March/April 1987
Vol. 3 No. 2

$3.95
U.S.A.

# Micro/Systems Journal™

*For the Advanced Computer User*


# Multitasking Between Programs

see page 22

*Also in this issue*



*and introducing the column*



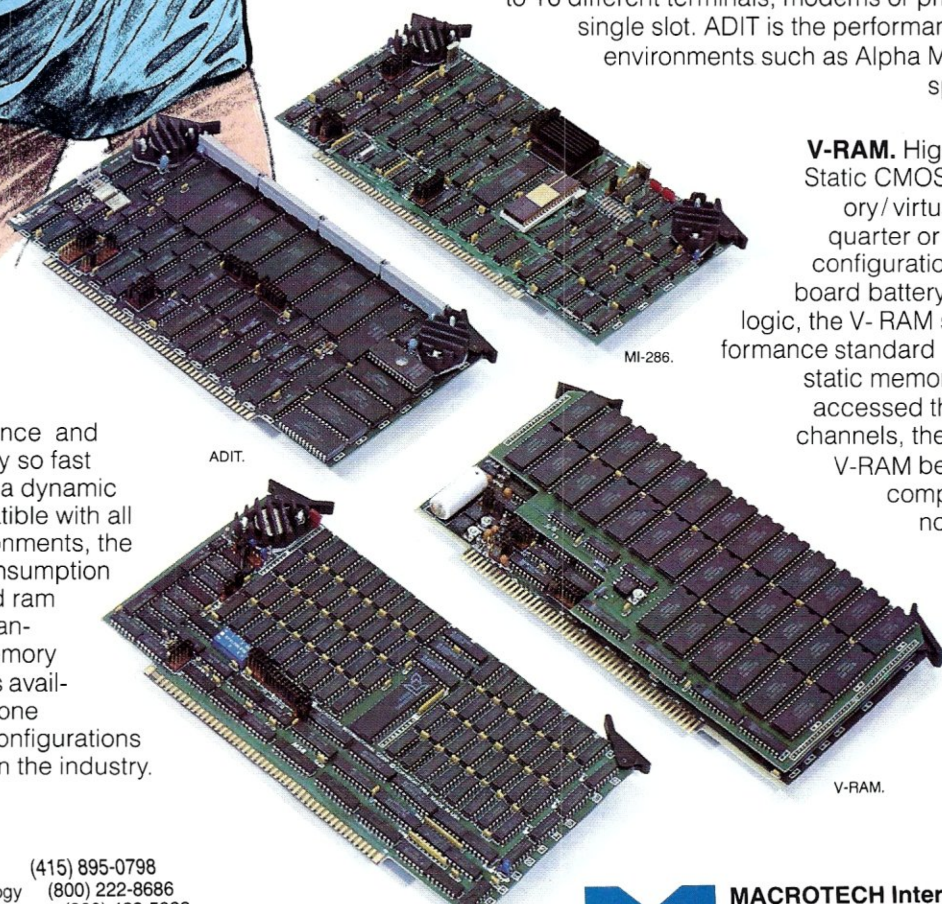
03
38351 19902 9

# 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:**

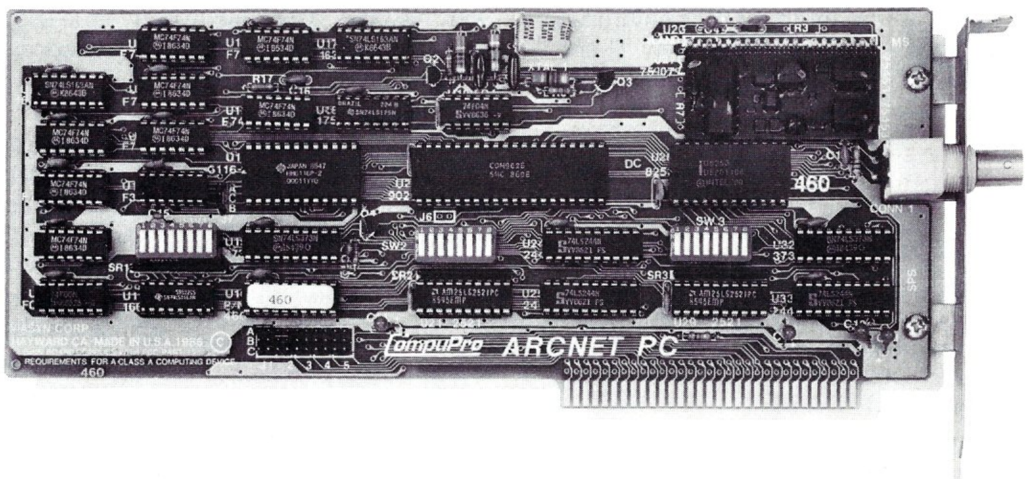
| | |
|---|---|
| Gifford Computer Systems | (415) 895-0798 |
| Custom Computer Technology | (800) 222-8686 |
| S-100 | (800) 423-5922 |
| John D. Owens & Associates | (212)448-6298 |
| In England; Fulcrum (Europe) Ltd. | (0621) 828763 |

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

## The CompuPro ARCNET® PC Board




### *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.

*Micro/Systems Journal* (ISSN #8750-9482) is published bimonthly 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.


*For the Advanced Computer User*

# Micro/Systems Journal™

March/April 1987
Vol. 3 No. 2

# IN THIS ISSUE

## FEATURE ARTICLES



## PRODUCT REVIEWS



## DEPARTMENTS

# Editor's Page

by Sol & Don Libes

## When is Multitasking Not Multitasking?

This issue of *Micro/Systems Journal* features a major article devoted to multitasking on IBM PCs, PC/XTs, PC/ATs, and compatible machines.

A true multitasking system allows more than one task to be in execution at a time. Most multitasking systems allow the user to switch attention from one task to another easily. This is useful for tasks such as editors and spreadsheets, which require the user to interact with them to get anything done.

On the other hand, compilers and assemblers need no such interaction. These and other tasks don't require user input or screens for output. Additionally, they don't require constant attention from the CPU. They can run off of whatever resources are left over while you are doing something at the console. Such tasks are considered to be "background" tasks, since they sit in the background of the computer's attention, getting the short share of its power.

Conversely, editors, command processors, and so forth are considered foreground tasks, since they get priority of CPU resources over the background processes.

In reality, of course, computers tasks are not run concurrently. The CPU is only executing one program at a time. It is possible, however, to switch the CPU between tasks automatically, so that it appears as if tasks are being executed concurrently. For example, while the computer is waiting for you to type the next character, it could also be compiling, or spell checking, or working on lots of background tasks.

MS-DOS was not designed to be a multitasking system. This introduces great inefficiency in what potentially could be an extremely powerful system. UNIX is an example of a more intelligent approach to multitasking. Not only multiuser processes, but also multisystem tasks, can execute at the same time. For example, UNIX device drivers allow asynchronous execution, while MS-DOS drivers are synchronous. That is, with MS-DOS, when you request a sector from the disk, everything in the system stops until the disk responds. Although DOS device drivers are much easier to write than UNIX device drivers, the result is a tragic waste of computer power.

Despite the design of the PC running MS-DOS/PC-DOS as a single-user, single-tasking system, a number of clever folks have found ways to implement multitasking on the systems.

I guess they got just plain tired of waiting for Microsoft to introduce its multitasking version of MS-DOS (which, rumor has it, will run only on 286- and 386-based machines and requires application programs to be written specifically to run in multitasking mode). Moreover, this new version is not expected to be released until late fall, at the earliest (see *News & Views* column).

Such homegrown multitasking systems provide some interesting features and a lot of power, although few of them provide automatic scheduling with their multitasking as does UNIX. And, of course, special hardware support (such as the 80286) is a necessity for a protected environment.

Nevertheless, some of the advertised multitasking options for DOS are quite useful. Some systems allow windows into the background applications so that one can observe their activity. On others, one can switch between *virtual consoles*—in effect, placing a foreground process temporarily in the background, and vice versa.

Digital Research's Concurrent PC-DOS, introduced almost three years ago, was probably the first true multitasking operating environment to become available for the PC.* In it, tasks run concurrently (hence the name), and the screen and keyboard can be switched between executing tasks. Its problem was that early versions provided only a low level of IBM PC compatibility. DRI has since updated the program, and the version in current release has a very high degree of PC compatibility. Concurrent PC-DOS' multitasking is so good that many users use Concurrent DOS as a multiuser system. I have heard of as many as 16 users on a Concurrent DOS system. (A review will appear in next month's *M/SJ*.)

There are a number of commercial programs, and several also in the public domain, that purport to provide multitasking. However, many of them are really "task-swapping" systems. In other words, one can have more than one task in memory, but only the foreground task is actually active. Only the task that has access to the screen and keyboard is running. All other tasks are in a suspended state. Although such a program is not a true multitasker, it is quite useful. These programs generally cost less than true multitasking systems and usually run faster than the latter.

