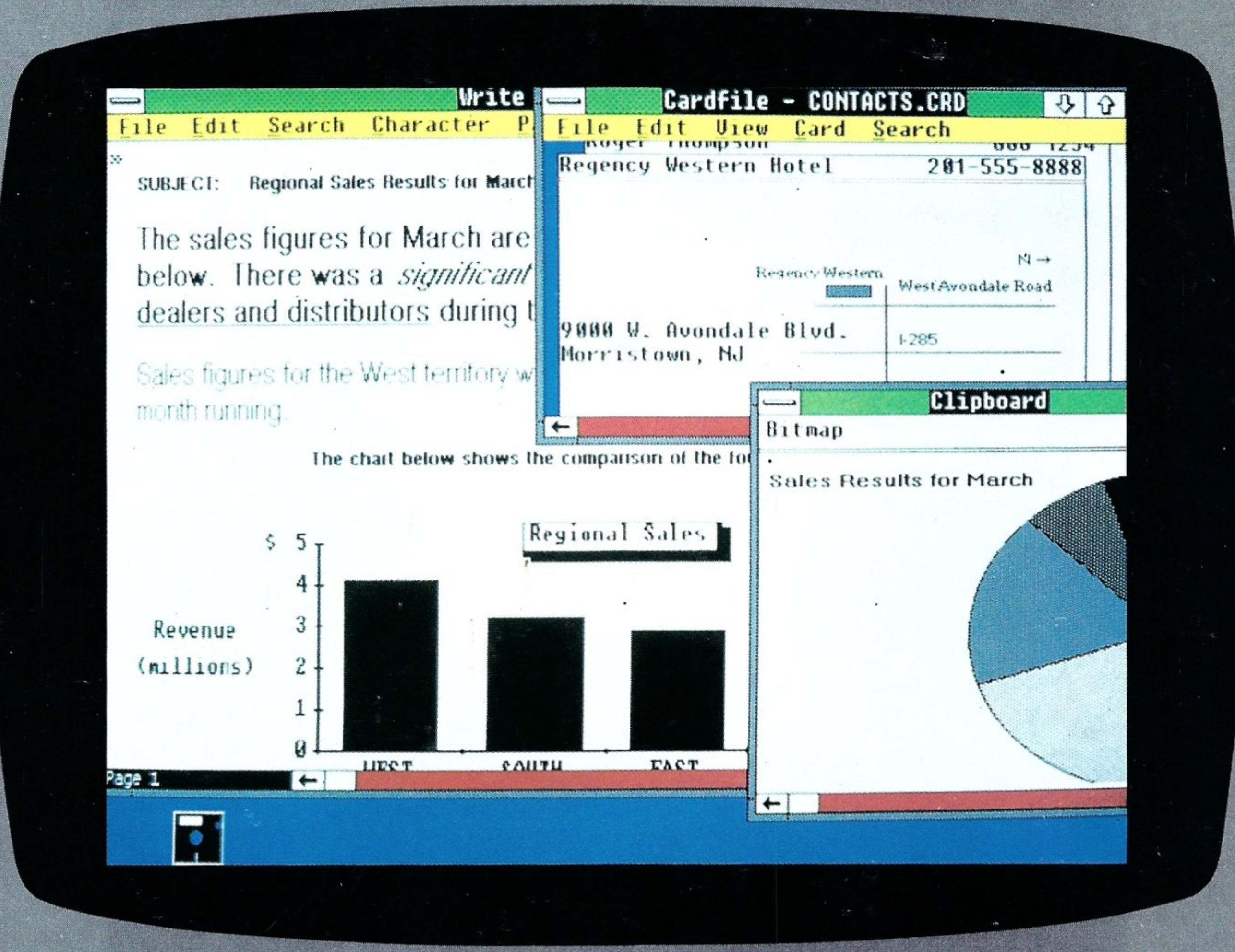
Vol. 3 No. 4 July/August 1987 $3.95 USA

# Micro/Systems Journal™

*For the Advanced Computer User*



SUBJECT: Regional Sales Results for March

The sales figures for March are below. There was a *significant* dealers and distributors during t

Sales figures for the West territory w month running.

The chart below shows the comparison of the fo

Roger Thompson 555-1234

Regency Western Hotel 201-555-8888

9000 W. Avondale Blvd.
Morristown, NJ

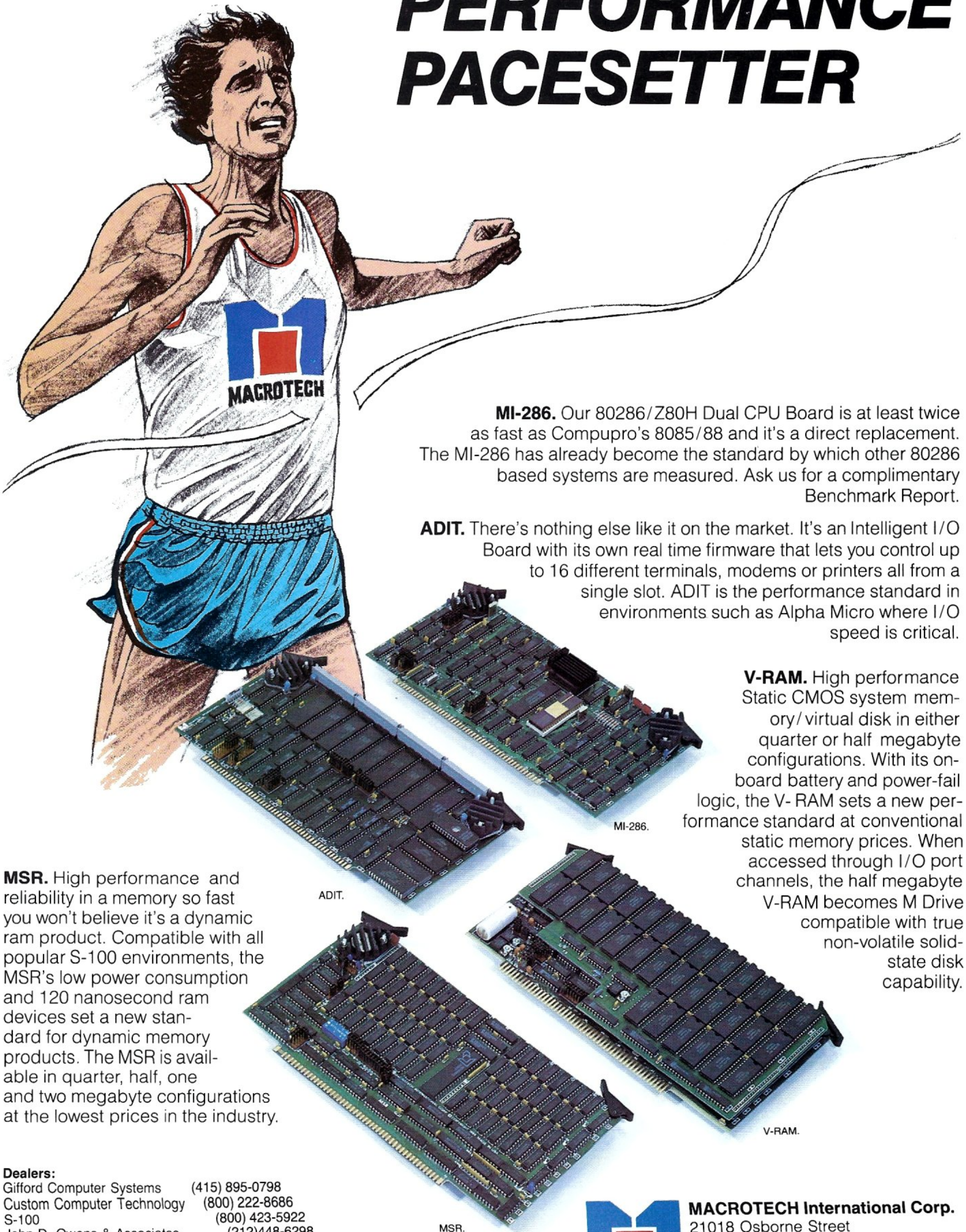
Sales Results for March

# *Program Interfacing to Microsoft Windows*

see page 44

*Also in this issue:*

# MACROTECH—STILL THE S-100 PERFORMANCE PACESETTER



**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 on-board 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.

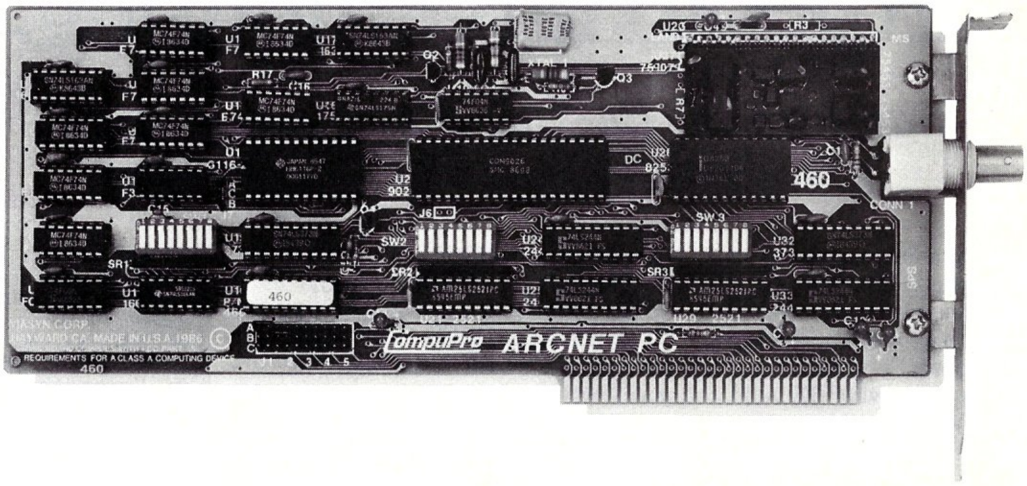
**MACROTECH International Corp.**
21018 Osborne Street
Canoga Park, CA 91304
**(800) 824-3181 • in Calif. (818) 700-1501**
**Telex: 9109970653**

# The Better Net

TESTED AND APPROVED FOR COMPLIENCE WITH FCC CLASS A (COMMERCIAL) USE

## The CompuPro ARCNET® PC Board

NOVELL NETWARE® TESTED


CompuPro ARCNET PC
460
VIASYN CORP.
HAYWARD, CA MADE IN U.S.A. 1986 ©
REQUIREMENTS FOR A CLASS A COMPUTING DEVICE

### Better Compatibility

The **CompuPro ARCNET PC** is compatible with all versions of Novell Netware and Advanced Netware™, *and* DR-NET™ running in all types of PC-compatibles. No other board can say that!

### Better Reliability

The **CompuPro ARCNET PC** board offers distinctly more reliability than other boards. Why? Read on.

### Better Design

The **CompuPro ARCNET PC** board uses unique circuit design to increase signal margins. Better interface circuitry insures operation with even the fastest PC-compatibles.

### Better Specs

The **CompuPro ARCNET PC** board meets all FCC requirements!

### Better Made

The **CompuPro ARCNET PC** board is manufactured, tested and burned in right here in the USA. It is not subject to the wide swings in quality often experienced with foreign-manufactured boards.

### Better Value

All of this adds up to the fact that the **CompuPro ARCNET PC** board is the best value in Novell compatible network boards.

### Better Call Today

Call us today for the name of your nearest **CompuPro** dealer. They are prepared to discuss a variety of network and multi-user solutions to meet your requirements.

CompuPro™

Viasyn Corporation
26538 Danti Court
Hayward, CA 94545-3999

(415) 786-0909
TWX: 510-100-3288
Easylink Mailbox 62877579

Trademarks: CompuPro: Viasyn Corporation. ARCNET: Datapoint Corporation. Netware, Advanced Netware: Novell, Inc. DR-NET: Digital Research Inc.

# On April 2, 1987 IBM and Quarterdeck announced the next generation in personal computing.

*Introducing DESQview 2.0. Improving the past and ready for the future right now.*

In one sweeping announcement from Miami Beach and New York City, IBM established new standards of performance for personal computers, with its new Personal System/2.™ Quarterdeck was there with IBM and simultaneously helped establish new standards for multi-tasking and multi-windowing.

We were there for them then. We're here for you now.

If you use two or more software programs, if you use a PC-compatible machine, if you own a new 80386 computer, if you've just bought one of the new Personal System/2 computers, or if you've tried Microsoft Windows and were disappointed but still need the power of graphics programs, DESQview 2.0 is the answer.

Consider this. InfoWorld voted DESQview's earlier version 1986 Product of the Year. SoftSector gave it the Editor's Choice Award. In PC Tech Journal's "System Builder Contest" at Comdex Fall it was voted best operating environment. And 450,000 dedicated users on four continents have voted yes with their dollars.

The new DESQview 2.0 is an order of magnitude better.

This unique software program enhances the power of your personal computer and makes it more convenient to use. It still gives your PC the power of many PCs. It still does windows. It still multi-tasks. It still breaks the DOS 640K barrier. It still transfers data. It still dials your phone. It still gives you menus for DOS. It still remembers your keystrokes (macros). It still runs your existing programs and your new programs soon to come. In fact now you can even run Windows-, GEM-, and Topview-specific programs too. And with 386 machines and our Expanded Memory Manager it still becomes a 386 control program, but now you can run text *and* CGA graphics programs in background.

The new DESQview 2.0.

For us it's the next logical step. For you it's windows of opportunity.

SYSTEM REQUIREMENTS

•IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286 or 80386 processors) with monochrome or color display; IBM Personal System/2 •Memory: 640K recommended; for DESQview itself 0-145K •Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage •Disk: Two diskette drives or one diskette drive and a hard disk •Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA), IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA) •Mouse (Optional): Mouse Systems, Microsoft and compatibles •Modem for Auto-Dialer (Optional): Hayes or Compatible •Operating System: PC-DOS 2.0–3.3; MS-DOS 2.0–3.2 •Software: Most PC-DOS and MS-DOS application programs; programs specific to TopView 1.1, GEM 1.1 and Microsoft Windows 1.03 •Media: DESQview 2.0 is available on either 5¼" or 3½" floppy diskettes

Rush me DESQview 2.0! Today! **MSJ787**

| No. of Copies | Media 3½"/5¼" | Product | Retail Price ea. | Total |
|---|---|---|---|---|
| | | DESQview 2.0 | $129.95 | $ |
| | Shipping & Handling | USA<br>Outside USA | $ 5.00<br>$ 10.00 | $ |
| | | Sales Tax (CA residents) | 6.5% | $ |
| Payment: □ Visa □ MC □ AMEX □ Check | | | Amount Enclosed | $ |

Credit Card: Valid Since ________/________ Expiration ________/________

Card Number: □□□□□□□□□□□□□□□□

Credit Card Name ____________________

Shipping Address ____________________

City ____________ State ________ Zip ________ Telephone ________

Mail to: Quarterdeck Office Systems, 150 Pico Boulevard, Santa Monica, CA 90405

NOTE: If you own DESQview call us for a special upgrade offer, or send in your DESQview registration card and we'll send you upgrade information.

DESQ view™
Quarterdeck

Quarterdeck Office Systems • 150 Pico Boulevard, Santa Monica, CA 90405 • (213) 392-9851

DESQview is a trademark of Quarterdeck Office Systems. AboveBoard is a trademark of Intel Corporation.Hayes is a trademark of Hayes Microcomputer Products, Inc. IBM, PC, Personal System/2 and TopView are trademarks of International Business Machines Corporation. Microsoft Windows and MS are registered trademarks of Microsoft Corporation. Mouse Systems is a trademark of Metagraphics/Mouse Systems. RAMpage is a trademark of AST Research, Inc. GEM is a trademark of Digital Research. Hercules is a trademark of Hercules.

July/August 1987 Vol. 3 No. 4

# Micro/Systems Journal™

*For the Advanced Computer User*

***About the cover:*** *Microsoft Windows allows application programs to run in overlapping windows on the screen. The catch is that the applications have to be written to work with Windows. This month's cover story shows how.*

## *Feature Articles*

## *Product Reviews*

## *Columns*

## *Departments*

**Editorial**

| | |
|---|---|
| **Founder and Editor** | *Sol Libes* |
| **Technical Editor** | *Don Libes* |
| **Associate Editors** | *Lennie Libes*<br>*Susan Libes* |
| **Contributing Editors** | *A.G.W. Cameron*<br>*Michael Cherry*<br>*Ian F. Darwin*<br>*Stephen R. Davis*<br>*B.J. Hall*<br>*Charles H. Strom* |
| **Consulting Editor** | *Michael Swaine* |
| **Editorial Coordinator** | *Kobi Morgan* |
| **Copy Editor** | *Rhoda Simmons* |

**Production**

| | |
|---|---|
| **Production Manager** | *Bob Wynne* |
| **Art Director** | *Kate Paddock* |
| **Production** | *Kobi Morgan* |
| **Typesetter** | *Lorraine Buckland* |

**Circulation**

| | |
|---|---|
| **Director of Circulation** | *Maureen Kaminski* |
| **Circulation Coordinator** | *Kathleen Shay* |
| **Circulation Assistant** | *Sarah Frisbie* |

**Administration**

| | |
|---|---|
| **Finance Manager** | *Kate Wheat* |
| **Business Manager** | *Betty Trickett* |
| **Accounts Payable Supv.** | *Mayda Lopez-Quintana* |
| **Accounts Payable Asst.** | *Aileen Abadam* |
| **Accounts Receivable Supv.** | *Laura Di Lazzaro* |
| **Accts. Receivable Clerk** | *Patricia Albert* |

**Advertising**

| | |
|---|---|
| **Advertising Director** | *Richard Mixter* |
| **National Account Mgr.** | *Tami Beaulieu*<br>*(213) 470-5959* |
| **Northwestern Account Mgr.** | *Patricia Bacci*<br>*(415) 366-3600* |
| **Advertising Coordinator** | *Charles Shively* |

**M & T Publishing, Inc.**

| | |
|---|---|
| **Chairman of the Board** | *Otmar Weber* |
| **Director** | *C. F. von Quadt* |
| **President and Publisher** | *Laird Foshay* |

*Micro/Systems Journal* (ISSN #8750-9482) is published bi-monthly by M & T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points.

**Article Submission:** Please write the editors at Micro/Systems Journal, Box 1192, Mountainside, NJ 07092 or call (201) 522-9347 to see if there is interest in a particular 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 writing.

Send a stamped self-addressed business size envelope (to NJ address) for a copy of our Author's Guide.

**Address Correction Requested:** POSTMASTER: send address changes to Micro/Systems Journal, 501 Galveston Dr., Redwood City, CA 94063. **ISSN 8750-9482**

**Change of Address:** Please send old label and new address to: Micro/Systems Journal, 501 Galveston Dr., Redwood City, CA 94063.

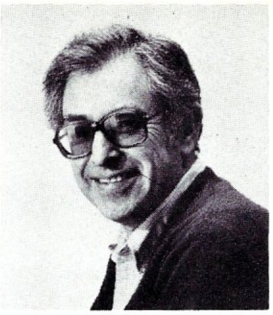
**Correspondence:** Editorial Office: Box 1192, Mountainside, NJ 07092; (201) 522-9347. The editor may also be contacted via MCI Mail (SLIBES). Business Office: 501 Galveston Dr., Redwood City, CA 94063; (415) 366-3600.

**Advertising Rates:** Available on request. Call (415) 366-3600 or write to: Micro/Systems Journal, 501 Galveston Dr., Redwood City, CA 94063.

**Subscription Rates:** U.S.: $20 for one year, $35 for two years. Canada & Mexico add $7 per year for airmail; $3 per year surface mail. Other countries add $12 per year for airmail.

# Editor's Page

Sol Libes



## Separating IBM's Fact from Fiction, or Whatever Happened to the PS/1 and OS/1

IBM's PS/2-OS/2 April 2 announcement was a masterpiece of showmanship that rivaled that of P. T. Barnum. It was preceded by months of leaked information, and when the announcement actually came, there was little if anything we did not already know. IBM accomplished what it set out to do, however—to make its major customers hold off from ordering competing systems.

The actual announcement came with presentations held in several cities across the country. Some rivaled rock concert spectaculars. This was followed by a three-month advertising campaign on TV, radio, and in the press costing more than $30 million.

Most of the announced products will not be available for months. Some will not appear until a year from the announcement. Some, if IBM is true to form, will never materialize. Customers will not be able to evaluate the products and place real orders for sometime. The result is to cause them to continue holding off from placing orders for competing systems that are currently available.

IBM followed the introduction with well-placed PR indicating that the PS-2 had gained quick momentum, that systems are in great demand, that it is receiving large orders and already has back orders. Few orders, however, appear to be coming from outside IBM's traditional large corporate customer base, to which the PS-2 has been aggressively marketed. Many of these early orders are based on volume purchase terms and IBM's promise finally to deliver mainframe connectivity. Distribution in the retail channel has been limited to key dealers. Thus sales of PC, PC/XT, and PC/AT systems are continuing at a robust pace.

IBM has again demonstrated a master talent for marketing. It reigns supreme in its ability to bring a product to the market's attention, to differentiate it from the competition, and to create barriers to direct competition. There is little doubt that PS/2-OS/2 will be a success in the corporate world. The question is whether it will also be successful in noncorporate environments. Many prognosticators express doubts and believe that DOS 3.x and the PC/XT/AT architecture will be around for a long time.

The thing that makes a computer system successful is application software written for it. Lotus 1-2-3, dBASE-III, and the many word-processing programs created a market for the PC. Desktop publishing software created a market for the Mac. We will have to wait for the release of OS/2 to see if new applications appear that make it attractive to move up to IBM's new, much more expensive 286 and 386 systems. This will surely happen, but it will take time—probably a year or more.

In the meantime corporate users are increasingly using the Apple Macintosh, a much more "user friendly" system. The new Mac-II and Mac-SE can provide MS-DOS, UNIX, Ethernet, and token ring features—all dear to the hearts of corporate PC users. The best of two worlds are coming together, and there is no doubt the Mac-II will do better in the corporate world than the original Mac did. Considering that Apple sold more than a million of the original Macs, mostly to business users, IBM will have a fight on its hands.

### What's New and What Isn't

Virtually all the new features of the PS/2 and OS/2 are already available with competing products. Improved performance is available from AT clones with higher clock rates and zero wait states. Optimum performance for Microsoft Windows is available with the Microsoft Mac-10 card.

Multitasking is available with many products—Quarterdeck's DESQview and DRI's Concurrent DOS, to mention a few. In fact, C-DOS also provides multiuser facilities. Expanded memory and large-size, hard-disk drives are also nothing new on the PC, and 640 × 480-pixel color graphics has been available for some time.

What is new is the "promise" of connectivity and SQL database facilities built into the operating system. These features are only announcements, however, and we will have to wait until next year to evaluate their significance. The likelihood is that bridge products will appear to provide these features for existing PC/XT/AT systems.

There are an estimated 10 million PC/XT/AT and compatibles already in use. You can expect to see many bridge products bring PS/2-OS/2 compatibility and performance to these systems.

Clone makers are studying the features of IBM's new Micro Channel carefully. Implementing this feature is more a legal problem than a technical one. The main feature of Micro Channel is the ID—a 2-byte signature word assigned by IBM to each plug-in card—that is used for arbitration and device selection. The system was actually created and patented by Computer Automation (Irvine, Calif.) to which IBM is paying a royalty for its use. Clone makers may therefore be able to license the system's use. There are also rumors that IBM may license PS/2 bus technology. Clone component suppliers such as Chips and Technologies (Milpitas, Calif.), Faraday Electronics (Sunnyvale, Calif.), and Phoenix Technologies (Norwood, Mass.) have already indicated that they will ship chip samples for the architecture by the end of the year. We may yet see another "clonefest." §

Will magazines such as *PC*, *PC Week*, *PC World*, et al; now change their names to *PS*, *PS Week* and *PS World*?

# News, Views & Gossip

by Sol Libes

## Random Gossip & Rumors

**IBM**, which is already shipping systems utilizing its new 1-Mbit RAM chip, has disclosed that it will be in volume production of a 4-Mbit RAM chip later this year. The chip has a reported access time of 65 ns, which should enable systems utilizing it to run at clock speeds of up to 25 MHz with zero wait states.

IBM has also reported that in 1986, for the first time, more profits came from its European operations than from its U.S. operations, although U.S. sales were greater.

**Apple**, which has already announced plans to ship an Ethernet interface option for its Mac-II that supports TCF/IP, is financing development of a Mac token-ring interface at Carnegie-Mellon University. Rumors are that Apple will include this on the next version of the Mac sometime next year and let a third-party market it for existing Mac-IIs.

**NEC**, the largest IC manufacturer in the world, is going into production of its V60 32-bit microprocessor rated at 3.5 MIPS. The V70, rated at 6 MIPS, is expected to be announced in July, and the V80, with on-board cache memory and rated at 10 MIPS, is expected next year. Hitachi and Fujitsu also have a joint effort to develop a 32-bit microprocessor.

**Apple** will soon have shipped its one millionth Macintosh and four millionth Apple II. Although far below the number of IBM PC, PC/XT, PC/AT, and compatible systems that have been made, it does indicate that Apple has a achieved a critical mass that ensures it can successfully compete with the likes of IBM and DEC. Apple has finally achieved a foothold in the business marketplace with its easy-to-use Macintosh and excellent desktop publishing facilities. There is no doubt that its new more powerful Macs and networking system will extend its ability to compete in the powerful workstation market.

**Sun Microsystems**, however, is slowly encroaching on Apple's territory with networking technology that links Macs, Apple IIs, PCs, and UNIX workstations. Sun's NeWS (Networking Windowing System) is a superset of the X Windows standard recently ratified by DEC and Apollo. Users will even be able to run Microsoft Windows in an X Windows.

**IBM** is rumored to be readying several new models for the PS/2 line. Included are a new portable, a diskless workstation, and a multiuser system running OS/2.

**Borland** is expected to release a 386 C compiler running under DOS 3.x and OS/2 early next year. It will break the 640K memory barrier and provide access to protected mode.

**Everex**, **Samsung**, and **Novell** are rumored to be readying 16-MHz AT-compatible file server systems. Novell is also rumored to be planning to release low-cost starter versions (up to five workstations) of its popular Advanced Netware. Pricing is expected to be less than $500 with the product becoming available in the fall.

Two Japanese companies (**Sumitomo Chemicals** and **Sumitomo Metal Mining**, Tokyo) will soon commence manufacture of erasable/rewritable optical magnetic disks. They should reach the marketplace next year. And **Sony** has announced that it will begin shipping erasable optical disk drives before the year's end.

**Atari** is promising shortly to begin shipment of a complete desktop publishing system for less than $3,000. The system will include a Mega-ST system, laser printer, and software.

## OS/2 & PS/2

What ever happened to OS/1? Was it ever a released product? Oh well, we should soon have OS/2.

Announced in April, the OS/2 "standard" version (1.0) is promised for year-end delivery. It will be able to support 16 megabytes of RAM, provide multitasking, and break the 32 megabyte disk limit of DOS. Version 1.1 will be released in the first quarter of next year and will add windowing. The delivery date for the "extended" version with communications and database support will be announced in December, however. The price of the standard version will be $325, and the extended version will be $795. And if you want to develop software to run under OS/2, note that the Microsoft OS/2 Toolkit costs $3,000—enough to discourage most of the small software developers.

OS/2's high price will no doubt prove a deterrent to single-system owners. Owners of a large number of systems will be able to purchase a low-cost (per system) site license. Hence OS/2 will most likely be limited mostly to large companies (IBM's most important market). Actually the price of OS/2 is now very close to the price of UNIX for the PC.

OS/2 will finally bring an operating system to the AT that takes advantage of the 80286 2½ to 3 years after its release. And, purchasers of IBM's new Models 50 and 60 (replacements for the AT) will only have to run old DOS for 6 months to a year before their operating system arrives. Purchasers of the Model 80 (80386) are going to have wait even longer for their operating system. New versions of DOS 3.x and Microsoft Windows were introduced for the Model 30 (8086-based) and XT systems.

On April 2, IBM introduced its Personal Systems/2 line of desktop systems, which replace the old PC/XT/AT systems. Aimed at the *Fortune* 500 (IBM's traditional customers), it left room for a lot of competition.

IBM has introduced some new features that are worth noting. The bus in its 286 and 386 systems is of a new design with bus manager hardware to improve throughput to plug-in devices. Thus clock rate will no longer be a meaningful measure of system performance. Several previously plug-in circuits are now on the motherboard, which should also improve performance. Color graphics has been significantly improved in resolution, colors, and speed with a lower price/performance ratio. Also, IBM has finally begun to resolve some of the desktop-to-mainframe connectivity problems with new hardware, software, and network offerings—most of which will not be available until next year. The new LAN products promise to reduce substantially the costs of connecting a station into a net.

This news should come as no surprise to readers of this column as these new products were almost exactly according to the rumors that appeared in my earlier columns.

The first reaction from competition was price cutting. Sun Microsystems slashed the price on its 3/50M workstation by 36 percent to $4,995. This was seen as a reaction to the new Mac as well as the IBM PS/2 Model 80, 80386-based system. Hewlett-Packard and Leading Edge cut

# Uninterruptible Power Systems

## Read what the experts say;

“Clary Corp's OnGuard 600VA has the makings of the perfect UPS: it's functionally invisible. Whether or not line voltage is available, it constantly supplies up to 600VA of nearly pure sine-wave power at an almost unvarying 120 volts. No UPS is more unobtrusive than the OnGuard.”

–Winn L. Rosch,<br>
PC Magazine,<br>
SEPTEMBER 16, 1986

For heavy-duty use with large systems, our choice is Clary Corp.'s OnGuard 600VA. It has sufficient capacity to run a network server with external hard disk(s) long enough to bridge most outages or shut down the network in an orderly manner. Also, it's quiet enough for an office area and simple enough to use that it can be forgotten.

–PC Magazine,<br>
Editor's Choice,<br>
SEPTEMBER 16, 1986

PROTECT YOUR IBM SYSTEM/34/36 FOR ONLY

**$4590.00***

**AVAILABLE IN 600VA, 800VA, 1.0 KVA, 2.0 KVA 2.5 KVA, & 3.0 KVA**

- Specifically designed for the IBM System/ 36 or equal.
- Capable of providing nearly twice its rated output capacity for today's non-linear computer loads.
- On-line operation, static bypass switch and sinewave output.
- Fifth generation, high frequency PWM MOSFET technology reducing size and weight by over 70%.
- Ideal for multi-user systems.
- Meets UL, FCC and IEEE587.
- Virtually silent.
- Plug compatible (no installation costs).
- Rack-Mountable.

CLARY®

ONGUARD™

**CLARY CORPORATION**<br>
320 West Clary Avenue<br>
San Gabriel, CA 91776 [818] 287-6111x658

***2.0 KVA IN QUANTITIES OF 4.**

their system prices by 10-15 percent, and 10-MHz AT clones are now expected to have starting prices of less than $1,000. XT clones will be around for a long time and prices should drop 5-15 percent. Don't look for any new XT systems to be introduced.

## IBM Micro Channel Specs

IBM has finally started shipping copies of its 110-page technical manual for PS/2-50/60/80 systems to selected OEMs on a "need-to-know-basis." The Micro Channel bus uses 116 pins. The model 50 has three bus slots and the 60 has seven (the model 80 had not yet been released when this was written). Both also have one slot dedicated to a video card; this slot has a second 20-pin connector. When a video card is plugged into this slot, it disables the video controller on the main board (IBM calls it the Planar board) and routes video output to the plug-in card.

The Micro Channel bus allows both 8- and 16-bit-wide data transfers between up to 16 megabytes of memory and up to 64K I/O ports. DMA can operate in both standard and burst modes. Bus arbitration is provided for up to 15 devices. IBM claims the Micro Channel provides improved error detection and recovery via level-sensitive interrupts shared on all levels.

The Micro Channel design provides Programmable Option Select (POS) registers that do away with DIP switches and jumpers. Thus all plug-in cards must have IDs that are recognized at boot time, allowing the system and software to configure themselves under program control. Cards can also be enabled and disabled by software. Peripheral-card producers must register their cards with IBM to receive their 2-byte IDs.

The Micro Channel bus also has two lines for the audio speaker, which can be used by modems, speech synthesizers, and so on. Also four lines are reserved by IBM for a future undisclosed use, and four of the pins are actually keys to prevent incorrect insertion. There are 29 power and ground lines distributed on the bus in such a way as reduce radio frequency radiation, and a steel bracket is provided on each plug-in card to act as an RF shield.

## Mac News

**Apple** is not sitting still. Look for it to announce a host of new products later this year that in many ways will make the IBM PS/2 and OS/2 announcements look sick. Look for a new Mac operating system with smooth context switching (from one application to another and return without having to close files), true multitasking, hierarchical menus (à la the Sun Microsystems workstations), the ability to address more than 8 megabytes RAM (the current Mac limit), and more. This new operating system is expected this fall and hence will appear at least 6 months (or more) before releases of the full version of OS/2.

While IBM is delivering 3½-inch disks capable of storing 1.44 megabytes (Models 50, 60, and 80 only), Apple will go to 1.6 megabyte floppies. Using compression techniques these disk can be used for hard-disk backup. The disks will also read and write IBM disks.

Apple will also introduce a CD-ROM (Toshiba XM 2000A) with SCSI interface using the High Sierra format drive. Also look for token-ring boards that can link Macs to IBM's new, faster, token-ring network announced by IBM in April.

## 386 Hardware Prices Dropping Fast

It's a paradox!—not the first and certainly not the last. With no software available to take advantage of the 386, people are rushing out to buy the things. It's like a repeat of the PC/AT. Although the PC/AT was introduced in 1984, IBM's 286 operating system (OS/2) is only just now being announced. Of course, several other software vendors have had 286 multitasking operating systems for quite a while—but who cares?

It didn't take the cloners very long at all to begin shipping 386 clones. For example, **Four Star Computers** (San Jose, Calif.; [408] 434-0735) is already offering a 386 motherboard with AT form factor and Phoenix BIOS for $1,500 and a complete system with 40-megabyte hard disk for less than $3,000. The system appears to be a clone of the Compaq 386 motherboard at a price that is almost half that of the Compaq 386 and pretty close to the price of an AT clone. And, I hear rumors of several more 386 systems in development that are ready to be modified into IBM 386 clones as soon as IBM ships its first systems.

**IBM**'s new 386 systems, when shipped, will be more than twice the price of most current systems. If the performance of the 386 clones compares favorably with that of the IBM 386 systems and Microsoft's 386-OS runs on them, the cloners may do well.

IBM is the only company to which Intel has given an 80386 license, and IBM is known to have developed several custom versions of the 386. The question is whether it is planning to use these chips. If so it will reduce production costs but should not prevent cloning.

## 386 Software Where Are You?

Although 386 systems have been in production now for ten months, no software is being shipped yet. Several multitasking 386 operating systems are being advertised and promoted, but official, released versions have yet to appear. Several are in beta test and should be out shortly. **Microsoft** and **IBM** do not expect to release their 386 operating system until late next year.

In the meantime IBM and Microsoft hope to hold the other operating systems at bay with an implementation of UNIX-386. This strategy will also provide connectivity to **Apple**, **DEC**, **Sun Microsystems**, and **Apollo Systems** via Ethernet. IBM has finally jumped on the connectivity/compatibility bandwagon. This, however, may turn out to be an interim strategy until OS/386 is released.

386 operating systems are far more sophisticated (meaning more complicated to develop) than are single-user systems. Hence, we can expect it to take a lot longer to bring application software utilizing the 386's features to market. Further, the development of 386 software will be much more expensive and the market will be more limited, and hence, expect that the pricing should be much higher than that of standard DOS software. The likelihood is that we will see little in the way of public-domain 386 software, just as we have not seen any public-domain software for Microsoft Windows. Writing software applications that run under these operating systems is much more difficult and beyond the ability of all but well-trained software developers.

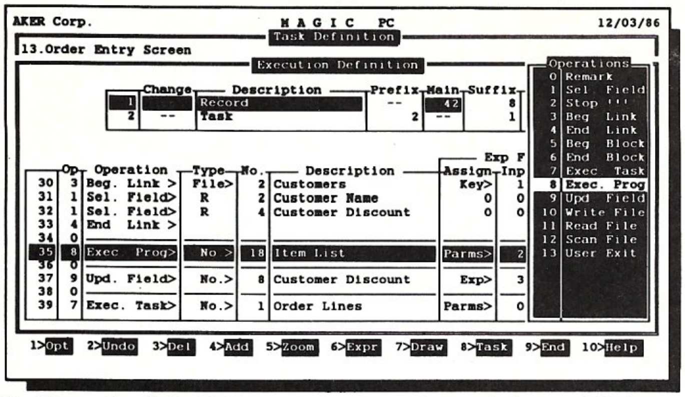
## Zilog Introduces the Z280

**Zilog** is sampling a new Z80-compatible CMOS processor called the Z280. Microcode compatible with the Z80, it has an expanded instruction set, operates at 10 MHz, has on-chip cache (256 bytes), memory management (system and user modes), and burst memory support. A 25-MHz version is expected to appear next year.

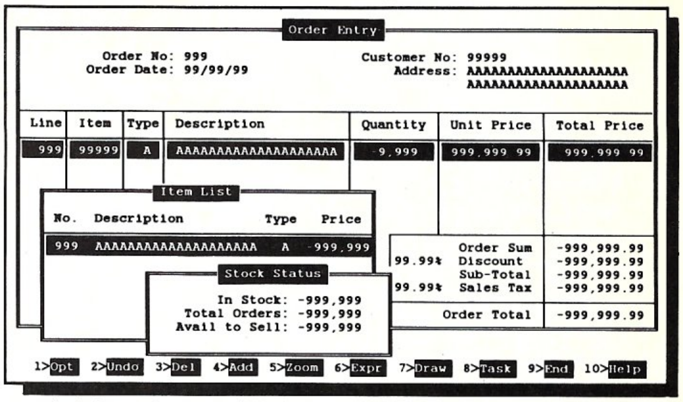
It has twelve 16-bit registers, four DMA channels with flow-through and fly-by modes, a DRAM controller, a full-duplex UART, and three 16-bit counter/timers. It is housed in a 68-pin package.

Zilog is also second sourcing the AT&T 32100 32-bit microprocessor and expects to second source the NEC V60/70/71 32-bit chips when they are released. Zilog is also in beta test on its own Z80,000 32-bitter. §

# Develop DB applications 10 times faster without the coding pain...you'll swear it's Magic



```
AKER Corp.                   M A G I C   PC                      12/03/86
                            Task Definition
13.Order Entry Screen
                        Execution Definition                  Operations
                                                               0 Remark
         Change   Description        Prefix Main Suffix        1 Sel   Field
      1          Record                --    42      8         2 Stop  '''
      2   --     Task                   2    --      1         3 Beg   Link
                                                               4 End   Link
                                                               5 Beg   Block
                                                   Exp F       6 End   Block
    Op  Operation    Type  No.  Description      Assign Inp    7 Exec  Task
 30  3 Beg.  Link >  File>   2 Customers           Key>   1    8 Exec. Prog
 31  1 Sel.  Field>   R      2 Customer Name          0   0    9 Upd   Field
 32  1 Sel.  Field>   R      4 Customer Discount      0   0   10 Write File
 33  4 End   Link >                                           11 Read File
 34  0                                                        12 Scan File
 35  8 Exec  Prog>   No >   18 Item List         Parms>   2   13 User Exit
 36  0
 37  9 Upd.  Field>  No.>    8 Customer Discount    Exp>   3
 38  0
 39  7 Exec. Task>   No.>    1 Order Lines        Parms>   0

 1>Opt  2>Undo  3>Del  4>Add  5>Zoom  6>Expr  7>Draw  8>Task  9>End  10>Help
```



```
                               Order Entry
        Order No: 999                     Customer No: 99999
      Order Date: 99/99/99                    Address: AAAAAAAAAAAAAAAAAAAA
                                                       AAAAAAAAAAAAAAAAAAAA

 Line  Item   Type  Description             Quantity  Unit Price   Total Price
  999  99999   A    AAAAAAAAAAAAAAAAAAAA     -9,999   999,999 99   999,999 99

          Item List
   No.  Description            Type   Price
   999  AAAAAAAAAAAAAAAAAAAA    A   -999,999
                                                          Order Sum  -999,999.99
                                               99.99%     Discount   -999,999.99
                      Stock Status                        Sub-Total  -999,999.99
                                               99.99%     Sales Tax  -999,999.99
               In Stock: -999,999
           Total Orders: -999,999                       Order Total  -999,999.99
          Avail to Sell: -999,999

 1>Opt  2>Undo  3>Del  4>Add  5>Zoom  6>Expr  7>Draw  8>Task  9>End  10>Help
```

Visual Programming with Magic PC looks as simple as this. To design an application you quickly place your design specifications into menu-driven Task Tables without having to write a single line of code. For example, just by highlighting the Execute Program operation on the left screen and also highlighting the Item List program in the Program Menu, you tell Magic PC to Zoom into the Item List program through the window shown on the right screen. The window will automatically scroll the Item List data horizontally and vertically, and allow query, "cut and paste" copy or even creation of new Items.

### Free yourself from coding

Database professionals throughout the world are discovering a new way to dramatically cut development time.

So can you! With **Magic PC, the Visual Database Language** by Aker.

Consultants, VAR's, Software Houses and DP MIS professionals: If you develop DB applications for a living, now you can tackle any database application 10 times faster than with your DBMS or 4GL.

What makes you so fast with Magic PC? It's not magic...it's simply because Magic PC finally frees you from coding. And doesn't coding take up most of your time right now?

Magic PC lets you leverage your design skills instead of wasting your time coding. Now you can generate a fully functional prototype in just hours for quick customer feedback, and easily refine the same prototype to a finished application.

All you do is enter your system design specifications directly into Magic PC's non-procedural menu-driven Tables, as ideas come to mind, and Magic PC generates the programs for you automatically.

Magic PC gives you a free hand to design powerful data management systems limited only by your own imagination. Without the time consuming mechanical details of conventional procedural programming. There's your competitive edge. The rest is up to you.

Your biggest time saving comes from Magic PC's dynamic adaptation to spontaneous design changes. You're free to change your design on the fly, and Magic PC automatically updates your programs and data files online. No more time wasted maintaining each program manually with every small change.

### Visual Programming Power

You program with Magic PC by describing your data elements with **Data Dictionary Tables** (Files, Fields, Keys), and placing your system design specifications into **Task Description Tables.**

The Tasks can be nested within one another or dynamically Link to satellite Tasks, to give you true One-to-Many relational database power.

Only 13 Task **Operations** harness the power of Magic PC. Operations are specific enough to eliminate the need for tiresome coding, yet elastic enough to produce robust custom applications.

Use the Task building blocks to quickly generate Online Programs: Screens, Window Zooms, Menus; or Batch Programs: Reports, Updates, Data Import/Export and much more.

You develop the Task Tables visually on the screen by highlighting selections from Window Zooms and pop-up menu-driven Tables. You're not forced to follow any particular Table sequence, and there's no coding to slow you down. It's that simple.

You can apply mathematical and logical Expressions, or use the built-in Functions directly in the Task Tables to automate conditional Task processing, to display custom error messages or even invoke external applications such as spreadsheet, word processing or communication programs, transparently from within your Magic PC application programs.

Magic PC generates your application by fusing all your Data Dictionary and Program Tables seamlessly into a single **Integrated Library,** and automatically maintains changes online for optimal, bug-free performance, so you always get it right the first time.

Your application is executed at runtime by a **Magic Run** engine for stand-alone operation, and you can distribute your applications at a low cost and protect your design. Magic Run has a built-in visual interface to manipulate data and get on-the-spot ad-hoc information without any commands or syntax, simply by highlighting selections from menus. Data validation, security and error-checking are done automatically for you without programming.

Magic PC has built-in support for File and Record Locking so you can design multi-user applications for a local area network, and share data with any number of Magic Run users.

Magic PC integrates the Btrieve file manager internally, supporting the B-Tree file structure for fast high performance data access, and fault tolerant recovery during power failures.

Magic PC's powerful **Window Zoom** lets you design composite screens with windows to probe deep into the application through nested windows and manipulate the data underneath. By Zooming from any field, your end-user can conveniently query, copy or even create data in other programs directly through the windows, without stopping their screen session. The window frame size does not limit the available Data View since each window has built-in horizontal and vertical scrolling.

### Magic PC is the professional's choice:

**IBM France:** "IBM encourages Magic PC and salutes such evolution..."

**Israeli Air Force:** "We were convinced that it was not possible to have a design tool powerful enough to implement real life applications without a programming language. Magic PC changed our mind..."

