP/M, and other applications programs means whatever the future holds, TurboLAN is ready.

And so is Intercontinental Micro.

You see, our S-100 BUS products revolutionized an industry. Intercontinental's slaves, masters and memory boards introduced advanced features like Direct Memory Access and Memory Management to micro environments. TurboLAN.™ The new networking solution from the company that's been building networking solutions for years.

Still think TurboLAN's impossible?

Then call, write or circle the bingo number below—we'll send you complete information on TurboLAN and descriptions of all our products, including the most complete S-100 BUS product line in the business.

OUR COMPLETE TURBOLAN AND S-100 BUS PRODUCT LINE

CPS4800X-SINGLE BOARD COMPUTER.
4 or 6 MHz processor (Z80A/B), onboard floppy disk controller, 64K RAM, 4 channel DMA controller, 24 line parallel I/O port, two serial I/O channels, real time clock. Memory mapped or I/O mapped capability.

CPS-16X—16-BIT 8086 SLAVE PROCESSOR.
256K, 512K or 1MB RAM, bank selectable, memory mapped, two serial ports, 24 line parallel I/O port, 8 MHz.

CPS-BMX/MX—Z80 SLAVE PROCESSORS
4 to 6 MHz processors, 64K RAM or 128K bank selectable memory, two serial ports, 24 line parallel I/O port.

256KMB—MEMORY BOARD.
Hard disk cache, linear addressable to two megabytes, bank selectable in 16K increments, configures for phantom deselection, parity error detection.

LANS-100—TURBOLAN FOR S-100 BUS SYSTEMS
ARCNET controller meets 696.2/D2S-100 spec, coax cable interface, 255 nodes per network segment, 2.5 megabit/sec. data rate.

LANPC TURBOLAN FOR THE IBM PC
Plug-in expansion ArcNet board. Custom software drivers integrates IBM PC into TurboLAN networks. 64K or 256K RAM options available.

WS80X-DISKLESS WORKSTATION
Converts almost any dumb terminal into intelligent workstation with networking capability. Floppy and hard disk options available.

PERSONALITY BOARDS—
SASI, Centronix, PRIAM, Clock/Calendar, RS232, Modem, RS422, long distance serial communications (up to 4000 Ft.)