True multitasking can be accomplished on PC/AT-class machines with programs such as DesqView, DoubleDOS, TopView, Windows, and TaskView. Philip Burns shows in his article "Multitasking Between Programs" how to get them to do this with some Turbo Pascal routines.

There are also some programs in the public domain that provide limited multitasking features. A review of these programs appears in Charlie Strom's column on public-domain software.

Hank Kee (former author of the PC/Blue public-domain column) currently is using DesqView on a PC/XT clone to support a two-line electronic bulletin board system that he also uses for word processing and other tasks. He will be describing this use of DesqView in an upcoming issue of *M/SJ*. Hank is also testing an eight-user, commercially available bulletin board system for the PC/XT, and this review will appear in *M/SJ* shortly. In the meantime, people who want to check out his two-user BBS, or the eight-user BBS, can try calling 718-539-3338 or 718-539-3560. §

* That is, if you do not count the various implementations of UNIX for the PC, PC/XT, and PC/AT. Several of these UNIX multitasking, multiuser implementations will be reviewed in the next issue of *M/SJ*.

## Moving?

Affix your present label here.

New address ______

City ______ State ______ Zip ______

Mail entire ad to:

**Micro/Systems Journal**
P.O. Box 3713
Escondido, CA 92025

Please allow 6–8 weeks for address change to take effect.

# News, Views & Gossip

by Sol Libes

## RUMORS & GOSSIP

**Sharp Corp.** is rumored to be readying a Macintosh-compatible system. If so, it will be the first such system. The system is expected to provide 512 × 512-pixel color graphics. While on the subject of the Mac, look for early release in the U.S. of a UNIX package for the Mac from SRL, of Pisa, Italy. The package is expected to be distributed in the U.S. by Plexus, of San Jose, CA. Called MacNIX, it features an icon-rich front end. It is already being sold in Europe.

**IBM** is rumored to have ceased production of the PC/XT and is clearing out inventory. Replacements for the PC/XT have already been shipped to software developers and shown to large volume purchasers. Thus, official introduction is expected as soon as current PC/XT inventories are exhausted. The features of the replacement have been discussed in prior columns.

**Microsoft** and **IBM** have pushed back the release of PC-DOS/MS-DOS version 5 until at least August. The operating system, in test for several months now, was scheduled originally for a spring release. IBM has a large programming staff, in Boca Raton, working on DOS 5.0. The rumor is that IBM will announce a new 286-based system at that time with some proprietary hardware, and the new version of DOS may be tailored to the hardware. If this is indeed the case, it means that IBM has finally figured out a way to lock-out the clone makers—at least for a few months. It is expected that several circuits currently on plug-in cards will be bundled on the motherboard (e.g., display, disk, and network controllers and ports for printer, modem, and mouse).

The prime new features of DOS 5.0 will be the ability to use more memory and more disk space, windows support, and multitasking (but not multiuser capability). However, the multitasking will only work with "well-behaved" programs, and Microsoft is recommending that developers use their C-compilers for development to ensure compatibility. Hence, most programs will need rewriting to do multitasking. The MS-DOS version that Microsoft will sell to other OEMs is expected to be significantly different from IBM's version. It is interesting to note the DRI's Concurrent PC-DOS will multiprocess most DOS applications, and it has been available for a long time now, for 8088/86/286 systems.

Rumors also abound that IBM shortly will introduce a WORM (Write Once Read Mostly) optical disk drive for the new PC/AT systems. The drive is rumored to be from Matsushita Electric Industries.

**Apple** is expected to release a new ultra-powerful graphics coprocessor shortly for its new open-Mac system. Based on a proprietary chip design, it is expected to provide megapixel monochrome resolution, as well as a lower-resolution color display. Apple is also expected to introduce another open-Mac system, this one based on the 68030 with several expansion slots. The system is expected to compete in the powerful work station market currently dominated by **Sun Microsystems**, **Apollo**, and **DEC**. And, late this year (or early next year), Apple is expected to introduce the portable Mac. A low-cost version of the LaserWriter is also expected this year for the Apple IIGS and low-end Mac systems. The printer is not expected to include PostScript. Postscript and networking features are expected to be reserved for the more expensive versions of the LaserWriter.

**Microsoft** and **Borland** are expected to announce low-cost C Compilers shortly. Look for names like Quick-C and Turbo-C.

**Novell**, the leader in networking software, is expected to announce software soon for internetworking PCs to UNIX systems. I hear talk of a new Universal Network Architecture (UNA).

**NEC**, the current leader in multiscan-type color displays, is rumored to be readying a new version capable of resolutions greater than 1000 × 1000 pixels. The current display handles up to 800 × 560 pixels. In the meantime, Thomson, Sony, Taxan, Mitsubishi, and Motorola are introducing multiscan-type color displays, so that prices should start dropping very soon.

**Atari** is expected to soon start selling a 2-megabyte version of the ST. The unit is being sold in Europe already. Also expected from Atari is an IBM-PC emulator, low-cost laser printer, and a new 68020 work station. Atari seems to have found a healthy niche market and is moving into high gear.

**UNIX** may be going public domain very soon. Richard Stallman, in charge of UNIX research at MIT, is heading up an effort to develop a UNIX-workalike system called GNU that will be in the public domain and available to anyone without charge. The effort is part of DARPA funded project to develop a distributed operating system called Mach, being worked on jointly by MIT and Carnegie-Mellon. The likelihood is that Mach may also be public domain.

## 386 UPDATE

**IBM** is up to its old tricks: undermining competitors' products by pre-announcing products it knows it will not be shipping for a long time. IBM has begun showing prototypes of a 386 system to selected corporate buyers who are considering buying 80386-based machines from Compaq, Zenith, and PC's Limited. The object is to dissuade these purchasers from buying non-IBM systems, even though IBM is a long way from shipping the product. The likelihood is that the IBM Entry Systems Division will not ship its first 386-based system until next year. In the meantime, a lot of people are turning to 386 systems such as the Compaq DeskPro-386 for applications such as CAD, desktop publishing, software development, and network servers, where processing speed is really important and cost is secondary. Compaq is reporting that sales of its units with 40Mb and 130Mb drives are the most popular. The 70Mb unit is the least popular.

It is estimated that over 15,000 386 systems are shipped monthly, with **Compaq** the dominant supplier. Most vendors are using the standard AT bus for expansion cards, with one or two proprietary 32-bit-wide connectors. Hence, there is as yet no bus standard in this area, a weakness that IBM may be able to capitalize on. Compaq is reported to be working hard to develop alliances that will establish its hardware

# 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,
PC Magazine,
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,
Editor's Choice,
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**

- Capable of providing nearly twice its rated output capacity (with switching power supply loads).
- Designed specifically for use with your IBM system/34/36, PC, XT and AT.
- On-line operation, static bypass switch and sinewave output.
- Fifth generation, high frequency PWM technology reducing size and weight by over 50%.
- Ideal for multiple XT and AT configurations
- Meets FCC Part 15-Class A, NEC and IEEE 587 standards.
- Virtually silent.

CLARY®

ONGUARD™

**CLARY CORPORATION**
320 West Clary Avenue
San Gabriel, CA 91776 [818] 287-6111 x633

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

system as a standard. There is no doubt that IBM's "leaks" on its 386 system are having an adverse effect on Compaq's efforts.

Also, **Intel** has introduced a bus for 386 machines that has already been adopted by PC's Limited (Austin, TX) and ALR (Irvine, CA). Intel is also marketing its own 386 system, using the bus. If a few more vendors adopt the bus, and IBM dallies long enough, it may have a chance to become a standard.

IBM's system is expected to have many proprietary features, as IBM attempts to lock out competition. The IBM 386 system is not expected to provide any performance features that are superior to currently available 386 systems. However, judging by IBM history, it should be a marketing success. That is, unless IBM introduces a really inferior product. After all, they did it with the portable, "convertible," PCjr, RT, etc.! (IBM = Inferior But Marketable.)

## REPORT FROM BIG BLUE

William Lowe, president of Big Blue's Entry Systems Division, recently said that IBM will "continue to combat the gray market through monitoring and enforcement . . . of dealer agreements."

He went on to say that IBM "will continue to support open-architecture interfaces for application providers to write to. This is extremely important to allow our customers to get full value from their purchase of an IBM system." (*I expect that IBM's new "open-architecture" will be open only to application software while locking out clone hardware.*)

He also said ESD is currently investing half its R&D money in PC development of new software that emphasizes connectivity (*no doubt to IBM mainframes*), ease of use, multitasking (*DOS 5.0*), enhanced graphics (*a new graphics controller to go beyond the EGA/PGA?*) and support of complete IBM systems for its customers (*lock out the peripherals and clone vendors*).