**PC Magazine:** "If the thought of programming database applications makes you tremble, Magic PC is for you. The applications generator saves users from the need to deal with much dreaded computer language code..."

**PC Tech Journal:** "Magic PC is probably the best integrated database application and screen generator that we have seen...very smooth system, and smoothness comes at a premium these days..."

**PC World:** "Relational data managers and application generators that offer power without programming are a bit like perpetual motion machines—very rare. Into that vacuum comes Magic PC, a data management tool without language that is ideal for turnkey applications..."

**PC Week:** "Rather than use a written programming language the user is given a great deal of freedom and power to create complex relational database applications...this package is a true time-saver..."

### Get your Magic Tutorial for only $19.95

See for yourself how fast and powerful visual programming can be for you. Order your copy of **Magic PC Tutorial** including Tutorial disks, and a step-by-step tutorial to quickly set up an Order Entry application without coding. All for only $19.95 and We'll credit the Tutorial cost towards the $695 Magic PC price for up to 30 days.

### Or save $500 at no risk!

Yes, for a short time only, qualified VAR's can save almost $500 off the list price and get the complete unprotected **Magic PC** software and documentation at a **special VAR price of $199*** with no risk. Keep it at this low price only if it's as good as we say, or return it within 30 days for a full refund if you're not completely satisfied. Act now and you'll also get 2 Magic Run (runtime modules) for only $95.

### Order now for immediate delivery

Call this toll-free number now with your credit card or COD charge, or send the Order Coupon below today with your check to Aker.

## 1-800-345-MAGIC
## in CA 714-250-1718

VISA MasterCard AMERICAN EXPRESS

Yes, please rush me the following

Prices include 2nd day shipping

| | |
|---|---|
| □ Magic PC Tutorial | $ 25.95 |
| □ Magic PC (VAR's only) | $209.00 |
| □ Magic PC | $705.00 |
| □ Magic Run | $100.00 |
| in CA add 6% tax | $____ |
| Total | $____ |

Ship to: ________________

Address: ________________

City/ST/Zip: ________________

Phone: ________________

OEM inquiries are welcome. Prices valid in North America only.

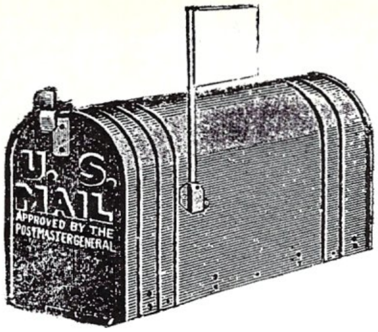
*Less $19.95 restocking fee. Limit one per customer, subject to availability. Not for resale.

System requirements: IBM PC, XT, AT and 100% compatibles, PC-DOS 2.0 or later, 512K and hard disk.

AKER

Aker Corp. 18007 Skypark Cir. B2, Irvine, CA 92714
Tlx 4931184 AKER UI
Aker S.A. 11 Route de Florissant CH-1206, Geneve
Switzerland Tlx 421792 AKER CH

**Trademarks:** Magic PC, The Visual Database Language, Window Zoom, Magic Run, Magic LAN and Magic PC Tutorial are trademarks of Aker Corp., IBM PC and PC-DOS are trademarks of IBM Corp., Novell is a trademark of Novell Inc., Btrieve is a trademark of Softcraft Inc.



# there is mail . . .

*We welcome your letters, with their comments, compliments, criticism, and suggestions. We do not have the staff to answer all letters personally. All letters become the property of* M/SJ *and may be subject to editing. 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.*

## 9600-BPS Modem Review Criticized

Dear *MS/J*:

Several of the staff at GammaLink are subscribers to *M/SJ*. We were particularly incensed by your magazine's article entitled "Communicating at 9600 baud" (January/February 1987). The authors clearly have little knowledge of the field and are misinformed. Even the title contains an unforgivable error. Transmission at 9600 bits/sec is available, but 9600 baud is beyond the theoretical limit of the telephone network with its voice bandwidth of 200-3,200 Hz. Baud rate is only equal to bps if pure binary coding is used.

For example, the CCITT V.29 uses 2400 baud with QAM, providing the 9600-bps. The article did not even acknowledge the advantages of synchronous modems, which dominate this end of the market.

GammaLink has provided 9600-bps V.29 on dial-up lines since 1984 and is therefore a leader in the field. Special features such as auto-fallback to lower speed, auto-dial, and auto-answer make us the leading supplier to this market. GammaLink was also the first company to create a 9600-bps modem for the PC using the RFAX chip featured in the article.

Simon McGrath, Product Manager
GamaLink Synchronous Comm.
Palo Alto CA

*Editor's Response:*
*Mr. McGrath is correct in pointing out that baud rate and bit rate are not the same. I take the blame for the article's title. Common usage of the word baud, however, has established it to be the same as the bit rate. Actually, the baud rate is signaling elements per second. Each signaling element carries at least one, but sometimes more, bits. Hence modems commonly referred to as "1200 baud" are actually 300-baud modems because each signaling element carries four bits, for a total of 1200 bits per second.*

*Author's response:*
*We are familiar with GammaLink products and other synchronous high-speed modems. We chose to focus on asynchronous modems because of their lower cost. We did not review all the asynchronous modems on the market, however, we felt the modems we chose to evaluate represented the most innovative, versatile implementations of asynchronous modems.*

## Looking for an HD64180 CPU Card

Dear *MS/J:*

I would like to use an S-100 CPU based on the Hitachi HD64180 microprocessor. I have been unable to reach Intelligent Computer Designs, which advertised such a product in *M/SJ*. I would appreciate your help in contacting the company.

Alex R. Borrell
Mexico City, Mexico

*I regret to say that ICD appears to be out of business. Also, Magnum Digital, whose HD64180 S-100 CPU card was reviewed in* M/SJ *last year, has gone out of the S-100 products business. I do not know of any company currently manufacturing such a product.*

## Clone Documentation

Dear *MS/J*:

Referring to your article "Cloning in the Fast Lane," (November/December 1986), a review of the PC's Limited 12-MHz AT clone, I find that there is a point that you, as most reviewers, are omitting. This is the availability of system schematics and BIOS source listings. It appears that the majority of clone manufacturers are not releasing this information, PC's Limited included. A phone call to the company revealed that it does not supply schematics or BIOS listings to anyone—period. When asked who can service its machines—the factory. Anyone else—no.

I feel that such systems should be declared unacceptable as end-users are left at the mercy of paranoid manufacturers who are, after all, copying someone else. IBM, with its long history of system secrecy, is providing this information. Users of CP/M-type systems have had this information from the beginning.

I ask that your publication begin including the availability (or lack) of schematics and BIOS source listings in its reviews. System documentation would be most useful to individuals considering PC hardware design and a requirement for those wishing to modify or make repairs to their systems. I feel that this is an absolute necessity when purchasing a system of this type.

Edward W. White
Bossier City, LA

*Editor's response:*
*Your points are well taken; however, I must correct you on a few of them. First, the vast majority of CP/M systems sold (Kaypro, Commodore, and Apples with Z80 cards) never included schematics or BIOS listings. The only manufacturers I ever saw include this information were the S-100 manufacturers who sold components and expected users to modify the BIOS to suit their systems' needs. The BIOS resided on the disk, and only a loader resided in ROM. Today the BIOS is in ROM and is not easily changed—and really does not have to be changed.*

*When we bought our first IBM PC in late 1981, the technical manual, including the BIOS source listing and schematics, was included. Those days are long gone. Today, these manuals must be purchased separately, and I can buy a basic PC clone for what IBM charges for the manuals.*

*Most PC/XT clone suppliers furnish schematics for some of the components (not all) in their systems. I have yet to see any furnish BIOS source listings. When it comes to AT clones, I have yet to see any supplier furnish either. I certainly do not agree with you that the BIOS source listing is "an absolute necessity when purchasing a system of this type."*

*When it comes to service, I Rate PC's Limited very highly. Its systems come with a 30-day unconditional return policy (don't like it, return it and get your money back), a one-year full warranty, and an 800-number technical support group. This is far better than the service that IBM and most other clone makers provide. I have had occasion to use both IBM and PC's Limited's service and technical support. I prefer PC's Limited's.*

### CDOS

Dear *MS/J*:

I would really like to thank your magazine for its continuous coverage of Digital Research's Concurrent PC-DOS (CDOS) operating system. You are one of the few magazines that has shosen to evaluate this operating system (OS) from a technical viewpoint and not soured your reviews or comments simply because CDOS does not support every function of Microsoft's MS-DOS.

It is my feeling that far too many so-called professional magazines fail to recognize or understand that MS-DOS was originally copied from CP/M nearly function for function, structure for structure. As such, carried over in both the MS-DOS operating system and the program structure, which runs under PC-DOS (.EXE and .COM files), are many primitive features that are ill-suited for processing use in a multitasking operating system design. Of course, this is the state of the world, and we all have to live with it now. . . . even Microsoft!

I find Bill Gates' comments about the exteme difficulty of producing a multitasking (not multiuser, mind you) MS-DOS operating system environment a "very" amusing situation for a company that has continually benefited from, and fostered the spread of, the idea of the inadequacies of CP/M (and the subsequent relation to DRI).

I wish to thank you for your keen insights into these inadequacies of the MS-DOS design structure and of your exposure of DRI's fine multiuser/multitasking OS to your readers. I have been using CDOS in the design of PC-based workstations for the government for the past three years and am quite pleased with its performance and capabilities.

Concurrent DOS-386 is to be released soon, and from discussions with several beta-test sites, it appears to have few problems that might hold up its introduction.

The problem DRI faces, however, is one of public ignorance of the capabilities of CDOS and of the repeated announcement of products by Microsoft long before they become available. Examples of these abuses are:

1. Microsoft's annoucement of Windows two years prior to its release.
2. Microsoft's announcement in November 1986 that it planned to release ADOS in December. Then in December it announced ADOS would be released in January/February. And finally, at a users' group conference in March, it announced that it would actually deliver ADOS late this year or early next year.
3. Microsoft's 80386 protected mode DOS will not be available until late 1988.

I find these publicity tactics disgusting and an abuse of the press and the trust of the computing public. It is hard enough to make intelligent decisions regarding corporate or customer project software needs without the false hopes promised by what is considered to be one of the pillars of the personal computer industry.

DRI certainly has my support for providing us all with a viable alternative to MS-DOS. CDOS has all the sophistication and technical capabilites normally found on only the best mini/mainframe operating systems. I am sure that with enough exposure this OS will gain the acceptance it deserves in the world of PC-based microsystems.

Again, thank you for your recent articles covering the internals of CDOS, and I hope to see more in the future.

Brian J. Mullan
McDonnell Douglas Corp.
Lutz, FL

§

# C Forum

Don Libes

## Byte Ordering
## yBetO drreni g
## etyBdr Oiner g

*This column features tips and techniques for using the C language productively. It discusses typical problems with using C and their solutions. Reader suggestions, comments, and questions are encouraged. Address them to "The C Forum,"* Micro/Systems Journal, *Box 1192, Mountainside, NJ 07092.*

```
/* unportable */
int c = 0;
read(0,&c,1);
printf("%d\n",c);
```

Try the above code fragment on your favorite machine. The output may surprise you. Even worse, it may surprise you later. If you include such code in a program that is ported to other machines, you will eventually find that it fails to behave in the same way on all the machines.

To save you some trouble, I ran it on Intel 80286, DEC VAX, and Motorola 68010 systems (they were handy). The input to the program was the character *a*. The results were:

```
80286:    97
VAX:      97
68010:    1627389952
```

No, the 68010 isn't broken or wrong. The program is, however. *Read( )* is defined to take the address of a *char*, but I passed it the address of an *int*. What's the big deal, you say—*int*s are bigger than *char*s, right? Yes—however, integers are stored "differently" on different machines.

Specifically, the least significant byte (lsb) is stored at the same address as the integer on the 80xx6 and VAX families. This is commonly referred to as "little-endian" addressing. The most significant byte (msb) is stored at the same address as the integer on the 680x0 family. This is commonly referred to as "big-endian" addressing. (The phrases refer to Jonathon Swift's *Gulliver's Travels,* in which the Brobdingnagians engaged in a frivolous dispute over "on which end an egg should be broken.") Another phrase describing the same problem is "byte sex," emphasizing the arbitrary but real difference between the two types of addressing, but begging the question as to which is better.

What happened in my example, then, was that the *97* got poked into the least significant byte of an *int* on the VAX and 80286, whereas it got poked into the most significant byte on the 68010. Hence, when we printed out the *int*, I found $97 \times 256^3$ on the 68010.

Looking at the following table, you see the results of the statement *c = 0x03020100* when *c* is declared as an *int* that lies at address *0* on the various machines:

| address | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| little-endian | 0 | 1 | 2 | 3 |
| big-little-endian | 2 | 3 | 0 | 1 |
| big-endian | 3 | 2 | 1 | 0 |

The big-little-endian is found on the PDP-11. On that machine, an *int* is only 2 bytes, little-endian style. To form a 4-byte integer, you must declare it as a *long*. The words (2-byte values) in the *long* are stored in big-endian order.

Although the 68010 order may seem strange and the PDP-11 positively bizarre, you should not discredit either of these schemes on the basis of one wrong piece of code. Unfortunately, I constantly hear people claim that one or another of these formats is "wrong" and "a mistake." This is just stubbornness. As long as it is done consistently, either approach is valid. (Indeed, many other machines work in the same way as the 68010 does, including the AT&T 3B20, the IBM 370, and the HP 9000.)

An extensive treatment of the issues is presented in a USC/ISI memo by Danny Cohen, "On Holy Wars and a Plea for Peace."[1] Interested readers are referred to this memo for further insight. A rather amusing article by Mike Higgins on the same subject appeared in the 1986 April Fool's issue of *Computer Language Magazine*. This also discusses the related issue of bit numbering within bytes, also often referred to in endian terms.

Depending on what you are doing, being able to choose one of these forms can be of great advantage in the efficiency of program execution. Once the CPU has been chosen or the architecture designed, however, there isn't much you can do but live with it. (Some machines are configurable—for example, microprogrammable ones).

Consider integer arithmetic, for example. The natural method is to represent integers as polynomials of powers of 2 and do the arithmetic starting with the lsb and working up—little-endian. On the other hand, if you are comparing integers, it is more useful to be handed a pointer to the msb. Then you can compare bytes and work down—big-endian.

These justifications are typical, but others I've heard discussed are based on multiprecision integers, strings, human readability, mathematical history, bit numbering within bytes, and so on. About the only thing that is clear, apart from the better fit of big-endian to block-oriented operations and of little-endian to stream-oriented ones, is that this is a "religious issue."

Most of the time I leave these matters to assembly-language programmers, who need to be very conscious of bit and byte ordering. Communications between different architectures, however, requires that even C programmers be aware of these issues. If you are on a network communicating between a 68010 and an 80286, for example, sending integers is, all of a sudden, not so simple as just handing the other guy a couple of bytes.

### Solutions

One solution is to have one or both of the programs recognize that they are communicating with a different endian processor and adapt to that. An alternative solution is to select either big- or little-endian or possibly even a different method and use that as a standard. I favor the latter solu-

# Microsoft Avoids Challenge

**We challenged Microsoft to a C compiler duel-to-the-finish, comparing compile, link and execution times, and we offered to stop advertising for two months if they won...**

by Roy Sherrill, President, Datalight

**Microsoft purchased our C-compiler during February 1987 and we still haven't heard from them. OK, Microsoft, we are extending our challenge deadline from April 1, 1987 to May 15, 1987. After all, the Microsoft ad claims "the fastest C you've ever seen." Your reply, Microsoft!**

### Walter says Optimum-C is better

Walter Bright, the developer of Optimum C, says that Optimum C would win 7 out of 10 benchmarks as compared to Microsoft C, V.4.0. Walter explained to me that Optimum C includes a unique global optimizer that helps create compact code while increasing execution speed up to 30%. By the way, Borland, Walter is still waiting for his copy of Turbo C® V.1.0. Borland's ad claims "the fastest, most efficient and easy-to-use C compiler at any price."

After reviewing Borland's benchmarks, Walter claims that Optimum C is faster. And, as for ease of use, all Datalight C compilers have been shipped with a free Learn C program for the last six months. Also, our new EZ Interactive Editor will show you each syntax error in your source code, then compile or "make" and run your program, all from within the editor. OK, so let the Microsoft challenge begin...

### We only ask the following...

The benchmark suite will consist of the set of programs that Microsoft supplied to *Computer Language* for their February 1987 C compiler review issue. Microsoft will make available the programs to Datalight at least two weeks prior to the benchmarking. The benchmarking will be between Microsoft C 4.0 and Optimum-C. It will occur at a mutually agreed upon time and place. Interested individuals will be allowed to attend. The benchmarks will be compiled and run on a standard IBM PC-AT.

There will be two separate tests for each program: compile and link speed, and execution speed. For each test, a representative from each company will set up the compiler so that it performs at its best.

The benchmarks will be adjusted so that they take sufficiently long to run, that the tolerance involved in timing them is insignificant. The winner is determined by the compiler with the faster execution times for the majority of the benchmarks. We'd like an answer from Microsoft no later than May 15, 1987.

### So what's a global optimizer?

A global optimizer looks at an entire function at once, analyzing and optimizing the whole function. A technique called data flow analysis is used by Optimum-C to gather information about each function. This enables your compute-bound programs to execute as much as 30% faster after global optimization. But, there is one catch...because the global optimizer ruthlessly searches for ways to speed-up execution speed and minimize memory usage, it has relatively slow compile times. No need to worry, though, because you can merely turn the global optimizer off. In fact, you can select all, none, or part of the following optimizations: constant propagation, copy propagation, dead assignment elimination, dead variable elimination, dead code elimination, do register optimizations, global common subexpression elimination, loop invariant removal, loop induction variables, optimize for space, optimize for time, and very busy expressions.

### ROM-it Speeds ROM Development

ROM-it provides the extra support needed to speed completion of your ROM applications. ROM-it includes a ROMable C start up routine, and the BLAZE Intel hex utility. ROM-it also includes a library of ROMable functions that allow full access to the 8086, including interrupt handling in C, access to all of memory, and reading and writing I/O ports.

Extra support is also available for remote debugging and multi tasking. The remote debugger supports full symbolic debugging of the application in the target machine, from the PC.

Please call for more information!

### Try Optimum-C risk free

Try Optimum-C for 30 days and if you are not 100% satisfied return it for a full refund. *Also available is a C tutorial which is a combination workbook and floppy disk to help lead you through the C language with tutorials, quizzes, and program exercises.

**O.K. Microsoft, it's up to you. We've put two months of advertising on the line that says you can't beat Optimum-C to a real test. Your answer, please?**

### PRICES

| | |
|---|---|
| Developer's Kit w/ C Tutorial | $ 99 |
| Optimum-C w/ C Tutorial | $139 |
| (both with library source) | |

Add $7 for shipping in US/$20 outside US
COD (add $2.50)

Not Copy Protected

## ORDER TOLL-FREE TODAY!
## 1-800-221-6630

### ATTENTION OEMs!

Contact us regarding arrangements.

*Limited offer available exclusively to readers who purchase directly from Datalight.

Microsoft and MS-DOS are registered trademarks of the Microsoft Corporation. Turbo C is a registered trademark of Borland International.

### Magazine Reviewers Shocked by DATALIGHT's Performance...

"Reviewing this compiler was quite a surprise for us. For such a low price, we were expecting a "lightweight" compiler. What we got was a package that is as good as or better than most of the "heavyweights." Datalight C implements a complete C language. It also compiles quickly, doesn't take up much disk space, and looks impressive in the benchmarks."

***DR. DOBBS*, August 1986**

"This is a sharp compiler!... what is impressive is that Datalight not only stole the compile time show completely, but had the fastest Fibonacci executable time and had excellent object file sizes to boot!"

***COMPUTER LANGUAGE*, February 1986**

### Optimum-C Version 3.0
**NEW!**
**EZ Interactive Development Environment**
**NEW!**
**Inline 8087/80287 Math Support**

- Full UNIX System 5 C language plus ANSI extensions
- Fast/tight code via powerful optimizations including common sub-expression elimination
- DLC one-step compile/link program
- Multiple memory model support
- UNIX compatible library with PC functions
- Compatible with DOS linker and assembler
- Third-party library support
- Automatic generation of .COM files
- Supports DOS pathnames, wild cards, and Input/Output redirection
- Compatible with Lattice C version 3.x
- Interrupt handling in C
- Debugger support
- ROMable code support/start-up source

**MS-DOS® Support Features**

- Mouse support
- Sound support
- Fast screen I/O
- Interrupt handler

**MAKE Maintenance Utility**

- Macro definition support
- MS-DOS internal commands
- Inference rule support
- TOUCH date manager

**Tools in Source Code**

- cat—UNIX style "type"
- diff—Text file differences
- fgrep—fast text search
- pr—Page printer
- pwd—Print working directory
- wc—Word count

**Datalight**

**17505-68th Avenue NE, Suite 304**
**Bothell, Washington 98011 USA**
**(206) 367-1803**

tion, and fortunately, recent history provides you with a clear choice.

Both ISO and DOD protocols use big-endian as a standard "network integer" format. Don't worry if you are on a little-endian machine. The proper technique is to convert all your integers to network standard integers before putting them on the network. Similarly, when receiving information from the network, you must convert from network standard form to the local host form. By encapsulating the conversion in functions or macros, it is possible to produce code portable to any-endian machines. Note that, if your machine stores integers in network standard form already, you can supply null macros with no loss of efficiency.

For example, to send the number *17* across a network, you might use the following code fragment:

```
long int c = 17;
/* convert host to network- */
/* long */
netc = htonl(c);
/* fictional send routine */
send(netc);
```

Here, *htonl( )* performs any byte swapping if necessary, and *send( )* takes the network standard information and actually delivers it to the recipient.

The opposite of *htonl( )* is *ntohl( ),* which applies the opposite conversion. Analogous functions exist for *shorts*, namely *htons( )* and *ntohs( ).*

## byteorder.c

The public-domain code in Listing 1 implements all of these conversion functions and was written by Dennis Bednar (rlgvax!dennis). Interestingly, it is machine independent—you don't even have to declare whether your machine is big- or little-endian! In order to do this, the program stores a *long* and then looks at the first byte. Based on that, it can tell what type of machine you are on and how to convert to/from network standard form.

## Note

[1] Danny Cohen, "On Holy Wars and a Plea for Peace," IEN 137 (April 1, 1980). University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90291. §

*Don Libes is a computer scientist working in the Washington, D.C. area. He works on artifical intelligence in robot control systems.*

### Listing 1

```
/* #define STAND 1 */
/*
 * byteorder.c
 * June 18, 1986
 * seismo!rlgvax!dennis              dennis@rlgvax.UUCP
 * Compile with -DSTAND for standalone program that should run,
 * compile without -DSTAND for normal library package.
 *
 * Public domain byte order routines.
 * You use this package to store binary shorts and ints
 * such that, when written to the disk, they will be in a network
 * independent order.
 * Tested on CCI Power6/32.
 * Tested on VAX.
 * Tested on 680X0
 *
 * The neat thing about this code is that it doesn't use
 * any #ifdef tricks, rather it figures out itself what
 * the byte ordering should be on the first call to the
 * package.
 *
 * The order chosen was such that if you step thru the
 * binary number using a char * pointer, you will go from
 * MSB to LSB.
 *
 * There are two good applications of this package.
 *      - You are trying to create binary disk files that
 *        are guaranteed to be able to be read on another
 *        machine (public domain cpio's, etc.).
 *        To do this you should do
 *               long     result = htonl(l);
 *        *just* before you write to the disk, and you should
 *        call
 *               long     result = ntohl(l);
 *        *just* after you read from the disk.
 *
 *      - and writing protocol programs which write to networks,
 *        applications which write binary data over a network.
 *
 * Presently ntohl() and htonl() are internally really the same thing,
 * but these interfaces were modelled on the 4.2BSD, and that's the
 * way they did it, so don't complain.
 *
 * Longs should be 4 bytes, and shorts should be 2 bytes.
 * ARE THERE ANY MACHINES ON WHICH THIS IS NOT TRUE???
 *
 * We don't use int's because they are 2 bytes on some machines,
 * and 4 bytes on other machines.
 *
 */

/* forward refs */
long     ntohl(),
         htonl();
short    ntohs(),
         htons();

static   int      called = 0;       /* none of the routines called yet */

/*
 * there are 4 possibilities
 *                                                   MSB   LSB
 *       Order on the disk                   3 2 1 0
 *
 *       Order in memory if you stepped a char * ptr thru the long
 *       - noswap:                           3 2 1 0          68K, 370
 *       - byteswap: (swap bytes in short)   2 3 0 1          PDP-11
 *       - halfswap: (swap shorts in long)   1 0 3 2          ?
 *       - bothswap: (swapbyte && halfswap)  0 1 2 3          VAX, 8086
 *
 * Init2() only checks the first byte to know which of these 4 cases
 * we are using.
 */

/* boolean flags, see the above comment to understand what they mean */
static char
         noswap,
         byteswap,
         halfswap,
         bothswap;
/*
 * just to play it safe, init passes the argument on the stack so
```

# 10 Important Reasons C Programmers Use Our File Manager

### 1. It's written in C.

Clearly the growing language of choice for applications that are fast, portable and efficient. All of db_VISTA's source code is written in C.

### 2. It's fast – almost 3 times faster than a leading competitor.

Fast access that comes from the unique combination of the B-tree indexing method and the "network" or direct "set" relationships between records. A winning combination for fast performance.

### 3. It's flexible.

Because of db_VISTA's combination of access methods, you can program to your application needs with ultimate design flexibility. Use db_VISTA as an ISAM file manager or to design database applications. You decide how to optimize run-time performance. No other tool gives you this flexibility without sacrificing performance.

db_VISTA is also well behaved to work with most any other C libraries!

### 4. It's portable.

db_VISTA operates on most popular computers and operating systems like UNIX, MS-DOS and VMS. You can write applications for micros, minis, or even mainframes.

### 5. Complete Source Code available.

We make our entire C Source Code available so you can optimize performance or port to new environments yourself.

### 6. It uses space efficiently.

db_VISTA lets you precisely define relationships to minimize redundant data. It is non-RAM resident; only those functions necessary for operation become part of the run-time program.

### 7. Royalty free run-time.

Whether you're developing applications for yourself or for thousands, you pay for db_VISTA or db_QUERY only once. If you currently pay royalties to someone else for your hard work, isn't it time you switched to royalty-free db_VISTA?

## db_VISTA™

**Features**

- ♦ **Multi-user** support allows flexibility to run on local area networks
- ♦ **File structure** is based on the B-tree indexing method
- ♦ **Transaction processing** assures multi-user consistency
- ♦ **File locking** support provides read and write locks
- ♦ **SQL-based db_QUERY** is linkable
- ♦ **File transfer** utilities included for ASCII, dBASE optional
- ♦ **Royalty-free** run-time distribution
- ♦ **Source Code** available
- ♦ **Data Definition Language** for specifying the content and organization of your files
- ♦ **Interactive database access** utility
- ♦ **Database consistency check** utility

**File Management Record and File Sizes**

- ♦ Maximum record length limited only by accessible RAM
- ♦ Maximum records per file is 16,777,215
- ♦ Maximum file size limited only by available disk storage
- ♦ Maximum of 256 index and data files
- ♦ Key length maximum 246 bytes
- ♦ No limit on number of key fields per record
- ♦ No limit on maximum number of fields per record

**Operating System & Compiler Support**

- ♦ **Operating systems:** MS-DOS, PC-DOS, UNIX, XENIX, UNOS, ULTRIX, Microport, VMS
- ♦ **C compilers:** Lattice, Microsoft, IBM, DeSmet, Aztec, Computer Innovations, Turbo C, XENIX and UNIX

### 8. SQL-based db_QUERY™

Add our new C-linkable, SQL-based, ad hoc query and report-writing companion product to provide a simple relational view of your db_VISTA applications. Without compromising speed.

### 9. Free tech support.

60 days of free technical and application development support for every Raima product. Of course, extended support and training classes are also available at your place or ours.

### 10. Upward database compatibility

Start out with file management in a single-user PC environment—then move up to a multi-user LAN or a VAX database application with millions of records. You'll still be using db_VISTA. That's why so many C programmers are choosing db_VISTA.

### But don't just take our word for it.

"Raima's customer support and documentation are excellent. Source code availability and royalty-free run-time is a big plus."

**Dave Schmitt, President**
**Lattice, Inc.**

"db_VISTA has proved to be an all-round high performer in terms of fast execution, flexibility and portability, and has undoubtedly saved us much time and development effort."

**John Adelus, Hewlett-Packard**
**Office Productivity Division**

### 30-day Money Back Guarantee!

Try db_VISTA in your environment for 30 days and prove it to yourself. If not completely satisfied, return it for a full refund.

### Price Schedule

| | db_VISTA | db_QUERY |
|---|---|---|
| □ Single user | $ 195 | $ 195 |
| □ Single user w/Source | $ 495 | $ 495 |
| □ Multi-user | $ 495 | $ 495 |
| □ Multi-user w/Source | $ 990 | $ 990 |
| **NEW:** | | |
| □ VAX Multi-user | $ 990 | $ 990 |
| □ VAX Multi-user w/Source | $1980 | $1980 |

### Order Now.

Put db_VISTA to work in your application program. Ordering is easy—simply call toll-free. We'll answer your technical questions and get you started. Call today.

## Call Toll-Free Today!

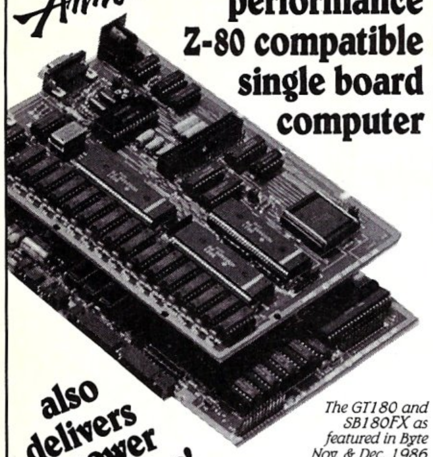
**1 (800) db-RAIMA**
(800/327-2462) or
206/828-4636

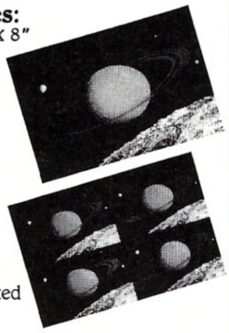
MasterCard VISA

RAIMA™
CORPORATION

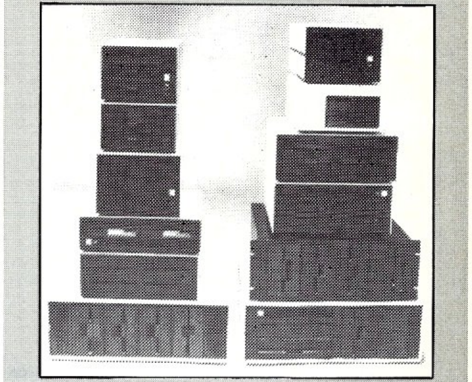
3055 - 112th NE, Bellevue, WA 98004 USA
(206) 828-4636 Telex: 6503018237 MCI UW

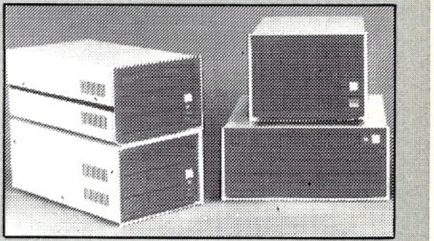
MJ77

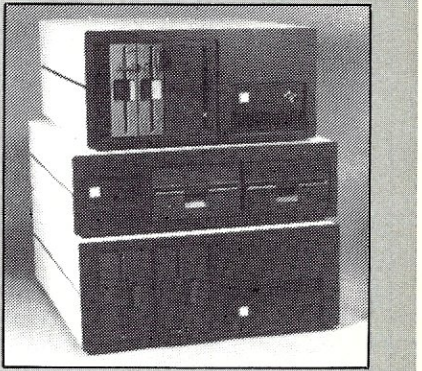




```
 * that there are no tricky problems caused by differences in byte
 * order for binaries on the stack vs. binaries in static variables.
 * Thus the byte-ordering on the stack for init() is the same as
 * the byte-ordering on the stack for ntohl(), ntonl().
 * Am I being too cautious???
 */
static
init()
{
        init2( (long) 0x03020100 );
}

/*
 * basically initialize the 4 boolean flags, so that the conversion
 * routines know what to do.
 */
static
init2(l)
long l;
{
        char *cp;

        if (sizeof(long) != 4) {
                write(1, "byteorder: sizeof(long) != 4\n", 29);
                exit(1);
        }

        if (sizeof(short) != 2)    {
                write(1, "byteorder: sizeof(short) != 2\n", 30);
                exit(1);
        }

        cp = (char *) &l;
        switch (*cp) {
        case 03:
                ++noswap;
                break;
        case 02:
                ++byteswap;
                break;
        case 01:
                ++halfswap;
                break;
        case 00:
                ++bothswap;
                break;
        default:
                write(1, "byteorder: Unknown machine\n", 27);
                exit(1);
        }
#ifdef STAND
        printf("%d %d %d %d\n",*cp,*(cp+1),*(cp+2),*(cp+3));
#endif
}

/*
 * network to host long
 * call this *just* after you read from the disk or the network.
 */
long
ntohl(l)
long l;
{
        return(htonl(l));
}

/*
 * host to network long
 * call this *just* before you write to the disk or the network.
 */
long
htonl(l)
long l;
{
        register char *sp,              /* source pointer */
                      *dp;              /* dest pointer in r */
        long r;                         /* result - cannot be register */

        if (!called) {
                init();
                called = 1;
        }

        sp = (char *) &l;
        dp = (char *) &r;

        if (noswap) return l;

        if (bothswap) {                 /* swap bytes within long
                                                   3<->0 && 2<->1 */
                *dp++ = sp[3];
                *dp++ = sp[2];
                *dp++ = sp[1];
                *dp++ = sp[0];
                return r;
        }
        if (byteswap) {               /* bytes swapped within shorts */
                *dp++ = sp[1];
                *dp++ = sp[0];
                *dp++ = sp[3];
                *dp++ = sp[2];
                return r;
        }

        if (halfswap) {               /* swap halfwords (shorts) within long */
                *dp++ = sp[2];
                *dp++ = sp[3];
                *dp++ = sp[0];
                *dp++ = sp[1];
                return r;
        }
}

/*
 * host to network short.
 * call *just* before writing to disk or network.
 */
short htons(s)
short s;
{
        return ntohs(s);
}

/*
 * network to host short.
 * call *just* after reading from disk or network.
 */
short ntohs(s)
short s;
{
        register char *sp,            /* src ptr */
                      *dp;            /* dst ptr */
        short r;              /* result - cannot be register */

        if (!called) {
                init();
                called = 1;
        }

        if (noswap || halfswap)    return s;

        if (byteswap || bothswap) {
                sp = (char *) &s;
                dp = (char *) &r;
                *dp++ = sp[1];
                *dp++ = sp[0];
                return r;
        }
}

#ifdef STAND

#include <stdio.h>

main()
{
        long l = 0x61626364;          /* ASCII abcd */
        short s = 0x6162; /* ASCII ab */
        long ldum;                    /* long dummy */
        short sdum;                   /* short dummy */

        /* so partial lines to stdout are printed immediately,
         * so that we can mix stdio with raw i/o.
         */
        setbuf(stdout, (char *)NULL);

        ldum = ntohl(l);
        printf("You should see \"abcd\" if ntohl() works: \"");
        write(1, &ldum, sizeof(ldum));
        printf("\"\n");

        printf("You should see \"ab\" if ntohs() works: \"");
        sdum = ntohs(s);
        write (1, &sdum, sizeof(sdum));
        printf("\"\n");

        printf("If you see any bugs, send a bug report to rlgvax!dennis\n");
        printf("Thanks.\n");
}
#endif STAND
```







# Turbo Pascal Corner

Stephen R. Davis

## Using In-line Code

*This column features tips and techniques for using Turbo Pascal productively on MS-DOS/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, Route 5, Box 107K, Greenville, TX 75401 or through MCI mail, 289-6124.*

Things are temporarily in one of those intermediate times. Turbo BASIC, Turbo C, and Eureka! have all be announced (and advertised) but have yet to appear. I will keep you informed. This time, I want to take another look at Turbo Pascal debuggers and present the Program of the Month on in-line programming.

### Turbo Pascal Debuggers

I have mentioned source code debuggers for Turbo Pascal before, but the topic is just too important to leave alone for long. Recently, I had occasion to reexamine the subject of Turbo Pascal debuggers. Of course, I have used TDebug-Plus, from TurboPower Software, for some time now to help me in the development of Turbo Pascal software, including that appearing in this column. What I did not know until recently, however, is that TurboPower Software has not been resting on its laurels, as well it might have done with a product as impressive as Version 1.00 of TDebug-Plus. Version 1.04, the version currently shipping, includes some significant improvements over its forebear.

For those of you who missed my earlier column, a source code debugger is a debugger that allows users to single-step, examine, and set breakpoints on Turbo Pascal program statements, rather than revert to the machine code that the Turbo Pascal compiler generates, as is the case with conventional compilers. This ability to work at the source level (to use the current jargon) is very exciting. Because you have written the source code, you are, ostensibly, familiar with it. It is therefore an easy matter to move around in it. The object code (the output of the compiler) bears only a tenuous relationship to the original source. With an object code debugger, therefore, it is usually quite difficult to figure out where you are in your source code at any given time.

Source code debuggers have been available for some time for other languages that go through a link step and can, therefore, generate a map file with the symbol addresses. By generating the executable file directly and avoiding the link step, Turbo Pascal cannot be used with these general debuggers (until now, as you will see in a moment).

Sometime early last year, I first saw a public-domain debugger called TDebug for Turbo Pascal. I was amazed at its elegance. Sometime during 1986 Kim Kokkonen's *TurboPower Software* bought the rights to TDebug. Fearing a "buy the shareware, jack up the price and rake in the money," I was slightly leary of the now-renamed TDebug-Plus when it first appeared. I am relieved to report that I needn't have worried.

TDebug-Plus now supports dual-screen operation, overlays, resizing of debug windows and overlays, and several new debug commands. It also has increased support for machine code debuggers, such as DEBUG, via a "get the address of label" command and an "examine registers" command. Just as impressive are some of the utilities now included on the TDebug-Plus distribution disk.

The most significant of these utilities, TMAP, can generate a map file for your Pascal program identical to the one generated by DOS' LINK. Not only is this a useful thing to have in and of itself, but also it can be used as the input for many symbolic debuggers, such as SYMDEB and Periscope. This allows users to use their favorite symbolic debugger for those really hairy problems.

But there is more good news for Turbo Pascal users. There is another product that has just appeared on the market, called GSI Pascal, that bills itself as a debugger for Turbo Pascal. GSI's advertising is carefully worded because it isn't really a debugger for Turbo Pascal in the sense that TDebug is, but rather a separate Pascal compiler with an integrated debugger. GSI Pascal claims to be "totally Turbo Pascal compatible." (You still need Turbo Pascal as GSI Pascal does not compile to machine code.) Because I don't believe that any thing as complicated as a compiler could be "totally compatible," I was dubious when I first cracked the shrink-wrap seal.

After the first ten minutes of figuring out which way was up, I discovered that GSI Pascal is indeed a very powerful Pascal debugger. It does have some compatibility worries—at least, as of Version 1.00—but by virtue of the fact that it does not stick to the limitations of working directly with TURBO.COM, GSI Pascal can include some sophisticated features. Unlike TDebug, GSI has a windows-oriented, "point and shoot" type of interface. Other than the name of the file to be debugged, you need never type in a single thing—the only keys you really need are the arrow keys, Return, F10, and Escape. This package begs for a mouse.

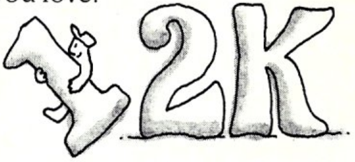
GSI (a French concern, by the way) has indeed created a powerful debugger that complements TDebug nicely. Which is better? TDebug does have the advantage of working with TURBO.COM and thus maintaining total Turbo Pascal compatibility. GSI, although not 100 percent compatible, has a wonderful user interface for those to whom remembering and typing commands is a chore. It may sound a little weak to say, but with something as important as a debugger, I suggest purchasing both and deciding for yourself. They are both inexpensive. Whatever you do, don't

*Turbo Programmers*

# THE OPTIMIZER. TAKE TURBO TO THE LIMIT.

Turbo OPTIMIZER™ helps you squeeze down, rev up, and fine tune your compiled Turbo Pascal programs. Now there's no need to write in assembler...you program faster and easier in the language you love.

## A LEANER TURBO

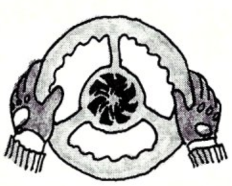


Small programs of 12K bytes are often reduced to only 2K. Medium-sized programs can shrink by 7K bytes.

The Object Code Compactor makes code smaller by automatically removing the unused portions of the run-time library as well as unused procedures and functions in your own code.

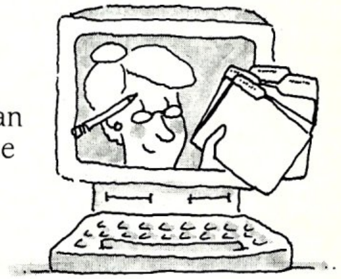
You save valuable disk and RAM space.

## A FASTER TURBO



The Object Optimizer scans the object code and removes unnecessary instructions like null jumps and register reloads. Your Turbo Pascal programs run up to 30% faster.

## CALL ON THE LIBRARIAN



Store compiled versions of your often-used procedures in relocatable form using The Object Librarian. The Object Librarian then creates External procedures that you include in your source code. You cut compile time.

*"A real lifesaver...I saved 10K bytes in the latest version of Note-It®"*
Dave Whitney, author, Note-It

Call or send today for Turbo OPTIMIZER. It's only $75. (With source code, $125) Call toll free for credit card orders:

**(800) 538-8157** x830
(800) 672-3470 x830 in CA.

Satisfaction Guaranteed or your money back within 30 days.

For other information call (408) 438-8608. Shipping and taxes prepaid for U.S. and Canadian customers. Others please include $6 per item for shipping.

More ways to take Turbo to the limit:
T-DebugPLUS, the symbolic run-time debugger, only $60,
Turbo EXTENDER breaks the 64K barrier, only $85,
TurboPower Utilities, nine powerful programs, only $95 with source.

TURBO Power

TurboPower Software, 3109 Scotts Valley Dr., Suite 122, Scotts Valley, CA 95066

These products require Turbo Pascal 3.0 (standard, 8087, or BCD), PC-DOS 2.X or 3.X, and IBM PC/XT/AT or compatibles.
Note-It is a registered trademark of Turner Hall Publishing, a division of Symantec.

continue debugging Turbo Pascal without a debugger of some sort.

### Fast Revisited

Toward the end of last year (November/December 1986), I presented two programs, called FAST and FASTEST, designed to perform fast screen output. (FASTER was embodied as a variation of FAST.) Both programs wrote directly to screen memory to achieve maximum display performance. As I noted then, the most time-consuming chore of any display routine is the scrolling of the screen to make space at the bottom available for the next line. This is due to the large amount of data that must be transferred for each single line scroll (roughly 16K).

FASTER used Turbo Pascal assign statements to scroll each line up. FASTEST improved performance by not scrolling the data at all but by instead reprogramming the CRT controller chip to move the "beginning of screen" pointer down in memory, making the resulting screen move up. Although the results were very fast indeed, they were not totally acceptable because the display was left in a strange condition, requiring a reset to clear it up. Besides, that program would only work for displays that used the 6845 CRT controller—newer generations of display adapters, including today's EGA, cannot use such an old display controller.

So what do you do when Pascal isn't fast enough? One way is to insert your own assembly-language code in place of Turbo Pascal's using the *Inline* statement. Compilers can never generate small sections of code as efficiently as humans can. It varies, but humans usually generate code that is two to three times faster. You can impart this speed advantage to your Turbo Pascal by using *Inline*.

As I noted earlier, the scroll operation took up the vast majority of FAST's time. Scrolling is performed solely within the procedure *Scroll*. This is partially a result of good planning and partially of serendipity, but it is also usually the case that more than 90 percent of a program's time is spent in just one or two procedures. To achieve a two to three times performance increase over FAST, assuming the normal ratios hold, all you must do is rewrite that one procedure using *Inline*.