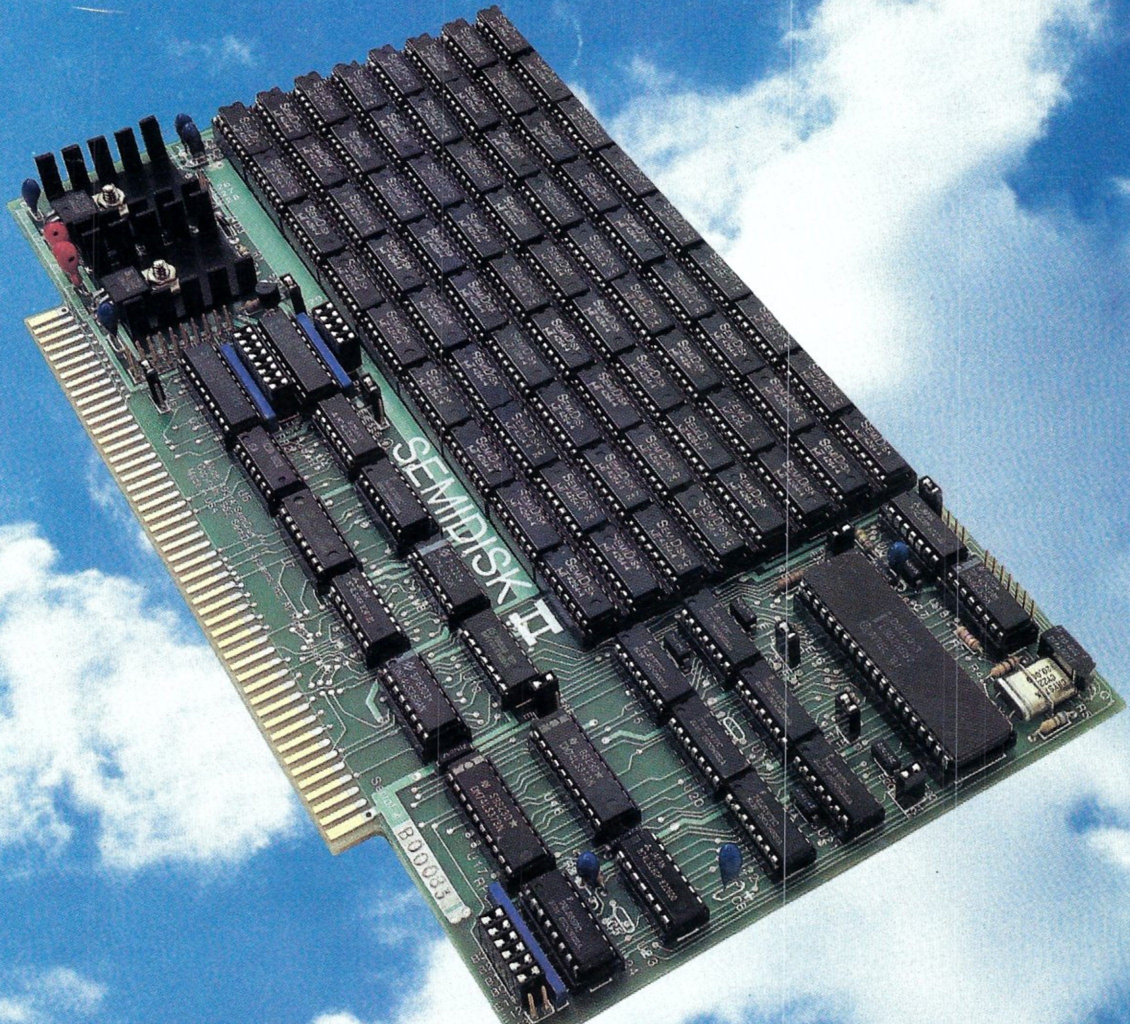
Intercontinental
Micro Systems

4015 Leaverton Ct., Anaheim, CA 92807, (714) 630-0964, TELEX: 821375 SUPPORT UD

TurboDOS is a Trademark of Software 2,000, Inc. ARCnet is a Trademark of Datapoint. CP/M, CP/M 86, MP/M, MP/M 86, CP/M PLUS, Concurrent CP/M 86 are Trademarks of Digital Research. PC-DOS, IBM-PC are Trademarks of International Business Machines. MS-DOS is a Trademark of Microsoft. TurboLAN is a Trademark of Intercontinental Micro Systems.

TIME & MONEY

SemiDisk is the disk emulator that saves you more of both.



We've just done something we know you'll like. We've made the SemiDisk far more affordable than ever before. With price cuts over 25% for most of our product line. Even our new 2 megabyte units are included.

COMPARE WITH THE OTHERS.

SemiDisk Systems builds fast disk emulators for more microcomputers than anyone else. S-100, IBM-PC, Epson QX-10, TRS-80 Models II, 12, and 16. Up to 2 megabytes per board, standard. Up to 8 megabytes per computer, using only four bus slots, max! Software drivers are available for CP/M 80, MS-DOS, ZDOS, TurboDOS, VALDOCS 2, and Cromix. SemiDisk turns good computers into great computers.

GREAT NEWS FOR IBM PC AT USERS.

New MS-DOS 3.0 software drives take full advantage of the 80286's expanded instruction set, moving data four times faster than can be done on PC or XT.

BATTERY BACKUP, TOO.

At 0.7 amps per 2 megabytes, SemiDisk consumes far less power than the competition. And you don't have to worry if the lights go out. The battery backup option gives you 5-10 hours of data protection during a blackout. Nobody else has this important feature. Why risk valuable data?

IT'S EXPANDABLE TO 8 MEGABYTES.

You can start with as little as 512K bytes, and later upgrade to 2 megabytes per board. At your own pace, as your needs expand.

In an IBM PC, XT, and AT you can have multiple drives on a single system. And the SemiDisk gives you mainframe performance on the top of your desk.

AND THE BEST NEWS IS SAVED FOR LAST:

	512K	1Mbyte	2Mbyte
SemiDisk I, S-100	**$695**	**$1395**	
SemiDisk II, S-100	**$995**		**$1995**
IBM PC, XT, AT	**$695**		**$1795**
QX-10	**$595**		**$1795**
TRS-80 II, 12, 16	**$695**		**$1795**
Battery Backup Unit	**$150**	**$150**	**$150**

SEMIDISK

SemiDisk Systems, Inc.
P.O. Box GG, Beaverton, Oregon 97075
503-642-3100

Call 503-646-5510 for CBBS/NW, 503-775-4838 for CBBS/PCS, and 503 649-8327 for CBBS/Aloha, all SemiDisk-equipped computer bulletin boards (300/1200 baud). SemiDisk, SemiSpool trademarks of SemiDisk Systems. CP/M trademark of Digital Research.