There are rumors that IBM will release a new multitasking product for 286/386 systems that also will include networking support. The product is expected to be tied to proprietary hardware on the new IBM 286/386 systems. Unlike TopView, its multitasker for its 8088-based systems, this new product is expected to be similar to Windows. There is some speculation that IBM may license it to clone vendors when they come out with clones of the new IBM systems. This would provide IBM with income from the cloners and control over networks.

## BLUES FOR THE RT

**International Data Corp.** reported that IBM shipped only 6,000 PC/RT systems last year. Thus, Big Blue has achieved less than an 11 percent share of the 32-bit microprocessor-based work station business. **Sun Microsystems** and **Apollo** are reported to be the market leaders, each with a 36 percent share, followed by **DEC** with an 18 percent share. Despite IBM's price cuts, many users feel the RT is still too expensive. Further, the lack of software, graphics, and networking features have drawn criticism from customers.

This is IBM's second attempt to break into this market. The first was its 68000-based System 9000 desktop scientific work station. This system was dropped after two years. IBM has put a larger marketing effort into the RT and provided more support than it did with the 9000, no doubt because the market has increased substantially in size, and Sun, Apollo and DEC are doing so well in it. But if IBM does not achieve a larger market share soon, we may see the RT follow the 9000 into oblivion, as IBM tries a different tactic to break into the work station market.

## MICROCOMPUTERNIKS MAKE FORBES 400 LIST

Every year *Forbes* magazine publishes a list of the 400 richest Americans. This year quite a few microcomputer people made the list. Highest on the the list is Bill Gates, worth at least $315 million, mostly in Microsoft stock. Paul Allen, his partner in establishing Microsoft (but who has since left to form his own software house), is worth $175 million, but did not make the Forbes list since it cuts off at $180 million.

Bill Millard also made the list with his $200 million holdings in ComputerLand, which he founded. Steve Jobs just barely made the list with a net worth of $185 million.

Other computer people who made the list include H. Ross Perot (EDS, worth $2.5 billion), William Hewlett (HP, worth $1 billion), An Wang (Wang, worth $550 million), Ken Olson (DEC, worth $275 million), and Max Palevsky (SDS, worth $200 million).

## LARGE LAYOFFS HIT THE COMPUTER BIZ

This year may go down as the first "year of shrinking employment" in the computer industry. **IBM** will push over 10,000 employees into early retirement; **AT&T** will terminate over 27,000, mostly from its computer division (it pushed out over 53,000 employees during 1984-1986), **Burroughs** and **Sperry** will lay off several thousand as a result of their merger; and several thousand other layoffs are expected from smaller companies.

IBM recently reported a third-quarter decrease in earnings of close to $400 million. To blame were poor market conditions compounded by loss of market share in mainframes to Amdahl and Hitachi, in mid-size systems to DEC, and in personal computers to the cloners.

IBM recently said that it saw "no signs of improvement" in business conditions for this year. Considering that IBM does more market analysis than any other company, it is probably an indicator of what employment conditions will be like this year. The computer industry, which until now has resisted the downward swings in the economy, is entering a new era.

## GET READY FOR THE GHZ CLOCK RATE

**Hughes Aircraft**'s research lab in Malibu, CA, is boasting that it has built the world's fastest digital IC chip. It is a D-type flip-flop (the heart of all registers and memory circuits) fabricated of gallium arsenide and clocked at 18 GHz, at room temperature. **Rockwell International** also reported a silicon divider circuit clocked at 10.38 GHz.

It is apparent that the current 10-25 MHz clock rates will be considered a snail's pace in the next decade. I can still remember my first 8080 system (in 1975) clocked at 1 MHz: Wow, was that slow!

## THE TRENTON COMPUTER FESTIVAL

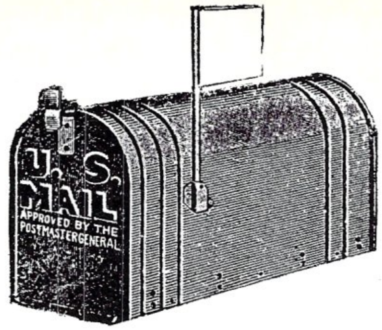
It happens every spring: close to 20,000 avid microcomputer hobbyists decend on Trenton, NJ, for the annual Trenton Computer Festival, the oldest of personal computer shows. And, TCF will happen again on April 11th and 12th. The big attraction is an outdoor flea market that covers about 20 acres. It also includes a large indoor commercial exhibitor area, speakers, and user group meetings. Sol Libes, Don Libes, Bill Wong, Hank Kee, Steve Leon, and Rich Conn (all *M/SJ* authors) will be attending and speaking, as well as other well-known authorities (e.g., Steve Ciarcia).

For more information, call (609) 771-2487, or write TCF-87, c/o Trenton State College, Trenton, NJ 08625.

## QUOTATION OF THE MONTH

"There are more than 7 million work stations out there today using MS-DOS."
—Bill Gates, Chairman, Microsoft Corp.

(*And who should know better?—Ed.*) §



# 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.*

## FURNACE CONTROL WITH A MICRO

Dear *MS/J:*

I really snapped up the September/October 1986 issue of *M/SJ* with enthusiasm because of Lynwood Wilson's article on process control. I am in the industrial furnace business and am totally involved with the control of heat. In fact, I am now working on a microcomputer program to control multiple industrial processes.

The central problem with PID control is setting the PID constants. It is so confusing and time-consuming to do this that rarely is it ever done. The modern method, called autotuning, is at least a partial answer to that. I have still not developed a good way to program this step—at least, I don't think I have. I have not yet found the mathematics of tuning the constants—then, lo and behold, Lynwood Wilson talked about that very subject.

In the real world of large-furnace control, we are finding that time proportioning is a real problem. The necessary accuracy just cannot be achieved because the sampling time with mechanical contactors is too great. With SCR power control, you can get even and close sample times. On a recent job of a 40-KW furnace at 985°F, we were able to get +1/−0 control as measured by the control TC. Curtis D. Johnson's book entitled *Microprocessor-Based Process Control* developes very efficient arithmetic for PID control and even presents the assembly language for the routines.

Roger Clark
Long Beach, Calif.

*Lynwood Wilson responds:*
*I am glad Roger Clark found my article useful. It's particularly rewarding to receive praise from a professional in the field.*

*I also am interested in the automatic setting of PID constants under software control. I know there are commercial controllers that do this, but I am not familiar with the algorithm they use. I believe that some of them begin by calling for a long pulse of heat, presumably to determine the relationship between thermal inertia and heat input for the system in question. I would be very interested in more information if anyone runs across it (or invents it!). I might include it in a future article. It might be entertaining to work it out empirically for an article such as the last one.*

*As far as the mathematics behind control systems goes, the best source I know of, short of a good college electrical-engineering course in control-system theory, is the book* Real Time Programming *by Caxton Foster, published by Addison-Wesley (ISBN 0-201-01937-X). This book also covers other areas that you may find useful. I recommend it highly.*

## PROGRAM INTERFACING TO MS-DOS PROBLEMS

Dear *MS/J:*

This letter is with regard to William Wong's *Program Interfacing to MS-DOS* series that you have been running.

Mr. Wong is certainly competent technically but could use some help with grammar, sentence structure, spelling, and organization. (*Editor—We may be to blame for much of that. Hopefully, with more time available now and a full staff, such errors will not occur.*) He often assumes a background knowledge that does not always exist. Even those people with the necessary background often find him quite confusing and difficult to follow.

The real confusion begins with Part VI, Device Drivers. He says, for example: "A standard device driver is linked as a .COM file with an origin of 0 instead of the normal 100 hex." I think I know what he means, but his logic is difficult to follow. The definition of a .COM file is that it originates at 100h, and to say that it is linked at 0 is nonsensical.

I typed in his source listing starting on page 57 of the July/August 1986 issue exactly as printed. As I expected, my assembler (MASM V1.27) barfed all over me with multiple errors. I combed out my own typos, which left two typos inherent in the listing: bootrecord instead of boot_record and rh_stats instead of rh_status. Fair enough.

I was left with three recurrent errors, all Code 57: Illegal Size for Item. These errors came on three lines only, as follows:

```
mov cs:rh_offset,bx
mov cs:rh_segment,bx
lea ax,mdisk_data
```

The problem turned out to be that the rh_address was a variable, not a label, and it was defined as a dd type. Mov instructions will not accept this type.

I tried breaking rh_address into two dw statements, which cured the mov problem all right but blew half a dozen other lines out of the water—lines which require rh_address to be dd type.