The rewritten program appears as Listing 1. Notice that I have changed the remaining program a little bit (none of my programs lie around unchanged for long), but it is easily recognizable. *Scroll*, however, is completely different. The first section of in-line code performs the scroll operation, and the second in-line section blanks out the now-abandoned lines at the bottom. I have vastly simplified the assembly-language code by performing all calculations of *target*, *source*, and *number* in Pascal.

My program is aided by the use of machine instructions, which compilers normally cannot use. Block move and block

***Listing 1***

```
{Higher speed output using INLINE code -
    Higher speed output using the REP MOVS command in an INLINE
    section of code.  Note that on a CGA the snow should be
    horrible (nothing we can do about that), but on EGA's and
    monochromes the results are slightly faster than 'FASTER'.

    PS.  This version is also a little more general using constants
    'screenh' and 'screenw' instead of hardcoding the screen dimensions,
    a good programming practice.
}


const
    cga = $b800;                     {offset of color graphics}
    ega = $b800;                     {          ega}
    mono= $b000;                     {          monochrome screen}
    attribute = $0300;               {this attribute is normal video}

    segmnt = ega;                    {set this to match the monitor type}
    screenh = 25;                    {lets make things a little more general}
    screenw = 80;                    {screen heigth and width}
    sch0 = 24;                       {zero relative of above}
    scw0 = 79;

type
    outline = array [0..scw0] of integer; {this defines the screen}
    display = array [0..sch0] of outline;
    strng = string [255];

var
    screen : display absolute segmnt:0; {currently set for monochrome}
    x_pos,y_pos : integer;         {x and y position of cursor}
    value : integer;
    i, j : integer;

{******scroll N lines from bottom of screen towards top*******}
var
    source, target, number : integer; {variables used in scroll}

Procedure Scroll (count : integer);
var
    index : integer;

begin
    if (count > 0) then
    begin

         {First we must scroll up the indicated number of lines --
          we will block move 'number' words from 'source' to 'target'}

         if (count > screenh) then
              count := screenh;
         target := 0;
         source := count * screenw;
         number := (screenh - count) * screenw;
         if (number > 0) then
              Inline(
                $8B/$3/TARGET     {MOV DI,[TARGET]}
                /$01/$FF           {ADD DI,DI}
                /$8B/$36/SOURCE    {MOV SI,[SOURCE]}
                /$01/$F6           {ADD SI,SI}
                /$8B/$0E/NUMBER    {MOV CX,[NUMBER]}
                /$06               {PUSH ES}
                /$B8/SEGMNT        {MOV AX,SEGMNT}
                /$8E/$C0           {MOV ES,AX}
                /$9C               {PUSHF}
                /$FA               {CLI}
                /$26/$F2/$A5       {ES: REP MOVSW}
                /$9D               {POPF}
                /$07               {POP ES}
              );

         {now let's clear out the lines scrolled from by block
          storing 'number' attributes into 'target'}

         target := (screenh - count) * screenw;
         number := count * screenw;
         if (number > 0) then
              Inline(
```

store are fantastic ways of moving and clearing large blocks of memory in a hurry. You will notice these as the *REP MOVSW* and *REP STOSW* near the bottom of the in-line blocks.

How do you generate in-line code? One way is to write down the desired program on a piece of paper and type it into DEBUG using the *A* command, then turn around and list what DEBUG generated. This is laborious, but if you are smart, you can limit the amount of assembly-language code to just the critical sections. Another way is to use the public-domain INLINE utility (this is on the utilities disk I send out, by the way). This program accepts an assembly-language source file and generates as its output in-line code ready to be inserted into the middle of your program.

```
                  $8B/$3E/TARGET     {MOV DI,TARGET}
                  /$01/$FF           {ADD DI,DI}
                  /$8B/$0E/NUMBER    {MOV CX,NUMBER}
                  /$06               {PUSH ES}
                  /$B8/SEGMNT        {MOV AX,SEGMNT}
                  /$8E/$C0           {MOV ES,AX}
                  /$B8/ATTRIBUTE+$20{MOV AX,ATTRIBUTE}
                  /$9C               {PUSHF}
                  /$FA               {CLI}
                  /$F2/$AB           {REP STOSW}
                  /$9D               {POPF}
                  /$07               {POP ES}

                );
    end
end;

{****initialize quick output variables and clear screen*********}
Procedure Init;
var
    index : integer;

begin
    x_pos := 0; y_pos := 0;
    Scroll (screenh)
end;

{*****move the cursor location down one line*************}
Procedure NextLine;
begin
    x_pos := 0;
    y_pos := y_pos + 1;
    if (y_pos >= screenh) then
    begin
         Scroll (1);
         y_pos := screenh - 1;
    end
end;

{*****write a string to the display screen***********}
Procedure QWrite (outstrng : strng);
var
    count : byte absolute outstrng;
    temp : intege;
    offset : integer;

begin
    for temp := 1 to count do
    begin
         value := attribute + Integer(outstrng [temp]);
         screen [y_pos][x_pos] := value; {put character on screen}
         x_pos := x_pos + 1;
         if (x_pos > (screenw - 1)) then       {wrap strings around}
              NextLine;
    end
end;

Procedure QWriteLn (outstrng : strng);
begin
    QWrite (outstrng);
    NextLine;
end;

  *main -- simple test program of quick write*********}
begin
    Init;
    for j := 1 to 20 do
    begin
         foQWrite ('this is so called hispeed output');
              QWriteLn ('       this is more')
         end;
         QWriteLn ('*****************************');
         for i:= 1 to 24 do
         begin
              Write ('this is normal Turbo output     ');
              WriteLn ('       this is more')
         end
    end;
end.
```

If you are planning on doing some "in-lining," I have some suggestions. First, get your program working with plain Jane Turbo Pascal. In-line code is difficult enough to debug without the rest of the program having problems. Second, figure out which sections need speeding up—don't bother with routines that are hardly ever executed unless they are very slow. Third, in-line code is particularly useful when instructions exist that can be applied to a particular problem—for example, in my case, the block-move instruction to perform scrolling. And, finally, do all the difficult stuff in Turbo Pascal. Notice in Listing 1 the calculations were left in Pascal because their contribution to the time required was not significant.

## Conclusion

Keep those suggestions, questions, complaints, and so on coming. Remember, I am just as interested in suggestions as I am in questions. (Complaints I'm not so excited about, but I still read whatever I get.) I am still sending out copies of the Turbo Pascal utility disk with both the public-domain TDebug and INLINE on it. §

*Stephen Randy Davis is a senior systems programmer for a defense contractor in Greenville, Texas, where he programs various microprocessors. He is also working on his Masters in physics.*

## Product Information

**TDebug-Plus**
Approx. $60
**TurboPower Software**
3109 Scotts Valley Rd., Ste. 122
Scotts Valley, CA 95066
(800) 538-8157 x830 (orders only)
(408) 438-8608

**GSI Pascal**
$79.95
**GSI**
1380 Old Freeport Rd.
PGH, PA 15238
(412) 963-7270

# Multitasking with Turbo Pascal

by Marshall Brain

Originally, I began thinking about multitasking systems while working on the design of a word processor for the IBM PC. As with most word processors, almost all of the CPU time used by the program was spent in a tight loop waiting for the user to enter keystrokes. Whenever the user wished to print a document, however, the machine would be tied up completely for several minutes while the program formatted each page and sent it to a print buffer. It seemed that if the time wasted waiting for keystrokes could be used for something more productive—such as formatting output being sent to the printer—a better product could be developed.

The ability to use previously wasted CPU time and to get more than one thing done by software at a time are strong motivators when attempting to build high-performance software. Recently, I completed a set of routines, collectively known as MTASKER, that implement a simple multitasking environment for Turbo Pascal programmers working on the IBM PC, PC/XT, and PC/AT.

The routines described in this article provide Turbo Pascal programmers with a complete, multitasking environment. The MTASKER environment includes multiple prioritized tasks, task pausing, and semaphores. It can be used to experiment with and learn about multitasking concepts and problems, as well as to add sophistication to Turbo Pascal programs. In this article, I will discuss the multitasking concepts, capabilities, and commands available under MTASKER, and the principles that make MTASKER possible.

Before going further, an important point should be made: the MTASKER routines present only one simple solution to the multitasking problem. For example, the priority scheme is simple, and the scheduler uses a round-robin approach. Also, a user working on an AT might wish to use many of the multitasking capabilities native to the 80286 to suit his own needs. This article, therefore, is intended to show you one possible approach, which you may wish to adapt or modify to suit your own specific requirements and facilities.

If you wish to add more sophisticated algorithms or techniques to MTASKER, there are many books and articles available on the subject of multitasking. One book is *The Mt. Xinu Approach.* Another is the textbook *Operating System Concepts* by Peterson and Silberschatz, which contains an excellent bibliography. Several articles are also listed in the following text. Because MTASKER is written in a high-level language where ever possible, it is a good system for experimenting with and learning about multitasking concepts.

## Multitasking Basics

Multitasking is an illusion created by software. This illusion causes the programmer and user to believe that a single CPU, which by design can do only one thing at a time, is doing several things at once. Multitasking is common on large systems, where many users can share a CPU simultaneously under operating systems such as UNIX, and is making its way rapidly into the world of microcomputers. One of the best-known multitasking systems for small computers is IBM's Topview for the IBM PC. It allows several programs to run at once on a single PC. Commadore's new Amiga computer also supports multitasking.

To create the illusion of multitasking, a program or operating system allows several different routines, called tasks, to be prepared for execution. The CPU is then made to switch between each of them so quickly that they appear to be running simultaneously. A more complete description of multitasking techniques and a good bibliography on the subject can be found

in the article "Add Multiple Tasks to Your Communications and Control Program" by Jerry Holter, in the September 1983 issue of *Byte* magazine.

Several interesting problems arise when multitasking systems are implemented. One of the most important of these is resource competition. Imagine that two tasks are running together, and both need to use the system's printer. If nothing is done to control the use of the printer, then it is possible that both tasks would begin sending their information to it at the same time. The result would be a garbled mess of intermixed lines and characters from the two tasks. The same holds true for other resources such as the disk drives, the screen, and even the individual variables used by several tasks. One device used to solve this problem is called a semaphore. MTASKER's use of semaphores is explained later in this article.

Another potential problem involves the efficiency of the system. To change from one task to another, some overhead processing must be done. If this processing takes too much time, the loss of performance for the system may be unacceptable. MTASKER is designed to be efficient and consumes only about 5 percent of the CPU's time to manage its multitasking capabilities (the exact percentage depends on your specific program).

### Using the MTASKER System

The MTASKER system consists of a set of routines that can be included at the beginning of a Turbo Pascal program. These routines manage the multitasking system. They allow the programmer to specify different procedures for simultaneous execution and then manage the concurrent operation of these tasks.

The heart of the MTASKER system consists of a structure called the dispatcher. The dispatcher keeps track of all of the tasks that are currently running and switches the attention of the CPU between them, using a basic round-robin scheme. The system will switch tasks for one of two reasons: either the system's hardware timer interrupt (which occurs approximately 18.2 times a second) causes a task switch, or a task requests a switch to a new task using the MTASKER's SWITCH_TASK command.

For MTASKER's dispatcher to work, it must first be initialized using the INIT_DISPATCHER command. Once the dispatcher is initialized, any number of tasks can be started by adding them into the dispatcher's list of running tasks. This is done using the START_TASK command. Once the program has finished using the dispatcher, the dispatcher should be removed using the REMOVE_DISPATCHER command.

Table 1 gives a complete list of the eight commands available for use under MTASKER and briefly describes the functions of each. The use of each of these commands is also demonstrated in the program listing shown in Listing 1. This code provides a realistic example of a simple multitasking program. The program includes a main task and four other tasks that run simultaneously with it.

Any program that uses the MTASKER routines must do two things. First, the stack checking (K) compiler directive must be turned off in the first line of the program. This must be done because of the way task stacks are implemented under MTASKER. Second, the program must include the MTASKER file right after any global variable declarations. These are both demonstrated in Listing 1. The MTASKER file can be obtained either by typing in the MTASKER listing shown in Listing 2 or by downloading the file from the subscriber network. During compilation, MTASKER adds only about ten seconds to the compilation time of the pro-

# THE DOCTOR MAKES HOUSECALLS!

## Get the diagnosis from the Doctor in your own home.

Subscribe to *Dr. Dobb's Journal* and enjoy the convenience of having your personal copy delivered to your home or office each month.

And you'll save over $5 off the cover price!

Every issue of *Dr. Dobb's* will bring you indispensable programming tools like algorithms, coding tips, discussions of fundamental design issues, and actual program listings. You'll find regular coverage of:

- Popular languages such as C, Assembly, Forth, Pascal, Ada, Modula-2, BASIC, FORTRAN, and Cobol.
- 68000 and 80x86 architectures
- The MS-DOS, and Unix operating systems
- Usable techniques and practical applications of AI and object-oriented programming research.
- New product reviews, and lively discussion of professional issues in software design.
- Compilers, cross assemblers and much more!

*Dr. Dobb's Journal of Software Tools . . .* the magazine that has lived up to its reputation as the foremost source of technical tools since 1976. One year (12 information-packed issues) is just $29.97—to subscribe simply mail in the attached card. But do it today . . . you won't want to miss *any* of the exciting issues we have planned.

gram being compiled (on a PC).

The demonstration program shown in Listing 1 is a rather liberal modification of the program called WINDOW.PAS that is shipped with Turbo Pascal. The main task, at the bottom of the program, is responsible for setup and take down of the MTASKER environment. It initializes the dispatcher and then starts the four tasks that run concurrently during the demonstration. Once the four tasks have been started, the main task pauses for 15 seconds (273 timer ticks/18.2 ticks per second = 15 seconds), and then removes the dispatcher before ending the program. This last step is important; the dispatcher MUST be removed before program completion, or the state of the system will be unreliable.

Under MTASKER, any Pascal procedure can be used as a task. The four tasks that run together in this demonstration are all fairly simple. The first displays random lines in the first window, much like the original window-demonstration program did. The second task waits for keys to be pressed by the user and then displays them in the second window. Note that whenever there are no pending keystrokes to be displayed, this task switches immediately to the next task. The third task opens a file and lists it in the third window. This task is unique in that it doesn't run forever like the other tasks. When it has completed its job, it deletes itself from the dispatch list. The fourth task is a background task (i.e., it has no output) that simply counts. Technically, the main task is also running in this demonstration program, but it is suspended for nearly the entire time.

**Multitasking is an illusion created by software. This illusion causes the programmer and user to believe that a single CPU is doing several things at once.**

Three of the tasks in this program make use of the subroutine FASTWRITE. This subroutine is similar to the many others that write characters directly into the screen buffer rather than using a BIOS call (as Turbo's WRITE command does). The use of this procedure speeds up the program, but also avoids another wait on BDOS, since FASTWRITE is reentrant.

Whenever a task is started, four parameters must be specified. The first parameter is the starting address of the task. This is demonstrated in the example program. The second parameter specifies the stack space that should be allocated for this task. The third parameter is the task's priority, 0 being the highest priority and 255 being the lowest. The background task in the demonstration program has a priority of 2, which means that it is only activated every third time the dispatcher finds it in the dispatch list. In contrast, priority 0 tasks are activated every time their turn arises in the dispatch list, priority 1 tasks every second time their turn arises, and so on. The final variable can be used to test whether or not the task started correctly.

These four tasks make extensive use of MTASKER's semaphore capability. Under MTASKER, a semaphore is a device that ensures that only one task uses a resource at a time (more generalized semaphore

| COMMAND NAME | PARAMETERS | DESCRIPTION |
|---|---|---|
| **Dispatcher Commands** | | |
| INIT_DISPATCHER | none | Initialize all dispatcher variables and interrupt vectors. |
| REMOVE_DISPATCHER | none | Completely remove the dispatcher. It must be used before the program terminates. |
| **Task Commands** | | |
| START_TASK | in : task address<br>in : stack space<br>in : task priority<br>out: OK | Place a task in the dispatch list so that it begins execution. The task address is the task's starting address, and the priority determines how often the task is used. Priority 0 is highest, 255 is lowest. You must also specify stack space needed. If task is started, OK = true. |
| SWITCH_TASK | none | Switch to the next task immediately instead of waiting for time slice to end. |
| PAUSE_TASK | in : time to pause | Suspend the current task for the specified number of timer ticks (approx 18.2/sec). |
| DELETE_TASK | none | Remove the current task from the dispatch list. |
| **Semaphore Commands** | | |
| WAIT | in : resource to wait for | Wait in queue for the specified resource to become available. Queue is FIFO. |
| AVAILABLE | in : resource to release | Signal that the current task is finished using the specified resource. Let next task in queue use it. |

**Table 1. Eight commands and their functions available under MTASKER.**

schemes might allow several tasks to access a resource at once). All other tasks wait in a first-in-first-out (FIFO) queue for their turn to use the resource. To signal that it wishes to use a resource, the task calls the WAIT command and specifies the resource that it wishes to use. If no other task is currently using the resource, the requesting task is given the resource immediately. Otherwise, it waits in line. When the task finishes using the resource, it signals this with the AVAILABLE command.

In this demonstration program, the resource that is in demand is the disk operating system (DOS) and BIOS. As described in the Turbo Pascal manual, DOS and BIOS are not reentrant. This means that only one task can write to the screen or use the disks or call almost any other Turbo I/O command at a time. Turbo Pascal adds another restriction: it seems that real-number calculations and I/O calls can not go on concurrently. Many other Turbo commands are not reentrant also. For all of these cases, it is important to use the BDOS semaphore to make sure that only one task uses DOS or BIOS at a time. It is also important that no task monopolizes a resource for long periods of time, or other tasks waiting for the resource will be unnecessarily delayed.

The user can specify any number of other semaphores for use under MTASKER. For example, a semaphore could be created to control access to a variable array. This array could then be used for intertask communication. Any resource or process that demands single-task access can make use of this semaphore capability by simply adding the resource name to the resource list in MTASKER.

**To use previously wasted CPU time and to do more than one thing at a time are strong motivators when attempting to build high-performance software.**

When creating your own programs using MTASKER, there are two things that should be remembered. First, debugging in a multitasking environment often demands a full bottle of aspirin. You are debugging not just one program but several different programs that are running together. All of these programs may also be interacting with one another. If something goes wrong, it is easy for your PC to become confused and lock itself up. To avoid headaches, it usually is best to develop and debug a task alone and then integrate it into the multitasking environment once it is working reliably. The demonstration program contains a special error-handling routine that will catch run-time errors and halt the system gracefully, but it is still easier to work with debugged tasks. Table 2 lists several common or easily made errors under MTASKER, as well as possible solutions, to help you with debugging.

The second problem is minor but important. Under MTASKER, it is possible to initialize the dispatcher, run several concurrent routines briefly, and then remove the dispatcher and continue under normal execution. This might be done many times during program execution. A problem arises if one of the tasks is nested within a

1. Failure to wait on BDOS—By far the most common problem is to use a routine that is not reentrant as though it were. My first experience with this came when I began placing real-number calculations in my programs. These routines are not reentrant and conflict with I/O commands, so you must wait on the BDOS semaphore for them. Be sure to wait on BDOS for all I/O commands also, since they call DOS or BIOS.

2. Insufficient stack space—This problem is especially bad on the AT. If a task requires more stack space than you allocate when the task is started, then the task will begin to fail eventually. On the PC, the task will simply begin writing into memory locations that do not belong to the task. On the AT, however, the 80286 will trap a stack overflow, using vector 0Eh as the trap vector. Under DOS, however, 0Eh is the disk interrupt, so one of the disks will begin spinning, and the machine will lock up. Be sure to allocate adequate stack space or more than enough space if you aren't sure how much you'll need.

3. Pop-up utilities—MTASKER has trouble working with some pop-up utilities, keyboard enhancers, and so forth, for all of the same reasons that these utilities have trouble working with each other. If you are having problems, try removing all memory-resident programs.

4. Bad compiler directives—Be sure that the K directive is off. Other directives have various effects. U+ doesn't seem to work at all under MTASKER, for example.

5. Dropping off the end of a task—Each task you start either should run as an infinite loop or use the DELETE_TASK command when it is finished (as Task 3 does in the demo). Do not allow any task simply to hit its final END statement.

6. Failure to remove the dispatcher—Before an MTASKER program terminates, the dispatcher must be removed with the REMOVE_DISPATCHER command. The variable MULTITASKING is true whenever the dispatcher is active.

7. Failure to call AVAILABLE—If a task waits on a resource, but then fails to release the resource using AVAILABLE (either because the AVAILABLE statement does not exist or because an infinite loop prevents it from executing), then other tasks waiting on that semaphore will be blocked permanently.

8. Deadlock—Deadlock is one of the biggest problems in any multitasking system. If Task A contains the following:

   ```
   WAIT(BDOS);WAIT(PRINTER);
   ```

   and Task B contains:

   ```
   WAIT(PRINTER);WAIT(BDOS);
   ```

   and if events occur in a certain sequence, then a deadlock situation could arise. Task A is granted BDOS, and task B is granted PRINTER, but both are waiting on the resource held by the other. This will block both of the tasks, and potentially many others, forever.

**Table 2. Common problems and solutions for MTASKER programs**

INSTANT ACCESS

# The COMMAND System™

NOW, the professional who doesn't need protection from DOS can organize files on a hard disk the right way without the hassle of long path names or DOS shells.

With The COMMAND System you get:

- Instant Access to Any Directory on Your Hard Disk
- Automatically Generated Short Names For Every Directory
- All These Programs Use Short Names
- Fast Programs to Copy, Erase, Rename, Move, Compare
  - Consistent Command Structure
  - Multiple Operations On a Line
  - Built-in Help
  - Super Programs to Show Files and Directories
- CompuMagic's SEARCH Program — Search ASCII or Word Processor Files by Line or Paragraph
- Uses Less than 20K of Memory

## INTRODUCTORY OFFER

**ORDER NOW FOR ONLY $75!**

**REGULAR PRICE $95**

**30-Day Money-Back Guarantee**

**Never Copy-Protected**

We still support CP/M!
The CompuMagic Utility Package $45
SEARCH $30

CALL NOW with your COD, VISA or MC order or send check or MO. The COMMAND System requires DOS 2.0 or higher and includes Full Documentation and a disk with 25 programs. Free UPS Ground Shipping. Add $2 for COD, $5 for overseas. (MD res. add 5%)

**CompuMagic, Inc.™**

P.O. Box 437
Severn, MD 21144
(301) 969-8068

*MAKING COMPUTERS WORK MAGIC SINCE 1983* procedure, and the task tries to access the nesting procedure's local variables. This cannot be done due to the way task stack space is handled. Global variables can always be accessed, however, as well as the task's own local variables. Tasks can also call other procedures and functions, each of which can have its own local variables.

### MTASKER Principles of Operation

The implementation of the MTASKER system is a bit involved. This is because the IBM PC, DOS, and Turbo Pascal were all designed to form a single-user, single-task system. The heart of the MTASKER environment is a routine called CHANGE_TASK. It is an interrupt-driven routine that creates a multitasking capability in this single-tasking setting.

The operation of the CHANGE_TASK routine is simple in concept. It is virtually invisible to the programmer and becomes active only when a task switch is needed. A task switch can be activated by two separate events: either through a hardware interrupt 1Ch, which occurs approximately 18.2 times a second to update the system's clock, or through a software interrupt using vector 60h. This vector is placed into a constant called SWITCH_TASK_VECTOR and can be changed by the user to another free vector if desired. The hardware interrupt determines a task's maximum *time slice*—no task will run longer than an eighteenth of a second unless it disables interrupts. The software interrupt is used by several MTASKER routines, including SWITCH_TASK, to force a task switch before the actual hardware time

Whenever CHANGE_TASK is activated by either interrupt, it does three basic things. First, it stores all of the current task's register values on the task's stack, thereby remembering the task's state so that the task can be restarted later. Second, it chooses the task that should be started next, taking into account the different task priorities, as well as the fact that some tasks will be suspended while they wait on a semaphore or pause. If there is no task that can be started (this only happens when all tasks are suspended), then CHANGE_TASK goes into a loop waiting. Otherwise, it starts the new task that has been chosen. This is done by switching the stack-pointer (both the segment and pointer register) to the new task's previous stack-pointer value and popping all of the task's previous register values. Once this has been done, an IRET will begin execution of the new task right where it left off.

CHANGE_TASK keeps track of all of the currently active tasks in a structure called the dispatch list. This list holds all of the information CHANGE_TASK needs for every task. For example, each task's priority, last stack value, pause count, and SUSPENDED flag reside in the dispatch list. The variable CURRENT_TASK points to the currently active task in this list. Whenever a task is first started, it is given a space in the dispatch list.

CHANGE_TASK does several other things that slightly complicate its design. If the current task needs to be deleted, then CHANGE_TASK handles the deletion so that no confusion results. CHANGE_TASK also checks the activity of hardware-interrupt handlers, such as the keyboard interrupt, which is handled by the BIOS. If, for example, the current task is in the middle of handling a keyboard interrupt, CHANGE_TASK will not cause a task switch. This allows the current task to finish handling the interrupt. If this were not done, it is possible that several tasks could be handling several closely spaced hardware interrupts at once, leading to a great deal of confusion. Currently, only the keyboard and disk interrupts are handled in this way.

Since CHANGE_TASK is an interrupt-driven routine, it must be placed in the 8088 interrupt vector table. (If you are not familiar with this table, you should read the article "A Peek into the IBM PC" by Tim Field in the March 1983 issue of *Byte*.) This is handled by the INIT_DISPATCHER routine.

INIT_DISPATCHER actually changes four interrupt vectors: it changes 1Ch and 60h to point to CHANGE_TASK, and it also changes the keyboard and disk vectors (09h and 0Eh) to point to two special routines that detect when they are active. INIT_DISPATCHER saves the initial values of these vectors to allow chaining and also to allow REMOVE_DISPATCHER to return the PC to its normal state when it is called. The changes made to the interrupt table cause the PC to be unstable if REMOVE_DISPATCHER is not called before an MTASKER program terminates.

The START_TASK routine is used whenever you wish to start a new task. It simply adds the task to be started into the dispatch list—the next time CHANGE_TASK gets around to that location in the list, the task will begin running. In order for this to work, however, START_TASK must place on the task's stack a copy of its initial register set for CHANGE_TASK to use. Most of the initial register values can be set to zero. The CS, IP, and flag values are set onto the stack so that an IRET will start a task at its first instruction.

In Turbo Pascal, the stack is used in several ways. As usual, it is used to hold flag, address, and register values whenever a subroutine call or interrupt occurs. The stack is also used for local variable storage for any procedure or function that needs it. To allow the different tasks under MTASKER to call other procedures and functions and to allow them to have their own local variables, each task is given its own stack. The stack space is allocated from the heap by START_TASK, with the amount of space being determined by the amount requested when START_TASK is called.

It is important to ensure that no task uses more than its allocated stack space, or havoc will result. To estimate the stack requirements for a task, start with the 24 bytes required for register storage space. Then determine the set of calls that will consume the most local variable space, and add in the number of bytes needed by this set of calls. Then add 8 bytes for each of these calls (address, flags, and BP and SP registers are pushed). Finally, add about 200 bytes to this for interrupts, etc., that can't be predicted. This number should then be passed in START_TASK. Sometimes it is easier simply to start with a large number (e.g., 5000) and not worry about it.

### Conclusion

Overall, the MTASKER environment is excellent for experimenting with and learning about multitasking. There are several enhancements that you may wish to make to MTASKER to match your own requirements. One of these might be a more advanced priority scheme. It would not be difficult to provide several different but completely separate priority levels to enhance the performance of the system. You might also want to add more formal task communication routines if intertask communication is something that you use often. If you use a lot of parallel routines that require a rendezvous point, you could add in a routine or structure to handle this. Whenever I need this capability, I normally use flags, but you might like a cleaner implementation. §

*Marshall Brain is an instructor at North Carolina State University, where he is finishing his master's degree in Computer Studies. He can be reached at Box 37224, Raleigh, NC 27627.*

### Listing 1—Assembly Language

```
;store 255 in entry_via_switch if entered from software int
;(switch_task_vector), 0 if entered via hardware interrupt
;
     mov byte ptr cs:entry_via_switch,255 ;entered from swch_task
     jmp short aaa
     mov byte ptr cs:entry_via_switch,0 ;entered thru timer tick
aaa: push ax                            ;save AX so it can be used
;
;if in any hardware service routine (BIOS), then wait until it
;is done
;
     mov al,cs:in_kbd ;in_kbd will 0
             if not in keyboard handler (BIOS)
     and al,al
     jnz yyy
     mov al,cs:in_disk ;in_disk will 0 if not in BIOS
     and al,al
     jz xxx
;
;jump back to task that was interrupted if that task is in a
;hardware service routine. If entered through hardware interrupt
;(entry_via_switch=0) then jump to rest of interrupts in the chain
;of interrupts normally hooked into vector $1c. If entered through
;software intr, then simply return.
;
yyy: mov al,cs:entry_via_switch   ;check value of flag
     and al,al
     pop ax                       ;get original AX back
     jz bbb
     sti;simple iret for sftwr int
     iret
bbb: sti                    ;jump to remainder of chain for hdwr int
     jmp cs:timer_tick_ofs
;
;if waiting, then don't push anything because no task currently
;active.
;
xxx:   mov al,cs:waiting
     and al,al
     jz zzz
     pop ax
     jmp short www
;
;save all register values for current task, so task can be
;restarted
;
zzz: push bx
     push cx
     push dx
     push si
     push di
     push bp
     push es
     push ds
;
; save stack info for next startup of this task
;
     mov cs:temp_SS,SS
     mov cs:temp_SP,SP
;
;set up so turbo can work normally with arrays in DS. This
;is a Turbo pascal quirk - without these instructions Turbo
;won't work properly with arrays.
;
www: push bp
     mov sp,bp
     push bp
;
;get turbo's current data segment so it can find global vars.
;
     mov ax,cs:data_segment
     mov ds,ax
```

### Listing 2—Assembly Language

```
;restore all variables of new task. CS, IP, and flags
;will be restored by eventual IRET instruction.
;
     mov ss,cs:temp_ss
     mov sp,cs:temp_sp
     pop ds
     pop es
     pop bp
     pop di
     pop si
     pop dx
     pop cx
     pop bx
;
;start up new current_task through 1Ch int chain or iret
;
yyy:  mov al,cs:entry_via_switch
      and al,al
      pop ax
      jz ccc
      sti
      iret
ccc:  sti
      jmp cs:timer_tick_ofs
```

### Listing 1—TASKDEMO source code

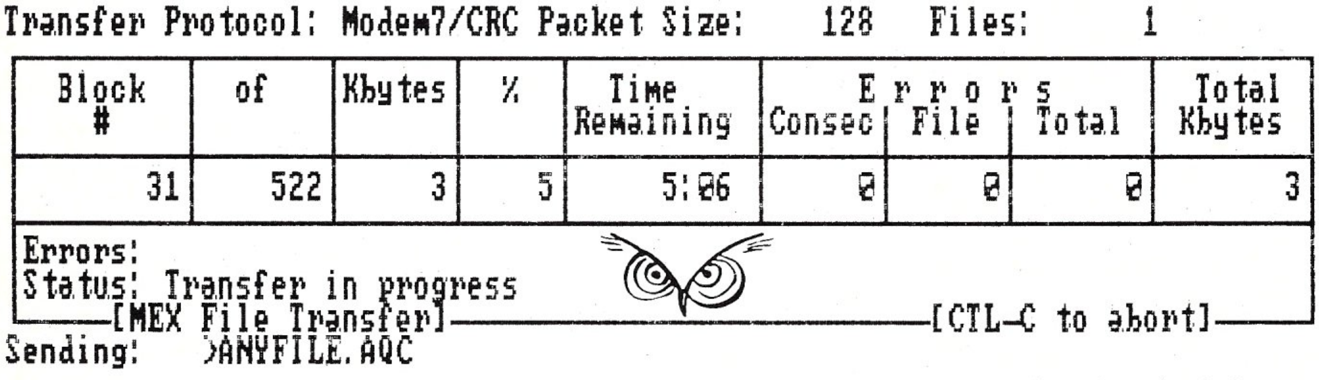
```
{TASKDEMO-

{by Marshall Brain    Apr 22, 1986}
```

```
{Modified version of WINDOW.PAS demo program shipped with Turbo}
{$K-}   {K directive MUST BE OFF}


type string80=string[80];
const
   Windows     = 3;
   Wtab        : array[1..Windows,1..4] of Integer
                 = (( 2,  2,  78,  7), { X0,Y0,X1,Y1 }
                    ( 2, 10,  78, 13),
                    ( 2, 16,  78, 21));
   normal_attr=$07;
var i:byte;ok:boolean;count:integer;
    result : record ax,bx,cx,dx,bp,si,di,ds,es,flags: integer; end;

{$I MTASKER.PAS}  {include mtasker right below global declarations}

   procedure Frame(UpperLeftX, UpperLeftY, LowerRightX, LowerRightY:
Integer);
   {produces square shape using graphics characters.
            All constants are
    ASCII graphics character numbers.}
   const
     top_left_corner=218;
     top_right_corner=191;
     bottom_left_corner=192;
     bottom_right_corner=217;
     horiz_line=196;
     vert_line=179;
   var
      i: Integer;
   begin
      GotoXY(UpperLeftX, UpperLeftY);  Write(chr(top_left_corner));
      for i:=UpperLeftX+1 to LowerRightX-1 do Write(chr(horiz_line));
      Write(chr(top_right_corner));
      for i:=UpperLeftY+1 to LowerRightY-1 do
      begin
         GotoXY(UpperLeftX , i);  Write(chr(vert_line));
         GotoXY(LowerRightX, i);  Write(chr(vert_line));
      end;
      GotoXY(UpperLeftX, LowerRightY);
      Write(chr(bottom_left_corner));
      for i:=UpperLeftX+1 to LowerRightX-1 do Write(chr(horiz_line));
      Write(chr(bottom_right_corner));
   end;

   procedure SelectWindow(Win: Integer);
   begin
      Window(Wtab[Win,1], Wtab[Win,2], Wtab[Win,3]; Wtab[Win,4])
   end  { SelectWindow };

   procedure fastwrite(col,row,attrib:byte;str:string80);
   { displays string STR at COL,ROW using attribute ATTR by writing
     directly into the memory space of mono adapter or CGA. See text
     for further explanation.}
   begin
     inline
       ($1E/$1E/$8A/$86/row/$B3/$50/$F6/$E3/$2B/$DB/$8A/$9E/col/
        $03/$C3/$03/$C0/$8B/$F8/$be/$00/$00/$8A/$BE/attrib/
        $8a/$8e/str/$22/$c9/$74/$3e/$2b/$c0/$8E/$D8/$A0/$49/$04/
        $1F/$2C/$07/$74/$22/$BA/$00/$B8/$8E/$DA/$BA/$DA/$03/$46/
        $8a/$9A/str/$EC/$A8/$01/$75/$FB/$FA/$EC/$A8/$01/$74/$FB/
        $89/$1D/$47/$47/$E2/$Ea/$2A/$C0/$74/$10/$BA/$00/$B0/
        $8E/$DA/$46/$8a/$9A/str/$89/$1D/$47/$47/$E2/$F5/$1F);
   end;

   procedure task1; {displays random lines}
   var x:integer;line:string[80];
   begin
     x:=1;
     while true do  {infinite loop}
     begin
       wait(bdos);  {WAIT is used here because selectwindow, GotoXY,
                     delLine, etc. all reference BIOS, which is not
                     reentrant. It is
             important that all calls such as
                     these be within wait-available pairs}
         SelectWindow(1);
         GotoXY(1,1);
         DelLine;
         str(x,line);
         line:='Line '+line+' |
0123456789012345678901234567890123456789';
       available(bdos);
       fastwrite(1,Wtab[1,4]-1,normal_attr,line);
       x:=x+1;
     end;
   end;

   procedure task2; {displays all keystrokes}
   var ch:char;x,y:byte;

     function keypress:boolean;  {wait before using keypressed}
     begin
(bdos);
      keypress:=keypressed;
    available(bdos);
  end;

begin
  x:=3;y:=9;
  while true do   {repeat forever}
  begin
    while not keypress do switch_task;
    x:=x+1;
    if x>70 then
    begin x:=3;y:=y+1; end;
    wait(bdos);
      Read(KBD, Ch);
      fastwrite(x,y,normal_attr,ch);
      available(bdos);
    end;
  end;

  procedure task3; {Lists the Turbo window demo on the screen}
  var f:text;line:string[255];

    function eofile(var f:text):boolean;  {wait before using eof}
    begin
      wait(bdos);
        eofile:=eof(f);
      available(bdos);
    end;

  begin
    wait(bdos);
      assign(f,'window.pas');
      {$I-}  reset(f);  {$I+}
    available(bdos);
    while not eofile(f) and (ioresult=0) do
    begin
      wait(bdos);
        SelectWindow(3);
        GotoXY(1,1);
        delline;
        readln(f,line);
        line:=copy(line,1,77);
      available(bdos);
      fastwrite(1,Wtab[3,4]-1,normal_attr,line);
    end;
    wait(bdos);close(f);available(bdos);
    delete_task;
  end;

  procedure background_task;
  { This tasks simply counts in the background.}
  var x:integer;line:string[40];
  begin
    count:=0;
    while true do
    begin
      for x:=1 to 10000 do;
      count:=count+1;
    end;
  end;

procedure error(errno,erraddr:integer);
{ run time error routine that determines if dispatcher
  is currently active and adjusts itself accordingly.}
var line:string[40];

  procedure hex(x:integer);
  const num:string[16] = '0123456789ABCDEF';
  begin
    line:=line+num[1+(hi(x) shr 4)]+num[1+(hi(x) and $0f)]+
               num[1+(lo(x) shr 4)]+num[1+(lo(x) and $0f)];
  end;

begin
  inline($fa); {cli}  {disable interrupts - keeps dispatcher
           from switching out of the erroneous task.}
  window(1,1,80,25);
  gotoxy(1,25);
  if (lo(errno)=1) then
  begin
    line:='**** USER BREAK AT ';hex(erraddr);
  end
  else
  begin
    case hi(errno) of
      1:line:='**** I/O ERROR ';
      2:line:='**** RUN-TIME ERROR ';
    end;
    hex(lo(errno));line:=line+' AT ';hex(erraddr);
  end;
  line:=line+' ****';
  fastwrite(1,24,normal_attr,line);
  if multitasking then      {if currently multitasking, remove
           dispatcher gracefully, so machine doesn't lock up.}
  begin
    if current_task>1 then
    begin
      temp_ss:=dispatch_list[1].last_ss;
      temp_sp:=dispatch_list[1].last_sp;
      inline
      ($2E/$8E/$16/temp_ss/   {mov ss,cs:temp_ss}
       $2E/$8B/$26/temp_sp);  {mov sp,cs:temp_sp}
    end;
    remove_dispatcher;
  end;
  inline($fb); {sti}
  halt;
end;

begin
   errorptr:=ofs(error);  { Must be in plce to allow error checking}
   clrscr;
   GotoXY(13,23);
   Write('TURBO PASCAL Multitasking Demo  -  Stops in 15 seconds');
   GotoXY(13,24);
   Write('Displays random lines, all keys hit, and file listing.');
   for i:=1 to Windows do
      Frame(Wtab[i,1]-1, Wtab[i,2]-1, Wtab[i,3]+1, Wtab[i,4]+1);
   count:=0;
   init_dispatcher;
   {start all tasks}
   start_task(ofs(task1),1000,0,ok);  {task name,stack space
          needed by task, priority, ok flag returned}
   start_task(ofs(task2),1000,0,ok);
   start_task(ofs(task3),1000,0,ok);
   start_task(ofs(background_task),1000,2,ok); {priority 2}
   pause_task(273); {wait for 15 seconds (273/18.2=15 seconds)}
   wait(bdos); {wait before removing dispatcher. this insures that no
                routine is in the middle of DOS or BIOS call when
                dispatcher is eliminated}
   remove_dispatcher;
   Window(1,1,80,25);
   gotoxy(1,25);
   {check how far background task got}
   writeln;writeln;writeln('Background Task got to ',count,'0000');
end.
```

### *Listing 2—MTASKER source code*

```
{MTASKER -
 by Marshall Brain    Apr 22, 1986
 This set of routines implements a simple multitasking system
 under Turbo Pascal.}

const
  max_tasks=5;              {Max tasks - make any size but watch
                             memory consumption due to stack space}
  initial_flags=$0200;      {flags register whenevr task first started]
  switch_task_vector=$60;   {interrupt vector used by SWITCH_TASK and
                             other routines to force a task switch via
                             a software interrupt}
  timer_tick_vector=$1c;    {timer tick interrupt vector}
  disk_vector=$0e;          {BIOS Disk handler vector}
  keyboard_vector=$09;      {BIOS keyboard vector}
type
  sem = record
        queue:array[1..max_tasks] of integer;  {array used to hold
                                 ID's of all tasks waiting on a certain
                                 semaphore. The array is used as a
                                 linked list.}
         link:array[1..max_tasks] of integer;   {array used to
                                 hold links for linked list.}
         num_in_q:byte;                  {number of tasks
                                 waiting on the semaphore.}
         free_first:byte;                {pointer to first
                                 free block}
         q_first:byte;                   {pointer to first
                                 task waiting on the semaphore}
         q_last:byte;                    {pointer to last
                                 waiting task}
        end;
  resources = (printer,bdos); {resource list - add any new ones
                               that you need}
                              {but bdos MUST remain the last one in
                               the list}
const {typed constants used for variable storage in code segment}
  data_segment : integer = 0;  {required to remember DS - see
                                      Turbo manual}
  number_of_tasks: byte=0;      {current number of tasks being
                                 handled}
  {flags}
  waiting : byte=0;             {indicates that no tasks running
                                currently}
  in_kbd: byte=0;              {indicates that currently servicing
                               keyboard interrupt in BIOS}
  in_disk: byte=0;             {indicates that currently servicing
                               disk in BIOS}
  multitasking: boolean=false; {indicates that multitasker is
                               currently operating}
  entry_via_switch: byte = 0;  {indicates if entered change_task via
                               hardware or software interrupt}
  temp_SS : integer = 0;       {temp holder for stack pointer}
  temp_SP : integer = 0;
  timer_tick_ofs: integer=0;   {saved values for changed interrupt
                               vectors.  The}
  timer_tick_seg: integer=0;   {old values are saved in these
                               locations so that}
  kbd_int_ofs: integer=0;      {they can be retrieved by
                               REMOVE_DISPATCHER}
  kbd_int_seg: integer=0;
  disk_int_ofs: integer=0;
  disk_int_seg: integer=0;
  switch_task_ofs: integer=0;
  switch_task_seg: integer=0;
var
  dispatch_list:array[1..max_tasks] of record  {array of all active
                               tasks}
                 task_name:integer;        {starting addr also}
                 last_SS:integer;          {task's last stack
                               pointer val}
                 last_SP:integer;
                 stack_space_alloc:integer;{size of stack allocated}
                 stack_ptr:^Integer;       {pointer to stack space
                               on heap}
                 suspended:boolean;        {indicates if task is
                               suspended}
                 priority:byte;            {task's priority}
                 priority_count:integer;
                 pause_count:integer;      {current val of pause
                               cntr}
               end;
  semaphores : array[resources] of sem;
  current_task:byte;          {number of the currently active task}
  disp_templ:byte;            {temp values used by CHANGE_TASK}
  disp_stopflag:boolean;
  next_dispatchable:byte;     {next task that will be dispatched}
  next_possible:byte;         {next suspended task that could be run
                               in case no immediately dispatchable
                               tasks}
  disp_result:record ax,bx,cx,dx,bp,si,di,ds,es,flags:integer end;
  disp_del_flag:boolean;      {"delete current task in CHANGE_TASKS"
                               flag}

{The following two procedures reroute the keyboard and disk ints so
 that CHANGE_TASK can know if one of these ints is currently being
 handled. If it is, CHANGE_TASK won't do a task switch. You may wish
 to add identical routines for RS-232 and printer hdwr interrupts if
```

**Multitasking with Turbo Pascal** *code continued on page 57*



```
Transfer Protocol: Modem7/CRC Packet Size:    128    Files:      1
```

| Block # | of | Kbytes | % | Time Remaining | Errors Consec | File | Total | Total Kbytes |
|---|---|---|---|---|---|---|---|---|
| 31 | 522 | 3 | 5 | 5:06 | 0 | 0 | 0 | 3 |

```
Errors:
Status: Transfer in progress
——[MEX File Transfer]——————————————[CTL-C to abort]——
Sending:   >ANYFILE.AQC
```

# NEW COMMON LISP Development System for Your PC or AT

## *Introducing TransLISP PLUS™, The Consultant's LISP*

## Over 400 Primitives, a C Interface, and Optional Runtime System

People call you because you're an expert. Your customers want to keep up with what's going on. They want software that is more intelligent and responsive, or software that does something that just wasn't possible before. So they call you.