I finally got around it by a clumsy ruse: I declared a variable as rh_test dw 1 dup (?), located just before the rh_address dd statement. Then I rewrote the rh_offset and rh_segment as follows:

```
rh_offset equ rh_test + 2
rh_segment equ rh_test + 4
```

This satisfied the assembler, and it happily accepted the mov statements. I was left with the lea statement, which just didn't like mdisk_data equ $—Code 57, of course.

I resorted to more clumsy patching. I converted the equ statement to:

```
mdisk_data db 1 dup (?)
```

and the assembler finally gave me a complete run—no errors. I linked the program, renamed it .SYS and put the necessary DEVICE statement in CONFIG.SYS.

The RAM-disk program works just great. Mr. Wong may have difficulty passing on his technical expertise, but he certainly knows how to write good software.

Gerald L. Hewett
Inyokern, Calif.

# THE PROGRAMMER'S SHOP

**helps save time, money and cut frustrations. Compare, evaluate, and find products.**

## RECENT DISCOVERY

**Turbo Expert** by Thinking Technologies - Menu driven expert system generation package details reasoning, comes with tutorial, manual, demos. "Corporate" supports up to 4000 rules. "Startup" up to 400. Corporate $359, PC Startup $129

## AI-Expert System Dev't

| | | |
|---|---|---|
| Arity System - use with C | MS | $ 259 |
| Auto-Intelligence | PC | $ 749 |
| Experteach - Powerful, samples | PC | $ 349 |
| Exsys | PC | $ 309 |
| Runtime System | PC | $ 479 |
| Insight 2+ | MS | $ 379 |
| Intelligence/Compiler | PC | $ 749 |
| SQL Dev't Package | MS | $ 259 |
| Texas Instruments: | | |
| PC Easy | PC | $ 439 |
| Personal Consultant Plus | PC | $2599 |

## AI-Lisp

| | | |
|---|---|---|
| Microsoft MuLisp 85 | MS | $ 179 |
| PC Scheme LISP - by TI | PC | $ 85 |
| TransLISP - learn fast | MS | Call |
| TransLISP PLUS | | |
| Optional Unlimited Runtime | | $ 150 |
| PLUS for MSDOS | | $ 179 |

Others: IQ LISP ($155), IQC LISP ($269)

## AI-Prolog

| | | |
|---|---|---|
| APT - **A**ctive **P**rolog **T**utor - build applications interactively | PC | Call |
| ARITY Standard - full, 4 Meg | | |
| Interpreter - debug, C, ASM | PC | $ 309 |
| COMPILER/Interpreter-EXE | PC | $ 699 |
| With Exp Sys, Screen - KIT | PC | $1129 |
| Standard Prolog | MS | $ 79 |
| MacProlog Complete | MAC | $ 295 |
| MicroProlog - intro | MS | $ 85 |
| MicroProlog Prof. | MS | $ 339 |
| MPROLOG P550 | PC | $ 175 |
| Prolog-86 - Learn Fast | MS | $ 89 |
| Prolog-86 Plus - Develop | MS | $ 229 |
| TURBO PROLOG by Borland | PC | $ 69 |

## AI-Other

| | | |
|---|---|---|
| Q'NIAL - APL with LISP. | PC | $ 349 |
| Smalltalk-80 - Xerox improved | PC | $ 995 |
| Smalltalk/V-graphics | PC | $ 89 |

## Atari ST & Amiga

| | | |
|---|---|---|
| Manx, Lattice, & Metacomco. | | Call |
| Amiga - LINT by Gimpel | Amiga | $ 79 |
| Cambridge LISP | Amiga | $ 200 |
| Lattice C | ST, Amiga | $ 139 |
| Lattice Text Utilities | Amiga | $ 75 |
| Megamax - tight, full | ST | $ 200 |

## FEATURES

**r-tree** - report generation for ctree. Multiple file handling, fixed or variable length. Many built-in functions like Boolean, computational functions, string, date handling, numeric to string conversion. Source in C. PC $ 249

**Advantage C++** - Object-oriented enhancements for most major C compilers also maintain compatibility with existing C code. Write reusable easily maintainable code, develop large applications with fewer bugs. Call

### Free Literature Compare Products

Evaluate products. *Compare* competitors. Learn about new alternatives. One *free call* brings information on just about any programming need. Ask for any ***"Packet"*** or Addon Packet □ AI □ ADA, Modula □ BASIC □ "C" □ COBOL □ Editors □ FORTH □ FORTRAN □ PASCAL □ UNIX/PC or □ Debuggers, Linkers.

**Our Services:**

- **Programmer's Referral List**
- **Compare Products**
- **Help find a Publisher**
- ***Evaluation Literature FREE***
- **BBS - 7 PM to 7 AM 617-826-4086**
- **Dealers Inquire**
- **Newsletter**
- **Rush Order**
- **Over 700 products**
- **National Accounts Center**

## Basic

| | | |
|---|---|---|
| Basic Development System | PC | $ 105 |
| Basic Development Tools by Sterling Castle | PC | $ 89 |
| Basic Windows by Syscom | PC | $ 95 |
| BetterBASIC | PC | $ 129 |
| 8087 Math Support | PC | $ 75 |
| Run-time Module | PC | $ 169 |
| Better Tools - for Better Basic | PC | $ 95 |
| CADSAM FILE SYSTEM-full | MS | $ 69 |
| Finally - by Komputerwerks | PC | $ 85 |
| GoodBas - maintain code | PC | $ 95 |
| LPI Basic - MS compatible | UNIX | $1100 |
| Prof. Basic - Interactive, debug | PC | $ 75 |
| 8087 Math Support | PC | $ 45 |
| QuickBASIC | PC | $ **69** |
| TRUE Basic - ANSI | PC | $ 119 |
| Run-time Module | PC | $ 129 |
| Turbo BASIC - by Borland | PC | $ 69 |

## Cobol

| | | |
|---|---|---|
| Macintosh COBOL - full | MAC | $ 459 |
| MBP - Lev. II, native | MS | $ 819 |
| Microfocus Professional Cobol | PC | $2295 |
| VS Workbench | PC | $3379 |
| Microsoft COBOL | MS | $ 439 |
| Microsoft Cobol Tools | PC | $ 209 |
| Realia - very fast | MS | $ 819 |
| Ryan McFarland COBOL | MS | Call |
| COBOL-8X | MS | Call |
| Screenplay-by Flexus. Interactive screen mgmt. | PC | $ 175 |

## Editors for Programming

| | | |
|---|---|---|
| BRIEF Programmer's Editor | PC | Call |
| EMACS by UniPress Source: $929 | | $ 299 |
| Epsilon - like EMACS, full C-like language for macros. | PC | $ 155 |
| KEDIT - like XEDIT | PC | $ 99 |
| Lattice Screen Editor - multiwindow, multitasking Amiga $ 89 | MS | $ 109 |
| Micro Focus Micro/SPF | PC | $ 49 |
| PC/EDT - macros | PC | $ 250 |
| PC/VI - by Custom Software | MS | $ 109 |
| Personal REXX | PC | $ 99 |
| PMATE - power, multitask | PC | $ 119 |
| SPF/PC - fast, virtual memory | PC | $ 139 |
| Vedit | MS | $ 107 |
| Vedit PLUS | MS | $ 139 |

## C Libraries-Communications

| | | |
|---|---|---|
| Asynch by Blaise | PC | $ 135 |
| Essential Comm Library | PC | $ 135 |
| With Debugger | PC | $ 199 |
| Greenleaf Comm Library | PC | $ 129 |
| Multi-Comm - add multitasking, use w/Multi-C | PC | $ 149 |

## RECENT DISCOVERY

**VXM** by Command Technologies - Allows intelligent program control sharing by different environments like PCDOS-VAX. "Software Robots" use resources as needed, convert formats automatically, transmit data. Innovative mini-micro integration. PC Call

## C Language-Compilers

| | | |
|---|---|---|
| AZTEC C86 - Commercial | PC | $499 |
| Datalight C - fast compile, good code, 4 models, Lattice compatible, Lib source. Dev's Kit | PC | $ 77 |
| Datalight Optimum - C | MS | $ 99 |
| with Light Tools by Blaise | PC | $168 |
| Lattice C - from Lattice | MS | $275 |
| Mark Williams - w/debugger | MS | $369 |
| Let's C Combo Pack | PC | $ 99 |
| Let's C | PC | $ 59 |
| Microsoft C 4.0- Codeview | MS | $**279** |
| Uniware Cross Assemblers | MS | $249 |
| Cross Dev't Tools | MS | Call |
| Rex - C/86 by Systems & Software - standalone | MS | $695 |
| Wizard C | MS | $359 |
| Rom Development Package | MS | $299 |