Now you can write and deliver a whole new category of software with TransLISP PLUS, a practical, efficient LISP system. You can also add Artificial Intelligence to your software. We include several features, like a C Interface and Optional Runtime System, so you can control the performance and security of your programs. And your users only need to have a PC (can even be non-compatible) with 320K and 1 floppy drive. Now, what could you write for a $700 PC? . . . or a $7000 PC? . . .

### Add AI Technologies to Your Software

Most of the programs you write are accessed by a user who doesn't have your expertise. Use intelligent interfaces to make your programs more responsive to the end user.

You can even use the C Interface included with TransLISP PLUS to customize LISP, or combine C functions with LISP programs.

Take advantage of AI technologies to make your programs smarter and more flexible.

### Extensive Development Environment Over 400 Primitives

TransLISP PLUS provides you with *over 400* primitives for development, including extras for hardware support and operating system access. Their spectrum ranges from control constructs, macros, and special forms, to multi-dimensional arrays, reader support for binary, octal, and hex constants, improved list processing, and system interrupts.

DOS commands and applications can be invoked from within TransLISP PLUS, as can the fast editor. Of course, you can use your own editor if you like.

A variety of debugging tools are provided. The trace facility tracks the evaluation of any built-in or user-defined function or macro.

Traceback, Break, Cross Reference, and Pretty Printer are also provided to help you spot problems.

### The ONLY Full Featured Common Lisp with a C Language Interface

The best of both worlds. The interface to Microsoft C gives you a powerful extension to TransLISP PLUS – now you can write code in LISP and C. And you don't need an AT, it will run on your PC!

The C Interface makes it practical for you to write a C program and add it as a new function to TransLISP PLUS. Your function can:

- extend and/or change the LISP syntax
- be an entire system of programs

Create your own BUILT-IN primitives which are directly tied to the system and called at full speed by the interpreter. Extend the functionality of your program by including features of your own like macros, functions, and special forms.

Code from C libraries produced by other vendors can be integrated into your program to perform tasks not normally part of LISP.

### Use PLUS for Your Applications. No Royalties.

Once you own TransLISP PLUS, you may want to use it to distribute applications. No problem.

The Optional TransLISP PLUS Runtime supplies you with a special interpreter. You can distribute an executable version of your program without distributing source code.

The Runtime is available for $150 and TransLISP PLUS is required.

### Don't Know LISP?

Get a solid understanding of LISP with the comprehensive, easy-to-understand tutorial. Each section walks you through a new concept and reinforces it with examples — both text and online.

Over 30 demo programs supplement the tutorial. Use them for an in-depth introduction to LISP programming techniques. Commented source is included so you can see how and why the program operates.

The demos cover a wide variety of applications including: Select a word processor, Read dBASE SDF files, Job Counselor, and many, many more.

### The Fundamentals

If you are interested in learning LISP but don't need all the extras in TransLISP PLUS, order TransLISP for $95. It's a full, easy-to-use introduction to LISP that includes the tutorial and demo programs described above, and over 300 primitives.

It is a solid subset of COMMON LISP; and you can write programs of up to 12000 lines.

#### COMMON LISP Standard

Programs written carefully with TransLISP PLUS will be completely "portable" to any other COMMON LISP system on a micro, mini, or mainframe computer. This allows you, for example, to write a program with TransLISP PLUS on your PC at home, and compile and run it on the VAX at work.

#### System Requirements

TransLISP PLUS requires 320K RAM and a 360K disk drive.
TransLISP requires 256K RAM and a 360K disk drive.

| | |
|---|---|
| TransLISP PLUS . . . . . . . . . . | $195 |
| TransLISP . . . . . . . . . . . . . . | $ 95 |
| Upgrade TransLISP to PLUS . . . | $158 |
| TransLISP PLUS Runtime . . . . . | $150 |

- Over 400 COMMON LISP Primitives
- Optional Runtime (No Royalties)
- Interpreter
- Program Editor
- Many Debugging Utilities
- Microsoft Mouse Support
- Supports IBM PC color graphics
- Supports 8087 math coprocessor
- Over 30 Demo Programs with Source
- C Language Interface
- Complete Manual with Tutorial, Indexed Reference Manual, and Quick Reference Card
- Online Help
- Lexically scoped
- Use your C Libraries
- System Interrupts
- **NOT COPY PROTECTED**

***Use TransLISP PLUS to program with and deliver to lots of machines . . . Use your existing C libraries . . . Distribute your applications***

**TO ORDER OR FOR DETAILS CALL**

MasterCard (617)337-6963 VISA

## MONEY BACK GUARANTEE

**Try TransLISP PLUS ($195) for 30 days — if not satisfied get a full refund.**

Solution Systems™

541 Main St., Ste. 410—M, So. Weymouth, MA 02190 (617) 337-6963

TransLISP and TransLISP PLUS are trademarks of Solution Systems. Solution Systems is a trademark of Solution Systems.

# 8086 Assemblers

by Stephen Randy Davis

## A Comparison of Five Popular 8086 Assemblers

Before the early 50s, programmers wrote programs on large paper pads coding with individual 1s and 0s. Though laborious, it was not too terrible because the monstrosities of the day could only handle programs of a few hundred bytes. As computer memory grew, the size of programs grew, making the job of writing programs more and more of a problem. Early on, programmers realized it was easier to write MOVE #1, A to move a 1 into the A register and let the computer translate this into 10011100, or whatever. So was born the first assembly language, and machine language disappeared almost immediately.

It is even easier to write programs in high-level languages, and so programmers invented FORTRAN, COBOL, ALGOL, and so on. But assembly language refuses to die. Why? Higher-level languages always involve some overhead. Take the same program for example, write it in assembly language and again in a high-level language. The assembly-language version will take less binary code and will run faster—usually twice as fast—sometimes more than that.

People enjoy performance. Take two programs that perform the same function. The faster version will outsell the slower one every time. Thus it is that the topic of assemblers remains relevant even in these days of Pascal, Ada, and C++. This review compares five of the most popular assemblers for the IBM PC and its compatibles. I have directed this review both to both non-assembly-language programmers thinking about taking the plunge and assembly-language programmers who are unhappy with their current assembler.

### Features of Modern Assemblers

So what's the big deal? Converting simple mnemonics into machine language does not sound all that difficult. Having written an assembler myself, I can tell you it isn't all that easy. But the point is well taken. If all that modern assemblers did was take simple mnemonics with simple addresses and generate an executable binary, there would not be much to differentiate one assembler from another. Of course, modern assemblers do much more: optional addressing modes, multiple segments, .OBJ file output, macros, structures, and even strong typing are all available in one or more of the assemblers I've reviewed. What are these capabilities?

It is often inconvenient to reassemble an entire project just to make one small change. It would be so much nicer to break a large project up into parts. Then, only an individual module would need to be edited, reassembled, and linked back in with the other unchanged parts. In the MS-DOS/PC-DOS world, these smaller parts are called object files and they carry the extension .OBJ. It is the job of the LINK program to connect the object files into a single executable .EXE file. Furthermore, LINK can be instructed to place different parts of the program into different segments. This segment information is stored in a header tacked onto the front of the .EXE file. Therefore, .EXE files are not limited to 64K of program or data. The ability of an assembler to generate an .OBJ file output is of critical importance for large projects.

On the other hand, on small projects it is much faster to reassemble the entire program, generating an executable binary file directly and avoiding the cumbersome linking process. Such a binary file carries the extension .COM. Because the .COM files have no header information, they are limited to single-segment programs, so .COM programs cannot be larger than 64K in length. The ability to create a .COM file directly is not critical, but its very convenient for small jobs.

The 8086 introduced some fairly complex addressing modes. For some reason, a consensus on the exact format of some of these modes was never reached. For example, all the following statements are equivalent:

```
MOV AX,[BX+SI+LABEL]
MOV AX,[BX+SI].LABEL
MOV AX,LABEL[BX+SI]
MOV AX,LABEL[BX][SI]
```

Beginners probably don't particularly care, but for those who have already adopted a standard, it is nice not to have to change. The best assemblers accept all of the above formats interchangeably.

Often, a particular sequence of assembler code is used again and again. For example, to output a string to the screen in PC-DOS, you could use the following code:

```
LEA    DX,OUR_MSG
MOV    AH,9H

INT    21H
```

Modern assemblers allow programmers to define a single label that is equivalent to a series of assembler instructions. Such a label is known as a macro. Not only does using a macro save coding time, but the resulting code is easier to read. In the preceeding example, you could define a macro WRITE and invoke it with:

```
WRITE OUR_MSG
```

# EVEN MORE POWER AND FLEXIBILITY

# BRIEF 2.0

Users and industry press alike have unanimously proclaimed BRIEF as the best program editor available today. Now, the best gets better, with the release of BRIEF 2.0.

Straight from the box, BRIEF offers an exceptional range of features. Many users find that BRIEF is the only editor they'll ever need, with features like real, multi-level Undo, flexible windowing and unlimited file size. But BRIEF has tremendous hidden power in its exclusive macro language. With it, you can turn BRIEF into your own custom editor containing the commands and features you desire. It's fast and easy.

Jerry Pournelle, columnist for BYTE magazine summed it all up by saying BRIEF is, "**Recommended. If you need a general purpose PC programming editor, look no further.**" His point of view has been affirmed by rave reviews in C JOURNAL, COMPUTER LANGUAGE, DR. DOBB'S JOURNAL, DATA BASED ADVISOR, INFOWORLD AND PC MAGAZINE.

One user stated "BRIEF is one of the few pieces of software that I would dare call a masterpiece." Order BRIEF now and find out why. **BRIEF 2.0 is just $195. If you already own BRIEF, call for upgrade information.**

**TO ORDER CALL: 1-800-821-2492**
**(in MA call 617-659-1571)**

As always, BRIEF comes with a 30 day money-back satisfaction guarantee.

Solution Systems™
335 Washington St.
Norwell, MA 02061
(617) 659-1571

## Look at these BRIEF 2.0 enhancements!

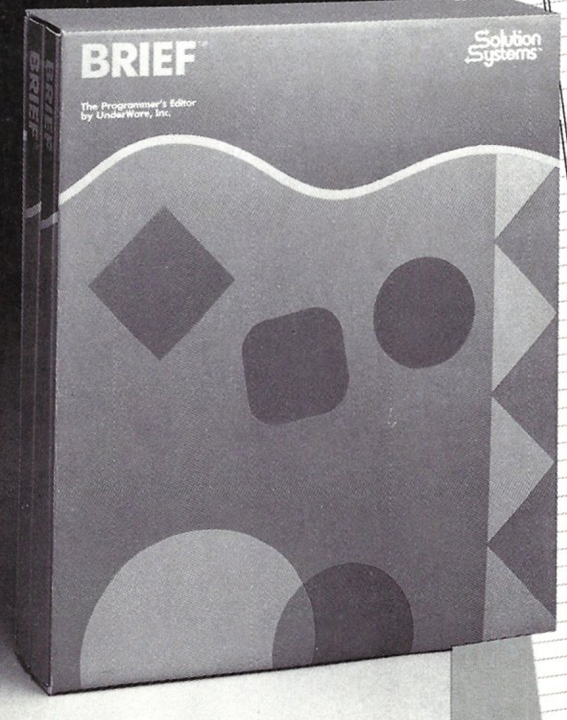
### Main Features:

- All new documentation with tutorials on basic editing, regular expressions **and** the BRIEF Macro Language.
- Setup program for easy installation and configuration. (Requires no knowledge of the macro language)
- Increased speed for sophisticated operations like Undo and Regular Expression Search.
- Expanded regular expressions, with matching over line boundaries.
- More block types, with marking by character, line or column.
- Command line editing (move cursor, add and delete characters, specify command parameters).
- Support for more programming languages.
- Optional borderless windows.
- Enhanced large display support, including wider displays.
- Reconfigurable indenting for C files (supports most indenting styles).

Plus the basic features that made BRIEF **SO** popular!

### Basic Features:

- Full multi-level Undo
- Windows
- Edit many files at once
- File size limited only by disk space
- Automatic language sensitive indentation



Requires an IBM PC or compatible with at least 192K RAM.
BRIEF is a trademark of UnderWare, Inc.
Solution Systems is a trademark of Solution Systems.

to replace the three lines above.

By the same token, it is often convenient to group data to highlight the relationship between them. Such a grouping is known as a structure. Grouping of bit fields within a word is known as a record. Macro, structure, and record capabilities are not absolutely required of an assembler, even when working on large jobs, but they can simplify the job of creating a working program. Even more important, they can add greatly to the readability of the resulting code.

Although people generally speak only of the 8086 and the software-equivalent 8088, there are actually several other members of the 86 line. Intel's 186, 286, and 386 processors and NEC's V-Series processors all have noteworthy extensions to the 86 instruction set. Of course, making use of these extensions reduces the potential audience of the resulting program, but if you are only writing code for yourselves, that is of little concern. The ability of an assembler to understand these processor extensions is a useful but not mandatory feature.

Somewhat more important are the instruction extensions necessary to access the 8087 and 80287 numerical data processors. These NDP's plug into the slot next to the 8088 in the IBM PC and most compatibles to increase numerical-processing performance. To have any effect, programmers must address these chips using the so-called floating-point instruction set. It is an absolute requirement of any assembler intended for numerical work that it understand 8087/287 opcodes.

As mentioned earlier, assemblers that do not generate object files cannot generate more than one segment. Those that do, however, allow even a single source file to contain more than one segment. Because a single segment can be up to 64K in length, this may seem odd, but remember that it is often convenient for the data and code to be in different segments. This capability is an important one, although not absolutely critical. With the idea of making the whole concept of multiple segments more palatable, Microsoft introduced early on the concept of the ASSUME statement. Too complicated to explain in detail, the net effect is to allow the assembler to generate segment override statements automatically. Personally, I find this ability not too significant.

When I first struggled to generate a simple say Hello program with Microsoft's MASM, Version 1.0, I was struck by the immense difficulty. Multiple segments are a powerful capability, but Microsoft made them into a liability by requiring even the rankist beginner to define segments, provide ASSUME statements, and so on. This problem has been addressed in some modern assemblers by the noname segment. The noname segment and associated ASSUME statements are automatically defined whenever users do not define their own. This allows beginning programmers to generate simple programs without the need to deal with these advanced concepts. Again, this is not a requirement, but I consider it a very important feature.

A good assembler should be able to generate symbol tables and cross-references files. It should be able to adapt its listing file to different-size page lengths and carriage widths, and it should allow INCLUDE files and multiline comments. Any other feature it adds is welcome, but whatever else, a good assembler must be fast. Even more than with higher-level languages, generating a working assembly-language program is an iterative pro-

| **Command Line** | **MASM** | **PASM** | **TASM** | **CHASM** | **WASM** |
|---|---|---|---|---|---|
| Invoking assembler (P, C, M) (Prompted, Command line, Menu) | P, C | C | P, C, M | P, C | P, C |
| Built-in editor | - | - | X | - | - |
| .OBJ file output | X | X | X | - | - |
| .COM file output | - | - | X | X | X |
| List file output | X | X | X | X | X |
| Symbol table output | X | X | X | X | X |
| Turbo INLINE output | - | - | - | X | - |
| BASIC BSAVE output | - | - | - | X | - |
| MAKE utility | X | - | - | - | - |
| Assem multiple sources | - | X | - | - | - |
| **Listing Format Control** | | | | | |
| Title/subtitle | X | X | X | - | X |
| Page | | | | | |
| user definable width/length | X | X | X | (b) | X |
| automatic printer mode control | - | - | - | (b) | X |
| Multiline comments | X | X | X | - | - |
| Listing on/off | X | X | X | - | X |
| INCLUDE files | X | X | X | X | X |
| RADIX command | X | X | X | X | - |
| **General Assembler Considerations** | | | | | |
| MASM compatible | X | X | (a) | - | - |
| $ Pseudo-op | X | X | X | X | - |
| Unnamed code segment | - | X | X | X | X |
| Multisegment | | | | | |
| supported | X | X | X | - | - |
| ASSUME pseudo-op | X | X | X | - | - |
| ? Pseudo-op | X | X | X | - | (c) |
| **Address Formats** | | | | | |
| Immediate values | | | | | |
| ASCII strings | X | X | X | X | X |
| expressions allowed | X | X | X | X | X |
| compound expressions | X | X | X | X | - |
| number of arith ops | 11 | 11 | 11 | 4 | 5 |
| Legal address formats | | | | | |
| [BX+SI+label] | X | X | X | - | X |
| [BX+SI].label | X | X | X | - | - |
| label [BX+SI] | X | X | X | X | - |
| label [BX][SI] | X | X | X | - | - |

cess—it is often assembled many times before it is complete. The perfect assembler would be infinitely fast and have a built-in editor and debugger.

## Individual Reviews

For a quick comparison of the most important features of each assembler I have reviewed, refer to Table 1. Use the preceding discussion of assembler concepts to decide which features you think you will need and which you will not. For example, if you have never written a program larger than 16K, multisegment capability is probably not an important feature. None of the assemblers I tested had every feature, so some weighting will be necessary to arrive at the best choice.

I gave much thought to arriving at a subjective ease-of-use factor for each assembler. I finally decided upon a plan. First, I coded up the well-known Sieve of Eratosthenes in a general assembler format (Listing 1). In this program I tried to use every feature I thought important, even though it might not be absolutely necessary. In addition, I tried to use all the legal addressing modes. The resulting General Sieve did not assemble properly on any of the assemblers I reviewed, although it came remarkably close on one.

I converted the General Sieve to meet the requirements of each of the assemblers. The more features, the better the error messages, and the less finiky the assembler, the easier was the conversion. Not only is ease of use an important consideration, but using each assembler to perform the conversion gave me a better feel for each one.

| **Address formats cont.** | **MASM** | **PASM** | **TASM** | **CHASM** | **WASM** |
|---|---|---|---|---|---|
| Ambiguous type references | | | | | |
| B, W suffix | - | - | - | X | - |
| BYTE, WORD pseudo-op | X | X | X | - | X |
| Typing (strong, weak, or none) | S | S | W | N | N |
| Segment override | | | | | |
| colon format | X | X | X | X | - |
| SEG pseudo-op | - | - | - | X | X |
| automatic override | X | X | X | - | - |
| Forward references allowed | X | X | X | X | X |
| **Advanced Topics** | | | | | |
| Macros | | | | | |
| macro capability | X | X | X | X | - |
| local labels | X | X | X | X | - |
| local comments | X | X | - | - | - |
| No. conditional assembly pseudo-ops | 13 | 13 | 12 | 12 | 0 |
| Assembler variables | X | X | X | X | - |
| Structures | X | X | X | X | - |
| Records | X | X | X | - | - |
| 186 instructions | X | X | X | - | - |
| 286 instructions | X | X | X | - | - |
| 8087/287 instructions | X | X | X | X | - |
| **Customer Support** | | | | | |
| Price | 150 | 195 | 99 | 40 | free |
| Manual | | | | | |
| printed in binder | X | X | - | - | - |
| softback manual | - | - | X | - | - |
| suitable for neophytes | - | - | X | X | X |
| Technical support | | | | | |
| written | X | X | X | X | - |
| telephone | - | X | X | - | - |
| 800 number | - | - | - | - | - |
| Debugger included | X | X | - | - | - |
| **Performance Figures** | | | | | |
| Assembly time on the standard SIEVE | 9 | 8 | 7 | 8 | 4 |
| Subjective difficulty factor<br>(1—easy, 5—eating rocks) | 3 | 2 | 1 | 3 | 4 |

X —*present*
- —*absent*
(a) —*present, but incomplete*
(b) —*on command line only*
(c) —*treats as a 0*
S —*strong*
W —*weak*
N —*none*

### Microsoft's MASM, Version 4.0

What can I say about MASM that hasn't already been said? Microsoft established the standard with MASM 1.0 in the early days of the PC. It wasn't that MASM was good. Although powerful, MASM was large, obtruse, and as slow as California's slide into the Pacific. But it was first (Intel's ASM came earlier but cost about as much as my car), and IBM put its beautiful three-letter stamp of approval on it. That assured its position as the standard against which all others must be measured.

The orginal MASM had all the features expected of a professional assembler. Directions on macros, structures, records, conditional pseudo-ops, and the like were all to be found within its manual. Nothing of importance was missing. Understanding what the manual said was something else again, though, and even when you understood it, getting it to work on your program was torturous.

Microsoft has made many improvements with Version 4.0. First, the price has dropped to a reasonable level. Additionally, Version 4.0's speed has improved so far as to actually be described as swift. My biggest problem with earlier versions was the manual. It has been completely rewritten. Still not for beginners, the instructions are clear and well indexed. Of course, MASM's command format makes no better provisions for novices than it ever did—no noname segment here, for example.

Converting the General Sieve to MASM was easy, probably because I had already had enough experience wth MASM to know what it wanted to see. I did notice a few problems, however. When I forgot to close the structure declaration, MASM generated a series of confusing messages, none of them having anything to do with structures.

Just like the PC itself, whatever you might think of it, MASM is the standard. At least now the pill is not nearly so bitter to swallow. Dropped price, increased speed, and a good bundled debugger have turned MASM into a pretty reasonable deal.

### Phoenix's PASM, Version 2.0

Phoenix bases its entire company upon PC compatibility. The Phoenix BIOS is a mark of distinction among PC clones. In like fashion PASM, Version 2.0, hangs its hat on MASM compatibility. Reacting to MASM's obscurities, Phoenix's PASM, Version 1.0, assembler set out on its own course. Although easier to use, it lacked that important mark of MASM compatibility, so Phoenix rewrote PASM into the currently MASM-compatible Version 2.

*Compatible with* does not mean *the same as*. First, PASM can assemble larger source files with more symbols. PASM contains a PDRIVE utility capable of reassembling large lists of source files rapidly. A powerful, full-screen debugger, PFixLite, is also present. (Inexplicably, PASM lacks a MAKE utility.) More important than these enhancements, though, PASM is a more powerful assembler than is MASM. Constructs that MASM accepts without complaint, quietly generating the wrong code, PASM notes with either warnings or error messages.

During the General Sieve conversion process, PASM's superiority showed itself. When confronted with the same improperly ended structure, PASM clearly and accurately marked the error. Subsequent code was not littered with irrelevant error messages as in the MASM case. More important to me, however, PASM allows a noname segment, even though it is not meant for beginners.

Known for professional packages, Phoenix's line of software products is enhanced by the addition of PASM 2.0, a powerful, fully MASM-compatible assembler.

### Speedware's Turbo EditAsm, Version 2.00B

On the surface, Turbo EditAsm (TASM) comes closest to my ideal assembler. TASM comes with a built-in editor, a menu-oriented interface, and the ability to generate either .COM or .OBJ file output. The only hole is the absence of a built in debugger and that Speedware has apparently already plugged with a RAM-resident debugger; however, because this is a separate product, so I did not consider it in this review.

TASM's user interface faithfully copies the interface developed by Borland for Turbo Pascal. When invoked, TASM presents you with a menu of commands, each of which you execute by entering the first letter. First you might (G)et a file and then (E)dit it, after which you can (A)ssemble it. Simple! By entering the option menu, you direct TASM to generate either a .COM file or a .OBJ file or assemble directly into RAM. The latter option only makes sense if you have a RAM-resident debugger loaded because TASM provides no debug features of its own. (A $50 version of TASM is available that lacks the .OBJ option.) TASM's editor is a Wordstar knockoff, like Turbo's, except that it cannot be installed to different commands. A second version of the assembler, which is invoked in the same way as is MASM, is included for those who prefer that method. The feel of TASM is good, but it's not quite as polished as Turbo Pascal.

***Listing 1***

```
       TITLE 'Assembler Review'
       PAGE 66,80
  COMMENT #
      Sieve of Erathosthenes (General Version)
          by Stephen R. Davis
  This is the famous Sieve benchmark, who's algorithm is
  both simple and well known.  This version was not written for
  speed but to make use of the following features for assembler
  evaluation purposes:
  - INCLUDE file
  - segment override with both : and seperate override
  - various address formats
  - arithmetic in place of constant
  - arithmetic expressions using parenthesis
  - ASCII literal in place of a constant
  - define variables on stack
  - define variables in code segment
  - use variables before they are defined
  - move constant into typed data field
  - forward unconditional jump
  - use of "?" pseudo_op
  - MACRO definition
  - conditional assembly
  - STRUCTURE definition

  This program, written in "general assembler", was converted to each
  assembler to generate a .COM file.  None of the assemblers tested
  supportedall of the features used below, but the more features
  supported and the less "finiky" the assembler the easier it was to
  convert this source program.  This was used to arrive at a subjective
  "difficulty of use" factor between 1 (like eating rocks) to 5 (piece
  of cake), inclusive.
  #

INCLUDE 'MSDOS.INC'                         ;DEFINE THE SYSTEM CALLS

NTIMES  EQU     8190                        ;STANDARD SIZE SEARCH
NLOOPS  EQU     10                          ;DEFINE THE NUMBER OF ITERATIONS

;DEFINE OUR LOCAL VARIABLES AS A STACK STRUCTURE

STACK   STRUC
        LOOP_CNT        DB ?                ;NUMBER OF ITERATIONS
        COUNT           DW 0                ;INCREMENT COUNTER
        PRIME           DW 0                ;NUMBER OF PRIMES
        ARRAY           DB NTIMES DUP (?);PRIME ARRAY
        ENDS

        ORG     100H

START:
        SUB     SP,SIZE STACK               ;ALLOCAE STACK SPACE
        MOV     BP,SP

;PERFORM THE SIEVE 'NLOOPS' TIMES

        MOV     [BP].LOOP_CNT,NLOOPS        ;INIT LOOP COUNTER
OUTER_LOOP:

;FIRST CLEAR THE BOOLEAN ARRAY

        XOR     AX,AX                       ;FIRST ZERO OUT THE FLAG ARRAY
        LEA     DI,[BP].ARRAY
        MOV     CX,(NTIMES/2)               ;CONVERT # BYTES -> # WORDS
        CLD
```

One of the advantages of TASM's menu user interface is the great speed of assembly. In order to be fair on the timing tests, I invoked all assemblers from the command line, including TASM. When TASM was invoked from the menu, however, assembly times were on the order of 2 seconds. When you are reiteratively assembling large files to remove assembly errors, this is a great advantage. TASM tries to keep things simple for beginners, such as allowing the noname segment and automatically ORGing to 100 when you select the .COM file option.

At the same time, TASM lacks nothing for more advanced users. A quick glance at the features table shows TASM matching MASM, feature for feature. TASM includes macros; structures; records; and support for the 186, 286, and 8087. Expert users can select TASM's .COM file output for those simple jobs and save the .OBJ file output for the larger projects.

```
        MOV     DX,SS
        MOV     ES,DX
        REP     STOSW

;NOW CHECK FOR PRIMES IN THE FIRST 'NTIMES' INTEGERS

        XOR     SI,SI                       ;START AT ARRAY SUBSCRIPT 0
        MOV     CX,NTIMES                   ;GET THE LOOP COUNT
        MOV     [BP+COUNT],0                ;ZERO THE PRIME NUMBER COUNT

INNER_LOOP:
        CMP     SS:ARRAY[BP+SI],0           ;SI = ARRAY SUBSCRIPT
        JZ      CONT                        ;IS THIS A PRIME?
        JMP     SKIP_AROUND
CONT:                                       ;YES -- COUNT IT
        MOV     AX,SI
        ADD     AX,AX
        ADD     AX,3
        MOV     [BP].PRIME,AX               ;SAVE OFF PRIME NUMBER

;MARK THE MULTIPLES OF THIS PRIME AS NONPRIME

        MOV     DI,SI
MARK_OUT:
       MOV     ARRAY[BP][DI],0FFH          ;MARK THIS MULTIPLE OUT
        ADD     DI,AX
        CMP     DI,NTIMES                   ;IS THAT IT?
        JNA     MARK_OUT

        INC     [BP].COUNT                  ;INCREMENT THE PRIME NUMBER COUNT

SKIP_AROUND:
        INC     SI
        LOOP    INNER_LOOP

;OUT A MESSAGE CONTAINING THE LOOP COUNT

        MOV     AL,[BP].LOOP_CNT            ;GET THE NUMBER OF LOOP
        ADD     AL,'0'                      ;CONVERT IT TO ASCII
        MOV     CS:LOOP_NUM,AL

        SYSTEM  OUTMSG,PREAMBLE

;LOOP AROUND AND DO IT AGAIN

IF NLOOPS NE 0                      ;ONLY ASSEMBLE FOLLOWING IF NECESSARY
        DEC     [BP].LOOP_CNT               ;DO IT N TIMES
        JNZ     OUTER_LOOP
ENDIF
                                            ;FINISHED!
        SYSTEM  OUTMSG,FINISH
        SYSTEM  TERMINATE                   ;OK -- STOP NOW

;DEFINE OUR STRINGS DWN HERE

PREAMBLE DB "NOW WE START LOOP NUMBER "
LOOP_NUM DB 0
        DB 0AH,0DH,"$"

FINISH  DB 0AH,0DH,"THAT'S ALL FOLKS!",0AH,0DH,"$"
        END START
LOOP NUMBER "
LOOP_NUM DB 0
        DB 0AH,0DH,"$"


FINISH  DB 0
```

During conversion of the General Sieve, however, I lost some of my excitement. First, the multiline COMMENT did not work in quite the same way as Microsoft's did. Second, when asked the size of a structure, TASM returned a 0, probably because no space was allocated to the declared structure. Both MASM and PASM returned the size of the structure properly. Further tests of TASM's Microsoft compatibility revealed a few more chinks in the armor. Finally, no update command was available in the command menu, forcing you to (W)rite the file back out by name. In general, the conversion process was quick with TASM's rapid assemblies and built-in editor.

In earlier days, when Microsoft's assembler was very expensive and slow, Speedware's TASM was an excellent choice, tarnished only by a few bugs in the editor and a few MASM incompatibilities. TASM still has a few minor problems that Speedware should address but not enough to spoil my excitement for the product.

**CHASM—Cheap Assembler, Version 4.08**

CHASM is actually two assemblers. A useable version with lesser capability is circulated in the public domain. You can obtain the full-capability version by sending $40 to the author. I reviewed the full-capability version. This version of CHASM has little to do with an earlier one written in BASIC, which was glacially slow. As you and I were busy with other things, CHASM was steadily undergoing improvements, until now CHASM is a worthy beginners' assembler.

CHASM fits somewhere in between WASM and the more professional assemblers, both in capability and price. CHASM lacks .OBJ file output capability, records, and Microsoft compatibility. Additionally, CHASM accepts only one of the common addressing modes. On the other hand, it includes most of the professional features such as macros, structures, assembler variables, and conditional assembly. It has two special output formats—one for direct insertion into a Turbo Pascal INLINE statement and the other for direct load into BASIC via the BSAVE statement.

Not slavishly following Microsoft's lead has both advantages and disadvantages. On the one hand, CHASM does several things better than does MASM—for example, many of CHASM's conditional assembly pseudo-ops are more powerful than MASM's. On the other hand, you will never be able to adapt listings found in magazines to CHASM without some rewriting. Still, CHASM is now acceptably fast and certainly powerful enough. The inclusion of an extensive 8086 assembler primer with the already extensive on-disk documentation serves to fill out a competent package.

### WASM

WASM is a public-domain assembler available to anyone for the cost of a call to any one of hundreds of bulletin-board systems (BBS) around the country or from any of several public-domain libraries. The version I reviewed had apparently come from The Public (Software) Library, P.O. Box 61565, Houston, TX 77208. The question is, how much can you expect from an assembler that is free?

WASM has few of the more advanced assembler features. It has no macros, structures, and records; no multiline comments; no support for 186, 286, or 8087 instructions; and no built-in editor. WASM supports only one addressing mode and requires segment overrides on a separate line. The absence of object file output is probably the biggest missing feature.

WASM does allow control of the listing file, such as page width and length, and allows INCLUDE files. WASM supports the unnamed segment, important for beginners. WASM's manual, present on the same disk as the assembler, is complete but just barely so.

In the conversion of the General Sieve, several things stood out. Anything in column 1 was assumed to be a label, even if it was a reserved word. Even though WASM supports expressions in place of a constant, the expression NTIMES/2 generated an "overflow error" and had to be replaced. One big problem with WASM was that it did not continue to look on a line beyond the instruction found. When I coded REP STOSW, WASM assembled the REP but did not notice the STOSW and generated no error message (WASM wants instructions on separate lines). I have used this type of assembler before, and it can lead to errors that are difficult to track down. One positive feature was WASM's speed. On the small SIEVE program, WASM was clearly faster than the rest of the pack when invoked from the command line.

Conversion from the General Sieve was difficult because of the extensive amount of recoding around WASM's missing features and not because it was all that difficult to use. All in all, WASM would be a good assembler for anyone interested in dabbling in assembly language, but it's not up to the requirements of serious coding.

### Conclusion

In the end it boils down to a question of price vs. performance. Each user must factor his or her requirements into the equation. I hope this review will help you with that decision. §

*Stephen Randy Davis is a senior systems programmer for a defense contractor in Greenville, Texas, where he programs various microprocessors. He is also working on his Masters in physics.*

### Product Information

**Cheap Assembler, Version 4.08**
$40
David Whitman
P.O. Box 1157
North Wales, PA 19454

**MASM, Version 4.0**
$150
**Microsoft Corp.**
16011 N.E. 36th Way
P.O. Box 97017
Redmond, WA 98073-9717

**PASM, Version 2.0**
$195
**Phoenix Computer Products Corp.**
320 Norwood Park South
Norwood, MA 02062
(800) 344-7200.

**Turbo EditAsm, Version 2.00B**
$99
**Speedware**
9719 Lincoln Village Dr.
Ste. 303
Sacramento, CA 95827

**WASM, Version 1.0**
Free
Eric Tauck
Wolfware
1304 Deerpass Rd.
Marengo, IL 60152

# LOMAS DATA PRODUCTS, INC.

## Quality S100 Bus Boards For S100 Microcomputers

### THUNDER/PLUS

**THUNDER/PLUS** is a complete 16 bit S100 computer on one card. Standard features:

- 512K bytes of no waitstate dynamic RAM expandable to 1 Megabyte on board.
- Two RS232 serial ports capable of communicating at up to 19200 baud.
- Centronics compatible parallel interface.
- Floppy disk controller capable of controlling a mix of 8", 5¼", and 3½" floppy drives.
- 10 MHz 80186 microprocessor.
- Concurrent DOS version 4.1 included.

PRICE: $1295.00

### LDP-SCSI ADAPTER

- Four RS232 ports capable of communicating at baud rates up to 38400.
- Battery protected clock/calendar.
- Drivers available for Concurrent DOS and MSDOS.

PRICE: $375.00

### CONTROL-IT-ALL

One of the most complete disk controllers, offers the following advanced features:

- Controls up to four floppy disk drives in any mix of 5¼" and 8" drives.
- Controls up to four 5¼" Winchester disk drives with up to 16 heads and 2048 cylinders.
- The 8 Kbyte onboard buffer relieves the system of realtime constraints of the disk drives.
- Error detection and correction of up to 11 bit burst errors on Winchester drives.
- Software drivers available for MSDOS, CP/M-86 and Concurrent DOS.

| Pricing | |
|---|---|
| CONTROL-IT-ALL | $649.00 |
| CONTROL-IT-ALL *(floppy only)* | $395.00 |
| CONTROL-IT-ALL *(hard disk only)* | $575.00 |

### MEGA S RAM

Static RAM up to 1 full megabyte on a standard S100 bus card without a daughter board. Optionally battery protected.

| Pricing | |
|---|---|
| 16K MEGA S RAM | $ 349.00 |
| 256K MEGA S RAM | $ 495.00 |
| 512K MEGA S RAM | $ 995.00 |
| 1 MEGABYTE | $1749.00 |

## QUALITY MULTI-USER SYSTEMS

### LO-MAX 2/4 USER SYSTEM:

**LO-MAX** is **LO**w on price and **MAX**imum on performance. **LO-MAX** will support two or four users with the following advanced features:

- 10 MHz 80186 Microprocessor.
- 512K bytes of no waitstate dynamic RAM.
- Supports two users *(four users optional).*
- CC-DOS version 4.1 runs both MS-DOS and CP/M-86 programs.
- 7 slot chassis *(only 2 slots used)* for future expansion.
- 20 Mbyte formatted capacity harddisk drive.
- SCSI interface for expansion.
- 360K floppy drive standard, 1.2 Mbyte PC/AT compatible drive optional.

Price $3795.00

### OPTIONS:

- 1 Mbyte of no waitstate dynamic RAM.
- 8087 option
- 1.2 Mbyte PC/AT compatible floppy drive.
- 50, 80 or 160 Mbyte harddisk drives.
- 4 user support.

All these features make this system too great a value to pass up. If you need a multiuser system, LO-MAX is the sensible solution.

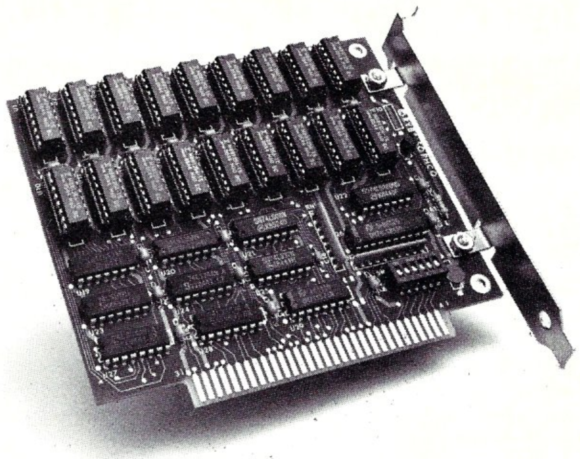
LOMAS DATA PRODUCTS offers other multiuser configurations supporting up to 18 users including our dual CPU system for a high reliability system. If you are an OEM of VAR with special needs, contact LOMAS DATA PRODUCTS for a quote on a specially configured system. All our systems are guaranteed for a full year, parts and labor. Additional maintenance options available.

**Prices are subject to change*

*LOMAS DATA PRODUCTS, Inc. 182 Cedar Hill Street, Marlboro, MA 01752 (617) 460-0333*

# HiCard

by Steve Leibson



## A Low-Cost Way to Expand the PC's Memory Beyond 640K

I'm always trying to eke more performance and capacity from my vintage IBM PC, and one of the problems I've fought is a shortage of memory capacity. The 640K ceiling looked mighty high five years ago when I bought my PC but barely suffices in today's software environments of device drivers, memory-resident programs, and RAM-hogging application software. The HiCard from Rybs Electronics is an ingenious little half-card and software product that helps you push your computer beyond MS-DOS/PC-DOS' 640K limits, giving you a maximum RAM capacity of 896K. Though similar in concept to EMS and EEMS block-switched memory boards for the IBM PC bus, the HiCard costs less and offers some improved features when compared to those other boards.

The hardware portion of the HiCard product is a memory card with a maximum RAM capacity of 512K. If you have not expanded your computer's memory to the 640K, unaugmented DOS limit, the HiCard will back fill your RAM. You can set the HiCard to start the back fill at any 64K boundary, aided by a program called HI, which looks at your address space and draws a picture on your screen (graphic or monochrome) showing you exactly how to set the switches. I wish all memory-board products came with this type of software aid.

Actually, the HiCard takes your contiguous RAM to 704K, just below the video display adapters' RAM address space. HIPAGE, an included program, expands DOS' memory limit to 704K. That is not the maximum usable memory you have with the HiCard, however. The HiCard can also fill unused segments above the display adapters in segments C, D, and E of the 8088's address space. If you have no expansion options occupying those address space segments, HiCard gives you an additional 192K. If you have a hard disk or an EGA board, you will already have ROMs occupying segment C, limiting the HiCard to segments D and E for an additional 128K of expansion memory.

Normally, DOS ignores RAM in those high segments. Rybs Electronics includes a range of programs with the HiCard allowing you to manage this block of high memory very effectively, however. HIDISK and HISPOOL provide a RAM disk and print spooler, respectively, that use this additional memory. You can allocate portions of the extra RAM to either or both programs. Two other programs, MOREMEM and LOADHIGH, give you the ability to load terminate-and-stay-resident (TSR) programs in this upper-segment RAM, freeing contiguous memory in the lower segments for your application programs. I tried LOADHIGH with Borland's SideKick and Turbo Lightning and experienced no problems.

For an even tougher test, I combined the HiCard memory board and software with AST's Rampage! EEMS memory card and Quarterdeck Office Systems' Desqview software. The Rampage!/Desqview combination allows you to run multiple programs under DOS in a 2-megabyte (per Rampage! card) memory space. I expected interesting conflicts between the Quarterdeck and Rybs software but all worked surprisingly well. In this configuration, the Rampage! board occupied all the space up to 704K and the HiCard provided an extra 128K in segments D and E. The LOADHIGH software put Turbo Lightning up above DOS, and I ran WordStar 2000 as a task under Desqview and activated Turbo Lightning successfully.

The one problem I did experience with the HiCard was because of a Dynatec Supercharger installed in my PC. The Supercharger runs its on-board 8088 at 9.54 MHz, and that outstripped the HiCard's memory access time. Switching the Supercharger to 4.77 MHz solved this problem. Rybs claims that it has experienced no problems with turbo motherboards running at 8 MHz so I guess the Supercharger just slightly exceeds the HiCard's access time. Rybs also reports operating the HiCard and its associated software successfully in an AT.

In my opinion, the HiCard represents one of those rare products that really gives you tremendously useful additional capacity at a reasonable price. §

*Steven Leibson has been working with microcomputers since 1975. He has been a development engineer, project manager and is now a regional editor for EDN magazine. He has authored more that 50 articles. He is also the author of the book titled* The Handbook of Microcomputer Interfacing.

### Product Information

**HiCard**
$149.95, with 0K
$199.95, with 256K
$249.95, with 512K

**Rybs Electronics Inc.**
5721 Arapahoe Ave., Ste. A
Boulder, CO 80303
(303) 449-9256

# Until now, LANs have had no parallel.

PC local area networks are a great idea, but they can be slow and complicated to install and use. That's because they use serial cables to communicate between processors.

QuickLink™ is different. It communicates on the high-speed IBM PC parallel bus. It's like driving your data on an eight-lane highway instead of a single-lane road. The performance advantage is significantly better than a typical LAN, multi-user system or "slave board" system.

Plug a QuickLink card into your PC, add a terminal, and you've created a multi-user MS-DOS system. Each QuickLink card is a complete PC on a card. For every card and terminal you connect, you've added another MS-DOS workstation. Without the expense or hassle of adding another PC with complicated LAN gear. And since each user's computer is placed inside your central PC, you get much better control and security.

QuickLink uses the industry-standard Novell Netware operating system, so you have access to all Novell multi-user applications plus the entire world of single-user PC programs. You get electronic mail, printer sharing, mainframe-like security, file and record locking, and much more.

So lower your LANing gear by calling TNL or your local dealer today. We'll show you how to turn your PC into a parallel LAN that has no parallel.

TNL

## QuickLink™

### From The Network Link™

3303 Harbor Blvd., Building H-10, Costa Mesa, CA 92626. (714) 549-9380

QuickLink developed by InterContinental Microsystems Corp. and manufactured for TNL.

# Program Interfacing to Microsoft Windows

by William Wong

## Part IV—Window Messages

*This series of articles has a twofold purpose. First, its intent is to give readers considering purchasing the Microsoft Windows Toolkit and writing a windows application an insight into what is involved. Its second purpose is to help readers who have purchased the Toolkit or attended a Microsoft Windows seminar to write a typical application. Some sample programs will be provided after sufficient groundwork has been laid.*

Windows uses a large number of messages to support a window. The response to one of these messages is a call to an associated Windows window function, as described in the previous part of this article.

The message identifier is a 16-bit constant. I use named identifiers, as does the Windows toolkit manual, although you can find actual values in the toolkit .H files. Messages include a word and a long-word parameter whose meaning is message dependent.

The messages discussed in this article are reserved by Windows and have a value between 0 and *WM_USER*. Applications can define their own message identifiers internally with values between *WM_USER* and 7FFF hex. Identifiers between 8000 and BFFF hex are reserved for Windows. Upon request Windows can allocate unique identifiers with values between C000 hex and FFFF hex, which are useful when exchanges occur between different programs. Also, there are a few messages that can be sent to an application instead of to a window.

Messages are processed by a window function that is associated with a window. The window function can ignore a message, process the message, or hand it over to *DefWindowProc*, the Windows default handler. *DefWindowProc* can handle window movement, sizing, and conversion between an icon and a tiled window.

The general window management messages (Figure 1) are sent in response to user interaction with a window. For example, *WM_CLOSE* is issued when the user selects the Close option on the system menu of a window or if the application itself wants to close a particular window. Only one window can be active at a time.

*WM_ACTIVATE* is sent to a window when it changes activity state. *BActive* is 0 when a window becomes inactive, 1 when it becomes active via a *SetActiveWindow* call, and 2 when it becomes active via user interaction. The high-order word of *lSpecial* is nonzero if the window is iconic, and the other word is the handle of the window that will become active/inactive.

Windows tells a window about changes of state so a window does not have to keep track of its current state. This approach makes it easier for a window class to handle many instances in different states.

*WM_ACTIVATEAPP* is like *WM_ACTIVATE*. The former is sent when the current active window belongs to another application. *BActive* is false if the application is becoming inactive, and *hTask* is the handle of the currently active task.

*WM_CREATE* is sent to an application before a *CreateWindow* call returns and before a window has its visibility set. No messages can be sent to a window until the handle is obtained. This type of initialization is normally done in the code that has the *CreateWindow* call.

The *DestroyWindow* function issues a *WM_DESTROY* message to the selected window. Child windows receive a *WM_DESTROY* message after their parent. A window receiving this message will not be visible.

A Windows session ends when the last DOS file window is closed, and *WM_QUERYENDSESSION* is sent to all remaining applications at that time. The session is terminated if all applications respond with a nonzero result. *WM_ENDSESSION* is then sent to all responding applications. *BEnd* is false if an application responded with a false result, indicating that the session should not be terminated. An application can check with the user upon receiving the *WM_QUERYENDSESSION* message.

*WM_ENABLE* is sent to a window when its enabled status changes. *BEnable* is false if the window has been disabled.

A window receives the *WM_ERASEBKGND* message when all or part of a window's region must be erased, as when a pop-up window is removed or a section of the window's client area is invalidated. The parameter is a device context describing the area that must be erased. The application does not have to respond to this message. In some instances, an application may have to align the brush being used for the erase operation with the actual origin of the window because the region being erased may not include this origin. This is accomplished using the *UnrealizeObject* call with the brush. This call will be discussed in a future part of this article dealing with graphic operations.

The *WM_ERASEBKGND* message is used in conjunction with

# COMPETITIVE EDGE

1421 E. GOLDSMITH, PLYMOUTH, MI 48170 (313) 451-0665

## SPEED SHOP

## *Summer Clearance*

SOME DEMO EQUIPMENT, SOME NEW OVERSTOCKS. ALL MUST GO. PRICES GOOD WHILE QUANTITIES ARE AVAILABLE. DEMO EQUIPMENT SOLD AS IS, NO RETURNS.

### PC EQUIPMENT

| | | | |
|---|---|---|---|
| MONOCHROME TEXT CARD W/PTR | $49. | MONOGRAPHICS W/PTR | $69. |
| STD COLOR 320 X 200 W/PTR | $69. | AT I/O CARD SERIAL/PARALLEL | 69. |
| PC XT FLOPPY CONTROLLER | 29. | PC XT DISK I/O CARDS | 69. |
| STB EGA CARD 640 X 350 | 149. | MULTITECH EGA CARD | 149. |
| STD AT STYLE KEYBOARDS XT/AT | 39. | MITSUBISHI COLOR MONITOR STAND | 19. |
| MICROPORT UNIX SYS/V FULL PKG | 395. | SCO XENIX RUNTIME | 395. |
| SCO XENIX DEV PACKAGE | 395. | SCO XENIX TEXT PROCESSING | 125. |
| SCO/FOXBASE M/USER DBASE | 395. | SCO PRO. M/USER 123 COMPAT. | 495. |
| DOS 2.1 VARIOUS MFG. | 35. | GWBASIC 2.1 | 25. |
| SUMMA OPTICAL MOUSE | 75. | CXI PCOX 3278 EMULATION BOARD | 350. |

**MITSUBISHI 19" 6920 LP/AG COLOR CAD MONITOR . . . . . . . . . . . . . . . . . . . . . . . . . . . 1895.**

### TERMINALS

| | | | |
|---|---|---|---|
| QUME 101 GREEN | 295. | QUME 108 (VT100 COMPAT) | 350. |
| QUME 101 AMBER | 295. | AMPEX 230 AMBER | 325. |

### S-100

| | | | | | |
|---|---|---|---|---|---|
| CompuPro® CDOS 4.1 E | 5" | 350. | 8MHz 8085/8088 CPU BOARD | | 99. |
| CompuPro CPM-86 | 8" | 75. | SPU-Z 256K Z80 SLAVE | | 195. |
| CompuPro CPM-68K | 8" | 125. | ENCLOSURE 2 21 SLOT CABINET | | 695. |
| DBASE II CPM 2.2 | 8" | 100. | T-MAKER II CPM 2.2 8" | | 50. |
| GENERIC MPM-86 | 8" | 75. | MACROTECH 512K MSR DRAM | | 195. |
| MICROSOFT MASM CPM | 8" | 50. | MICROSOFT BASIC COMPILER | 8" | 75. |
| LOMAS 6MHZ LIGHTNING 286 | | 295. | THUNDER 186 256K (1) ONLY | | 295. |
| LOMAS 8MHZ LIGHTNING 8086 | | 250. | THUNDER 186 256K (1) ONLY | | 595. |

### INTEGRAND CABINETS

INTEGRAND 4 SLOT CABINET, POWER SUPPLY, SPACE FOR 4 1/2 HT DRIVES . . . . . . . . . . . . . . 195.

### SUMMER SPECIAL ON SYSTEMS

AT COMPATIBLE SYSTEMS BELOW INCLUDE 1.2 MB FLOPPY, FLOPPY/HD CONTROLLER

| | |
|---|---|
| VELOCITY 286-12 12MHz OPERATION, 1024K, FULL SIZE MOTHERBOARD | 1199. |
| VELOCITY 286-12A 12MHz OPERATION, 640K, REDUCED SIZE MOTHERBOARD | 1149. |
| VELOCITY 386-20 20MHz OPERATION, 1024K, FULL SIZE MOTHERBOARD | 2695. |
| VELOCITY 386-16 16MHz OPERATION, 1024K, FULL SIZE MOTHERBOARD | 2195. |
| VELOCITY 386-16I 16MHz OPERATION, 512K, INTEL MOTHERBOARD | 2695. |
| BASE AT 8MHz 80286 SYSTEM, 512K, KEYBOARD, 1.2MB FLOPPY, FDC/HDC | 995. |
| BASE AT 10MHz 80286 SYSTEM, 512K, KEYBOARD, 1.2MB FLOPPY, FDC/HDC | 1125. |
| 12MHz BASIC SCO XENIX SYSTEM, 3 USER, 1024K, 30MB HARDDISK, MONO | 2695. |

*LIMITED ONE YEAR PARTS & LABOR WARRANTY ON ABOVE SYSTEMS*

### BLOW OUT SPECIAL ON TURBO XT COMPATIBLES

| | |
|---|---|
| MONOGRAPHICS SYSTEM, 640K, 8MHz,360K FLOPPY, 20MB HD,MONITOR | 995. |
| EGA GRAPHICS SYSTEM, 640K, 8MHz,360K FLOPPY, 20MB HD, MONITOR | 1495. |

*LIMITED 120 DAY PARTS & LABOR WARRANTY ON X'S*

### HARD DISK DRIVES

| | | | |
|---|---|---|---|
| SEAGATE ST225 65ms | $299. | MINISCRIBE 6053 28ms 42 MB | 675. |
| SEAGATE ST238 65ms | $299. | MINISCRIBE 6085 28ms 70 MB | 995. |
| SEAGATE ST251 40ms 1/2 HT 40MB | 595. | TOSHIBA 70MB 25ms 70 MB | 1095. |
| SEAGATE ST4038 39 ms 30 MB | 560. | SEAGATE 4096 28ms 80 MB | 1095. |
| ST225 KIT HD/CONTROLLER XT | 349. | ST238 KIT HD/CONTROLLER XT | 419. |

### MONITORS

| | | | |
|---|---|---|---|
| NEC MULTISYNC EGA 800 X 560 | 569. | SAMSUNG EGA MONITOR | 425. |
| SAMSUNG FLAT SCREEN AMBER | 119. | MITSUBISHI 1409C COLOR | 369. |

### I/O AND ADD ON CARDS

| | | | |
|---|---|---|---|
| MAGIC I/O CARD | 69. | MONOGRAPHICS 720 X 348 | 69. |
| EVERCOM II 1200 MODEM INT. | 117. | EVERCOM 2400 MODEM INT. | 229. |
| BOCA RAM AT EMS, & EXTENDED RAM | 169. | BOCA EGA, CGA, MDA, HERC. | 169. |

Xenix is a trademark of Micosoft, CompuPro and listed boards are either trademarks or registered trademarks of Viasyn Corp. UNIX is a trademark of AT&T.

repaint messages so that a window can handle the redrawing of its contents in a suitable fashion. In many instances, the background tends to be consistent, and it is easier to redraw it first before recreating the rest of the drawing. This approach works well when portions of a window are uncovered because the areas first have the background erased, which is normally quicker than redrawing the rest of the window.

*WM_CTLCOLOR* is sent to a parent window when a child is used to display a system-defined message, control, or dialog box. The *hDC* parameter is the device context handle used for this purpose. The drawing color can be changed by the parent window. The low part of *lTypeHandle* is the child window handle, and the high part is a window type. Values of *WM_CTLCOLOR* parameters are listed in Figure 2.

Dialog boxes can also be controlled by responding to the *WM_GETDLGCODE* message. *DefWindowProc* responds with 0; other responses are listed in Figure 3.

The text for a particular window is controlled using the *WM_GETTEXT*, *WM_GETTEXTLENGTH*, and *WM_SETTEXT* messages. A window's text depends on the window type. Button controls have text to the right of the button, and edit controls display their text within the window. Other windows have the text in the caption area if it is displayed. *WM_GETTEXT* receives the size and location of a buffer in *nSize* and *lpString*. The corresponding text, if any, should be placed into the buffer. The *WM_GETTEXTLENGTH* message should reply with the current length, in bytes, of the window's text. New strings are set using *WM_SETTEXT*, where *lpString* points to the new text.

*WM_KILLFOCUS* and *WM_SETFOCUS* are sent before a window loses or gains the input focus. The input focus is the window to which normal keyboard input is directed. Caret and cursor visibility are normally handled through these messages. Windows with captions normally change the background color, whereas other windows ignore these messages.

*WM_MOVE* is sent when a window's position changes. *LPoint* contains the x and y position in the low and high parts, respectively. Units are screen coordinates for tiled and pop-up windows and client coordinates for child windows.

## WRITING WINDOWS APPLICATIONS

*Windows applications cannot be written in MASM or C alone. This is because the linker must understand Windows objects (for example, icons, dialog boxes) and linking to Windows is through a special call linkage, not an* INT *instruction.*

*To write a windows application program, you need the Windows Toolkit ($500 list, but discounted to as little as $309) and the latest version of either Microsoft C, MASM, or Pascal. Microsoft conducts seminars for windows application programmers at which the Windows Toolkit package is distributed. Programmers who buy the Toolkit without attending the seminars are handicapped because the manuals contain little theory. This series of articles should assist programmers who buy the Toolkit without attending the seminars to write Windows application programs. It should also help seminar attendees.*

*The Whitewater Group (Technology Innovation Center, 906 University Pl., Evanston, IL 60201; (312) 491-2370) is expected shortly to introduce Actor, a lower-cost alternative to the Microsoft Windows Toolkit. Its documentation is poorer than that of the Toolkit, however, because it deals primarily with the Actor programming language.*

**Figure 1. Window management messages**

| Message | Word Param | Long Param |
|---|---|---|
| WM_ACTIVATE | bActive | lSpecial |
| WM_ACTIVATEAPP | bActive | htask |
| WM_CREATE | | lpCreateStruct |
| WM_CLOSE | | |
| WM_DESTROY | | |
| WM_ENDSESSION | bEnd | |
| WM_ENABLE | bEnable | |
| WM_ERASEBKGND | hDC | |
| WM_CTLCOLOR | hDC | lTypeHandle |
| WM_GETDLGCODE | | |
| WM_GETTEXT | nSize | lpString |
| WM_GETTEXTLENGTH | | |
| WM_KILLFOCUS | hWindow | |
| WM_MOVE | | lPoint |
| WM_QUERYENDSESSION | | |
| WM_QUERYOPEN | | |
| WM_QUIT | bPostQuit | |
| WM_PAINT | | lpPaintStruct |
| WM_SETFOCUS | hWindow | |
| WM_SETTEXT | | lpString |
| WM_SETREDRAW | bRedraw | |
| WM_SETVISIBLE | bVisible | |
| WM_SHOWWINDOW | bShow | lShowReason |
| WM_SIZE | nType | lWidthHeight |

**Figure 2. *WM_CTLCOLOR* parameters**

| Value | Type |
|---|---|
| CTL_BTN | Button |
| CTL_DLG | Dialog box |
| CTL_EDIT | Edit control |
| CTL_LISTBOX | List box |
| CTL_MSGBOX | Message box |
| CTL_SCROLLBAR | Scroll bar |
| CTL_STATIC | Static control |

**Figure 3. *WM_GETDLGCODE* responses**

| Value | Application Controls |
|---|---|
| DLG_HASSETSEL | EM_SETSEL messages |
| DLG_WANTARROWS | Arrow keys |
| DLG_WANTALLKEYS | All keys |
| DLG_WANTTAB | Tab key |

**Figure 4. *WM_SETVISIBLE lShowReason* values**

| Value | Reason |
|---|---|
| 0 | Change due to *ShowWindow* call |
| WM_OTHERUNZOOM | Another window is being unzoomed |
| WM_OTHERZOOM | Another window is being zoomed |
| WM_PARENTCLOSING | The parent window has closed |
| WM_PARENTOPENING | The parent window has opened |

**Figure 5. *WM_SIZE* parameters**

| Value | Description |
|---|---|
| SIZEFULLSCREEN | Window is now full-screen |
| SIZEICONIC | Window is now iconic |
| SIZENORMAL | Any other reason |
| SIZEZOOMSHOW | Sent to tiled windows when another is unzoomed |
| SIZEZOOMHIDE | Set to tiled windows when another is zoomed |

**Figure 6. Initialization messages**

| Message | Word Param | Long Param |
|---|---|---|
| WM_INITMENUPOPUP | hMenu | lPopupIndex |
| WM_INITMENU | hMenu | |
| WM_INITDIALOG | hItem | |

**Figure 7. Input messages**

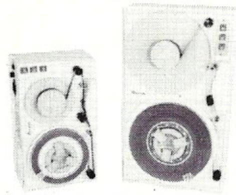
| Message | Word Param | Long Param |
|---|---|---|
| WM_MOUSEMOVE | nButton | lPoint |
| WM_LBUTTONDOWN | nButton | lPoint |
| WM_LBUTTONUP | nButton | lPoint |
| WM_LBUTTONDBLCLK | nButton | lPoint |
| WM_MBUTTONDOWN | nButton | lPoint |
| WM_MBUTTONUP | nButton | lPoint |
| WM_MBUTTONDBLCLK | nButton | lPoint |
| WM_RBUTTONDOWN | nButton | lPoint |
| WM_RBUTTONUP | nButton | lPoint |
| WM_RBUTTONDBLCLK | nButton | lPoint |
| WM_KEYDOWN | nKey | lStatus |
| WM_KEYUP | nKey | lStatus |
| WM_CHAR | nKey | lStatus |
| WM_DEADCHAR | nKey | lStatus |
| WM_SYSKEYDOWN | nKey | lStatus |
| WM_SYSKEYUP | nKey | lStatus |
| WM_SYSCHAR | nKey | lStatus |
| WM_SYSDEADCHAR | nKey | lStatus |

**Figure 8. Button status bits**

| Value | Button/Key Down |
|---|---|
| MK_SHIFT | Shift key |
| MK_CONTROL | Control key |
| MK_LBUTTON | Left mouse button |
| MK_MBUTTON | Middle mouse button |
| MK_RBUTTON | Right mouse button |

**Figure 9. System input messages**

| Message | Word Param | Long Param |
|---|---|---|
| WM_HSCROLL | nControl | lPosition |
| WM_VSCROLL | nControl | lPosition |
| WM_TIMER | nIDEvent | lpfnTimer |
| WM_COMMAND | nID | lParam |
| WM_SYSCOMMAND | nCommand | |



Iconic windows receive *WM_QUERYOPEN* when a window should be converted into a tiled window. A window that wants to remain iconic should return a false result; otherwise, subsequent messages set up the tiled window.

*WM_QUIT* is sent to an application by the *PostQuitMessage* function. *BPostQuit* is the *PostQuitMessage* parameter. An application should terminate when it receives this message and not pass it onto any of its windows.

A portion or the entire contents of a window is redrawn when the window receives the *WM_PAINT* message. The *lpPaintStruct* indicates the area to be redrawn and will be covered in the next part of this article, on GDI functions. The window should have received a *WM_ERASEBKGND* message for the same area. *WM_ERASEBKGND* can be ignored if *WM_PAINT* draws everything.

*WM_SETREDRAW* is sent to windows such as list boxes, which update themselves when they receive new information. *BRedraw* is true if the window should redraw itself.

*WM_SETVISIBLE* marks a change in a window's visibility. *BVisible* is false if a window will be invisible. A window's visibility could affect how an application operates.

*WM_SHOWWINDOW* handles a forced changed change of visibility, and *lShowReason* indicates why the change occurred. The meanings of the values of *WM_SHOWWINDOW lShowReason* are listed in Figure 4. Parent windows usually control visibility of child windows. A zoomed window causes all other windows to be hidden. The windows should show themselves when the zoomed window is unzoomed. Likewise, a child window should not be shown if its parent is hidden; however, a child window can be hidden if its parent is shown.

*WM_SIZE* is sent to a window when its size changes. The *nType* parameter indicates why the size of the window has changed. Values of *nType* are listed in Figure 5. *LWidthHeight* has the new width and height in the low and high part if the window is displayed.

### Initialization Messages

The initialization messages listed in Figure 6 are generated when a menu is selected or a dialog box is created. Window functions that cause these messages to be sent will be covered later in this article.

*WM_INITMENUPOPUP* is called after *WM_INITMENU* has been sent because

**Figure 10. Scroll control status**

| Value | Scroll | Value | Scroll |
|---|---|---|---|
| SB_BOTTOM | To bottom | SB_PAGEUP | Up one page |
| SB_ENDSCROLL | End of scroll | SB_THUMBPOSITION | Absolute position |
| SB_LINEDOWN | Down one line | SB_THUMBTRACK | Tracking thumb |
| SB_LINEUP | Up one line | SB_TOP | To top |
| SB_PAGEDOWN | Down one page | | |

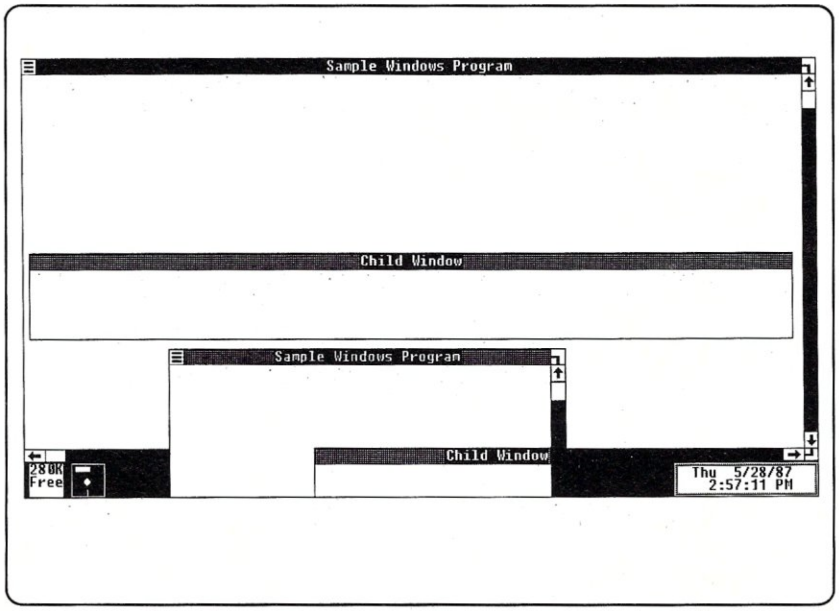
**Figure 11. *WM_SYSCOMMAND* parameters**

| Value | Request | Value | Request |
|---|---|---|---|
| SC_CLOSE | Close window | SC_NEXTWINDOW | Next window |
| SC_HSCROLL | Horizontal scroll | SC_PREVWINDOW | Previous window |
| SC_ICON | Make iconic | SC_SIZE | Size window |
| SC_KEYMENU | Key menu selection | SC_VSCROLL | Vertical scroll |
| SC_MOUSEMENU | Mouse menu selection | SC_ZOOM | Zoom window |
| SC_MOVE | Move window | | |

**Figure 12. System information messages**

| Message | Word | Long | Item Changed |
|---|---|---|---|
| | Param | Param | |
| WM_DEVMODECHANGE | | lpString | Device mode changed |
| WM_FONTCHANGE | | | Font added or deleted |
| WM_SYSCOLORCHANGE | | | System color changed |
| WM_SYSTEMERROR | nError | | System error occurred |
| WM_TIMCHANGE | | | Time or time attributes changed |
| WM_WININICHANGE | | lpString | WIN.INI section changed |

**Figure 13. The TOGGLE program is shown here in the initial popup and tiled modes. The clock is another program.**



# A POWERFUL UTILITY

## For under $20, get a powerful programmer's utility that will bring you a year of:

- **hot technical tips** from experts like Tom Swan, Ray Duncan, Michael Abrash, William Hunt and Rex Jaeschke.
- **guidance for consultants** from Paul Barkley— **industry news** from Frank Greco and Hal (*DTACK Grounded*) Hardenbergh.
- **valuable and efficient code** in ASM, C, Pascal, BASIC
- **expert tips** on operating systems, the 386, graphics drivers, the EGA, and more!

## RESPECTED BY PROFESSIONALS

"For hot programming tips look to... the excellent but relatively little known **Programmer's Journal**."
Peter Norton —*Inside the IBM PC*

"...stuffed with useful information, it definitely deserves a look."
Ray Duncan—*DDJ* 6-86

"I really love this magazine!"
Harry Miller —Editor, *PC WORLD*

## GET ONE FREE!

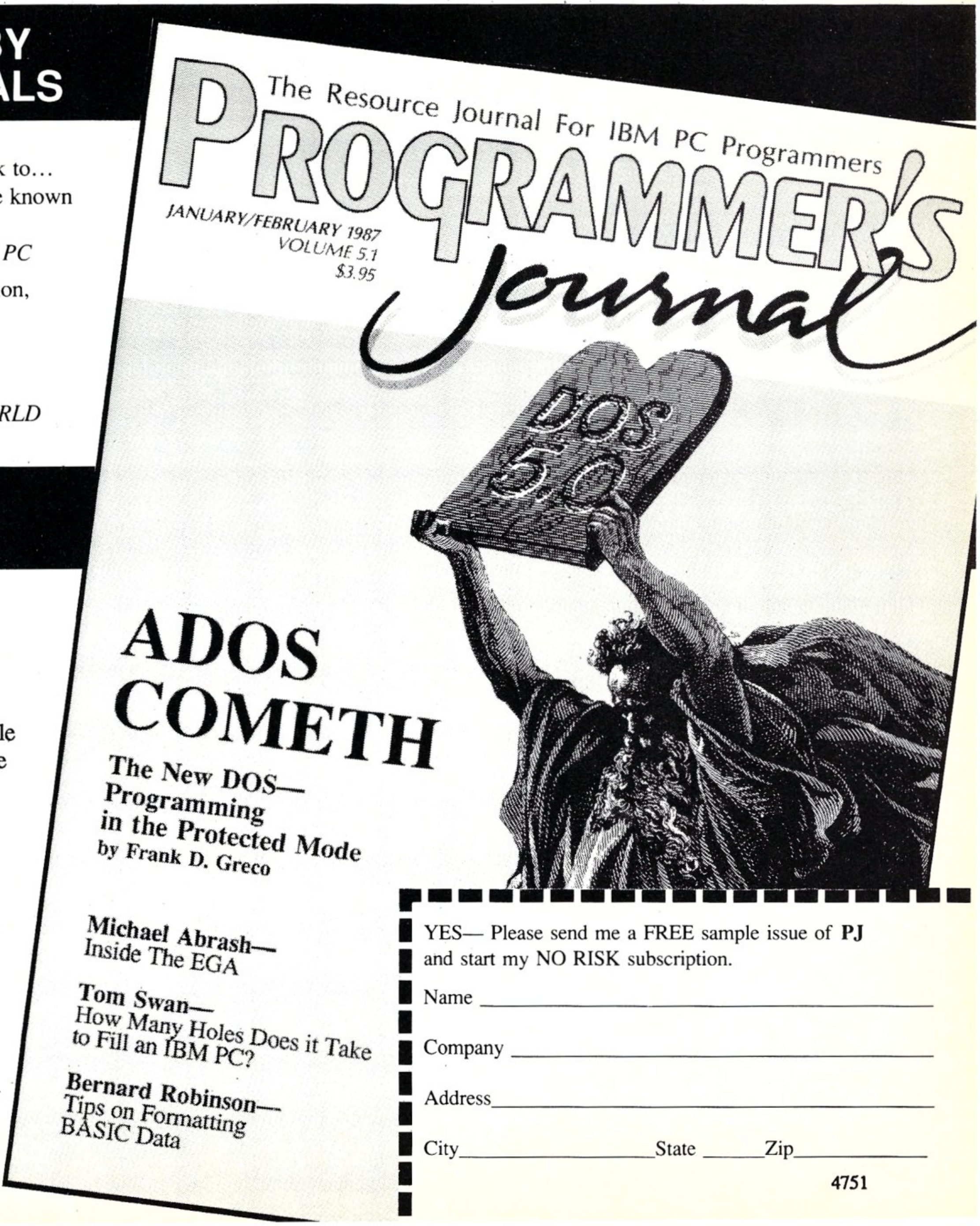
To see what ***PJ*** has to offer just do one of the following:

- call us at (503) 484-2162
- return the coupon at right

We'll send you a **FREE** sample copy of our latest issue, reserve a 1 year subscription (6 issues in all) and invoice you for $19.95. If **PJ** is not the utility you need, simply write "CANCEL" on the bill and return it. Keep your free issue and owe nothing.

**Programmer's Journal**
**P. O. Box 30160**
**Eugene, OR 97403**

Please allow 4-6 weeks for delivery of first issue. Offer good in U.S. only. Foreign subscriptions must be prepaid in U.S. funds. Can/Mex $29.95, elsewhere $39.95.



The Resource Journal For IBM PC Programmers

PROGRAMMER'S Journal

JANUARY/FEBRUARY 1987
VOLUME 5.1
$3.95

ADOS COMETH

The New DOS—
Programming
in the Protected Mode
by Frank D. Greco

Michael Abrash—
Inside The EGA

Tom Swan—
How Many Holes Does it Take
to Fill an IBM PC?

Bernard Robinson—
Tips on Formatting
BASIC Data

YES— Please send me a FREE sample issue of **PJ** and start my NO RISK subscription.

Name ______________________________

Company ______________________________

Address______________________________

City________________State _______Zip___________

4751

*WM_INITMENU* is sent when the user selects the menu bar of a window. Neither message is sent if a window has no menu bar. Access to the menu handle, *hMenu*, allows an application to change the state of items in the menu. The *lPopupIndex* is the index of the pop-up menu item within the main menu. The high part of *lPopupIndex* is 0 if the pop-up menu is not the system menu.

The *WM_INITDIALOG* message allows an application to control a dialog box instead of letting Windows handle everything, which is the conventional way of using dialog boxes. A zero reply value tells Windows not to set the input focus to the dialog box. It is up to the application to control the input focus. Windows handling this message can call the *EndDialog* function. This function will be discussed in a future part of this article.

### Input Messages

Figure 7 lists the input messages. *WM_MOUSEMOVE* and *WM_xBUTTONx* messages are sent when the mouse moves or a button is pressed or released. The mouse position is kept in *lPoint*. *NButton* has the current button status, which is a combination of items listed in Figure 8.

*WM_KEYDOWN* and *WM_KEYUP* are sent when a key is pressed or released. These are normally translated into *WM_CHAR* and *WM_DEADCHAR* messages. *WM_SYSKEYDOWN, WM_SYSKEYUP, WM_SYSCHAR*, and *WM_SYSDEADCHAR* messages are similar except that the Alt key is depressed. *NKey* contains the virtual key code. *LStatus* is divided into the following bit fields:

| Bit | Field |
|---|---|
| 1-16 | repeat count |
| 17-25 | scan code |
| 25-28 | reserved |
| 29 | Alt key down |
| 30 | key was down |
| 31 | key being released |

### System Input Messages

The messages in Figure 9 are sent to the window with the input focus. The exception is *WM_TIMER*, which can be sent at any time. *WM_HSCROLL* and *WM_VSCROLL* are sent when the corresponding scroll bar is selected. *NControl* can have any of the values listed in Figure 10. The *SB_THUMBx* messages use the second parameter containing the current position of the thumb in the low-order part. The scrolling done in the client area is application dependent—for example, it can be on a line or page basis for text.

*WM_TIMER* is sent in response to a timer request after a specified amount of time. *NIDEvent* identifies the event, and *lpfnTimer* points to the function passed to the initiating *SetTimer* function.

*WM_COMMAND* is sent in response to selection of an item from a menu when a keystroke is an accelerator that has been translated or when a control, such as a button, passes a message to the parent window. *NID* has the ID of the menu item, control, or accelerator. *LParam* is 0 for a menu item, 1 in the high-order part for an

## A Windows Programming Example

Listings 1-5 contain our first Windows program, called TOGGLE. It begins with a pop-up window containing a single child window. An option is added to the system menu telling TOGGLE to convert itself to a tiled window. The constants *IDSTILED* and *IDSPOPUP* have been added to the system menu entry so they can be processed in *MyWndProc*. *New_window* is included to support the more complex creation of the windows. The constant and menu strings are included in the system menu depending upon whether the window is being created as a pop-up or tiled window.

Note that the child window always has a parent whereas the main window does not. This allows a pop-up window to be displayed until it wants to disappear, as when the *IDSTILED* system menu entry is selected.

The *position* variable retains the location and size of a pop-up window. Switching from a tiled to a pop-up window in TOGGLE causes the pop-up window to be placed in the position the window was in when it was converted from a pop-up to a tiled window. This variation is easy to work with because TOGGLE can be started many times. Each instance of TOGGLE starts in the center of the screen and overlays the previous instance unless the prior window is converted to a tiled window or the window is moved.

*MainInit* is essentially the same function as in the first example except that two window classes are initialized—one is for the main pop-up/tiled window, and the second is for the child window. *WinMain* is also similar to the *WinMain* function in the first example except that there are additional strings to load and the initial position of the pop-up window is computed.

*MyWndProc* has been extended to handle the *IDSTILED* and *IDSPOPUP* messages. Also the movement of the *PostQuitMessage* function has been changed from the *WM_DESTROY* switch statement entry to the *SC_CLOSE* entry, which prevents TOGGLE from terminating early. This would occur if the initial example's code were used because the first thing the *IDSTILED* and *IDSPOPUP* entries do is destroy the existing parent and child windows, send the *WM_DESTROY* message to a window, and invoke the *PostQuitMessage* function to terminate the program.

The *WM_DESTROY* message is now ignored. Note, that it must be explicitly included or the default would be handled by *DefWindowProc* in *MyWndProc*. The *SC_CLOSE* function issues *PostQuitMessage*. This message is sent when the *CLOSE* system menu entry is selected.

The *SC_CLOSE* message would have invoked a *WM_DESTROY* message if left to the *DefWindowProc* support. TOGGLE's approach produces the same effect while letting windows be created and destroyed at will by the application. An alternative is to have a global flag handle the state information and not terminate unless the flag is set.

The other new addition is *MyChildWndProc*, the window procedure for the child window, which simply contains a call to *DefWindowProc* to handle all the messages. This function will be enhanced in future examples.

The parent windows handle all movement and resizing requests. The child window will always be clipped when moved within the parent window.

The windows do not display any text or graphics as yet, only the child window within the parent. TOGGLE shows how windows are related to parents and children. Also, it presents a mechanism for converting between a pop-up and a tiled window. Notice the minor difference between a pop-up and a tiled window in terms of support—most of the support is handled by Windows internally. An example is shown in Figure 13.

TOGGLE is a starting point for future examples. The next example will add graphics to the windows to show how clipping and drawing are handled.

accelerator, or a control indication. The latter has the handle of the window of the control in the high-order part and a control ID in the low-order part.

The *WM_SYSCOMMAND* occurs when an item is selected from the system menu. Values of nCommand are listed in Figure 11. *NCommand* has Windows-specific information in the four least-significant bits. These bits must be 0 to match the values listed in Figure 11. Applications can add items to the system menu, but the associated parameter values must differ.

## System Information Messages

The messages listed in Figure 12 are sent to windows or can be processed by the task, so they may alter their current state when the user changes some attribute of the system such as the system colors or items in the WIN.INI file. *LpString* references the name of the device that is changed in *WM_DEVMODECHANGE* and the name of the section for *WM_WININICHANGE*. The only system error is an out-of-memory error with *nError* set to 8.

## Summary

The sample program, TOGGLE, in Listings 1-5, which is discussed in more detail in the sidebar, and Figure 13 show how the functions covered in the previous part of this article and the messages mentioned in this one are tied together. Some of these functions were used in the first sample application. The functions and messages described up to now are appropriate only after a window or class has been created. Input focus—a resource that is handed from one window to another—is a concept unique to a multitasking environment. A single caret tracks the input focus under application control, and the cursor tracks the mouse.

Windows can be used without a mouse, but applications must be set up to process keyboard input and translate it into mouse movement. Menus and dialog boxes can be manipulated via the keyboard.

The next part of this article will cover the GDI functions and supporting window functions and messages, including the ideas of device contexts and paint structures. A primitive application can be created at this time. Part VI will discuss menus and dialog boxes, allowing full applications to be designed. §

*Bill Wong is president of Logic Fusion, Inc., 1333 Moon Dr., Yardley, PA 19067, a systems software development firm.*

### *Listing 1—Toggle*

```
toggle.res: toggle.rc toggle.ico toggle.h
    rc -r toggle.rc

toggle.obj: toggle.c toggle.h
    cc -d -c -u -W2 -Asnw -Gsw -Os -Zpe toggle.c

toggle.exe: toggle.obj toggle.res toggle.def
    link4 toggle, toggle/align:16, toggle/map, slibw, toggle.def
    mapsym toggle
    rc toggle.res
```

### *Listing 2—Toggle.def*

```
NAME     Toggle

DESCRIPTION 'Logic Fusion Inc. Basic Windows Program'

STUB     'WINSTUB.EXE'

CODE     MOVEABLE
DATA     MOVEABLE MULTIPLE

HEAPSIZE    4096
STACKSIZE   4096

EXPORTS
    MyWndProc     @1
```

### *Listing 3—Toggle.h*

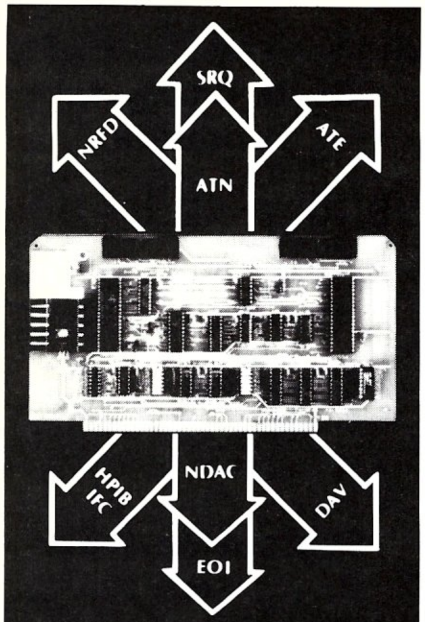
```
/* ----  Windows Popup Program Header File  02-16-87 WGW  ----*/
/**/
/* ----  String table constants  ----*/

#define IDSNAME      100/* ID for program name string*/
#define IDSTITLE     200/* ID for window title string*/
#define IDSPOPUP     300/* ID for POPUP  menu  string*/
#define IDSTILED     400/* ID for TILED  menu  string*/
#define IDSCHILD     500/* IE for child  title string*/


/* ----  End of Windows Popup Header File  ----*/
```

### *Listing 4—Toggle.rc*

```
#include "windows.h"
#include "toggle.h"

toggle  ICON    toggle.ico

STRINGTABLE
BEGIN
    IDSNAME,     "Toggle"
    IDSTITLE,    "Sample Windows Program"
    IDSPOPUP,"Popup"
    IDSTILED,"Tiled"
    IDSCHILD,"Child Window"
END
```