## C Language-Interpreters

| | | |
|---|---|---|
| C-terp by Gimpel - full K & R | MS | $229 |
| C Trainer - by Catalytix | PC | $ 89 |
| INSTANT C - Source debug, Edit to Run-3 seconds, .OBJs | MS | $379 |
| Interactive C by IMPACC Assoc. | PC | $209 |
| Introducing C-self paced tutorial | PC | $105 |
| Run/C Professional | MS | $169 |
| Run/C Lite | MS | $ 89 |

## C Libraries-General

| | | |
|---|---|---|
| Blackstar C Function Library | PC | $ 79 |
| C Essentials - 200 functions | PC | $ 83 |
| C Function Library | MS | $119 |
| C Tools Plus (1 & 2) - Blaise | PC | $135 |
| C Utilities by Essential | PC | $137 |
| 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

| | | |
|---|---|---|
| FILES: C Index by Trio - full B+ Tree, vary length field, multi compiler | | |
| /File is object only | MS | $ 89 |
| /Plus is full source | MS | $319 |
| CBTREE - Source, no royalties | MS | $ 99 |
| CTree by Faircom - no royalties | MS | $319 |
| rtree - report generation | PC | $249 |
| dbQUERY - ad Loc, SQL - based | MS | $159 |
| dbVISTA - full indexing, plus optional record types, pointers, Network. | | |
| Object only - MS C, LAT, | C86 | $145 |
| Source - Single user | MS | $399 |
| Source - **Multiuser** | MS | $799 |
| dBx - translator | MS | $315 |
| w/source to library | MS | $349 |

## FEATURE

**Sapiens V8** - virtual memory management for C programmers on PC's provides 8M workspace, 64-bit emulation, virtual stack library and heap. Link to MS, Lattice, Aztec. PC $300

**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.**

## dBASE and SUPPORT

Extend your dBASE programming power with FAST code, LAN power, and the FLEXIBILITY of C. These tools create a more productive environment. Call our specialist TODAY.

**Order before 4/30/87 and mention this ad for these special prices:**

| | List | Normal | SPECIAL |
|---|---|---|---|
| dBASE III Lan Pack | $995 | $649 | **$609** |
| dBASE Tools for C | $ 90 | $ 75 | **$ 59** |
| dBC Isam by Lattice | $250 | $199 | **$159** |
| dBXL Interpreter by Word Tech | $169 | $139 | **$119** |
| Genifer by Bytel | $395 | $299 | **$249** |
| QuickCode for III Plus | $295 | $249 | **$199** |
| QuickSilver by Word Tech | $599 | $499 | **$389** |

## C Support-Systems

| | | |
|---|---|---|
| Basic-C Library by C Source | PC | $139 |
| C Sharp - **realtime,** tasks. | PC | $600 |
| C ToolSet - DIFF, xref, source | MS | $ 95 |
| The HAMMER by OES Systems | PC | $149 |
| Lattice Text Utilities | MS | $ 89 |
| Multi-C - multitasking | PC | $149 |
| PC LINT-Checker. Amiga $89 | MS | $ 99 |
| Quickshell - script compiler | PC | $349 |
| Pfantasy Pac - by Phoenix | PC | $849 |
| Pre-C - Lint-Like | MS | $155 |
| Programmer's Extender, Vol. I | MAC | $ 79 |
| SECURITY LIB-add encrypt to MS C, C86 programs. Source $229 | PC | $115 |
| Time Slicer - R/T | PC | $265 |

## C-Screens. Windows. Graphics

| | | |
|---|---|---|
| C Power Windows by Entelekon | PC | $109 |
| dBASE Graphics for C | PC | $ 69 |
| C-Scape - capture Dan Bricklin | PC | $179 |
| Curses by Lattice | PC | $ 89 |
| ESSENTIAL GRAPHICS - fast | PC | $199 |
| GraphiC - mono version | PC | $209 |
| GraphiC - new color version | PC | $285 |
| Greenleaf Data Window | PC | $159 |
| w/source | PC | $319 |
| Multi-Windows - use w/ Multi-C | PC | $295 |
| Screen Ace Form Master | PC | $195 |
| Vitamin C - screen I/O | PC | $199 |
| **Windows for C** - fast | PC | $149 |
| **Windows for Data** - validation | PC | $239 |
| ZView - screen generator | MS | $175 |

## Debuggers

| | | |
|---|---|---|
| 386 Debug - by Phar Lap | PC | $149 |
| Breakout - by Essential | PC | $ 89 |
| CODESMITH - visual | PC | $ 99 |
| C SPRITE - data structures | PC | $129 |
| DSD87 - by Soft Advances | PC | $ 79 |
| Periscope I - own 16K | PC | $239 |
| Periscope II - Reset Box | PC | $109 |
| Periscope II-X - software only | PC | $ 85 |
| Pfix-86 Plus - by Phoenix | PC | $229 |
| Showcase - test software | PC | $135 |
| SoftProbe II - by Systems & Software, embedded systems | PC | $695 |

## FEATURE

**The Documentor** - for dBASE program flow chart, tree diagrams, .DBF documentation, variable/field concordance, hierarchy charts. Macros, searches, configure options. MS $295

## 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 | $**299** |
| No Limit - Fortran Scientific | PC | $115 |
| PC-Fortran Tools - xref, pprint, screen | PC | $179 |
| RM/Fortran | MS | Call |
| Scientific Subroutines - Matrix | MS | $139 |
| Statistician by Alpha | MS | $249 |
| Strings and Things - register, shell | PC | $ 55 |

## Multilanguage Support

| | | |
|---|---|---|
| BTRIEVE ISAM | MS | $199 |
| BTRIEVE/N-multiuser | MS | $465 |
| Flash-Up Windows | PC | $ 79 |
| GSS Graphics Dev't Toolkit | PC | $375 |
| HALO Graphics | PC | $209 |
| I/O Pro - screens, full char. | PC | $349 |
| Informix - by RDS | PC | $639 |
| Informix 4GL-application builder | PC | $799 |
| Informix SQL - ANSI standard | PC | $639 |
| Opt Tech Sort - sort, merge | MS | $115 |
| PANEL | MS | $215 |
| Pfinish - by Phoenix | MS | $229 |
| PolyLibrarian by Polytron | MS | $ 79 |
| PolyBoost - speed I/O, keyboard | PC | $ 69 |
| PVCS Version Control | MS | $329 |
| QMake by Quilt Co. | MS | $ 84 |
| Rtrieve - Xtrieve option | MS | $119 |
| Screen Sculptor | PC | $ 95 |
| SRMS - source control | MS | $109 |
| Xtrieve - organize database | MS | $199 |
| ZAP Communications - VT 100 | PC | $ 89 |

## Pascal and Supporting

| | | |
|---|---|---|
| ALICE - learn Pascal | PC | $ 68 |
| Exec - Chain Programs | MS | $ 79 |
| MetaWINDOWS-graphics toolkit bit-mapped, fast | PC | $115 |
| MetaWINDOWS PLUS | PC | $185 |
| Microsoft PASCAL - faster | MS | $189 |
| Pascal Extender | MAC | $ 65 |
| Pascal Pac with Tidy - formatter, utilities | PC | $ 69 |
| Pascal Tools PLUS | PC | $139 |
| Pascal 2 - by Oregon Software, tight, fast | MS | $329 |
| TurboHALO - 150 routines | PC | $105 |

## RECENT DISCOVERY

**F2C** by Solution Systems - Fortran 66 or 77 to C. No max program size $3000. Pioneer. 1000 line max: $ 795

## Other Languages

| | | |
|---|---|---|
| APL*PLUS/PC | PC | $ 429 |
| CCS Mumps - Singleuser | PC | $ 50 |
| CCS Mumps - Multiuser | PC | $ 369 |
| Lattice RPG II Compiler | PC | $ 719 |
| MasterForth - Forth '83 | MAC or PC | $ 109 |
| Microsoft MASM - faster | 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 |
| SNOBOL4 + - great for strings | MS | $ 80 |
| UR/Forth | MS | $ 279 |

## Xenix/Unix

| | |
|---|---|
| Basic - by Microsoft | $ 239 |
| C-Terp by Gimpel Software | $ 449 |
| Cobol - by Microsoft | $ 639 |
| Cobol Tools - by Microsoft | $ 319 |
| Fortran or Pascal - by Microsoft | $ 439 |
| MicroFocus Lev. II Compact COBOL | $ 795 |
| Panel | $ 539 |
| RM/Cobol | Call |
| RM/Fortran | Call |
| Xenix Complete System | $1049 |