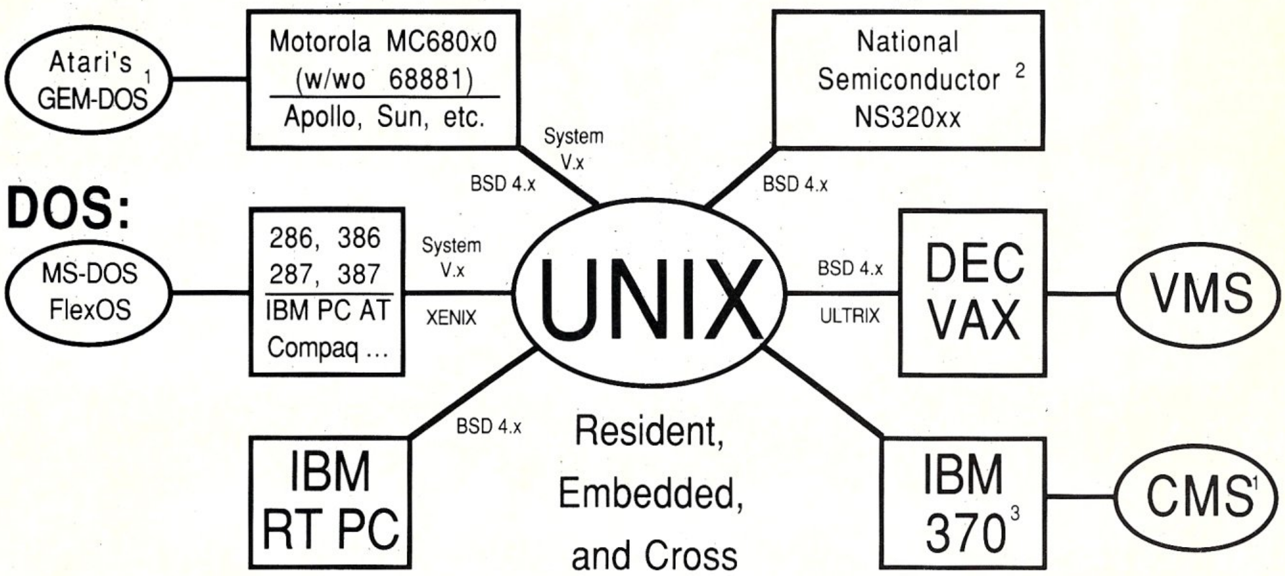
### *Listing 5—Toggle.c*

```
/*  ----  Basic Windows Toggle Program     02-07-87 WGW  ----*/

#include "windows.h"/* common Windows  definitions*/
#include "toggle.h"/* common TOGGLE.EXE definitions*/


/* ----  Forward Function Definitions  ----*/

long FAR PASCAL MyWndProc       ( HWND, unsigned, WORD, LONG) ;
long FAR PASCAL MyChildWndProc ( HWND, unsigned, WORD, LONG) ;


/* ----  Internal Global Variable Definitions  ----*/

char    szAppName     [ 10 ] ;
char    szWindowTitle [ 30 ] ;
char    szPopup       [ 20 ] ;
charszTiled        [ 20 ] ;
charszChild        [ 20 ] ;

RECTposition ;/* window position in popup mode*/


/* ----  Function Definitions  ----*/
/**/
/* ----  Paint the work area of the window the background color  ----*/

void MainPaint ( pps )
  PAINTSTRUCT * pps ;
{
  HBRUSH hbr, hbrOld ;
  HDC    hDC = pps ->  hdc ;


  if ( pps ->  fErase )/* check if erase flag is set*/
  {
    /* ----  Erase update rectangle with background color  ----*/

    hbr    = CreateSolidBrush ( GetSysColor (COLOR_WINDOW) ) ;
    hbrOld = (HBRUSH) SelectObject ( hDC,   (HANDLE) hbr ) ;

    FillRect     ( hDC,   (LPRECT) & pps ->  rcPaint, hbr ) ;
    SelectObject ( hDC,   (HANDLE) hbrOld ) ;
    DeleteObject ( (HANDLE) hbr ) ;
  }
}


/* ----  Initialize program  ----*/
/**/
/* Return TRUE if an error occurs.*/

BOOL MainInit ( hInstance )
  HANDLE hInstance ;
{
  BOOL      result ;
  PWNDCLASS pTypeClass ;


  /* -----  Copy strings from resource section  ----*/

  LoadString ( hInstance, IDSNAME,    (LPSTR)szAppName,     10 ) ;
  LoadString ( hInstance, IDSTITLE,   (LPSTR)szWindowTitle, 30 ) ;
  LoadString ( hInstance, IDSPOPUP,   (LPSTR)szPopup,       20 ) ;
  LoadString ( hInstance, IDSTILED,   (LPSTR)szTiled,       20 ) ;
  LoadString ( hInstance, IDSCHILD,   (LPSTR)szChild,       20 ) ;


  /* ----  Get buffer area for setup parameters  ----*/

  pTypeClass =  (PWNDCLASS) LocalAlloc ( LPTR, sizeof (WNDCLASS)) ;


  /* ----  Setup child window class for registry  ----*/

  pTypeClass ->  hCursor       = LoadCursor ( NULL, IDC_CROSS ) ;
  pTypeClass ->  hIcon         = NULL ;
  pTypeClass ->  lpszMenuName  = (LPSTR) NULL ;
  pTypeClass ->  lpszClassName = (LPSTR) szChild ;
  pTypeClass ->  hbrBackground = (HBRUSH) GetStockObject ( WHITE_BRUSH ) ;
  pTypeClass ->  hInstance     = hInstance ;
  pTypeClass ->  style         = CS_VREDRAW | CS_HREDRAW ;
  pTypeClass ->  lpfnWndProc   = MyChildWndProc ;


  /* ----  Register child window class  ----*/

  RegisterClass ((LPWNDCLASS) pTypeClass ) ;


  /* ----  Setup window class for registry  ----*/

  pTypeClass ->  hCursor       = LoadCursor ( NULL, IDC_ARROW ) ;
  pTypeClass ->  hIcon         = LoadIcon ( hInstance,  (LPSTR)szAppName ) ;
  pTypeClass ->  lpszMenuName  = (LPSTR) NULL ;
  pTypeClass ->  lpszClassName = (LPSTR) szAppName ;
  pTypeClass ->  hbrBackground = (HBRUSH) GetStockObject ( WHITE_BRUSH ) ;
  pTypeClass ->  hInstance     = hInstance ;
  pTypeClass ->  style         = CS_VREDRAW | CS_HREDRAW ;
  pTypeClass ->  lpfnWndProc   = MyWndProc ;


  /* ----  Try to register class and then deallocate structure  ----*/

  result = ( ! RegisterClass ((LPWNDCLASS) pTypeClass )) ;

  LocalFree ((HANDLE) pTypeClass ) ;/* Deallocate structure*/

  return result ;/* return initialize flag*/
}


/* ----  Create a new window of the specified type  ----*/

HWND new_window ( hInstance, type, cmdShow )
  HANDLE hInstance ;
  long   type ;
  int    cmdShow ;
{
  HWND  hWnd, hChildWnd ;
  HMENU hMenu ;
  LPSTR lpSelection ;
  int   nSelection ;
  RECT  clientRect ;


  /* ----  Create a window instance of my class  ----*/

  if ( type == WS_POPUPWINDOW )
  {
    /* ----  Create a popup window  ----*/

    lpSelection = szTiled ;/* allow other window*/
    nSelection  = IDSTILED ;
    hWnd = CreateWindow ( (LPSTR) szAppName/* application name*/
                        , (LPSTR) szWindowTitle/* window title*/
                        ,   WS_POPUPWINDOW/* window type*/
                          | WS_CAPTION | WS_SIZEBOX
                          | WS_VSCROLL | WS_HSCROLL
                        , position.left/*  x - position*/
                        , position.top/*  y - position*/
                        , ( position.right - position.left ) + 1
/* cx - width*/

                        , ( position.bottom - position.top ) + 1
/* cy - height*/
                        , (HWND)   NULL/* no parent window*/
                        , (HMENU)  NULL/* use class menu*/
                        , (HANDLE) hInstance/* handle to window */
                        , (LPSTR)  NULL/* no parameters*/
                        ) ;
  }
  else
  {
    /* ----  Create a tiled window  ----*/

    lpSelection = szPopup ;/* allow other window*/
    nSelection  = IDSPOPUP ;
    hWnd = CreateWindow ( (LPSTR) szAppName/* application name*/
                        , (LPSTR) szWindowTitle/* window title*/
                        ,   WS_TILEDWINDOW/* window type*/
                          | WS_CAPTION | WS_SIZEBOX
                          | WS_VSCROLL | WS_HSCROLL
                        , 0/*  x - position*/
                        , 0/*  y - position*/
                        , 0/* cx - width*/
                        , 0/* cy - height*/
                        , (HWND)   NULL/* no parent window*/
                        , (HMENU)  NULL/* use class menu*/
                        , (HANDLE) hInstance/* handle to window */
                        , (LPSTR)  NULL/* no parameters*/
                        ) ;
  }


  /* ----  Insert menu selection into system menu  ----*/

  hMenu = GetSystemMenu( hWnd, FALSE ) ;
  ChangeMenu( hMenu, 0, NULL, 999, MF_APPEND | MF_SEPARATOR ) ;
  ChangeMenu( hMenu, 0, lpSelection, nSelection, MF_APPEND | MF_STRING ) ;


  /* ----  Make window visible depending upon how app was started  --*/

  ShowWindow   ( hWnd, cmdShow ) ;
  UpdateWindow ( hWnd ) ;


  /* ----  Create a child window  ----*/

  GetClientRect ( hWnd, (LPRECT) & clientRect ) ;

  if (( clientRect.bottom * 2 ) > (int) GetSystemMetrics ( SM_CYCAPTION ))
  {
    /* ----  Create a child window  ----*/

    clientRect.bottom /= 2 ;/* make a small window*/
    clientRect.right  -= (int) GetSystemMetrics ( SM_CXVSCROLL ) ;

    hChildWnd =
           CreateWindow ( (LPSTR) szChild/* application name*/
                        , (LPSTR) szChild/* window title*/
                        ,   WS_CHILDWINDOW/* window type*/
                          | WS_CAPTION | WS_VISIBLE
                        , 10/*  x - position*/
                        , 10/*  y - position*/
                        , clientRect.right + 1/* cx - width*/
                        , clientRect.bottom + 1/* cy - height*/
                        , hWnd/* parent window*/
                        , (HMENU)  NULL/* use class menu*/
                        , (HANDLE) hInstance/* handle to window */
                        , (LPSTR)  NULL/* no parameters*/
                        ) ;

    UpdateWindow ( hChildWnd ) ;
  }


  /* ----  Return handle to main window  ----*/

  return hWnd ;
}


/* ----  Main Windows Entry Point for program  ----*/

int PASCAL WinMain ( hInstance, hPrevInstance, lpszCmdLine, cmdShow )
  HANDLE hInstance ;/* this program's instance handle*/
  HANDLE hPrevInstance ;/* previous module or NULL*/
  LPSTR  lpszCmdLine ;/* command line parameter string*/
  int    cmdShow ;/* display mode (hide, icon, window)*/
{
  int x_size, y_size ;
  MSG msg ;


  /* ----  Initialize class and get instance data  ----*/

  if ( hPrevInstance != NULL )/* see if instance active*/
  {
    /* ----  Copy data from previous instance  ----*/

    GetInstanceData ( hPrevInstance, (PSTR) szAppName,     10 ) ;
    GetInstanceData ( hPrevInstance, (PSTR) szWindowTitle, 30 ) ;
    GetInstanceData ( hPrevInstance, (PSTR) szPopup,       20 ) ;
    GetInstanceData ( hPrevInstance, (PSTR) szTiled,       20 ) ;
    GetInstanceData ( hPrevInstance, (PSTR) szChild,       20 ) ;
  }
  else
  if ( MainInit ( hInstance ))/* try to initialize*/
    return FALSE ;/* exit if error occurs*/

    /* ----  Setup popup parameters  ----*/
```



```
  x_size = GetSystemMetrics ( SM_CXSCREEN ) / 2 ;
  y_size = GetSystemMetrics ( SM_CYSCREEN ) / 2 ;

  position.right  = (( position.left = x_size / 2 ) + x_size ) - 1 ;
  position.bottom = (( position.top  = y_size / 2 ) + y_size ) - 1 ;


  /* ----  Create a window instance of my class  ----*/

  new_window ( hInstance, WS_POPUPWINDOW, cmdShow  ) ;


  /* ----  Loop until no more messages in event queue  -----*/

  while ( GetMessage ((LPMSG) & msg, NULL, 0, 0))
  {
    TranslateMessage ((LPMSG) & msg) ;
    DispatchMessage  ((LPMSG) & msg) ;
  }

  return (int) msg.wParam ;
}


/* ----  Main window procedure for handling messages to window  ----*/

long FAR PASCAL MyWndProc ( hWnd, message, wParam, lParam )
  HWND     hWnd ;
  unsigned message ;
  WORD     wParam ;
  LONG     lParam ;
{
  HANDLE      hInstance ;
  PAINTSTRUCT ps ;
  HBRUSH      hbr, hbrOld ;
  RECT        rect ;


  switch  (message)
  {
    case WM_SYSCOMMAND :/* process system command*/
      switch (wParam)
      {
        case IDSTILED :/* switch to tiled window*/
          GetWindowRect ( hWnd, (LPRECT) & position ) ;
          hInstance = GetWindowWord ( hWnd, GWW_HINSTANCE ) ;
          DestroyWindow ( hWnd ) ;/* destroy popup window*/
          new_window ( hInstance, WS_TILEDWINDOW, SHOW_OPENWINDOW ) ;
          break ;

        case IDSPOPUP :/* switch to popup window*/
          hInstance = GetWindowWord ( hWnd, GWW_HINSTANCE ) ;
          DestroyWindow ( hWnd ) ;/* destroy tiled window*/
          new_window ( hInstance, WS_POPUPWINDOW, SHOW_OPENWINDOW ) ;
          break ;

        case SC_CLOSE :/* Close selection made*/
          PostQuitMessage ( 0 ) ;/* exit from program*/
          break ;

        default :
          return DefWindowProc(hWnd, message, wParam, lParam );
      }
      break ;

    case WM_DESTROY :/* window being destroyed*/
      break ;

    case WM_PAINT :/* Paint work area*/
      BeginPaint ( hWnd, (LPPAINTSTRUCT) & ps ) ;
      MainPaint  ( (PAINTSTRUCT *) & ps ) ;
      EndPaint   ( hWnd, (LPPAINTSTRUCT) & ps ) ;
      break ;

    case WM_ERASEBKGND:/* Erase window*/
      hbr    = CreateSolidBrush ( GetSysColor (COLOR_WINDOW)) ;
      hbrOld = (HBRUSH) SelectObject (  (HDC) wParam,  (HANDLE) hbr ) ;

      GetClientRect ( hWnd,  (LPRECT) & rect ) ;
      FillRect       ((HDC) wParam,  (LPRECT) & rect, hbr ) ;
      SelectObject  ((HDC) wParam,  (HANDLE) hbrOld ) ;
      DeleteObject  ((HANDLE) hbr ) ;
      break ;

    default:
      return DefWindowProc ( hWnd, message, wParam, lParam ) ;
  }

  return (0L) ;
}

/* ----  Child window procedure for handling messages to window  ----*/

long FAR PASCAL MyChildWndProc ( hWnd, message, wParam, lParam )
  HWND     hWnd ;
  unsigned message ;
  WORD     wParam ;
  LONG     lParam ;
{
  return DefWindowProc(hWnd, message, wParam, lParam ) ;
}

/* ----  End of Basic Windows Toggle Program  ----*/
 ram ;
{
  return DefWindowProc(hWnd, message, wParam, lParam ) ;
}

/* ----  End of Basic Window
```

# C and Pascal on:



We cut our teeth on UNIX, but have become famous on MS-DOS, which we enhanced with our UNIX-like **DOS Helper** ™ utilities: find (including tar), fgrep, cat, ls, mv, tail, uniq, and wc; and our superior optimizing compilers:

**Professional Pascal** ™ and **High C** ™ on the PC are now well-respected by organizations such as Ansa, Ashton-Tate, AutoDesk, Boeing (BCS), Daisy Systems Corp., Deloitte Haskins & Sells, Digital Research, GE, IBM, Lifetree, Migent, Multimate, NYU, Silvar-Lisco, Sky Computers, Symantec, Xerox/Ventura, ...and *Computer Language* magazine; *Dr. Dobbs' Journal; PC Tech Journal; PC Magazine;* the *Journal of Pascal, Ada, and Modula-2...*

We supply the **first, and still only,** compilers generating **32-bit protected-mode code for the 80386** under MS-DOS. And our newly upgraded MS-DOS real-mode compilers were used by Symantec for their Q&A™ product to exploit the power of the 80386 real-mode instruction set. (**Just released:** HC v1.4 and PP v2.7, May 1987.)

Our **C Validation Suite** will blow your C compiler out of the sea, while our C compiler tracks the emerging ANSI Standard and generates tighter code with far better lint-like feedback help than competing compilers.

And you'll love **Professional Pascal**'s Ada-like packages, true data abstraction, C-like bit manipulation, and much more, along with the tight code that is linkable with **High C**, or other C, object modules (and vice versa).

Our **Translator Writing System (TWS)** goes far beyond LEX and YACC, with fully automatic error recovery...

All uniformly implemented on UNIX, VMS, CMS, MS-DOS, FlexOS...

**Professional software developers** in need of **industrial-strength tools** should contact:

Meta Ware™
INCORPORATED

MetaWare Incorporated
903 Pacific Avenue, Suite 201
Santa Cruz, CA 95060-4429
(408) 429-6382 (META)
Telex: 493-0879 (META UI)

## *The Clear Choice for Large Programming Projects.*

Name ____________________
Company ____________________
Address ____________________
City, ST ____________ Zip ______
Phone: (____) ______________

MSJ 07/87
Product:
Platform:

*Circle what interests you:*
PP HC TWS DOS Helper (DOS only
V.x 4.x DOS FlexOS VMS CMS
Sun Apollo Atari VAX 370
8086-family 80386pm 680x0 320xx

**Footnotes**: 1. Atari, CMS versions available 7/87. 2. NS320xx version by special order. 3. UNIX not yet available on 370.

# YOUR SYSTEM'S KEY COMPONENT

## The Only Magazine By And For Advanced Micro Users.

$3.95
U.S.A.
March/April 1987
Vol. 3 No. 2
Micro/Systems Journal
For the Advanced Computer User
Multitasking Between Programs
see page 22
Also in this issue
Communicating at 9600 bps . . 32
Program Interfacing to Microsoft Windows . . 36
Classic Technology's 286 Speed Pak . . 45
and introducing the column
LANScape . . 76

At last there is a magazine that brings you the strictly technical but practical information you need to stay up-to-date with the ever changing microcomputer technology . . . *Micro/Systems Journal. Micro/Systems Journal* is written with the needs of the systems integrator in mind—the individual who's involved in putting together the hardware and software pieces of the microcomputer puzzle.

In each issue of *Micro/Systems Journal* you'll find such useful and progressive articles as:

- Interfacing to Microsoft Windows
- Unix on the PC
- 80386 Programming
- High Resolution PC Graphics
- Using 80286 Protected Mode
- Multiprocessing and Multitasking

You'll get the hands-on, nuts and bolts information, insight and techniques that *Micro/Systems Journal* is famous for . . . in-depth tutorials, reviews, hints . . . the latest information on computer integration, networks and multi-tasking, languages, and operating systems . . . hard-hitting reviews.

To start your subscription to *Micro/Systems Journal*, simply fill out one of the attached cards or write to *Micro/Systems Journal*, 501 Galveston Dr., Redwood City, CA 94063. You'll receive a full year (6 issues) of *Micro/Systems Journal* for just $20, and enjoy the convenience of having M/SJ delivered to your doorstep each month. Don't wait . . . subscribe today!

## Multitasking with Turbo Pascal *code continued from page 32*

```
 they are used.}
procedure kbd_int;
begin
 inline
    {set in_kbd flag to indicate that the current task is handling
         an int}
 ($2E/$C6/$06/in_kbd/$FF/  {mov byte ptr cs:in_kbd,255}
   {pushf so that an iret will work}
  $9C/                     {pushf}
   {call the int handler}
  $2E/$FF/$1E/kbd_int_ofs/ {call cs:kbd_int_ofs}
   {reset inkbd flag}
  $2E/$C6/$06/in_kbd/$00/  {mov byte ptr cs:in_kbd,0}
   {switch tasks now}
  $CD/switch_task_vector/
  $CF);                    {iret}
end;

procedure disk_int;
begin
 inline
 ($2E/$C6/$06/in_disk/$FF/  {mov byte ptr cs:in_disk,255}
  $9C/                      {pushf}
  $2E/$FF/$1E/disk_int_ofs/ {call cs:disk_int_ofs}
  $2E/$C6/$06/in_disk/$00/  {mov byte ptr cs:in_disk,0}
  $CD/switch_task_vector/
  $CF);                     {iret}
end;

procedure change_tasks;
{ This routine switches the CPU from one task to the next
  in the dispatch list. It is called by the timer tick interrupt
  (1CH), which occures 18 times a second, or by the SWITCH_TASK
  command through int switch_task_vector.}
begin
  { Interrupts will be disabled when you get here }
  inline  {see Assembler Listing 1 for an explanation of
          the following}
  ($2E/$C6/$06/entry_via_switch/$FF/$EB/$06/$2E/$C6/$06/
  entry_via_switch/$00/$50/$2E/$A0/in_kbd/$22/$C0/$75/$08/
  $2E/$A0/in_disk/$22/$C0/$74/$11/$2E/$A0/entry_via_switch/
  $22/$C0/$58/$74/$02/$FB/$CF/$FB/$2E/$FF/$2E/timer_tick_ofs/
  $2E/$A0/waiting/$22/$C0/$74/$03/$58/$eb/$12/$53/$51/$52/$56/$57/
  $55/$06/$1E/$2E/$8C/$16/temp_ss/$2E/$89/$26/temp_sp/
  $55/$8B/$EC/$55/$2E/$A1/data_segment/$8E/$D8);
  if waiting=0 then {not waiting, so save tasks stack pointer in
                     preparation for switching tasks}
  begin
    with dispatch_list[current_task] do
    begin
      last_ss:=temp_ss;
      last_sp:=temp_sp;
    end;
  end;
  waiting:=0;  {reset waiting flag}
  { delete current task if requested by DELETE_TASK}
  if disp_del_flag then
  begin
    disp_del_flag:=false;
    {close up task descriptor currently occupied by current task}
    for disp_templ:=current_task to number_of_tasks-1 do
        dispatch_list[disp_templ]:=dispatch_list[disp_templ+1];
    with dispatch_list[disp_templ] do
      freemem(stack_ptr,stack_space_alloc);  {release stack space}
    number_of_tasks:=number_of_tasks-1;
    current_task:=1;
  end;
  { decrement all pause counters by one }
  if entry_via_switch=0 then {if entered through hdwr interrupt}
  begin
    for disp_templ:=1 to number_of_tasks do
    begin
      with dispatch_list[disp_templ] do
      begin
        if pause_count>0 then
        begin
          pause_count:=pause_count-1;
          if pause_count=0 then
            suspended:=false;
        end;
      end;
    end;
  end;
  { Check for dispatchable tasks. Go through list of all available
   tasks starting with current_task+1. The first one found that is
   not suspended and has priority_count=0 will be dispatched (next_
   dispatchable=task number).
    If none found, then first task not suspended is started
(next_possible=task#).
    If none found, then system simply waits in loop for next timer
         tick.}
  disp_templ:=current_task;
  next_dispatchable:=0;
  next_possible:=0;
  disp_stopflag:=false;
  while not disp_stopflag do
  begin
    disp_templ:=disp_templ+1;
    if disp_templ>number_of_tasks then
      disp_templ:=1;
    with dispatch_list[disp_templ] do
    begin
      if not suspended and (next_possible=0) then
        next_possible:=disp_templ; {records first task not suspended}
      if priority_count>0 then   {update priority counter}
        priority_count:=priority_count-1
      else
      begin
        if not suspended then
      begin
        next_dispatchable:=disp_templ;  {records first task not
         suspended with priority_count=0}
        priority_count:=priority;       {reset priority counter}
        disp_stopflag:=true;            {done searching for next
         task}
      end;
    end;
  end;
  if disp_templ=current_task then
    disp_stopflag:=true
end;
if entry_via_switch=0 then
begin
  {turn interrupt chip back on if it was a hardware interrupt}
  inline
  ($b0/$20/$e6/$20); {mov al,eoi , out 020h,al}
end;
{dispatch something or wait}
if next_dispatchable=0 then
begin
  if next_possible=0 then
  begin
    {if everything is suspended, then do nothing but wait in
        infinite loop}
    waiting:=255;
    {wait}
    inline($FB/            {sti - interrupts on so loop can be
        interrupted}
      $eb/$FE);            {qqq:jmp short qqq}
  end
  else
  begin
    {if only task unsuspended has low priority, let it run}
    current_task:=next_possible;
    with dispatch_list[current_task] do
      priority_count:=priority;
  end;
end
else
  current_task:=next_dispatchable;
{ restore context of new task to be started }
temp_ss:=dispatch_list[current_task].last_ss;
temp_sp:=dispatch_list[current_task].last_sp;
 inline   {see Assembler Listing 2 for explanation}
 ($2E/$8E/$16/temp_ss/$2E/$8B/$26/temp_sp/$1F/$07/$5d/$5f/$5e/
  $5A/$59/$5b/$2E/$A0/entry_via_switch/$22/$C0/$58/$74/$02/
  $FB/$cf/$FB/$2E/$FF/$2E/timer_tick_ofs);
end;

procedure set_vector(segment,offset,vector:integer);
{sets an interrupt vector (DOS function 25 hex)}
begin
  disp_result.ds:=segment;
  disp_result.dx:=offset;
  disp_result.ax:=$2500+vector;   { $25 in AH requests "set vector"
function}
  intr($21,disp_result); {call DOS function $25}
end;

procedure get_vector(var segment,offset:integer; vector:integer);
{gets an interrupt vector (DOS function 35 hex)}
begin
  disp_result.ax:=$3500+vector;   { $35 in AH requests "get vector"
function}
  intr($21,disp_result.ax); {call DOS function $35}
  offset:=disp_result.bx;
  segment:=disp_result.es;
end;

procedure init_dispatcher;
{initializes the dispatcher so it can be used}
var x,y:byte;
begin
  data_segment:=dseg;
  number_of_tasks:=1;
  current_task:=1;
  disp_del_flag:=false;
  { Init all elements in dispatch list }
  for x:=1 to max_tasks do
  begin
    with dispatch_list[x] do
    begin
      if x=1 then task_name:=$ffff else task_name:=0;
      last_sp:=0;last_ss:=0;
      suspended:=false;pause_count:=0;
      priority:=0;priority_count:=0;
    end;
  end;
  { Init all semaphor locations }
  for x:=0 to integer(bdos) do {bdos must remain last element
          in rescources list}
  begin
    with semaphores[resources(x)] do
    begin
      num_in_q:=0;free_first:=1;
      q_first:=0;q_last:=0;
      for y:=1 to max_tasks do
      begin
        queue[y]:=0;
        link[y]:=y+1;
      end;
      link[max_tasks]:=0;
    end;
  end;
  { save all old interrupt vectors for chaining }
  get_vector(timer_tick_seg,timer_tick_ofs,timer_tick_vector);
  get_vector(switch_task_seg,switch_task_ofs,switch_task_vector);
  get vector(kbd_int_seg,kbd_int_ofs,keyboard_vector);
  get_vector(disk_int_seg,disk_int_ofs,disk_vector);
  { replace interrupt vectors with new routines in this code.
          Note that change_tasks address has 7 added to it to get
          past a turbo pascal generated preamble for all subroutines,
          or 15 added to it so that entry_via_switch variable can be
          set correctly.}
  set_vector(cseg,ofs(change_tasks)+15,timer_tick_vector);
  set_vector(cseg,ofs(change_tasks)+7,switch_task_vector);
  set_vector(cseg,ofs(kbd_int)+7,keyboard_vector);
  set_vector(cseg,ofs(disk_int)+7,disk_vector);
  multitasking:=true;
end;

procedure start_task(task_addr,stack_space:integer;task_priority:byte;
                     var ok:boolean);
const register_space=24;   {Space needed on stack for original
          variables}
var x:byte;stack_seg,stack_ofs:integer;

  function mem_avail:boolean;
  var x:real;
  begin
    x:=maxavail;
    if x<0 then x:=65536.0+x;  {see Turbo manual under MaxAvail}
    if x*16.0<stack_space then mem_avail:=false else mem_avail:=true;
  end;

{Add a task to the dispatch list and set up it's registers for
          start up}
begin
  {insure that it's OK to start task}
  if ((number_of_tasks+1>max_tasks) and
      multitasking and mem_avail) then
    ok:=false
  else
  begin
    ok:=true;
    with dispatch_list[number_of_tasks+1] do
    begin
      task_name:=task_addr;           {save task name}
      {get space for stack from heap and save address it}
      stack_space_alloc:=stack_space;
      getmem(stack_ptr,stack_space);
      stack_seg:=seg(stack_ptr^);
      stack_ofs:=ofs(stack_ptr^)+stack_space-2;  {add stack space
             since stack grows toward zero}
      last_ss:=stack_seg;
      last_sp:=stack_ofs-register_space;
      suspended:=false;               {mark as dispatchable}
      priority:=task_priority;        {set up priority}
      pause_count:=0;                 {init counters to 0}
      priority_count:=0;
      {set up a register set image on the stack}
      memw[stack_seg:stack_ofs-2]:=initial_flags;
      memw[stack_seg:stack_ofs-4]:=cseg;
      memw[stack_seg:stack_ofs-6]:=task_addr;
      for x:=1 to 8 do
        memw[stack_seg:stack_ofs-x*2-6]:=0;
      memw[stack_seg:stack_ofs-register_space]:=dseg;
    end;
      {allow task to run by incrementing number of tasks}
      inline($fa);   {cli - interrupts off to avoid any problems }
      number_of_tasks:=number_of_tasks+1;
      inline($fb);   {sti}
    end;
end;
```

```
procedure remove_dispatcher;
{Takes out all changes made by INIT_DISPATCHER. Leaves sytem
              just as it was found}
var x:byte;
begin
  inline($fa);   {cli}  {interrupts off to avoid problems}
  number_of_tasks:=1;
  {replace vectors to original values}
  set_vector(timer_tick_seg,timer_tick_ofs,timer_tick_vector);
  set_vector(switch_task_seg,switch_task_ofs,switch_task_vector);
  set_vector(kbd_int_seg,kbd_int_ofs,keyboard_vector);
  set_vector(disk_int_seg,disk_int_ofs,disk_vector);
  multitasking:=false;
  inline($fb);  {sti}
end;

procedure wait(res:resources);
{Wait on semaphore for the specified resource}
var free:integer;
begin
  inline($fa);   {cli}  {interrupts off to avoid problems}
  with semaphores[res] do
  begin
    { Get first free element from free queue }
    free:=free_first;
    free_first:=link[free_first];
    { enter task name in element }
    queue[free]:=dispatch_list[current_task].task_name;
    { link element into q_last }
    if num_in_q=0 then
    begin
      q_first:=free;
      q_last:=free;
    end
    else
    begin
      link[q_last]:=free;
      q_last:=free;
    end;
    num_in_q:=num_in_q+1;
    {suspend current task if it has to wait in line and force
             task switch}
    if num_in_q>1 then
    begin
      dispatch_list[current_task].suspended:=true;
      inline($fb/$cd/switch_task_vector);  {sti, switch tasks}
    end;
    {otherwise let task continue running}
    inline($fb);  {sti}
  end;
end;

procedure available(res:resources);
{Signal that task is done with resource, so others can use it}
var free:integer;
begin
  inline($fa);   {cli}  {interrupts off to avoid problems}
  with semaphores[res] do
  begin
    { take q_first element off q }
    free:=q_first;
    q_first:=link[q_first];
    link[free]:=free_first;
    free_first:=free;
    num_in_q:=num_in_q-1;
    if num_in_q>0 then
    begin
      { unsuspend next task named queue }
      free:=1;
      while free<=number_of_tasks do
      begin
        if dispatch_list[free].task_name=queue[q_first] then
        begin
          dispatch_list[free].suspended:=false;
          free:=number_of_tasks+1;
        end else free:=free+1;
      end;
    end;
  end;
  inline($fb);  {sti}
end;

procedure delete_task;
{removes a task from the dispatch list}
var x:byte;
begin
  if (number_of_tasks>1) and (current_task<>1) then
  begin
    inline($fa);   {cli} {interrupts off to avoid problems}
    disp_del_flag:=true;  {signal to dispatcher that this task
                           should be deleted}
    inline($fb/$cd/switch_task_vector);  {sti, int switch_
           task_vector}
  end;
end;

procedure pause_task(ticks:integer);
{suspends a task and sets pause count as specified}
begin
  inline($fa);   {cli}  {interrupts off to avoid problems}
  {signal to dispatched that task should be suspended for
   the specified number of ticks}
  dispatch_list[current_task].pause_count:=ticks;
  dispatch_list[current_task].suspended:=true;
  inline($fb/$cd/switch_task_vector);  {sti, int switch_task_vector}
end;

procedure switch_task;
begin
  {force a software interrupt that will switch tasks}
  inline($cd/switch_task_vector);  {int switch_task_vector}
end;
```

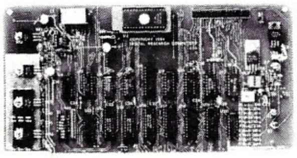
ALL SALES ARE MADE SUBJECT TO THE TERMS OF OUR 90 DAY LIMITED WARRANTY. A COPY OF THIS WARRANTY IS AVAILABLE FREE, ON REQUEST.

# DIGITAL RESEARCH COMPUTERS

(214) 225-2309

## S100 EPROM PROGRAMMER

*OUR NEWEST DESIGN, FOR FAST EFFICIENT PROGRAMMING OF THE MOST POPULAR EPROM'S ON YOUR S100 MACHINE. COMES WITH MENU DRIVEN SOFTWARE THAT RUNS UNDER CP/M 2.2 (8 INCH). PC BOARD SET CONSISTS OF (S100) MAIN LOGIC BOARD REMOTE PROGRAMMING CARD AND SIX PERSONALITY MINI BOARDS FOR 2716, 2532, 2732, 2732A, 2764, AND 27128. SOLD AS BARE PC BOARD SET ONLY WITH FULL DOC. SOFTWARE FEATURES "FAST" PROGRAMMING ALGORITHM. FOR Z80 BASED SYSTEMS.*



PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

**NEW!** $\$69^{95}$

## 128K S100 STATIC RAM/EPROM BOARD

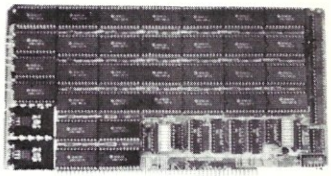
*JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.*

**NEW!**

| $\$59^{95}$ | $\$219^{00}$ | $\$139^{00}$ |
|---|---|---|
| BARE PC BOARD | 128K RAM KIT | 128 EPROM KIT |

## 64K S100 STATIC RAM

$\$99^{00}$ KIT



LOW POWER!

150 NS ADD $10

BLANK PC BOARD WITH DOCUMENTATION
$49.95

SUPPORT ICs + CAPS
$17.50

FULL SOCKET SET
$14.50

FULLY SUPPORTS THE NEW IEEE 696 S100 STANDARD (AS PROPOSED)

ASSEMBLED AND TESTED ADD $50

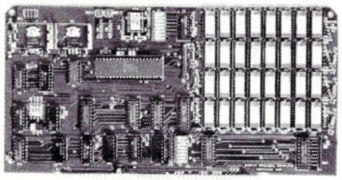
FEATURES: *PRICE CUT!*

* Uses new 2K x 8 (TMM 2016 or HM 6116) RAMs.
* Fully supports IEEE 696 24 BIT Extended Addressing.
* 64K draws only approximately 500 MA.
* 200 NS RAMs are standard. (TOSHIBA makes TMM 2016s as fast as 100 NS. FOR YOUR HIGH SPEED APPLICATIONS.)
* SUPPORTS PHANTOM (BOTH LOWER 32K AND ENTIRE BOARD).
* 2716 EPROMs may be installed in any of top 48K.
* Any of the top 8K (E000 H AND ABOVE) may be disabled to provide windows to eliminate any possible conflicts with your system monitor, disk controller, etc.
* Perfect for small systems since BOTH RAM and EPROM may co-exist on the same board.
* BOARD may be partially populated as 56K.

## 256K S-100 SOLID STATE DISK SIMULATOR!

***WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.***

**PRICE CUT!**



FEATURES:

* 256K on board, using +5V 64K DRAMS.
* Uses new Intel 8203-1 LSI Memory Controller.
* Requires only 4 Dip Switch Selectable I/O Ports.
* Runs on 8080 or Z80 S100 machines.
* Up to 8 LS-100 boards can be run together for 2 Meg. of On Line Solid State Disk Storage.
* Provisions for Battery back-up.
* Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
* Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB
(WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE)
$\$24^{95}$
(8203-1 INTEL $29.95)

*CLOSE OUT!* BLANK PCB ONLY:

$\$24^{95}$

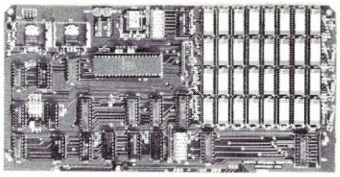
#LS-100

## 1 MEG. S-100 SOLID STATE DISK SIMULATOR!

***WE CALL THIS BOARD THE "LIGHT-SPEED-100" BECAUSE IT OFFERS AN ASTOUNDING INCREASE IN YOUR COMPUTER'S PERFORMANCE WHEN COMPARED TO A MECHANICAL FLOPPY DISK DRIVE.***

**LS 100 II**

***NEW!***



FEATURES:

* 1 Meg. on board, using +5V 256K DRAMS.
* Uses new Intel 8203-1 LSI Memory Controller.
* Requires only 4 Dip Switch Selectable I/O Ports.
* Runs on 8080 or Z80 S100 machines.
* Up to 4 LS-100 boards can be run together for 4 Megs. of On Line Solid State Disk Storage.
* Provisions for Battery back-up.
* Software to mate the LS-100 to your CP/M* 2.2 DOS is supplied.
* The LS-100 provides an increase in speed of up to 7 to 10 times on Disk Intensive Software.
* Compare our price! You could pay up to 3 times as much for similar boards.

BLANK PCB
(WITH CP/M* 2.2 PATCHES AND INSTALL PROGRAM ON DISKETTE)
$\$59^{95}$
(8203 1 INTEL $29.95)

(ADD $50 FOR A&T) $\$259^{00}$

#LS-100 II (FULL 1 M.B. KIT)

***1 MEGA BYTE!***

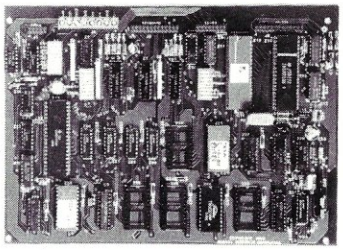
## ZRT-80 CRT TERMINAL BOARD!

*A LOW COST Z-80 BASED SINGLE BOARD THAT ONLY NEEDS AN ASCII KEYBOARD, POWER SUPPLY, AND VIDEO MONITOR TO MAKE A COMPLETE CRT TERMINAL. USE AS A COMPUTER CONSOLE, OR WITH A MODEM FOR USE WITH ANY OF THE PHONE-LINE COMPUTER SERVICES.*

FEATURES:

* Uses a Z80A and 6845 CRT Controller for powerful video capabilities.
* RS232 at 16 BAUD Rates from 75 to 19,200.
* 24 x 80 standard format (60 Hz).
* Optional formats from 24 x 80 (50 Hz) to 64 lines x 96 characters (60 Hz).
* Higher density formats require up to 3 additional 2K x 8 6116 RAMS.
* Uses N.S. INS 8250 BAUD Rate Gen. and USART combo IC.
* 3 Terminal Emulation Modes which are Dip Switch selectable. These include the LSI-ADM3A, the Heath H-19, and the Beehive.
* Composite or Split Video.
* Any polarity of video or sync.
* Inverse Video Capability.
* Small Size: 6.5 x 9 inches.
* Upper & lower case with descenders.
* 7 x 9 Character Matrix.
* Requires Par. ASCII keyboard.

FOR 8 IN. SOURCE DISK (CP/M COMPATIBLE) ADD $10



$\$89^{95}$ #ZRT-80 A&T ADD $50

(COMPLETE KIT, 2K VIDEO RAM)

BLANK PCB WITH 2716 CHAR. ROM. 2732 MON. ROM
$\$49^{95}$
SOURCE DISKETTE - ADD $10
SET OF 2 CRYSTALS - ADD $7.50

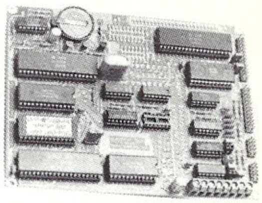
## THE NEW 65/9028 VT ANSI VIDEO TERMINAL BOARD!

★ FROM LINGER ENTERPRISES ★

**A second generation, low cost, high performance, mini sized, single board for making your own RS232 Video Terminal. This highly versatile board can be used as a stand alone video terminal, or without a keyboard, as a video console. VT100, VT52 Compatible.**

FEATURES: **MICRO SIZE!**

* Uses the new CRT9128 Video Controller driven by a 6502A CPU
* On-Screen Non-Volatile Configuration
* 10 Terminal Modes: ANSI, H19, ADM-5, WYSE 50, TVI-920, KT-7, HAZ-1500, ADDS 60, QUME-101, and Datapoint 8200
* Supports IBM PC/XT, and Parallel ASCII Keyboards
* Supports standard 15.75 kHz (Horiz.)
* Composite or Split Video (50/60 Hz)
* 25 X 80 Format with Non-Scrolling User Row
* Jump or Smooth Scroll
* RS-232 at 16 Baud Rates from 50 to 19,200
* On Board Printer Port
* Wide and Thin Line Graphics
* Normal and Reverse Screen Attributes
* Cumulative Character Attributes: De-Inten, Reverse, Underline and Blank
* 10 Programmable Function Keys and Answerback message
* 5 X 8 Character Matrix or 7 X 9 for IBM Monitors
* Mini Size: 6.5 X 5 inches
* Low Power: 5VDC @ .7A, ± 12VDC @ 20mA.



$79^{95}$ FULL KIT

w/100 Page Manual
ADD $40 FOR A&T

OPTIONAL EPROM FOR PC/XT STYLE SERIAL KEYBOARD: $15

SOURCE DISKETTE: PC/XT FORMAT 5¼ IN. $15

## Digital Research Computers

P.O. BOX 381450 • DUNCANVILLE, TX 75138 • (214) 225-2309

**TERMS:** Add $3.00 postage. Orders under $15 add 75¢ handling. No C.O.D. We accept Visa and MasterCard. Tex. Res. add 6-1/4% Tax. Foreign orders (except Canada) add 20% P & H. Orders over $50 add 85¢ for insurance.

*TM OF DIGITAL RESEARCH INC. (CALIF.)

WE ARE NOT ASSOCIATED WITH DIGITAL RESEARCH INC. (CALIF.) THE SUPPLIERS OF CPM SOFTWARE

# THE PROGRAMMER'S SHOP

**helps save time, money and cut frustrations. Compare, evaluate, and find products.**

## RECENT DISCOVERY

**Periscope III** - debugger with 64K protected RAM and breakout switch; breakpoints for hardware, memory, port, data. Real-time trace buffer, pass counter. PC $ 829

## AI-Expert System Dev't

| | | |
|---|---|---|
| Arity Combination Package | PC | $ 979 |
| System - use with C | MS | $ 229 |
| SQL Dev't Package | MS | $ 229 |
| Auto-Intelligence | PC | $ 739 |
| Exsys | PC | $ 309 |
| Runtime System | PC | $ 469 |
| Insight 1 | MS | $ 75 |
| Insight 2+ | MS | $ 379 |
| Intelligence/Compiler | PC | $ 739 |
| T.I.: PC Easy | PC | $ 435 |
| Personal Consultant Plus | PC | $2589 |
| Personal Consultant Runtime | PC | $ 85 |
| Turbo Expert-Startup(400 rules) | PC | $ 129 |
| Corporate (4000 rules) | PC | $ 359 |

## AI-Lisp

| | | |
|---|---|---|
| Microsoft MuLisp 85 | MS | $ 159 |
| PC Scheme LISP - by TI | PC | $ 85 |
| Star Sapphire | MS | $ 459 |
| TransLISP - learn fast | MS | $ 89 |
| TransLISP PLUS | | |
| Optional Unlimited Runtime | | $ 139 |
| PLUS for MSDOS | | $ 179 |

Others: IQ LISP ($155), IQC LISP ($269)

## AI Prolog

| | | |
|---|---|---|
| APT - Active Prolog Tutor - build applications interactively | PC | $ 49 |
| ARITY Prolog - Interpreter | PC | $ 229 |
| COMPILER/Interpreter-EXE | PC | $ 569 |
| Standard Prolog | MS | $ 77 |
| MicroProlog - Prof. Entry Lev. | MS | $ 85 |
| MicroProlog Prof. Comp./Inter. | MS | $ 439 |
| MPROLOG P550 | PC | $ 175 |
| Prolog-86 - Learn Fast | MS | $ 89 |
| Prolog-86 Plus - Develop | MS | $ 229 |
| TURBO PROLOG by Borland | PC | $ 69 |
| Turbo Prolog Toolbox | PC | $ 69 |

## Basic

| | | |
|---|---|---|
| BAS_C - economy | MS | $ 179 |
| BAS__PAS - economy | MS | $ 135 |
| Basic Development System | PC | $ 105 |
| Basic Development Tools | PC | $ 89 |
| Basic Windows by Syscom | PC | $ 95 |
| BetterBASIC | PC | $ 129 |
| Exim Toolkit - full | PC | $ 39 |
| Finally - by Komputerwerks | PC | $ 85 |
| Inside Track | PC | $ 50 |
| Mach 2 by MicroHelp | PC | $ 55 |
| Peeks n Pokes | PC | $ 35 |

## FEATURES

**COBOL I/Q** - Ad hoc queries, customizable reports, and graphs from MicroFocus, RM, Microsoft, mbp COBOL, Lotus, dBASE, Btrieve, ASCII files. PC $ 539

**NET-TOOLS** - Access NETBIOS-compatible network systems from Microsoft C, Pascal, FORTRAN, Assembler, Lattice C. Full Source, No Royalties. PC $ 129

### FREE Newsletter

Insightful commentary, guest columnists, survey results, and valuable resource listings. Interviews, technical articles, predictions — even cartoons. No wonder 96% of our readers pass *The Programmer's Letter* on to their friends; no wonder 72% make sure they get their copy back to keep for reference! You can request a FREE sample copy today by calling our toll-free number. A personal subscription is just $25 per year.

**Our Services:**

- Programmer's Referral List
- Compare Products
- Help find a Publisher
- Evaluation Literature FREE
- BBS - 7 PM to 7 AM 617-740-2611
- Dealers Inquire
- Newsletter
- Rush Order
- Over 700 products
- National Accounts Center

## Basic Cont.

| | | |
|---|---|---|
| QuickBASIC | PC | $ **69** |
| Stay-Res | PC | $ 75 |
| Turbo BASIC - by Borland | PC | $ 69 |

## Cobol

| | | |
|---|---|---|
| Microfocus Professional Cobol | PC | $1699 |
| VS Workbench | PC | $3379 |
| Microsoft COBOL | MS | $ 439 |
| Microsoft Cobol Tools | PC | $ 209 |
| Personal COBOL | PC | $ 129 |
| Realia - very fast | MS | $ 819 |
| Ryan McFarland COBOL | MS | Call |
| COBOL-8X | MS | Call |
| Screenplay - screen mgmt. | PC | $ 129 |

## C Libraries-Communications

| | | |
|---|---|---|
| Asynch by Blaise | PC | $ 125 |
| Essential Comm Library | PC | $ 125 |
| With Debugger | PC | $ 189 |
| Greenleaf Comm Library | PC | $ 129 |
| Multi-Comm - add multitasking | PC | $ 135 |

## dBASE Language

| | | |
|---|---|---|
| Clipper compiler | PC | Call |
| dBASE II | MS | $ 329 |
| dBase III Plus | PC | $ 429 |
| dBASE III LanPack | PC | $ 649 |
| DBXL Interpreter | PC | $ 139 |
| FoxBASE+ - single user | MS | $ 349 |
| QuickSilver by Word Tech | PC | $ 499 |

## dBASE Support

| | | |
|---|---|---|
| dBase Tools for C | PC | $ 65 |
| dBrief with Brief | PC | Call |
| dBC ISAM by Lattice | MS | Call |
| dBx Translator to C | MS | $ 319 |
| dFlow - flowchart, xref | MS | Call |
| Documentor - dFlow superset | MS | Call |
| Genifer by Bytel-code generator | MS | $ 299 |
| QuickCode III Plus | MS | $ 239 |

## Editors for Programming

| | | |
|---|---|---|
| BRIEF Programmer's Editor | PC | Call |
| EMACS by UniPress Source: $895 | | $ 265 |
| Epsilon - like EMACS | PC | $ 149 |
| KEDIT - like XEDIT | PC | $ 99 |
| Micro Focus Micro/SPF | PC | $ 139 |
| PC/EDT - macros | PC | $ 229 |
| PC/VI - by Custom Software | MS | $ 109 |
| Personal REXX | PC | $ 99 |
| PMATE - power, multitask | PC | $ 119 |
| SPF/PC - fast, virtual memory | PC | $ 189 |
| Vedit | MS | $ 99 |
| Vedit PLUS | MS | $ 129 |

## RECENT DISCOVERY

**CxPERT** - Expert systems shell, translates to C code to integrate with your application. Certainty factors, explanations, inheritance, frames, help. MS $295

## C Language-Compilers

| | | |
|---|---|---|
| AZTEC C86 - Commercial | PC | $499 |
| C86 PLUS - by CI | MS | $379 |
| Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev'rs Kit | PC | $ 77 |
| Datalight Optimum-C | MS | $109 |
| with Light Tools by Blaise | PC | $168 |
| Lattice C - from Lattice | MS | $269 |
| Let's C Combo Pack | PC | $ 99 |
| Let's C | PC | $ 57 |
| Microsoft C 4.0- Codeview | MS | $**275** |
| Rex - C/86 - standalone ROM | MS | $695 |
| Turbo C by Borland | PC | $ 69 |
| Uniware 68000/10/20 Cross Compiler by SDS | MS | Call |
| Wizard C | MS | $299 |
| ROM Development Package | MS | $259 |

## C Language-Interpreters

| | | |
|---|---|---|
| C-terp by Gimpel - full K & R | MS | $219 |
| C Trainer - by Catalytix | PC | $ 89 |
| INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs | MS | $369 |
| Interactive C by IMPACC Assoc. | PC | $209 |
| Run/C Professional | MS | $155 |
| Run/C Lite | MS | $ 79 |

## C Libraries-General

| | | |
|---|---|---|
| Blackstar C Function Library | PC | $ 79 |
| C Essentials - 200 functions | PC | $ 75 |
| C Function Library | MS | $109 |
| C Tools Plus (1 & 2) - Blaise | PC | $125 |
| C Utilities by Essential | PC | $129 |
| C Worthy Library - Complete, machine independent | MS | $249 |
| Entelekon C Function Library | PC | $119 |
| Entelekon Superfonts for C | PC | $ 45 |
| Greenleaf Functions-portable, ASM | | $139 |
| LIGHT TOOLS by Blaise | PC | $ 69 |

## C Libraries-Files

| | | |
|---|---|---|
| C Index by Trio - full B + Tree. | MS | $ 89 |
| /File is object only | MS | $ 89 |
| /Plus is full source | MS | $319 |
| BTree by Soft Focus | MS | $ 69 |
| CBTREE - Source, no royalties | MS | $ 99 |
| CTree by Faircom - no royalties | MS | $315 |
| rtree - report generation | PC | $239 |
| dbQUERY - ad Loc, SQL - based | MS | $129 |
| dbVISTA - full indexing, plus optional record types, pointers, Network. | | |
| Object only - MS C, LAT, | C86 | $129 |
| Source - Single user | MS | $389 |
| Source - **Multiuser** | MS | $799 |
| dBx - translator | MS | $299 |
| w/source to library | MS | $349 |

## FEATURE

**UI Programmer** - Quickly generate dBASE User Interfaces, prototypes. Use supplied templates or create own. Pop-up help, bounce bar menus, screen forms. II, III, FoxBASE+, Quicksilver, Clipper PC $249

**We support MSDOS (not just compatibles), PCDOS, Xenix-86, CPM-80, Macintosh, Atari ST, and Amiga.**

# THE PROGRAMMER'S SHOP

**provides complete information, advice, guarantees and every product for Microcomputer Programming.**

## What Every Hard Disk User Needs . . .

Everyone with a fixed disk needs a set of reliable utilities that are more than just tools. The products at the right can increase your productivity by protecting your data, maintaining your hard disk, and making DOS directories easy to use. We use these , tools in-house, and they really work. GUARANTEED.

**Order before July 31, 1987 and mention this ad for these SPECIAL PRICES:**

| | List | Normal | SPECIAL |
|---|---|---|---|
| Disk Technician - smart disk upkeep | $ 99 | $ 89 | **$ 79** |
| Xtree - classic graphic tree | $ 50 | $ 45 | **$ 35** |
| Back-It - flexible, fast | $100 | $ 89 | **$ 69** |

## C Support-Systems

| | | |
|---|---|---|
| Advantage C + + | PC | $479 |
| C Sharp - **realtime,** tasks. | PC | $495 |
| C ToolSet - DIFF, xref, source | MS | $ 89 |
| The HAMMER by OES Systems | PC | $129 |
| Lattice Text Utilities | MS | $ 89 |
| Multi-C - multitasking | PC | $135 |
| PC LINT-Checker. Amiga $89 | MS | $ 99 |
| Quickshell - script compiler | PC | $349 |
| Pfantasy Pac - by Phoenix | PC | $799 |
| PforCe + + | PC | $259 |
| Pre-C - Lint-Like | MS | $155 |
| Sapiens V8 - 8M workspace | PC | $269 |
| SECURITY LIB - Source $229 | PC | $115 |
| Timeslicer | PC | $265 |

## C-Screens, Windows, Graphics

| | | |
|---|---|---|
| C Power Windows by Entelekon | PC | $109 |
| C-Scape - capture Dan Bricklin | PC | $179 |
| Curses by Lattice | PC | $ 89 |
| dBASE Graphics for C | PC | $ 69 |
| ESSENTIAL GRAPHICS - fast | PC | $185 |
| GraphiC - new color version | PC | $285 |
| Greenleaf Data Windows | PC | $159 |
| w/source | PC | $299 |
| LightWINDOWS/C-for Datalight C | PC | $ 79 |
| Multi-Windows - use w/ Multi-C | PC | $295 |
| Screen Ace Form Master | PC | $195 |
| Vitamin C - screen I/O | PC | $159 |
| **Windows for C -** fast | PC | $189 |
| **Windows for Data -** validation | PC | $319 |
| View Manager - by Blaise | PC | $179 |
| ZView - screen generator | MS | $169 |

## Debuggers

| | | |
|---|---|---|
| 386 Debug - by Phar Lap | PC | $129 |
| Breakout - by Essential | PC | $ 89 |
| CODESMITH - visual | PC | $ 99 |
| C SPRITE - data structures | PC | $129 |
| DSD87 - by Soft Advances | PC | $ 79 |
| Periscope I | PC | $289 |
| Periscope II | PC | $139 |
| Periscope II-X | PC | $105 |
| Pfix-86 Plus - by Phoenix | PC | $229 |
| Turbo TDebug | PC | $ 55 |
| Showcase - test software | PC | $125 |
| SoftProbe II - embedded systems | PC | $695 |

## FEATURE

**I/O Pro -** FORTRAN screen development system includes text/graphics editor, screen, string, graphics toolkit, communications, keyboard. Color support. IBM, MS, R-M, Supersoft FORTRAN. PC $129

## Fortran & Supporting

| | | |
|---|---|---|
| 50:More FORTRAN | PC | $ 99 |
| ACS Time Series | MS | $399 |
| Forlib + by Alpha | MS | $ 59 |
| MACFortran by Microsoft | MAC | $229 |
| MS Fortran - 4.0, full 77' | MS | $279 |
| No Limit - Fortran Scientific | PC | $115 |
| PC-Fortran Tools - xref, pprint | PC | $165 |
| RM/Fortran | MS | Call |
| Scientific Subroutines - Matrix | MS | $139 |

## Multilanguage Support

| | | |
|---|---|---|
| BTRIEVE ISAM | MS | $185 |
| BTRIEVE/N-multiuser | MS | $455 |
| Flash-Up Windows | PC | $ 79 |
| GSS Graphics Dev't Toolkit | PC | $375 |
| HALO Graphics | PC | $205 |
| Development Package | MS | $389 |
| Informix 4GL-application builder | PC | $789 |
| Informix SQL - ANSI standard | PC | $639 |
| Opt Tech Sort - sort, merge | MS | $119 |
| PANEL | MS | $215 |
| Pfinish - by Phoenix | MS | $229 |
| PolyLibrarian by Polytron | MS | $ 79 |
| PolyBoost - speed I/O, keyboard | PC | $ 69 |
| PVCS Corporate-source control | MS | $309 |
| PVCS Personal | MS | $109 |
| QMake by Quilt Co. | MS | $ 79 |
| Report Option - for Xtrieve | MS | $109 |
| Screen Machine | PC | $ 59 |
| Screen Sculptor | PC | $ 95 |
| SRMS - source control | MS | $109 |
| Synergy - create user interfaces | MS | $375 |
| VXM - multi-env. link | MS | $195 |
| Xtrieve - organize database | MS | $199 |
| ZAP Communications - VT 100 | PC | $ 89 |

## Pascal and Supporting

| | | |
|---|---|---|
| ALICE - learn Pascal | PC | $ **49** |
| Exec - Chain Programs | MS | $ 79 |
| MetaWINDOWS-graphics toolkit bit-mapped, fast | PC | $115 |
| MetaWINDOWS PLUS | PC | $185 |
| Microsoft PASCAL - faster | MS | $189 |
| Programmer's Extender | MAC | $ 79 |
| Pascal Pac with Tidy | PC | $ 49 |
| Pascal Tools PLUS | PC | $119 |
| Pascal 2 - by Oregon Software | MS | $329 |
| Turbo Extender by Turbo Power | PC | $ 75 |
| TurboHALO - 150 routines | PC | $ 99 |
| Turbo Optimizer | PC | $ 69 |

## DataBase & File Management

| | | |
|---|---|---|
| DataFlex by Data Access | PC | $ 899 |
| Dataflex multiuser | PC | $1149 |

## RECENT DISCOVERY

**SSP/PC -** Fast math subroutine library in C and Assembler for C, Fortran, Pascal, and BASIC. 145 + routines include trig, elementary, hyperbolic, and gamma; chi square, polynomials, more. 8087 Support. PC $ 269

## Other Languages

| | | |
|---|---|---|
| APL*PLUS/PC | PC | $ 429 |
| CCS Mumps - Singleuser | PC | $ 50 |
| CCS Mumps - Multiuser | PC | $ 359 |
| MasterForth - Forth '83 | MAC or PC | $ 109 |
| Microsoft MASM | MS | $ 98 |
| Modula-2 - by Pecan | MS | $ 79 |
| Modula-2/86 by Logitech | PC | $ 62 |
| Pasm - by Phoenix | MS | $ 115 |
| PC Forth + - by Lab Micro | PC | $ 199 |
| Smalltalk/V | MS | $ 85 |
| SNOBOL4 + - great for strings | MS | $ 80 |
| UR/Forth | MS | $ 279 |

## Xenix/Unix

| | |
|---|---|
| Basic - by Microsoft | $ 239 |
| C-Terp by Gimpel Software | $ 379 |
| Cobol - by Microsoft | $ 639 |
| Cobol Tools - by Microsoft | $ 319 |
| Fortran or Pascal - by Microsoft | $ 439 |
| MicroFocus Lev. II Compact COBOL | $ 795 |
| Panel | $ 535 |
| RM/Cobol | Call |
| RM/Fortran | Call |
| Xenix Complete System | $ 999 |
| Xenix Development System | $ 499 |

## Other Products

| | | |
|---|---|---|
| 386 Assembler/Linker | PC | $ 389 |
| Advantage Link | PC | $ 359 |
| ASMLIB - 170 + routines | PC | $ 125 |
| asmTREE - B + tree file mgmt. | PC | $ 339 |
| Compact Source Print | PC | $ 45 |
| Dan Bricklin's Demo Program | PC | $ 59 |
| Help/Control - on line help | PC | $ 99 |
| Interactive Easyflow-HavenTree | PC | $ 125 |
| Link & Locate - tools to work with Intel and Tektronix projects. | MS | $ 329 |
| LMK - like UNIX make | MS | $ 139 |
| Microsoft Windows | PC | $ 69 |
| Software Development Kit | PC | $ 319 |
| MKS Toolkit - Unix, vi, awk | PC | $ 99 |
| Norton Commander | PC | $ 55 |
| Numerical Analyst by Magus | PC | $ 269 |
| PDisk - cache, tree | PC | $ 99 |
| PLink - 86 PLUS - overlays | MS | $ 299 |
| Polymake by Polytron | MS | $ 119 |
| PolyShell by Polytron | MS | $ 119 |
| PolyXREF by Polytron | PC | $ 99 |
| Quelo 68000 X-ASM | PC | $ 509 |
| Sapiens V8 - 8M virtual mgr. | PC | $ 269 |
| Taskview - ten tasks | PC | $ 55 |
| Tom Rettig's Library - dBASE | PC | $ 89 |
| Tree Diagrammer | PC | $ 45 |
| Visible Computer: 8088 | PC | $ 65 |

Note: Mention this ad. Some prices are specials. Ask about COD and POs. Formats: 3" laptop now available, plus 200 others. UPS surface shipping add $3/item. All prices subject to change without notice.

## Call for a catalog, literature, advice and service you can trust

VISA **HOURS** MasterCard

8:30 AM - 8:00 PM EST.

# 800-421-8006

**THE PROGRAMMER'S SHOP** ™

*Your complete source for software, services and answers*

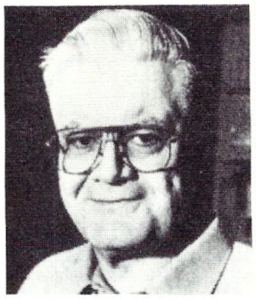
**5-M Pond Park Road, Hingham, MA 02043**
**Mass: 800-442-8070 or 617-740-2510 5/87**

"I like the way you do business. I ordered two programming packages from you and before I had time to wonder when I would receive them, they were on my desk. Good prices and fast delivery, - not a bad way to do business. - Thanks"

Jeff Schropfer
Bytrek

# The Scientific Computer User

A. G. W. Cameron



## Computer Modern Fonts, METAFONT, and TeX

### METAFONT Released

Donald Knuth, of the Stanford University Computer Science Department, has now finished a monumental eight-year detour into digital typesetting, after finding that the early books in his series of tomes on computer algorithms were unpleasing to his eye. This detour has encompassed two main tasks. The first of these resulted in the TeX program, a digital typesetting program that is designed to position characters, symbols, and other material on a page for maximum readability and beauty. This has become the primary means for formatting scientific text at my institution and at many others that I know about. The second task was to provide a means for designing digital type, including the great variety of symbols that are used in mathematics. This led to Knuth's development of the METAFONT program.

The end-products of this effort are a set of five volumes in the Computers and Typesetting series by Donald E. Knuth, published by Addison-Wesley. Volume A is *The TeXbook*, the standard reference for TeX users. Volume B is *TeX: The Program,* which contains the complete source code for TeX with appropriate documentation. The next two volumes do the parallel thing for METAFONT: volume C is *The METAFONTbook*, which is the standard reference for METAFONT users, and volume D is *METAFONT: The Program*, which contains the complete source code for METAFONT. The fifth book in the series, volume E, is *Computer Modern Typefaces*. It contains the complete specifications (METAFONT source code) for the 75 Computer Modern fonts that constitute the new font set for use with TeX. It also contains large blown-up pictures of each character in the different families of fonts, together with the reference points for that character.

I have discussed the use of TeX repeatedly in this column. Now it is METAFONT's turn. This program is available for use on PCs; Personal TeX has released its version, which is called pcMF ($195). The implementation of the program is still somewhat ragged, but I have been using it successfully for font generation.

For many years METAFONT's availability has been restricted to mainframe and minicomputer users. It has been used by some font designers and was used by Donald Knuth to design preliminary versions of fonts to use with TeX. To emphasize the preliminary nature of the font designs, Knuth called them "almost modern," and the names of these fonts begin with *am*. The final font design is called "computer modern," and the names of the fonts begin with *cm*. You might think that changing over to cm fonts would be a straightforward matter, but it turns out to be very complicated. I will get to the reasons for this in due course.

### About METAFONT

METAFONT is more than just a computer program. It is also a language. In the ordinary computer languages usually used by scientists, such as FORTRAN, a series of actions are commanded; numbers are to be transformed in various ways. In METAFONT, declarations are made about where the major components of a desired shape are located and how they relate to one another. It is left to the internal workings of the program to work out the details of where lines are drawn and where they cross. Thus, each letter, character, or symbol in a font has its own program containing these declarations. This same program is used for each font in a family of fonts; each font in a family differs from another usually in only a few parameters among a great many that need to be set. The many parameters that do not change comprise the distinctive characteristics of that family of fonts.

It is not my purpose here to give a detailed review of METAFONT. In the first place, font design is an art, and what Knuth has succeeded in doing is to analyze the techniques used by the artist, to quantify them, and to invent tools that enable the artist to express a design philosophy consistently across the range of characters in a font. Thus learning to use METAFONT is much more than just an exercise in reading a manual; you need to have a lot of experience, to exercise a lot of judgment, and to develop a flair for design. Right now I am still a beginner—I have used METAFONT successfully, I have produced standard fonts with new digital resolutions, and I have experimented a little bit by making minor changes in the font parameters. I will not have gone beyond the beginner stage until I have produced an entirely original design.

TeX users will find many familiar techniques used in METAFONT. *The METAFONTbook* is laid out in much the same way as is *The TeXbook*. It is a running tutorial that gradually introduces new topics and puts a lot of the technical details in appendices. It gives lots of exercises. It tells many "little white lies" in the course of doing this, so the first descriptions of certain operations are often oversimplified and you learn about the fine print later. METAFONT uses macro expressions. There is a great deal of similarity with TeX in the notation used.

The design of a character starts with the specification of the positions of some reference points. The lines that are drawn do not necessarily go through these reference points, but they are related to them. Usually the equations that describe the lines are polynomials of degree 3 in the position coordinates; these are called Bézier cubics. The lines themselves, however, can be of variable width. Here Knuth has followed established principles in calligraphy. He has adopted the notion of the pen that traces out a curve. Pens can have arbitrary shapes, and they can be rotated during the course of their motion, but they leave a trail of "ink" as they move over the paper. This ink can also be selectively erased. The ultimate image that is painted in this way must then be further transformed to give the best possible representation of that image in terms of pixels of finite size. It is this mathematical problem that forms the core of the work that METAFONT has to do, and it explains why the calculation of each character in a font may take a significant fraction of a minute, even with a 286 machine.

METAFONT users can take their output as a large display on the screen (in graphics mode). They can also produce hard-copy output with large characters in several ways. These are the normal ways of working in the midst of character design. Such displays are dispensed with during production, when you want to produce generic font files at a characteristic resolution and the corresponding font metric files that contain all the necessary information about character heights, depths, widths, kerning, and ligatures.

In order to use pcMF, you must initialize a LOCAL.MF file that contains some macro definitions that you will write to define the resolution that you want for a given style of character and also to set some hardware-dependent parameters for METAFONT, such as whether the pixels should be unusually black. PcMF has been prepared by Doug Henderson, who has as yet prepared only a preliminary set of notes about how to use the program. These suffice for computer users with a bit of hacker mentality, but average users may need a bit of help in getting started, and these users should be prepared to spend a lot of time thumbing the pages of volumes C and E of the Computers and Typesetting series. But the net result of doing all this can be a lot of fun. You, too, can be what Donald Knuth calls a "font freak."

### Font Confusion

TeX and METAFONT were originally developed on mainframe computers, on which disk storage was not a terribly scarce commodity and word sizes were fairly long. The original pixel files for TeX fonts were called .PXL files and were stored with raster representations 32 bits long. This was an extremely wasteful method for raster resolutions of 300 dots per inch or less. ArborText (formerly Textset), developer of several TeX drivers for laser printers, converted these .PXL files to a more compact form based on a raster word length of 8 bits, but this format was not widely adopted in the TeX community. Then, a much more compact PK form of storage based on compression was introduced by Tomas Rokicki at Stanford University. This was a step in the right direction, but it led to further confusion recently when the conversion algorithm was changed and the drivers that had been designed to read the original PK format no longer worked.

Now a further source of confusion has been added in connection with the naming of directory structures containing computer modern fonts as distinct from almost modern fonts. Drivers had been developed by Personal TeX (for use with PCTeX) and Addison-Wesley (for use with MicroTeX) to look for font files in a standard directory structure in which the ultimate subdirectories containing almost modern pixel files at a given resolution had standard names. Those names have now been changed on the grounds that the new names are more logical. The argument is undoubtedly correct, but the process is painful. I have an alpha copy of MicroTeX, Version 2.0, which has a built-in screen previewer developed by David Fuchs and which was demonstrated at the last TeX Users' Group meeting, and here the preview fonts use generic font formats, which are again different. It is not at all clear that the TeX community will adopt the generic font format for general use, so that may not be the form in which the product is eventually released.

Now the nature of the font confusion is apparent. If you have am fonts and want to change to cm fonts, not only do you need the new sets of fonts at the appropriate resolutions, but you also need new drivers for all your output devices as well. Those output devices may include a screen for screen preview, a dot-matrix printer for draft output, and a laser printer for better-looking (and usually final) output. So you will need fonts at several resolutions and several drivers to be all in hand before you make this change. Thus a fair amount of organization (and probably a fair amount of money) is involved.

### Screen Previewers

Screen previewers tend to be written to use low-resolution fonts with low-resolution screens. The standard font distributions that have been widely available have been at 80 and 118 dots per inch. These are not well suited to the use of monitors that are used with PCs. This is one of the reasons why I generally prefer to use the preview program called Maxview (Personal TeX, $125), which uses the actual pixel files that will be used with a printer and condenses several pixels into one on the screen. Thus a 300-dot-per-inch (dpi) font for laser printers will have a screen reduction factor of 3 and combine 9 pixels into 1, and a 180-dpi font for 24-pin dot-matrix printers will have a screen reduction factor of 2 and combine 4 pixels into 1. These combinations give nicely readable text on the screen. The use of a higher reduction factor allows you to see more of the page layout. Right now the maximum screen reduction factor is 3, but 4 will become available in the future. Maxview requires that the resolution in dpi be divisible by the screen reduction factor. Thus screen reduction factors of 2, 3, and 4 require a resolution that is divisible by 12. Neither 80- nor 118-dpi meets this requirement, so only some of the screen reduction factors could be used with them.

For this reason I decided that my first METAFONT project would be the production of computer modern fonts designed for use with PC screens and usable with Maxview at all screen reduction factors. A standard manuscript page normally has 1-inch margins and thus has text 6½ inches wide. Many different PC screen adapters display 640 pixels in the horizontal direction, so a font with 98 dpi would just about nicely fill the width of the screen. Actually it is desirable to see a small distance beyond the right margin in order to see if there are overfull horizontal lines, so you can take care of that and the Maxview requirements nicely with 96-dpi fonts. Similarly, a Hercules screen displays 720 pixels in the horizontal direction, so font resolution should not exceed 110 dpi and thus 108 dpi works nicely.

Hence my first METAFONT project became the production of a full set of computer modern fonts at 96 and 108 dpi for use with Maxview and in all the standard magnifications. This whole process was easily automated. I now have a bėta copy of Maxview that reads the new PK formats and can look in both the old and the new directory structures. These new fonts look surprisingly good at screen reduction factors of 1 on the two types of screen pixel format. I can now also see the outline of a full page using the new fonts and a screen reduction factor of 3. This is an example of the new freedom that METAFONT brings—you now don't have to rely on the font decisions of others.

### Font Examples

In general, the new computer modern fonts differ from the almost modern ones only in rather subtle ways. Nevertheless, I find these fonts generally more pleasing to the eye. These comparisons cannot be quantified. Previously, I preferred the internal LaserWriter Times Roman fonts to the almost modern Roman ones used by TeX. Now, however, I find the Times Roman and the computer modern Roman fonts to be about equally attractive. I can-

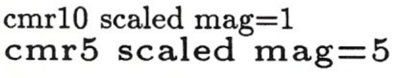

cmr10 scaled mag=1
cmr5 scaled mag=5

Figure 1

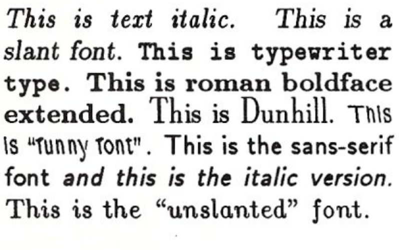

*This is text italic.* *This is a slant font.* This is typewriter type. **This is roman boldface extended.** This is Dunhill. This is "funny font". This is the sans-serif font *and this is the italic version.* This is the "unslanted" font.

Figure 2

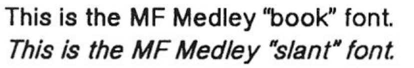

This is the MF Medley "book" font.
*This is the MF Medley "slant" font.*

Figure 3

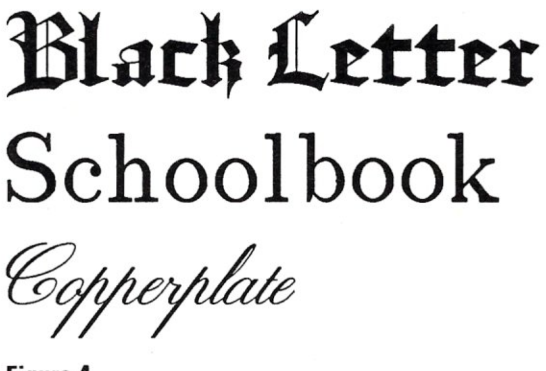

Black Letter
Schoolbook
Copperplate

Figure 4

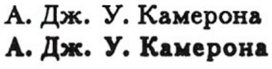

А. Дж. У. Камерона
**А. Дж. У. Камерона**

Figure 5

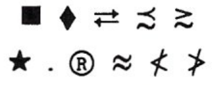

■ ♦ ⇄ ≾ ≳
★ . ® ≈ ≮ ≯

Figure 6

not be specific about why this is. This is why font design is really an art, and Donald Knuth's artistry has clearly improved with the passage of time.

In the LaserWriter the internal fonts are stored as outlines that are scaled and filled with dots as needed in the printing process. Thus the letter shapes are invariant to the magnification process. Computer modern fonts are different in this respect. The METAFONT parameters generally change slightly with changes in design size, so the smaller fonts tend to be relatively wider. This is illustrated in Figure 1, which compares computer modern Roman with a design size of 10 points with the same font at a design size of 5 points but scaled up to nearly the same actual size.

There are 75 fonts in the standard computer modern distribution. The 75 fonts include more varieties both of font families and within such families than did the am fonts. Figure 2 shows some examples.

In addition to Donald Knuth's standard TeX fonts, several independently designed fonts have been produced using the older METAFONT. I will show two sets of these here. First, the MF Medley fonts distributed by Personal TeX ($100). Most of these are sans serif fonts designed for 10 or 12 points and not available in magnification steps. Figure 3 shows the two cases that correspond to the examples of sans serif fonts shown in Figure 2. Figure 4 shows the 36 point fonts available in the MF Medley series; these consist of Black Letter, the only Gothic font available from anyone, Schoolbook (also available at 48 points), and Copperplate (also available at 60 points). MF Medley also contains sans serif math fonts.

The American Mathematical Society has also produced a series of fonts for general mathematical use and for abstracting purposes. The AMS owns the copyright to the TeX logo and is heavily involved in the use of TeX. It has a Cyrillic font in a variety of styles. Figure 5 shows how the Russians spell my name in two of these styles. The math fonts are two in number; they contain various symbols that Donald Knuth left out of his math fonts. Figure 6 shows some symbols that I find useful.

Although these examples show that a rich variety of fonts are now available for use with TeX, the font proliferation will continue for a while. Personal TeX and Bitstream have just announced an agreement whereby many of the great variety of Bitstream fonts will be provided for use with TeX. These fonts are somewhat like those internal to the LaserWriter, in that they consist of outline shapes that are scaled to size and filled with dots. Bitstream states that the strength of its FontWare software lies in its ability to choose pleasing pixel patterns to represent the resulting shapes. I understand that it will be possible to do this outline filling on the fly, thus allowing much more compact disk storage than is presently possible with TeX fonts. This will be an interesting development. §

*A. G. W. Cameron is professor of astronomy at the Harvard-Smithsonian Center for Astrophysics.*

## Addresses

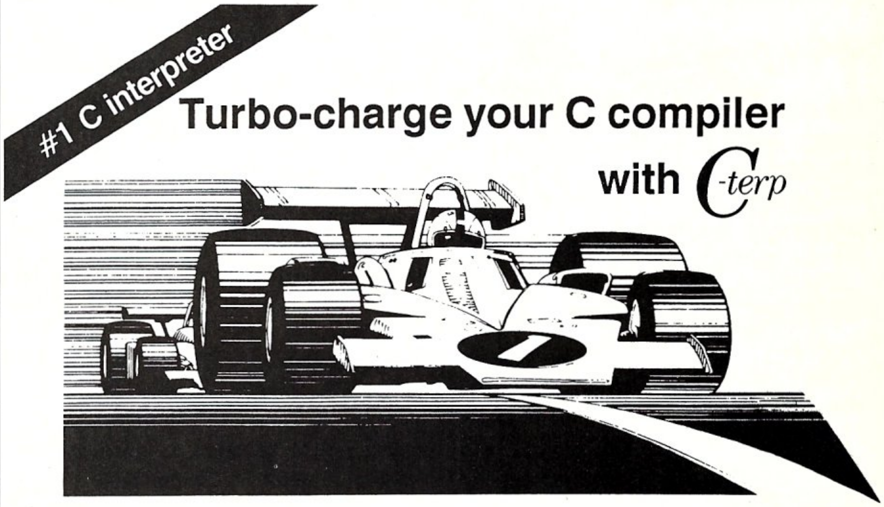
**Addison-Wesley Publishing Co.**, 1 Jacob Way, Reading, MA 01867; (617) 944-3700.

**American Mathematical Society**, 201 Charles St., Providence, RI 02904; (401) 272-9500.

**ArborText Inc.**, 416 Fourth St., P. O. Box 7993, Ann Arbor, MI 48107; (313) 996-3566.

**Bitstream Inc.**, Athaeneum House, 215 First St., Cambridge, MA 02142; (617) 497-6222.

**Personal TeX Inc.**, 12 Madrone Ave., Mill Valley, CA 94941; (415) 388-8853.



# LANScape

B. J. Hall and Michael Cherry

## Part III—LAN Topology: Interconnecting Devices on the LAN

In the last issue of LANScape, we looked at the various media used to connect devices into a local-area network (LAN). Now that you know about LAN cabling, let's look at the ways in which the devices are interconnected.

### Topologies

The description of the physical connections of a network is referred to as the topology. The three most common topologies for a LAN are the bus, the star, and the ring. There are two other LAN topologies, but to a certain extent, they are variations of the previous three. These are the tree and the series of stars.

### The Bus

In the bus topology, all the devices on the LAN are connected to a single cable, which is referred to as a bus. The cable is terminated at both ends, as shown in Figure 1. The advantage of the bus is that all the devices are attached to a single cable, which makes the wiring easy. The disadvantage is that all the devices must share the cable, and therefore, there has to be rules about who uses the cable and when. We'll discuss these rules, or protocols, later in this article.

### The Star

In the star topology (Figure 2), all the devices on the LAN are connected by their own cables to a centralized device, or server. This is typical of the topology used by mini or mainframe computers.

The advantage of the star configuration is that no two devices share the same cable or path into the centralized device, and so the failure of any one device only impacts that single device. The disadvantage is that the centralized device can become a communications bottleneck.

### The Ring

In the ring topology, each device is connected to two neighboring devices, forming a ring. This is shown in Figure 3.

The advantage of the ring is that it can be easy to wire and can conserve the amount of cable needed. The disadvantage is that the failure of a single device can break the ring and therefore stop communications to the other devices.

### The Tree

The tree (Figure 4) can be viewed as an extension of the bus. In the tree topology, the main bus forms the trunk, the buses attached to the trunk are the branches.

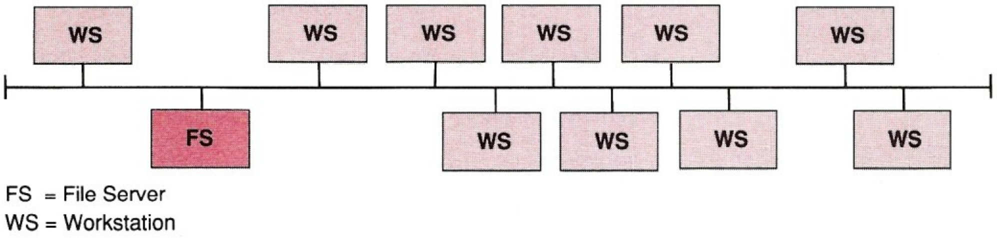


Figure 1. Bus Topology

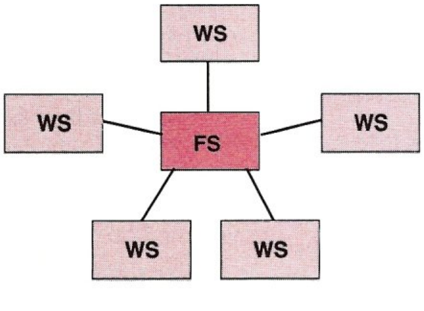


Figure 2. Star Topology

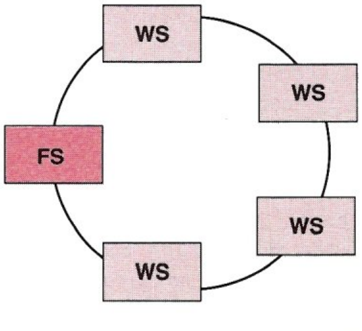


Figure 3. Ring Tolology

| LAN | Topology | Protocol | IEEE Standard |
|---|---|---|---|
| AT&T STARLAN | Star / Bus | CSMA | 802.3 |
| ARCNET | Star | Token | —— |
| Ethernet | Bus | CSMA | 802.3 |
| IBM PC NET | Tree | CSMA | —— |
| IBM Token-Ring | Star / Ring | Token | 802.5 |
| PRONET 10 | Star / Ring | Token | —— |
| PRONET 4 | Star / Ring | Token | 802.5 |
| S-NET | Star | Polled | —— |

**Table 1.**

| LAN Success Factor | Token | CSMA / CD |
|---|---|---|
| Performance | ■ | ■ |
| Expandability | ■ | ▨ |
| Maintainability | ■ | ▨ |
| Reliability | ■ | ■ |

■ = Excellent ▨ = Good

**Table 2.**

### The Series of Stars

In the series of stars topology (Figure 5), the devices radiate from a hub, and the hubs are connected.

### The Token-Ring

There is one other aspect of topology that we should examine before we look into how devices share the common cable. Sometimes, a look at the simple configuration of the LAN does not tell you what the topology is—for example, the IBM Token-Ring LAN appears to be a star topology, but actually it is a ring.

The IBM Token-Ring is wired around a hub called a multiple access unit, or MAU. From the MAU, the attached devices appear to be forming a star, and the interconnection of the MAUs creates a series of stars. Using the IBM Type 1 cable, four wires within a single jacket run from the MAU to the network interface card (NIC) of the attached device. These four wires allow for two wires to send signals from the MAU to the NIC and for the remaining two wires to connect between the NIC and the MAU. This wiring creates a ring within the insulating sheath of the cable. This configuration is sometimes referred to as a physical star but is actually a logical ring topology.

If any one device fails, the MAU can reconfigure the LAN, eliminating the failed device from the ring. As shown in Figure 6, the wiring between the NICs and the MAU create a ring. When other MAUs are connected or other devices are added, the physical ring is reconfigured and grows in circumference.

## Protocols

In examining the Bus topology, we indicated that a means of controlling when devices use the cable is necessary. The rules for control are called protocols, and they can be divided into two types—contention and deterministic.

### Contention Protocols

With a contention protocol, all devices on the LAN contend for use of the cable. The most common contention protocol is the CSMA/CD technique.

CSMA/CD stands for Carrier Sense Multiple Access/Collision Detection. Its operation is simple. When a device needs to send a message, it first listens to see if

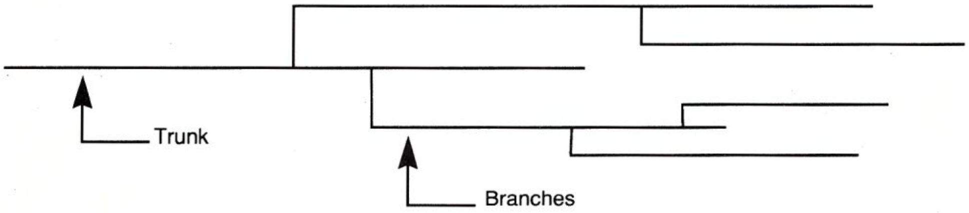


**Figure 4. The Tree Tolology**

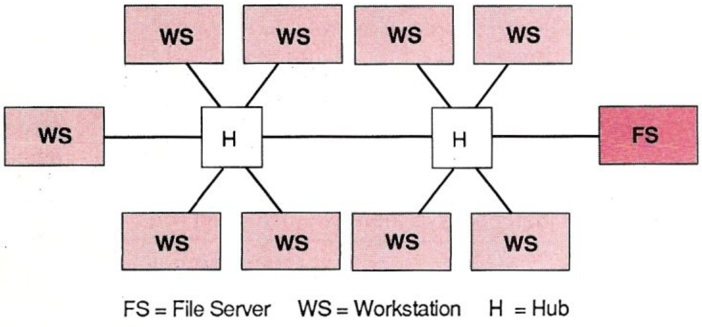


**Figure 5. The Stars Tolology**

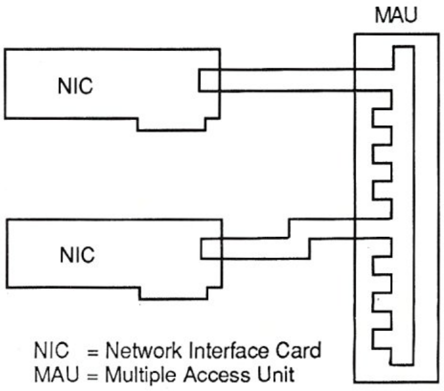


**Figure 6. NIC/MAU Interconnections**

any other device is transmitting. If all is quiet, the device transmits its message. If no other device attempts to transmit at the same time, then the device on the network for which the message was intended receives it. If two devices send at the same time, a collision occurs and the two signals are destroyed. Both sending devices detect the collision and both begin a random number count. Each device on the network has its own unique number for such an occurrence. When a device reaches its random number, it again listens, and if no other device is transmitting, it retries its transmission.

The down side of this protocol is that while the two devices whose messages collided are counting to their random numbers, other devices on the network are listening, and if they hear silence, they will attempt to transmit their messages.

The carrier sense multiple access protocol is the basis for the Ethernet style of LAN.

### Deterministic Protocols

Essentially, there are two types of deterministic protocols. The first is the polled technique, in which each device is polled, or called, in turn and asked if it has any signals to transmit. The device only speaks when spoken to by the polling host. The polling technique is the basis for the Novell Star-Net (S-Net) style of LAN.

The second technique, which is growing in usage, is the token protocol. With this protocol a special signal, called a token, is placed on the network. The simple rule is that whichever device has the token can use the network. When a device needs to send a message to another device, it waits for the free token to come to it. When it receives the free token, it attaches its message to the token and sends it off. The token, with the message attached to it, continues along through the network. When the device to which the message is being sent receives the token, it takes the message off the token and regenerates a free token.

### Comparing Topologies and Protocols

Table 1 summarizes topologies and protocols of some common LANs in use today.

In Part I, we discussed four elements of LAN success. How do the factors of LAN success relate to the topologies and protocols that are in use for LANs? Let's first look at the protocols, which have essentially been divided into two camps—CSMA/CD and token.

The pro and con groups for both these protocols are fairly evenly split. The pro-token camp will tell you that with too many devices on a bus, the number of collisions that occur increases to the point at which response time to all the devices deteriorates. The CSMA/CD camp will tell you that the time spent waiting for a free token is wasteful and that, again, response time increases.

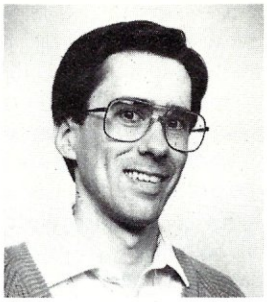
The tendency that we at HallComm NetWork Services see is an increase in token-based systems, although both protocols appear to work well in the field. Some of the token ring's popularity may in fact come from the endorsement of IBM. Table 2 summarizes HallComm's opinion of the two protocols.

In conclusion, then, both protocols are well suited for LANs. Often, in addition, the topology, and the type of cable that will be used, is based on the selection of the protocol. §

*B.J. Hall and Michael Cherry operate HallComm Network Services, a company devoted exclusively to designing and implementing LAN systems. HNS is located at 8101 E. Prentice Ave., Ste. 304, Englewood, CO 80111; (303) 770-6387.*

# The UNIX File

Ian F. Darwin



## Awk—A General-Purpose Language

*This column discusses the UNIX operating system. If you have comments or questions about UNIX or this column, please write to Ian Darwin at Box 603, Station F, Toronto, Ontario, Canada M4Y 2L8. If you have UNIX mail access to the uucp network, mail "ihnp4!darwin!ian." Internet users should try "ian@darwin.uucp" or "ian@sq.com." I can't always answer immediately, but I will get back to you; electronic mail gets answered first!*

Many personal computer users get started in programming with BASIC. About the only things to be said for BASIC are that it is an interpretive language and that it's widely available. But UNIX users have a widely available interpretive language, too. It's called *awk*, named after its authors: Aho, Weinberger, and Kernighan. *Awk* is tremendously useful for prototyping in the UNIX environment because it is interpretive, because its syntax is patterned after the C programming language, and because it can be used as a filter. But the similarities to C are sometimes deceiving. There are no declarations—beware of name conflicts! The input language is free form—you should use the same programming style as you do in C. Comments are allowed, but they begin with #—use them as plentifully as you would in C.

You always have to tell *awk* what function to perform on each input line or record. In its simplest invocation, *awk* behaves much like *cat*—that is, it just prints its input:

```
awk '{print}' myfile
```

prints all the lines in *myfile*. A similar form behaves like *grep*—that is, it prints all lines with a given pattern:

```
awk '/fred/' myfile
```

prints all lines with the string *fred* anywhere in file *myfile*.

*Awk* is field-oriented—input lines or records are broken into words (or fields) as each line is read. As is usual in interpretive languages, names for commonly used variables are short. The names *$1*, *$2*, *$3*, and so on refer to individual fields within the current record. So:

```
awk '{print $1}'
```

prints the first word of each line in the file.

Here are the command line options of *awk*:

```
awk ['program' ¦ -f pgmfile] [-Fchar] file . . .
```

There must be a *'program'*, either on the command line or in the file named by *-f* file. As you have seen, a 'program' can be as little as the statement *'{print}'*; on the other hand, *awk* programs can easily grow to several pages or more in length.

The fields are normally delimited by white space; you can change this with *-F*. A common use of *awk* for system administrators is to print the user IDs listed in the system password file:

```
awk -F: '{print $1}' /etc/passwd
```

lists just the log-in names, whereas:

```
awk -F: '{print $1, $5}' /etc/passwd
```

prints both the log-ins and the accounting field entries.

Let's look briefly at the way *awk* breaks records into fields. If there is no *-F* option, the field separator is any white-space character (tab, blank, and so on), and multiple spaces separate one field. But if there is a field-separator character given with *-F*, then multiple *-F* characters in a row result in multiple fields. For example, by default:

```
word1bbword2
```

(where *b* is a blank) is treated as two fields—*word1* and *word2*. But if a field separator of # is specified with *-F'#'*:

```
words1##word2
```

*Awk* would treat this line as if it were composed of three fields—*word1*, a null field, and *word2*.

A program can be preceded by a condition, in which case the program will be run only if or when the condition is true. One type of condition is a pattern match on a particular input field; the program will be run if the pattern matches. Like most UNIX tools that match patterns, *awk* uses a pattern syntax called "regular expressions" (if you aren't familiar with regular expressions, see the manual page on the UNIX text editor for an introduction). For example, to look for *the* or *The* in the first word of a line, use this:

```
$1 ~ /[Tt]he/ {print}
```

Another type of condition is *awk*'s special pseudopatterns. In the following, the first and last program section have special patterns. This program prints "about to start" before it does anything, then prints the first field of each input, then prints "all done":

```
BEGIN   {print "about to start"}
        {print "in record, field1 = ", $1}
END     {print "all done"}
```

There are also some special (read-only, or nonsettable) variables. *NR* refers to the number of the current record (its line number). *NF* contains the number of fields in the current record. These can be used in conditions. One common use of *awk* is as a simple data validator, such as this quickie program to check that all input records have the right number of fields:

```
awk 'NF != 8 { print
```

### Listing 1

```
# mkpath - function to make all components needed for a path
func
mkpath(dr)
{
        n=split(dr,dircomps,"/")
        for (i=1; i<=n; i++) {
                mkdir = mkdir dircomps[i]
                print "mkdir", mkdir
                mkdir = mkdir "/"
        }
}

# main action - build directories and get file.
NF == 3 {
        ...
        dir=substr($3,1,n)      # up to but not incl last "/"
        if (madedir[dir] == 0) {
                mkpath(dir)
                madedir[dir]=1
        }
        print "sccs get", "$SCCSROOT" "/" FROM
}
```

```
    \"Wrong number of fields on line", NR }'
```

*Awk* is a block-structured language with a strong family resemblance to the C language. The *if* statement has a syntax very like that of C:

```
if (expr)
        stmt
[else
        stmt]
```

(where the characters [ and ] mean "is optional"; they are not part of the *awk* language). As an example, to select all records with the wrong number of fields:

```
if ($1 < OLD1)
        print "input out of order, line", NR
else
        OLD1 = $1
```

The *for* statement is also like that of C:

```
for (expr)
        stmt
```

For example, to print all the fields of one record, in a tabular form:

```
for (i=1; i<=NF; i++)
        print "field", i, " = ", $i
```

*Awk*'s string handling is quite flexible. Concatenation is requested just by putting the strings (either literal strings or by name) together:

```
a = "Dear Mr/Mrs" $SURNAME
```

is not quite right; you must insert a space for readability:

```
a = "Dear Mr/Mrs " $SURNAME
```

There is a *substr* function reminiscent of the PL/1 language:

```
first = substr(a, 1, 5)
rest  = substr(a, 6)
```

puts the first five characters of string *a* into string *first* and the sixth to last characters into string *rest*. Note that string *a* is not affected at all by this operation. The *index* function works as you would expect: it returns the numeric index of one string in another.