## Other Products

| | | |
|---|---|---|
| 386 Assembler/Linker | PC | Call |
| ASMLIB - 170+ routines | PC | $ 129 |
| asmTREE - B+tree file mgmt. | PC | $ 369 |
| BSW Make - like UNIX make | MS | $ 85 |
| Compact Source Print | PC | $ 59 |
| Dan Bricklin's Demo Program | PC | $ 59 |
| dBrief - Customize BRIEF for dBASE development. with BRIEF $275. | PC | $ 95 |
| dFlow - .PRG diagram | MS | $ 149 |
| Help/Control - on line help | PC | $ 109 |
| Interactive Easyflow-HavenTree | PC | $ 129 |
| 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 | $ 329 |
| MKS Toolkit - Unix, vi, awk | PC | $ 119 |
| Numerical Analyst by Magus | PC | $ 295 |
| PDisk - cache, tree | PC | $ 125 |
| PLink - 86 PLUS - overlays | MS | $ 319 |
| PMaker - by Phoenix | PC | $ 79 |
| Polymake by Polytron | MS | $ 129 |
| PolyShell by Polytron | MS | $ 119 |
| PolyXREF by Polytron | PC | $ 99 |
| Sapiens V8 - 8M virtual mgr. | PC | $ 300 |
| Synergy-Create user interfaces | MS | $ 375 |
| Taskview - by Sunny Hill Software, ten tasks | PC | $ 55 |
| Tom Rettig's Library - dBASE | PC | $ 89 |
| Tree Diagrammer | PC | $ 59 |
| 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

**HOURS**
8:30 AM - 8:00 PM EST.

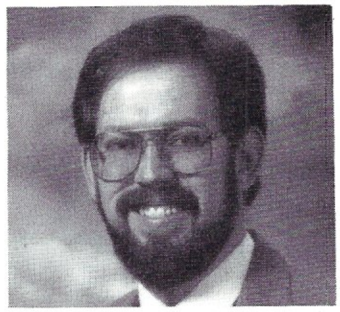
**800-421-8006**

**THE PROGRAMMER'S SHOP™**
*Your complete source for software, services and answers*

**5-D Pond Park Road, Hingham, MA 02043**
**Mass: 800-442-8070 or 617-740-2510 2/87**

"We at Sunspot are thrilled to know that there is a store that can cut through all the "bull", and find us the products that most computer stores know nothing about. Keep up the good work."
— Arland Hensler
Sunspot

# Turbo Pascal Corner



by Stephen R. Davis

## Compiler Options

*This column features tips and techniques for using Turbo Pascal productively on MS/PC-DOS and CP/M microcomputer systems. It discusses typical problems and their solutions. Reader suggestions, comments, and questions are encouraged. Address them to Turbo Pascal Corner, Route 5, Box 107K, Greenville, TX 75401 or through MCI mail, 289-6124.*

I am sure that you will already have read the news of Comdex 86—reports of desktop publishing, fully functional MS-DOS laptops the size of a Model 100, 386-based barn burners, and the rest. Crowds gathered around these modern marvels, but don't let them fool you. Just as many crowds gathered around Borland's booth to see the new offerings there.

Did you ever notice the claim on the cover of the manual for Turbo Pascal Version 3: "Before the end of the year, Turbo Pascal will be running on most 68000-based microcomputers?" I don't know about *most*, but Borland has at least kept part of its promise and delivered Turbo Pascal for the Mac, as it calls it.

Mac Turbo Pascal, not to be confused with MacTurbo, or TurboMac, or Mac anything else—blends very well into the Macintosh interface. The lack of a good interface has been a source of complaint with Borland's other Mac package, Reflex. In providing a good interface for the Macintosh version, Borland has omitted many of the options available to other Turbo Pascal users, though. A Borland representative tried to convince me that most of these options had no meaning on the 68000. On the other hand, the whole idea of the Smalltalk interface is to be as easy to use as possible and Mac Turbo keeps within that spirit. I will give a more careful analysis in my next column. In the meantime, I want to welcome Mac Turbo Pascal users into the fold and invite them to join me in future columns.

Borland also announced Turbo Basic at Comdex. This is a very impressive package. Not being a BASIC type, I was expecting to look as far down my nose as possible, but as soon as I took one glimpse, I was there gawking with the best of them. For one thing, Turbo Basic has a different user interface from that of Turbo Pascal 3.0. Retaining all the advantages of an integrated compiler and editor, it adds more control options and some debug features to a windowed interface that is easier to use—if you can believe that! Indications are that this is the interface for all future Borland PC-DOS languages. If so, bring on Turbo Pascal 4.0! (There were even some rumors afloat that Turbo C might not be too far away—hope springs eternal.)

Additionally, Borland was showing its Numerical Methods Toolbox. Like other toolboxes, this consists of a collection of carefully documented, carefully written Pascal source programs. The Numerical Methods Toolbox contains routines to calculate zeros of a function, spline interpolation, discrete differentiation and integration, matrix inversion, matrix eigenvalues, Fourier transformations, and more. As you read this, I will be feverishly working on a review of the toolbox for this column—if Borland sticks to the promised January 1987 release date.

### TODAY'S TOPIC

Quick programming tip: When trying to disassemble a section of Turbo Pascal code, often your biggest task is just finding the procedure in question in the generated .COM file. Edit the procedure and make it into an overlay (see the Turbo Pascal manual for details on how to do this). Disassembling the resulting overlay reveals that Turbo Pascal tacks a mere four or five instructions in front of your code. This makes finding the code for your procedure a breeze, and it does not alter the assembler code generated.

The Question of the Month comes from Farrell Chown of Amprior, Ontario, who is having trouble understanding the various addresses presented by Turbo Pascal, both at compile time and in the Options menu. Chown reports that after compiling his favorite program, Turbo Pascal reports the following:

Code: 00BA paragraphs (2,976 bytes),
0C7D paragraphs free
Data: 017B paragraphs (8,064 bytes),
0E5D paragraphs free

First of all, what is a paragraph? A paragraph is 16 bytes, a strange unit of memory that stems from the architecture of the 8086 and 8088. People who work with 8086 assembler often use this unit because they can then express 1 megabyte—the address space of the 8086/88—in four hex digits instead of in five. For some reason, people much prefer four digits to five. As long as you stay in Turbo Pascal, you needn't go any deeper than that. (Interesting note: a paragraph has a completely different meaning in the 286 and 386, which is why these chips currently cannot run DOS except in 8086 emulation mode.)

The above report means that the compiled program generates BAH paragraphs of code, which it reports as 2,976 bytes. Similarly, this same program declares some 8,064 bytes of global data. To make this calculation work out, remember that the *paragraph* numbers are in hex (which I indicate by adding an *H* to the number, but Borland does not) and the *byte* numbers are decimal. Add up the 17BH and the E5DH, and you get FD8H, which is reasonably close to the 1000H (64K) maximum data size. But, add up BAH and C7DH, and you get only D37H, which is some 11,408 bytes short of the magic 64K limit. So where did the 400 bytes of data and 11K of code go?

Every program you compile, no matter

how small, must have the Turbo Pascal library attached to it. This library defines all the standard functions, such as Write, WriteLn, GetMem, and so on. It also contains the floating-point library, which knows how to perform real-number calculations. Apparently, this version of Turbo Pascal contains some 11K of library code, which defines some 400 bytes of global variables.

This explains why, if you were to generate a .COM file from this program and then examine it with the DIR command, it would not be 2,976 bytes long but rather 14,124 bytes. Not only must any .COM file that Turbo Pascal generates contain your code, but it must also contain the 11.4K of library necessary to support your program.

At this point, Chown turns his attention to the Options menu. Noting that the maximum code size is D37H, he selects A6EH as his minimum code size. He is understandably surprised when the compiler returns with D37H rather than the A6EH he entered. I was taken aback myself—until I entered a 0, and Turbo Pascal responded with 288H. When I entered 100H, it came back with 388H, and so on. Whatever I entered, it added 288H—the size of its Turbo library. It was assuming that my number included only my code. In the example case, Turbo Pascal added some number slightly larger than 280H to the entered A6EH and came up with a number that it then rounded down to the maximum possible D37H.

Note also that the displayed results are different between RAM compilations and .COM compilations. In the former case, Turbo Pascal knows how much memory is available and can calculate the size of Stack and Heap accurately. When compiling to a .COM file, however, Turbo Pascal has no idea what size machine the resulting program will be run on. Remember that such memory questions must be resolved at run time and not at compile time. Therefore, when compiling to a .COM file, the maximums are often displayed as A000 paragraphs. This is actually the famous 640K address limit of DOS. Turbo Pascal is just assuming that it has access to all available memory when it runs, which is the case for single-user, single-tasking systems. This number can be reduced from the Options menu for multitasking systems.

Hopefully, this will clear up some questions that you and Farrell Chown have about the Turbo Pascal interface. If not, send in your question and get your free disk of Turbo Pascal programming tools. After all, Chown shouldn't be the only one to *chow'n* down on all those Turbo Pascal utilities! §

*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.*

# The C Forum

by Don Libes

## The International Obfuscated C Code Contest

The results of the 1986 IOCCC (no relation to the IOC, at least not yet) are in, and we are all in for a treat. This year's winners are even better (or maybe that should be "worse") than last year's.

The International Obfuscated C Code Contest is run annually by Landon Noll (Amdahl Corp.) and Larry Bassel (National Semiconductor), who collect C code that is so awful to read, it is actually funny. Viewed in the right light, you might even call it educational. The 1986 winners are presented toward the end of the column. (The first and second contest results were published in *M/SJ* in September 1985 and May 1986, respectively.)

The 1987 contest is now open. The goal is to write the most obscure C program, within the following rules:

1. The source must be 1024 bytes or less.
2. Include in your letter:
   a. name (or anonymous), company/school, city, state, and country
   b. your path from a major network site, if applicable
   c. a brief statement of what the program should do
   d. the machine(s)/operating system(s) on which it runs
   e. enclose your source between the following lines:
      ---start of program---
      <place obfuscated source of 1024 bytes or less here>
      ---end of program---
3. The entry should be written in common C (K & R plus common extensions).
4. The program must be of original work. All programs must be in the public domain. Copyrighted programs will be rejected.
5. Entries must be received before May 30, 1987 (00:00 GMT). Electronically mail your entries to: decwrl!amdahl!chongo
   Entries sent by electronic mail will be confirmed starting May 1, 1987. You are encouraged to submit entries via electronic mail; however, you may mail entries to the following address:

   International Obfuscated C Code Contest
   Amdahl Corp.
   Attn: Landon Noll, M/S 158
   1250 E. Arques Ave.
   Sunnyvale, CA 94088-3470

### JUDGING

Awards will be given to the best entry in a number of categories. The actual category list will vary, depending on the types of entries received. At the moment, the following categories are being considered:

- most obscure algorithm
- worst abuse of the C preprocessor
- strangest source layout
- best one-line program
- most useful program
- anything else needed to give recognition to a good entry

The judges will attempt to run each of the entries on a variety of systems. Try to avoid operating-system/machine-specific code. Extra points will be given for programs that:

a) pass lint without complaint
b) do something quasi-interesting
c) are portable

If you don't need 1024 bytes, don't use them. Given two programs with similar techniques and similar levels of obfuscation, the judges will tend to favor the more concise entry.

Good luck!

The contest judges will bend the rules to recognize outstanding obfuscation wherever it is found. So, if you absolutely can't get your program shorter than 1025 bytes, don't worry about it. Or if your one-year-old refused to include a single comment in the piece of code she banged out one day by rolling her head across the keyboard, but it solves Fermat's last theorem, send it anyway. The judges will give it its fair due.

After all, it can't be any worse than much of the C code the judges have seen, and, who knows, it might even end up as an essential part of the next release of the operating system.

Here are the winners from 1986. These are truly impressive. As with last year's, I decided I should come to terms with at least one of these programs (the smallest, naturally) and stare with astonishment for a while at the rest. But after studying the program for about 20 minutes, I was hopelessly confused.

My usual tactic was to look at the output of the C preprocessor and then run that through a C indenter. I struggled a little and then slowly succumbed to the preprocessor and indenter, in turn. My next tactic was to rename all the variables with more meaningful names. Running the program several times, I saw changing output, convincing me that time( ) was being called. Supported by most C libraries, time( ) returns the current time in seconds, as well as storing it at the address pointed to by its argument.

What helped the most was to add a couple of printf( )s and a cover function for time( ), so that I could see the arguments with which each function was being called. Much to my dismay, I found that main( ) really was calling itself. By looking at the arguments, however, I was able to figure out what was going on. With these hints, you should also be able to figure out what the program is doing.

## THE 1986 WINNERS

Note that several entries had lines so long that they had to be broken up in order to fit in the magazine. Lines that originally continued will appear with a backslash at the end.

The envelope, please . . .

In the category of "best layout," the winner is Eric Marshall of System Development Corp., a Burroughs Company, Paoli, PA.

```
                                                          extern int
                                                              errno
                                                                ;char
                                                                   grrr
                                  ;main(                            r,
  argv, argc )                  int    argc                          ,
   r        ;                 char *argv[];{int                      P( );
#define x  int i,             j,cc[4];printf("      choo choo\n"     ) ;
x  ;if    (P(  !              i               )       |  cc[  !      j ]
&  P(j    )>2  ?              j               :       i  ){*  argv[i++ +!-i]
;              for    (i=                     0;;    i++                 );
_exit(argv[argc- 2    / cc[1*argc]|-1<<4 ]    ) ;printf("%d",P(""));}}
  P  (    a  )   char a   ;  {    a  ;   while(    a  >      "  B   "
  /* -    by E            ricM    arsh             all-      */);    }
```

Comments: Good for wallpaper. Little of this program makes sense; even my compiler hated it.

---

In the category of "worst abuse of the C preprocessor," the winner is Jim Hague, University of Kent at Canterbury.

```
#defineDIT(
#defineDAH)
#define__DAH++
#define_DITDAH*
#defineDAHDITfor
#defineDIT_DAHmalloc
#define DAH_DITgets
#define_DAHDITchar
_DAHDIT _DAH_[]="ETIANMSURWDKGOHVFaLaPJBXCYZQb54a3c2g16g7c8a901?e'b.s;i,d:"
;mainDITDAH{_DAHDIT
DITDAH_DIT,DITDAHDAH_,DITDAH DIT_,
DITDAH_DIT_,DITDAHDIT_DAH DIT
DAH,DITDAHDAH_DIT DITDAH;DAHDIT
DIT _DIT=DIT_DAHDIT 81DAH,DIT_=_DIT
__DAH;_DIT==DAH_DITDIT _DITDAH;__DIT
DIT'\n'DAH DAHDAHDIT DITDAH_=_DIT;DITDAH
DAH_;__DITDITDITDAH
_DIT_?_DAH DITDITDAHDIT_ DAH:'?'DAH,__DIT
DIT' 'DAH,DAH_ __DAHDAH DAHDITDIT
DITDAHDIT_=2,_DIT_=_DAH_;DITDAH _DIT_&&DIT
DITDAH _DIT_!=DITDITDAH DAH_>='a'?DITDAH
DAH_&223:DITDAHDAH_ DAH DAH;DIT
DITDAHDIT_ DAH __DAH,_DIT___DAH DAH
DITDAH DIT_+=DIT DITDAH _DIT_>='a'?DITDAH _DIT_-'a':0
DAH;}_DAH DIT DIT_DAH{__DIT DIT
DIT_>3?_DAHDIT DIT_>>1 DAH:'\0'DAH;return
DIT_&1?'-':'.';}__DIT DITDIT_ DAH _DAHDIT
DIT_;{DIT void DAH write DIT1,&DIT,1 DAH;}
```

Comments: Feed it characters from standard input. It uses the international standard rather than the U.S. standard.

---

In the category of "best one liner," the winner is Jan Stein, Chalmers Computer Society, Gothenburg, Sweden.

```
typedef char*z;O;o;_=33303285;main(b,Z)z Z;{b=(b>=0||(main(b+1,Z+1),
*Z=O%(o=(_%25))+'0',O/=o,_/=25))&&(b<1||(O=time(&b)%0250600,
main(~5,*(z*)Z),write(1,*(z*)Z,9)));}
```

Comments: Join the preceding lines into one line before compiling. Rename as a.out and supply ^M^L (return/formfeed) as argument.

---

In the category of "most adaptable program," the winner is Jack Applin, Hewlett-Packard, Ft. Collins, CO.

```
cat =13 /*/ >/dev/null 2>&1; echo "Hello, world!"; exit
*
*  This program works under cc, f77, and /bin/sh.
*
*/; main() {
      write(
cat-~-cat
     /*,'(
*/
     ,"Hello, world!"
     ,
cat); putchar(~-~-~-cat); } /*
     ,)')
      end
*/
```

Comments: Execute in any of the methods documented.