Arrays are as flexible as you'd expect in an interpretive language. The arrays do not have to be predeclared; array elements are created as needed:

```
totals[1] = "Fred"
totals[$1] = $2
```

Also, the array indices do not have to be numeric:

```
fish["tuna"] += 35000
```

or more generally:

```
fish[$1] += $2
```

works perfectly well. There is a special *for* syntax to handle this type of associative array:

```
for (index in array)
```

For example, the complete program to total records of the form:

```
tuna     10000
salmon   30000
tuna     45000
salmon   10000
etc.
```

looks like this:

```
        {
        fish[$1] += $2
        }
END     {
        for (i in fish)
                print i, fish[i]
        }
```

This might print:

```
tuna   55000
salmon 40000
etc.
```

The order of retrieval for the elements in an associative array is not defined. But you can always pipe the output of *awk* into *sort*. Say you wish to collapse a list of interests in the form:

```
fred    cars
john    horses
john    dogs
john    cars
mary    dogs
mary    fish
```

into one entry per line:

```
fred cars
john horses dogs cars
mary dogs fish
```

Here is a script to do it, using *awk*'s associative arrays:

```
# collapse.awk -- collapse records with same $1
BEGIN   {
        old1 = -1
        }
        {
        if ($1 == old1)
                printf "%s ", $2
        else {
```

```
                    printf "\n"
                    printf  "%s  %s
", $1, $2
          }
          old1 = $1
          }

END       {
          printf "\n"
          }
```

There is a special function for making an array out of a string. *Split* takes as arguments the string you want split up, the name of the new array, and a set of delimiter characters. For example:

```
n   =   split("/u/ian/course/
assgt2/data",
    \pathcomps, "/")
```

creates an array called *pathcomps*, with five elements numbered from 1 to 5, with values *u*, *ian*, *course*, and so on.

One common application of *awk* is to generate tables. Here is a table of common to metric temperature conversions. And here is a trick: *When there is no need for input, put the action in the* BEGIN *rule and process a null input file.*

```
awk '
BEGIN     {
          for (i=5; i<25; i+=5)
                    print i,((i*9)/
5)+32
          }
' /dev/null
```

*Awk* has been used for many other applications—for example, a full microcomputer cross-assembler has been written in *awk*. Several UNIX sites have their complete usage accounting systems written in *awk*. At SoftQuad we have a typesetting simulator—an *awk* script that adjusts our output files for different device resolutions. Several specialized *troff* preprocessors have been written in *awk*; the best known is *chem*, a program that takes written descriptions of chemical formulas and uses the typesetting system to draw symbolic forms of the molecular structure. There is even an *awk*-to-C translator—it will be described in a forthcoming book by Kernighan, Aho, and Weinberger.

Brian Kernighan and others at Bell Labs Computer Science Research have continued to enhance *awk* since its first release in 1979 (most UNIX systems in circulation have the V7 UNIX version of *awk* from 1979). One of the most valuable of the recent additions is user-definable functions. For example, Listing 1 is a simplified part of a program maintenance system. It generates a shell script to make all necessary directories and then check out (from SCCS) the latest versions of source files. There are several other enhancements, including access to the *argv[ ]* array. This *awk* is in V8 UNIX (not generally available) and in System V, Release 3.1, and one can hope that it will be included in a future release of 4BSD.

There is also an *awk* for MS-DOS. Mortice-Kern Systems wrote and distribute the MKS Toolkit described previously in this column. The toolkit includes a full implementation of *awk* that is compatible with the current (V8 Research) version.

Will there ever be a public-domain *awk*? Maybe—one part of the GNU project is to reimplement all the standard UNIX commands. And the developers claim to have a working *awk* ready for beta testing, but I haven't seen it. What is certain is that *awk* will continue to be used on UNIX for years to come, providing useful service as the standard interpretive language. §

*Ian Darwin lives north of Toronto with his wife and their new son, Benjamin. Ian is Director of Research and Development for the Toronto-based firm SoftQuad Inc., maker of quality publishing software.*

# Lahey Computer Systems, Inc. Sets a New FORTRAN Standard!

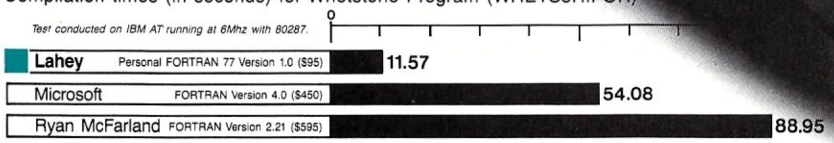
Introducing the latest addition to our line of FORTRAN Language Systems—
Lahey Personal FORTRAN 77.

## What you Get With Lahey Personal FORTRAN 77:

### Lahey Experience.

We are experts in designing and implementing FORTRAN Language Systems. Lahey has been producing mainframe implementations since 1967 and a PC FORTRAN (F77L) since 1984. In fact, F77L was named the "EDITOR'S CHOICE" among PC FORTRANs by *PC Magazine*. This 20 year span of specialization has been incorporated into the design of our revolutionary Lahey Personal FORTRAN 77.

**LAHEY SLASHES COMPILATION TIME.**
Compilation times (in seconds) for Whetstone Program (WHETS3H.FOR)



## Customer Support:

Our philosophy is that customer relationships begin, rather than end, at the point of sale. Services include free technical support, electronic bulletin board for fast service and information access, and newsletters to keep you up to date.

Purchasing the Lahey Personal FORTRAN 77 gives you our FORTRAN experience, a feature loaded product, industry leading compilation speed, and quality technical support; all for **$95.**

**International Representatives:** Canada: Barry Mooney & Assoc., Tel. (902)6652941 • England: Grey Matter Ltd., Tel. (0364)53499 • Switzerland: DST Comp. Services, Tel. (022)989188 • Denmark: Ravenholm Computing, Tel. (02)887249 • Australia: Comp. Transitions, Tel. (03)5372786 • Japan: Microsoftware Inc., Tel. (03)813822

MS-DOS & MS FORTRAN are trademarks of Microsoft Corporation.

## Feature Loaded:

- Full implementation of the ANSI X3.9-1978 FORTRAN Standard
- Fast Compilation (see chart)
- Popular Language Extensions highlighted in the manual
- Source On-Line Debugger
- English Diagnostics and Warning Messages
- LOGICAL*1, LOGICAL*4
- INTEGER*2, INTEGER*4
- REAL*4, REAL*8, and DOUBLE PRECISION
- COMPLEX*8, COMPLEX*16
- Recursion
- 31-Character Names
- Trailing Comment
- Cross Reference and Source Listings
- 64 KB Generated Code
- 64 KB Stack Storage
- 64 KB Commons, Constants and Saved Local Data
- Math coprocessor requirement gives maximum performance
- 350 Page User Manual

SYSTEM REQUIREMENTS:
256K Ram MS-DOS (2.0 or later)
Math Coprocessor Chip (8087 or 80287)

# $95

Lahey is setting the
PC FORTRAN Standard.
TO ORDER

**1-800-548-4778**

Lahey Computer Systems, Inc.
P.O. Box 6091
Incline Village, NV 89450
Telephone: (702) 831-2500
TELEX: 9102401256

**If you want to program in FORTRAN - Lahey is the Source**

□ Please send me ___ Lahey Personal FORTRAN 77 copies @ $95.00 each
Disk Format ___ 5¼″ ___ 3½″ ______
Add $2.50 for shipping and handling per unit ______
Nevada residents add applicable tax ______
TOTAL ENCLOSED ______

SEND ME INFORMATION:
□ Lahey Personal FORTRAN 77
□ A future release not requiring math coprocessor
□ LCS Software Products

ENCLOSED
__ Check __ Visa __ Master Card __ AMEX
Number ______
Expiration Date ______
Signature ______
Name ______
Company ______
Address ______
City ______
State ______ Zip ______
Phone ______
Send to: Lahey Computer Systems,
POB 6091, Incline Village, NV 89450-6091

Lahey
Computer Systems Inc.

# The Public Domain Software Forum

Charles H. Strom

This column is being brought to you courtesy of WordStar Professional, Version 4.0. Surprising as it may seem, I think the sleeping giant may have finally been awakened! Because it is fair to expect a plethora of reviews of the new WordStar, I will resist the temptation and jump right into an excellent shareware offering called EZFORMS.

## EZFORMS

As the name implies, EZFORMS is a forms-generating program. It is published as shareware by EZX Corp., 203 NASA 1 East, Webster, TX 77598; (713) 488-0210. Version D.13, the latest, is written in Turbo Pascal. Basically, the program allows users to create forms on the screen for subsequent output to a printer. The distribution also contains several sample forms that you can modify using the program. In addition, users can fill in the blanks, generating a completed form for printout. Forms can be as large as 126 lines by 255 columns but are stored on disk in a compressed format. The program is, however, more of a forms designer than a data entry program, as the data entry mode is a bit on the clumsy side. EZFORMS sports a 1-2-3-like menu interface that is comfortable to use, and there is a complete on-line manual that provides context-sensitive help every step of the way. This is fortunate, for the hard-copy manual offered with registration is terse in the extreme and leaves a lot to the imagination.

The shareware version of EZFORMS is identical to the registered version (available for $49.95) with one significant exception—the shareware version offers support only for dumb, nongraphics printers, whereas the registered version handles the Epson, HP Thinkjet and Laserjet (with Y cartridge), IBM Graphics, Toshiba P351, Prowriter M8510M, C.Itoh Starwriter, and Oki 92A printers. The printer support offers use of the graphics characters for line drawing, shading, and so on. In addition, the registered version includes many additional blank forms for direct use or modification by users.

EZFORMS is available as shareware through General Electric's GEnie IBM RoundTable or the Yellow Rose bulletin-board system ((713) 326-2999). I guess the best testimonial I can give to this program is that I purchased it. Given the volume of software at my disposal, I am very choosy about shelling out my hard-earned funds for such purposes!

## NSWEEP

Most of my fellow refugees from CP/M know Dave Rand and his famous disk file manager called NSWEEP, or NEWSWEEP. Dave has joined us along the inevitable path to MS-DOS, and I am pleased to see that he has just released a DOS version of his classic. The current incarnation is called NSWP3 and is available in the archive NSWPPC18.ARC as of this writing.

I would guess that the majority of experienced MS-DOS users have a disk file manager in their arsenal of utilities. There are literally dozens of these, in commercial, shareware, and public-domain forms. My personal favorite is VFILER, a translation of a program originally written by Rich Conn for use under ZCPR2, later translated into 8086 under CP/M-86 by Harry Van Tassell and still later converted to MS-DOS. VFILER is elegant, compact, fast, and screen-oriented. On the down side, it is limited in that it does not have features such as file moves, squeezing/unsqueezing, archive manipulation, and so on. NSWEEP addresses many of these shortcomings. Highlights include the ability to easily go up or down the directory tree; create, rename, and delete a directory; jump to a particular file; and zoom in on an archive. The latter feature is especially handy—the member files of the ARC are treated just as if they were stand-alone, permitting copying, viewing, printing, and so on, and the extraction occurs transparently. You can also perform a series of "mass operations" on groups of files, which you select by "tagging" them. Mass copying, setting file attributes, squeezing, and so on are features available in NSWEEP.

I have used the DOS NSWEEP but briefly and am already addicted to it. I hope to see more MS-DOS contributions from Dave in the near future. NSWEEP has been released to the public domain and certainly deserves your consideration.

## RGB-Techwriter

My last offering this issue is another newcomer to the software scene—RGB-Techwriter, by Paul A. Basore of BA^EL Software, 13016 Bear Dancer Trail, Albuquerque, NM 87112. As the name implies, RGB-Techwriter is a scientific/technical word processor that depends on a color graphics adapter to display special characters or character attributes. The shareware file, called RGBTEXT.ARC, contains a 44-page manual, the program, and a small selection of printer drivers, including Epson, AT&T 473, DEC LN03, HP Laserjet and Thinkjet, IBM Graphics and Proprinter, Oki 92/192, Texas Instruments 855, and Toshiba P341. The thorough documentation explains how to adapt the program for other printers, so using it should not be a problem for people armed with a technical description of their printer's control codes.

The basic word-processing functions are quite complete. The cursor pad along with the control keys are used for rapid movement within the document, there are toggles for insert/overtype, tab stops are easily set, and so on. Function keys invoke search, search/replace, block move or copy, and more. Specialized technically oriented functions include boldface, superscripts/subscripts, an alternate character set (defined by your printer setup file), and insertion of extended ASCII codes. These latter special character attributes are flagged in intense white, magenta, blue, and red, respectively. A useful feature is equation mode, which treats equations as units that are never split between pages, relaxes line length limitations, and right-justifies equation numbers.

RGB-Techwriter is efficient, extremely fast, and well planned. It is a viable alternative to those programs that use complex strings of control characters to represent special character attributes (à la WordStar). I prefer the WYSIWYG (what you see is what you get) display of Greek characters and mathematical symbols available with some of the more expensive technical products, but the undeniable beauty of RGB-Techwriter is the low shareware asking price of only $20. Best of all, you can try before you buy. §

*Charles Strom was bitten by the micro bug in 1977. He is an avid user of both MS-DOS and CP/M systems, with a particular interest in public-domain software. He is the author of numerous review articles and a sysop on the GEnie national time-sharing service.*

The PC/Blue disks are available from the New York Amateur Computer Club Inc., Box 106, Church Street Station, New York, NY 10008, for $7 per volume; foreign orders are an additional $2 per volume.

## New PC/Blue Releases

The following are the most recent releases in the PC/Blue library of public-domain MS-DOS software. I regret to report that SIG/M has no new releases.

**Volumes 277-278**
Mr. Bill, Legal Time,
and Billing,
Version 3.12

**Volume 279**
Classical Classifier outline
and text processor
Label Master
Finditem string finder

**Volume 280**
Commando disk manager
MasterKey disk catalog program
Mapantoc document processor

**Volume 281**
AMTAX86 tax preparation
TAX87 1987 income-tax projection

**Volume 282**
Miscellaneous games
Intercept (CGA)
Flightmare
Monopoly, Version 6.2
Original Adventure

**Volume 283**
BetterWay Calc program
I-Ching, Version 2.1

**Volume 284**
Bible-Q quiz program
Stock market simulator

**Volume 285**
Genealogy on Display,
Version 5

**Volume 286**
Other text editors: Dosedit, Edwin,
Ezedit, Mre, Pro-edit, Qedit,
Version 4.7

**Volume 287**
Miscellaneous screen handling
and subdirectory utilities

**Volume 288**
BBS directory assistance programs

**Volume 289**
Property Manager, Version 2.0
for tenanted real estate

**Volume 290**
As Easy—1-2-3 clone

**Volume 291**
Desk management utilities
SideKick clones

# New Products

*When contacting vendors, please mention that you read about their products in* Micro/Systems Journal.

*Manufacturers who would like to have their hardware products listed here should send their news releases to The Editor,* Micro/Systems Journal, *P.O. Box 1192, Mountainside, NJ 07092.*

## PC-COMPATIBLE HARDWARE

### Novell-Compatible LAN Card

CompuPro has announced the ARCNET-PC card for Novell and other networks using the ARCNET LAN interface. It is designed to work with high-speed PCs, PC/XTs, PC/ATs, and compatible computers. It also supports DRI's DR-NET.

The card costs $550 and is available from CompuPro, 26538 Danti Ct., Hayward, CA 94545; (415) 786-0909.

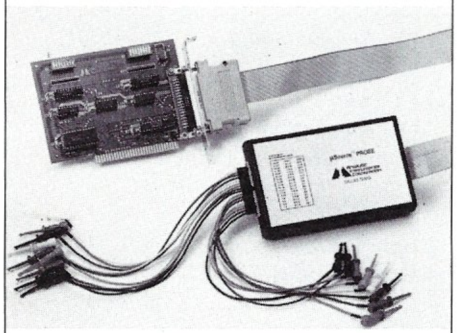
### 16-Channel Modem Card

Galacticomm has released a 16-channel modem card for PC, PC/XT, and PC/AT systems and compatibles. Included is the Galacticomm Breakthrough software, which supports up to 16 simultaneous users communicating at rates up to 1200 baud. The software can handle up to four cards, for a total of up to 64 simultaneous users. The software bypasses the operating system and requires only 128K RAM. It is intended for applications such as multiuser bulletin-board and electronic-mail systems, credit-card verification centers, CD-ROM access nodes, and on-line banking systems. A demo system is accessible via modem at (305) 922-3901.

For more information contact Galacticomm Inc., 11360 Tara Dr., Plantation, FL 33325; (305) 472-9560.

### Digital Pattern Generator/Analyzer/Recorder

The uSource probe, from Analytic Instruments, is a 16-channel (expandable up to 64) digital debug tool for PCs, PC/XTs, PC/ATs, or equivalents. It can produce arbitrary digital patterns up 6.5K long and record digital data input. It includes software to draw test patterns on the screen exactly as they appear at the probe input.



Pricing is $895 (16 channels) or $1,295 (32 channels). It is available from Analytic Instruments Corp., 9995 Monroe Dr., Ste. 205, Dallas, TX 75220; (214) 357-3882.

### Hi-Res 19-Inch MS-Windows Display System

The Viking 1 is a high-resolution display system for PCs, PC/XTs, PC/ATs, and RT/PCs that includes a 19-inch monitor (1,280 × 960 pixels refreshed at 66 Hz), a controller card, and an implementation of Microsoft Windows. The controller also supports programs written to run with CGA, monochrome, or Hercules display controllers.

Ventura Scoop
Shuttle
Xerox Shows Off Ventura Publisher at Conference
Ventura Publisher Edition Redefines Desktop Publishing

The Viking 1 is available from Moniterm, 5740 Green Circle Dr., Minnetonka, MN 55343; (612) 935-4151.

## Other Hardware Products

### 3-D Pointing Device

FastTRAP, from MicroSpeed, provides a trackball (x and y axes) plus a fingerwheel (z axis) and three buttons. It can be used with programs such as AutoCAD, which currently has 3-D capability but requires all z-axis information to be entered via the keyboard. It is compatible with existing mouse software and a software designers' kit is available. Versions are available for PC-compatibles and the Apple Macintosh.

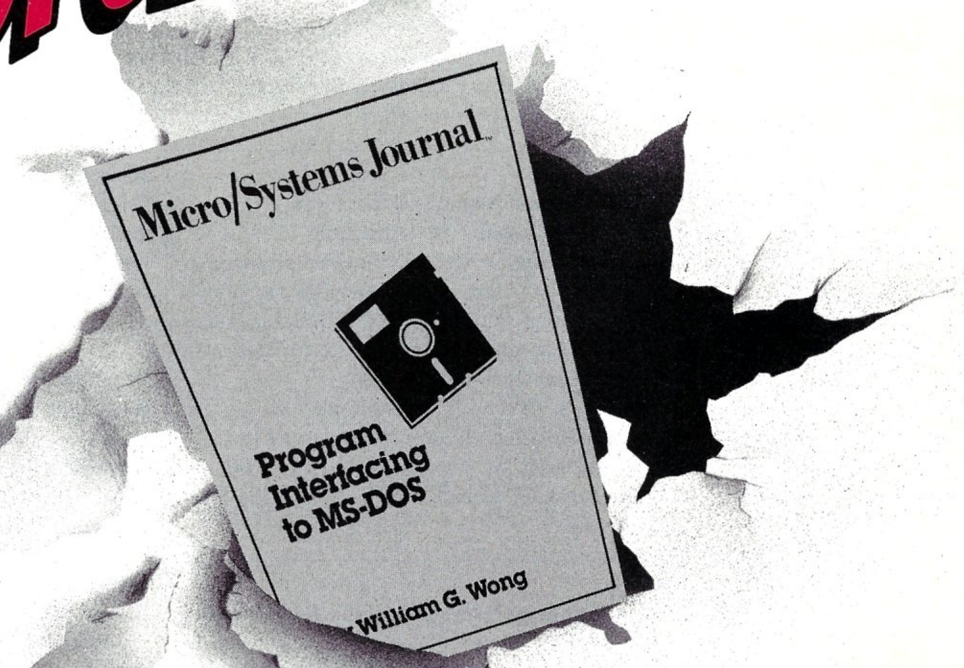
It has a resolution of 200 pulses/inch on x, y and z inputs with adjustable drag, an 18-square inch footprint, and a standard RS-232C interface. The suggested price is $149, and it is available from MicroSpeed Inc., 5307 Randall Pl., Fremont, CA 94538; (415) 490-1403.

### Uninterruptible Power Supply for Video Display Terminals

L/F Technologies has introduced an uninterruptible power supply for video display terminals. Called the VDT Guardian, it fits under the base of a terminal elevating it only two inches. It is designed to work with most popular VDTs and includes a 2.7-ampere-hour battery that provides about 15 minutes of battery backup. When the line voltage drops, it automatically switches to battery operation. When power is restored, the unit switches back to battery charger/maintenance operation.

The price is $149, and it is available from L/F Technologies, 2800 Lockheed Way, Carson City, NV 89701; (702) 883-7611. §

# BACK BY POPULAR DEMAND



Because you asked for it, **Program Interfacing to MS-DOS** was originally featured in *Micro/Systems Journal.* This reprint provides ten concise chapters guaranteed to orient any experienced programmer to the MS-DOS environment.

These articles reprinted together for the first time:

- The Program Segment Prefix, Access to DOS, and How to Terminate a Program
- Dealing with Character Input and Output Functions
- Basic File Access Functions
- Basic File Access
- Programming MS-DOS
- Device Drivers—Why and How
- Device Drivers, Structures
- A Memory Disk Device Driver and a Printer Device Driver (both provided on disk with full source code)
- Why Move to DOS 3.X?

Read in-depth discussions of:

• program construction • character base input and output functions • file access, including CP/M style vs. Unix style DOS file access • other useful functions related to memory allocation and program execution.

**Program Interfacing to MS-DOS** also contains sample program files and a detailed description of how to build device drivers.

The complete ten-part **Program Interfacing to MS-DOS** reprint is now available for only $29.95. It includes the device driver for a memory disk and a printer device driver, both on disk with macro assembly source code.

To order, return this coupon to: M&T Publishing, Inc., 501 Galveston Drive, Redwood City, CA 94063. **Or, CALL TOLL-FREE!**

***Yes!*** Please send me **Program Interfacing to MS-DOS**

for $29.95 ________

Tax (CA only) ________

Shipping, add $2.25 per item ________

TOTAL ________

□ Check Enclosed. **Make payable to M&T Publishing.**

Charge my: □ VISA □ Master Card □ Am Ex

Card # ____________ Exp. Date ________

Name ____________________

Address ____________________

City ________ State ______ Zip ______

34A

*800-533-4372 (IN CA 800-356-2002) ASK FOR PROGRAM INTERFACING TO MS-DOS, ITEM #166.*

# The Software Directory

*When contacting software publishers, please mention that you read about their products in* Micro/Systems Journal.

**Program Name:** CALLME and CALLUS, Version 3.1
**Requirements:** Concurrent DOS-386, Concurrent PC-DOS, and Concurrent CP/M (Digital Research)
**Description:** A remote console supervisor for single-user and multiuser/multitasking operation. Includes a mail facility, privileged access levels to subdirectories and applications, multimessage section, and a script language for creation of custom menus and commands for unattended operation. Designed for remote product support, remote order entry, and bulletin-board applications and more. To try software, call (305) 727-0331 (name = GUEST DEMO; password = CALLME).
**Price:** CALLME (single modem), $275; CALLUS (multimodem), $495.
**Publisher:** Concurrent Research Inc., 1592 Highway A1A, Satellite Beach, FL 32937; (305) 777-7080.

---

**Program Name:** CP/TREE
**Requirements:** CP/M 2.2
**Description:** A CCP replacement that implements a search path for both commands and files. Also provides UNIX-like programs to transform user areas into tree-structured name directories.
**Price:** $29, plus $4 shipping and handling
**Publisher:** Precise Electronics, 486 California St., Newton, MA 02160; (617) 332-3977.

---

**Program Name:** CSharp Realtime Toolkit 3.0
**Requirements:** PC compatible or DEC PDP-11
**Description:** A package of real-time tools for C. Provides interrupt handling, event scheduling, procedure scheduling, device-independent graphics, and state system control. Works with the following C compilers: Lattice 2.15/3.10 and Computer Innovations C-86. Special version is available for use with Rational Systems' Instant-C. A version is also available for the PDP-11 under Whitesmith-C.
**Price:** $495, source license; $495, object-production license (25 units); $2,475, object-production license (no limit).
**Publisher:** Systems Guild, P.O. Box 1085, Kendall Square Station, Cambridge, MA 02142; (617) 451-8479.

---

**Program Name:** Disk Explorer
**Requirements:** PC compatible
**Description:** Recovers accidentally deleted or erased files quickly. Displays and allows changes to contents of disks. Status of files can be displayed and changed. Data can be changed in any sector.
**Price:** $75, U.S.; $105, Canada
**Publisher:** Quaid Software Ltd., 45 Charles St. East, Third Floor, Toronto, Ontario, Canada M4Y 1S2; (416) 961-8243.

---

**Program Name:** MultiLink Advanced and Clustercomm
**Requirements:** PC, PC/XT, PC/AT, or compatible
**Description:** Provides SNA cluster (3270) emulation. Makes a single PC, XT, or AT act as a 3274 communications controller attached to an SNA host. Supports up to 17 logical units (fifteen 3270 screens plus two host printer sessions). Screen sessions operate on ASCII terminals attached via standard RS-232 ports. Printer sessions use standard PC printers (parallel or serial). Printer spooling is also provided. Communication to the host is either SDLC (via SDLC adapter) or X.25 protocol across a packet-switching network (e.g., Tymnet or Telenet). Users can hot key between emulation and DOS.
**Price:** MultiLink Advanced, $595; Clustercomm (two 3270 sessions and one printer), $545.
**Publisher:** The Software Link Inc., 8601 Dunwoody Pl. NE, Ste. 632, Atlanta, GA 30338; (404) 448-5465.

---

**Program Name:** NVRD: Non-Volitile RAM-disk
**Requirements:** PC-DOS/MS-DOS system, hard disk, and EMS or V-EMM board
**Description:** NVRD maintains two copies of its RAM disk—a working copy in expanded memory and a backup copy on hard disk. When a program writes to the RAM disk, NVRD updates both copies. When a program reads from RAM disk, there is no disk access. The RAM disk can be up to 35 megabytes.
**Price:** $49.95 (includes 30-day, money-back offer)
**Publisher:** Fort's Software, P.O. Box 396, Manhattan, KS 66502; (913) 537-2897.

---

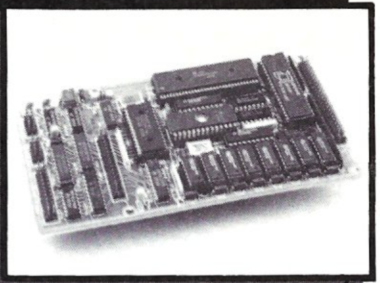
**Program Name:** Plotit
**Requirements:** PC or PC-compatible with MS-DOS 3.1, 640K RAM, hard disk.
**Description:** Interactively integrates sophisticated graphics with powerful statistical analysis. Can analyze data, put analysis in a report format, and produce more than 30 distinct graph types, including three-dimensional, pie, bar, scatter, and histogram charts. This program can also store finished graphs and data sets in a library. Includes on-line help and a tutorial mode. Popular printer and plotters are supported.
**Price:** $550
**Publisher:** Gracon Services, Inc., 4632 Okemos Rd., Okemos, MI 48864; (517) 349-4900.

---

**Program Name:** Turboref, Version 4.0
**Requirements:** PC compatible and Pascal compiler (Turbo Pascal, MS Pascal, or SBB Pascal)
**Description:** Cross-reference and lister utility for Pascal source programs. Features include Pascal control blocks enclosed in boxes, current procedure names indicated for each source line, source file indicated for each line, and more.
**Price:** $49.95
**Publisher:** Gracon Services Inc., 4632 Okemos Rd., Okemos, MI 48864; (517) 349-4900.

## THE ULTIMATE IN COMPACT CP/M COMPATIBLE COMPUTERS

## DSB-8100



***Features:***

- *Hitachi 64180 CPU running at 6MHz (executes a superset of Z80 instruction set)*
- *256K dynamic RAM*
- *8K EPROM with boot / monitor program standard, up to 32K EPROM optional*
- *1773 Floppy controller supports 40 and 80 track 5¼" and 3½" drives*
- *Host / target SCSI port can use DMA for all transfers*
- *Two RS-232 serial ports support asynchronous communications up to 38,400 baud*
- *Centronics type parallel printer port*
- *CP/M 2.2 optional*
- *Power requirements: + 5V at 1.0A*
  *+ 12V at .05A*
- *Size: 6-3/4" x 3-7/8"*

**$365.00** Quantity discounts

Compatible board with 512K RAM and 6 serial ports also available.

Davidge Corporation
P.O. Box 1896
94 Commerce Drive
Buellton, CA 93427
(805) 688-9598

Davidge

COMMAND YOUR MODEM WITH Voyager©

ADVANCED TECHNOLOGY MODEM SOFTWARE

VOYAGER LETS YOU COMMUNICATE . . . EFFECTIVELY!

- Window driven user interface • File Transfer Protocols (Xmodel/CRC/Checksum, Ymodem, Compuserve-B, ASCII) • Terminal Emulation (ANSI-BBS, ADM3A, QVT-102, TV925, VT-52-100, & more)
- VIDTEX (Compuserve) Emulation • Versatile Script Language allows unattended operation • Sample scripts for many services included
- User definable modem configuration • Multi-MODEM Support
- Screen Dumps, Capture Buffer, and many more features . . .

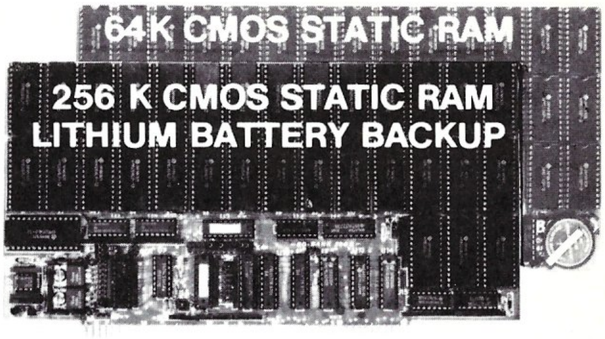
SPECIAL INTRODUCTORY PRICES!

- PC-DOS (IBM PC/XT/AT & 100% Compatible) $ 69.95
- Concurrent PC DOS (w/WINDOWS on SERIAL Terminals written for CDOS native mode!) $139.95
- Demo Disk (includes $5.00 Credit toward purchase) $ 5.00
- Custom Emulation Modules $ CALL

Concurrent Research

1592 Hwy. A1A
Satellite Beach, FL 32937
(305) 777-7080
*Dealer Inquiries Invited*

## HIGH PERFORMANCE RAM



## ✓ COMPARE

| | 8/16 BIT DATA | Bank Selection MPM, OASIS, CROMIX-D | 10 Megahertz Speed | Software Write Protect | Battery Backup Option |
|---|---|---|---|---|---|
| Compupro Ram 22 | ✓ | NO | ✓ | NO | NO |
| Octagon 256K | ✓ | NO | ✓ | NO | NO |
| Cromemco 256KZ II | ✓ | NO | NO | NO | NO |
| Dynamic Boards | ✓ | ✓ | NO | NO | NO |
| **BG-Bank 256S** | ✓ | ✓ | ✓ | ✓ | ✓ |

**GUARANTEED IN YOUR SYSTEM**
**CROMIX-D • MPM • CCS • OASIS • AMOS**

✓**PLUS:** 8/16 BIT TRANSFERS • 24-BIT EX. ADDRESSING
8-12 MHZ • 2K DESELECTS • RAM-EPROM MIX
IEEE 696/S-100 • LOW POWER • FULLY STATIC

LITHIUM BATTERY BACKUP avoids power failure crashes intelligently. Unique POWER-FAIL-SENSE circuit allows processor to save register information and disable board before POWER FAILURE CRASHES memory.

| | | | |
|---|---|---|---|
| **BG BANK 256S........** | **$~~695~~** | **Battery Backup .........** | **$~~70~~** |
| **BG BANK 64S.........** | **$~~290~~** | **Battery Backup ........** | **$~~50~~** |

MEGABYTE MADNESS
FOUR 256S CARDS
~~$2295~~
CALL FOR NEW LOWER PRICES

**BG COMPUTER APPLICATIONS,** 206 Brookside, Bryan, Texas 77801. International orders add 30%.
**(409) 775-5009**

# Classifieds

**Micro/Systems Journal** *accepts Classified Ads. The charge is $6/line (3 lines minimum, 7 lines maximum); 40 characters max./line. Three times frequency $15/line; six times $25/line; non-profit clubs $2/line. Logos, special type, etc. are extra charge. Check must accompany ad copy. Send to M & T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063.*

**POOR MAN'S NETWORK**
A true local area network that lets you share resources between 2 CP/M machines, using standard RS-232 or parallel ports. Works with CP/M 2.2, ZCPRx, Echelon's ZRDOS, Micro Methods' RP/M. Only $69 (U.S.). For info, call (613) 722-0690 Tues-Sat 10AM-5PM or write to Anderson Techno-Products Inc., 947 Richmond Road Dept. S, Ottawa, Ontario K2B 6RI, Canada.

**NO SOURCE CODE?**
REL/PAK converts Microsoft REL files to 8080 and Z80 source code MAC files including symbols under CP/M-80 and TurboDOS. Now $99.95 on 8″ SSSD. Microsmith Computer Technology, POB 1473, Elkhart, IN 46515. (800) 622-4070 or in IL (800) 942-7317.

**NEW**

**FOR YOUR BIGBOARD**

ONE MEGABYTE RAM DISK ON THE STD BUS
Includes: STD adapter pcb & connector, 1 MB RAM pcb & connector and software. Price: $125.00.
INTEGRATED BIOS, reads and writes any 5″ and/or 8″ format.

**INCLUDES:**

- CONFIGuration program that lets you install any new floppy disk format INTERACTIVELY.
- FORMATTER allows you to format almost any diskformat.
- PC-COPY reads and writes PC diskettes on your Bigboard II.
- MONITOR EPROM with serial keyboard and translate table.
- 300 page ZCPR2 manual.
- 60 page Bigboard II tech manual.
- BOOTABLE DISK contains free ZCPR2 and P2DOS system. With TIME and DATE stamping. BIOS also has provisions for 256K RAM disk, Centronics, System in EPROM AND 1 MB RAM DISK.
- WINCHESTER FORMATTER and SYSGEN. Supports XEBEC, W-D and Adaptec type controller. Subdivides into any specified number of drives. Price: $99.95 (specify disk).

**TAKE BOTH 1 MB AND BIOS FOR**
**$199.95**

**ANDY BAKKERS**
de Gervelink 12 ● 7591 DT Denekamp
The Netherlands ● Tel: 31-5413-2488
FIDO Net 500 Node 100. Please pay with
US$ Money Order. MC or VISA welcome.

**IBEX BUS SYSTEMS**, CP/M, 2-1.2MB, 8″ flpy, b/w Mon., Wordstar, Calcstar, Turbo Pascal & more. Like new. Cost $3500. Make offer. John Anderson (805) 496-6408, (213) 336-7573.

**JOIN PC-SMUG.** Get 4 disks of IBM-PC software, quarterly catalogs on disk with free software and other offers. $12 dues for membership thru June 1988. **PC-SMUG**, 39 Hanover, Asheville, NC 28806.

**FILE LISTING PROGRAM** List your source code files with or without line numbers, page breaks, offset, header with file date and time. Adjustable tab settings, wildcard file selection, input & output can be redirected. Will print a pre-listing showing maximum line length after tab expansion. APTECH SYSTEMS INC., P.O. Box 6487, Kent, WA 98064; (206) 631-6679. Program: $25; with C source code: $50.

**DISK CONVERSION**: Most CP/M and MS-DOS formats. 48 hour turnaround. Reasonable rates. Personal service. For information call or write: RH Associates, 2211 Mark Ct., Silver Spring, MD 20910; (301) 587-6230.

**ENGINEERING SOFTWARE**
Circuit Design and Analysis Programs. FREE CATALOG and TUTORIAL GUIDE. BV Engineering, 2200 Business Way, #207, Rverside, CA 92501; (714) 781-0252.

**RAM DISK**
*S-100, 2 MEG, PORT I/O.* New, Warranteed, $725. S. Lugert, 439 Peck Slip Sta., N.Y.C., N.Y. 10272, or call (718) 622-0654.

**DOS User's Group**
Quarterly Newsletter discounts on Third Party Products. Latest DOS news demo diskettes. DOS Tutorials. Annual membership fee: U.S. $25, Canada & Europe $35. DOS User's Group, P.O. Box 26601, Las Vegas, NV 89126.

**DISK SERVICE MANUAL** ($20)
"Printer/Plotter Manual" ($15); "Super Retrieving Method" ($7); "Computer Pbreaking" ($15); "Absolute Computer Security" ($15). 40+ computer & electronics manuals & software, Catalog: $1. By former NMSU CS Professor. CONSUMERTRONICS, 2011 Crescent, Alamogordo, NM 88310.

**FOR SALE:** TurboDOS/S-100 w/4-TV-912 terminals, 90MB micropolis drive, 2-8″ drives + CP/M software. $3,500/offer. (800) 992-0412.

**NEW! 68000 COMPUTERS** w/software $395. 68000 Operating System w/source $50. **HAWTHORNE TECHNOLOGY**, 8836 SE Stark, Portland, OR 97216; (503) 254-2005.

# Advertiser Index

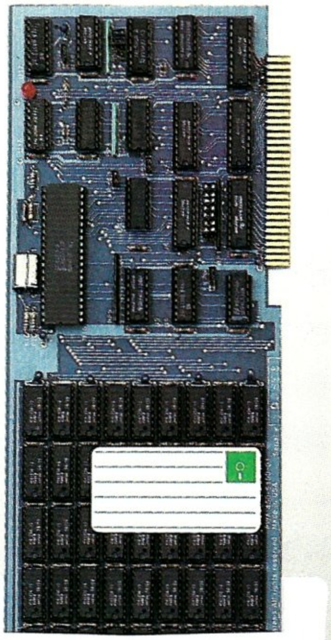
*Subscribers only.

# Think fast! Pick the better fit...

Our 5th Year Bonus!
Mention this ad when ordering
and get your choice of a V 20-8
(replaces 8088) or $20 off on
your Battery Backup purchase!

**FLOPPY DISK.**

- Fills time between coffee breaks
- Makes a hard disk seem *fast*
- Your computer appears busy (even if you aren't!)
- Wears out moving parts



**SEMIDISK Disk Emulator.**

- Gets that job done NOW
- Makes a hard disk seem *slow*
- Maximizes your productivity with anything from databases to compilers
- Totally silent operation

## ...for YOUR demanding tasks.

**SURPRISE!** *Neither* is memory mapped, so they don't affect your precious Main Memory. *Both* retain data indefinitely - even with the computer turned off.

**THE SEMIDISK SOLUTION.** You could invest in a series of "upgrades" that turn out to be expensive band-aids without solving your real problem. Even those "Accelerator" and "Turbo" boards do little to speed up disk-bound computers. If your applications spend too much time reading and writing to disk (and whose don't?), you won't want to settle for anything less than a SemiDisk disk emulator. The SemiDisk comes in 512K and 2Mb capacity. More boards may be added to make up to an 8 Megabyte SemiDrive!

**SPEED THAT'S COMPATIBLE.** PC, XT or AT, if you need speed, the SemiDisk has it. How fast? Recent benchmarks show the SemiDisk is from 2 to 5 times faster than hard disks, and from 25% faster (writing) to several times faster (random reads) than VDISK and other RAMdisk software that gobble up your main memory.

**MEMORY THAT'S STORAGE.** Using our small external power supply, with battery backup, your data remains intact through your longest vacation or even a seven-hour power failure!

**CELEBRATE WITH US!** Now, SemiDisk celebrates its fifth birthday with a special offer for IBM-PC owners. Buy a SemiDisk now and we'll include an 8 MHz V-20 microprocessor (replaces the 8088) to make your new SemiDisk run even faster. Don't need the V-20? We'll take $20 off the price of your Battery Backup Unit!

| | 512K | 2Mbyte |
|---|---|---|
| **IBM, PC, XT, AT** | **$495** | **$ 795** |
| **Epson QX-10** | **$495** | **$ 995** |
| **S-100 SemiDisk II** | **$795** | **$1295** |
| **S-100 SemiDisk I** | **$299** | ------------- |
| **TRS-80 II, 12, 16** | **$495** | **$ 995** |
| **Battery Backup** | **$130** | **$ 130** |

**Someday you'll get a SemiDisk.**
**Until then, you'll just have to...wait.**

***SemiDisk***

MasterCard VISA

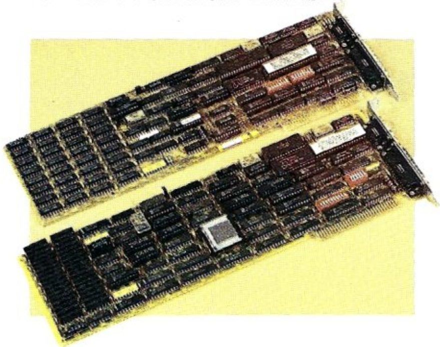
SemiDisk Systems, Inc., 11080 S.W. Allen Blvd., Beaverton, Oregon 97005 (503) 626-3104

# PC-PLUS™ The No-Nonsense Network

**Unlike traditional LANs, PC-PLUS' on-the-bus architecture puts the computing power where you need it — with the data — inside the server.**

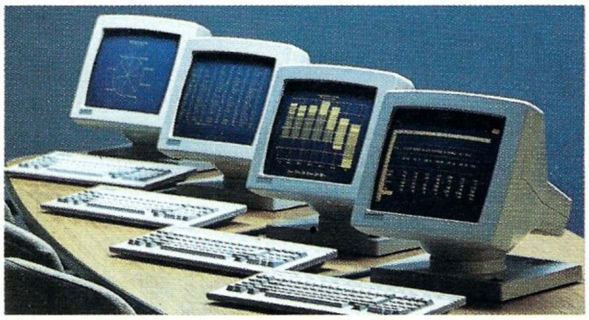
## PC-PLUS: The LAN Engineered for Shared Data

Any local area network can connect PCs. PC-PLUS, however, is engineered to optimize multiuser access to shared databases. We put a PC — or AT — on a card that *plugs into the server.* Putting your computing power on the same bus as your data. Giving you high bandwidth *where you need it.* Without the transmission overhead that steals performance on most LANs.



## PC-PLUS Gives You the Power You Need

Our plug-in cards come as 8 MHz PCs or 8 MHz ATs — to match your computing needs exactly. But the real test is in the results — and our benchmarks are second to none! Power to get your job done. Power to grow into new applications without sacrificing performance.



## PC-PLUS is Easy to Install and Use

We put the computing engines all in one place — in the server. Not spread all over the building. The easiest software installation of any LAN. Use PC-PLUS with your choice of AT compatible servers. Add a card and terminal to grow. Simple RS-232 data cables — no expensive rewiring with coax. You can often use existing telephone cable. Convenient packaged solutions, with preconfigured hardware and software.

## PC-PLUS Runs the Software You Want

Our network operating software runs most popular software — like Lotus 1-2-3 and word processors. And the multiuser software that runs on Novell, IBM and 3COM networks — like dBASE III PLUS w/LANpacks, R:BASE System V, the SMART Series, Revelation.

## PC-PLUS: Find Out Before You Invest

A local area network is a long term investment. Today's applications are just the beginning. Make sure your choice is ready for tomorrow — with mainframe connections, remote access, graphics and the power for growing database applications. Find out more about PC-PLUS today. Ask for our free 24 page brochure **"Choosing the *Right* LAN,"** and get all the facts you need for a wise investment.

ALLOY
Computer Products

Lotus 1-2-3 is a trademark of Lotus Development Corp., Novell is a trademark of Novell, Inc., IBM is a trademark of International Business Machines Corp., 3COM is a trademark of 3COM Corp., dBASE III PLUS is a trademark of Ashton-Tate, R:BASE System V is a trademark of Microrim, the SMART SERIES is a trademark of Innovative Software, Inc., Revelation is a trademark of COSMOS.

**Alloy Computer Products, Inc., 100 Pennsylvania Avenue, Framingham, Massachusetts 01701. (617) 875-6100, TWX: 710-346-0394**
**Alloy Computer Products, Inc., 9 Executive Circle, Suite 240, Irvine, California 92714. (714) 261-7661**