---

In the category of "most useful obfuscation," the winner is Walter Bright, FutureNet, Redmond, WA.

```
#include <stdio.h>
#define O1O printf
#define OlO putchar
#define O10 exit
#define Ol0 strlen
#define QLQ fopen
#define OlQ fgetc
#define O1Q abs
#define QO0 for
typedef char lOL;

lOL*QI[] = {"Use:\012\011dump file\012","Unable to open file '\x25s'\012",
 "\012","   ","");

main(I,Il)
lOL*Il[];
{FILE *L;
unsigned lO;
int Q,OL[' '^'0'],llO = EOF,

O=1,l=0,lll=O+O+O+l,OQ=056;
lOL*llL="%2x ";
(I != 1<<1&&(O1O(QI[0]),O10(1011-1010))),
((L = QLQ(Il[O],"r"))==0&&(O1O(QI[O],Il[O]),O10(O)));
lO = I-(O<<l<<O);
while (L-l,1)
{QO0(Q = 0L;((Q &~(0x10-O))== l);
OL[Q++] = OlQ(L));
if (OL[0]==llO) break;
O1O("\0454x: ",lO);
if (I == (1<<1))
{QO0(Q=Ol0(QI[O<<O<<1]);Q<Ol0(QI[0]);
Q++)O1O((OL[Q]!=llO)?llL:QI[lll],OL[Q]);/*"
O10(QI[1O])*/
O1O(QI[lll]);{}
}
QO0 (Q=0L;Q<1<<1<<1<<1<<1;Q+=Q<0100)
{(OL[Q]!=llO)? /* 0010 10lOQ 000LQL */
((D(OL[Q])==0&&(*(OL+O1Q(Q-l))=OQ)),
OlO(OL[Q])):
OlO(1<<(1<<1<<1)<<1);
}
O1O(QI[01^10^9]);
lO+=Q+0+l;}
}
D(l) { return l>=' '&&l<='\~';
}
```

Comments: Execute with a file name as an argument. And don't say you didn't get anything useful out of this article!

---

In the category of "best simple task performed in a complex way," the winner is Bruce Holloway, Digital Research, Monterey, CA.

```
#include "stdio.h"
#definee 3
#defineg (e/e)
#defineh ((g+e)/2)
#definef (e-g-h)
#definej (e*e-g)
#define k (j-h)
#definel(x) tab2[x]/h
#definem(n,a) ((n&(a))==(a))

long tabl[]={ 989L,5L,26L,0L,88319L,123L,0L,9367L };
int tab2[]={ 4,6,10,14,22,26,34,38,46,58,62,74,82,86 };

main(m1,s) char *s; {
    int a,b,c,d,o[k],n=(int)s;
    if(m1==1){ char b[2*j+f-g]; main(l(h+e)+h+e,b); printf(b); }
    else switch(m1-=h){
case f:
    a=(b=(c=(d=g)<<g)<<g)<<g;
    return(m(n,a|c)|m(n,b)|m(n,a|d)|m(n,c|d));
case h:
    for(a=f;a<j;++a)if(tabl[a]&&!(tabl[a]%((long)l(n))))return(a);
case g:
```

```
        if(n<h)return(g);
        if(n<j){n-=g;c='D';o[f]=h;o[g]=f;}
        else{c='\r'-'\b';n-=j-g;o[f]=o[g]=g;}
        if((b=n)>=e)for(b=g<<g;b<n;++b)o[b]=o[b-h]+o[b-g]+c;
        return(o[b-g]%n+k-h);
    default:
        if(m1-=e) main(m1-g+e+h,s+g); else *(s+g)=f;
        for(*s=a=f;a<e;) *s=(*s<<e)|main(h+a++,(char *)m1);
    }
    }
```

Comments: Execute. Understand.

---

In the category of "best nonsimple task performed in a complex way," the winner is Lennart Augustsson, Dept. of Computer Science, Chalmers University of Technology, Gothenburg, Sweden.

```
typedef struct n{int a:3,
b:29;struct n*c;}t;t*
f();r(){}m(u)t*u;{t*w,*z;
z=u->c,q(z),u->b=z->b*10,
w=u->c=f(),w->a=1,w->c=z->
c;}t*k;g(u)t*u;{t*z,*v,*p,
*x;z=u->c,q(z),u->b=z->b,v
=z->c,z->a=2,x=z->c=f(),x
->a=3,x->b=2,p=x->c=f(),p
->c=f(),p->c->a=1,p->c->c=
v;}int i;h(u)t*u;{t*z,*v,*
w;int c,e;z=u->c,v=z->c,q(
v),c=u->b,e=v->b,u->b=z->b
,z->a=3,z->b=c+1,e+9>=c&&(
q(z),e=z->b,u->b+=e/c,w=f(
),w->b=e%c,w->c=z->c,u->c=
w);}int(*y[4])()={r,m,g,h};
char *sbrk();main(){t*e,*p,*o;
o=f(),o->c=o,o->b=1,e=f(),
e->a=2,p=e->c=f(),p->b=2,
p->c=o,q(e),e=e->c,(void)write
(1,"2.",2);for(;;e=e->c){q(e),
e->b=write(1,&e->b["0123456789"],
1);}}t*f(){return i||(i=1000,
k=(t*)sbrk(i*sizeof(t))),k+--i;
}q(p)t*p;{(*y[p->a])(p);}
```

Comments: Execute. Works best with an infinite amount of stack space. Execute with as much stack as you can get.

---

In the category of "most illegible code," the winner is Michael H. Pawka, Naval Ocean Systems Center, San Diego, CA.

```
#include "stdio.h"
#define xyxx char
#define xyyxx putchar
#define xyyyxx while
#define xxyyyx int
#define xxxyyx main
#define xyxyxy if
#define xyyxyy '\n'
xyxx *xyx [] = {
"]I^x[I]k\\I^o[IZ~\\IZ~[I^|[I^l[I^j[I^}[I^n[I]m\\I]h",
"]IZx\\IZx[IZk\\IZk[IZo_IZ~\\IZ~[IZ|_IZl_IZj\\IZj]IZ}]IZn_IZm\\IZm_IZh",
"]IZx\\IZx[I^k[I\\o]IZ~\\IZ~\\I]|[IZl_I^j]IZ}]I^n[IZm\\IZm_IZh",
"]IZx\\IZx[IZk\\IZk[IZo_IZ~\\IZ~_IZ|[IZl_IZj\\IZj]IZ}]IZn_IZm\\IZm]IZh",
"]I^x[I]k\\IZo_I^~[I^|[I^l[IZj\\IZj]IZ}]I^n[I]m^IZh",'\0'};/*xyyxyxyxxxyxxxyy*/
xyxx *xyyx; xxyyyx xyyyx,xyyyyx,xyyyyyx=0x59,xyyyyyyx=0x29,/*yxxyxyyyxxyyyxyy*/
xxyx=0x68;xxxyy(){xyyyyx=0;xyyyxx(xyx[xyyyyx]){xyyx=xyx[xyyyyx++];/*xyyyxxyx*/
xyyyxx(*xyyx){xyyyx= *xyyx++-xyyyyyx;xyyyxx(xyyyx--)xyyxx(*xyyx-xyyyyyyx);/*x*/
xyxyxy(*xyyx==xxyx)xyyxx(xyyxyy);*xyyx++;}}}/*xyxyxyyyxxyxxxyyyxyyyxyxxyyy*/
```

Comments: This program speaks for itself.

---

And the grand prize (for "most well-rounded in confusion") goes to Larry Wall, System Development Corp., Santa Monica, CA.

```
#define _c(C)_ (C)&('|'+3):c_()(C)>>('\n'-3) __; /**/
#define C char*
#define keyboard ",,B3-u;.(&*5., /(b*(1\036!a%\031m,,,,,\r\n"
#define main(o,oo)oo(o){
#define _ ;case
C
#define c_(cc)c cc=
#define C_(sand)_O(sand)witch
o=keyboard;
#define __ ;break;
C
ccc(
cc)
C
cc;
{
C
cccc=
cc;int
#ifndef lint
```

```
#define keyboard "dijs QH.soav Vdtnsaoh DmfpaksoQz;kkt oa, -dijs"
#endif
c;
main(;c_(=(*cc);*cc++)c,for);
#define _O(s)s
main(0xb+(c>>5),C_(s))
_'\v'
:__ _'\f':
main(c,C_(s));
_c(8098)_c(6055)_c(14779)_c(10682)
#define O_(O)_O(O)stem(ccc(
_c(15276)_c(11196)_c(15150)
#define _C ;return
_c(11070)_c(15663)_c(11583)
}
__
default
:c_(+)o[c&__LINE__-007];
main(c_(-)'-'-1,C_(s))_
0214
:_
0216
:c_(+)025 _
0207
:c_(-)4 _
0233
:c_(+)' '-1;
}}c_(&)'z'+5;
}_C cccc;
}main(,cc)
C
#define O write(1,
c="O";
O_(sy) keyboard));
main(;;,for);
read(0,
c,1);*
c_(&)'~'+1
;O ccc(
c),
'\0');
main(*c,
C_(s));_
4
:O_(sy)";kkt -oa, dijszdijs QQ"))_C
_
13
:O o+' ',
3
)
#undef main
__ _ 127:O"\b \b",3)__
default
:O
c,1)
__}}}main(){
cc();
}
 oa, dijszdijs QQ"))_C
_
13
:O o+' ',
3
)
#undef main

__ _ 127:O"\b
```

Comments: Execute. Feed it characters from standard input. The C compiler documents this program during compilation. §

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

# Turbo Pascal Tools

## TURBO Advantage: Source Code Libraries for Turbo Pascal

This library of more than 220 routines, complete with source code, sample programs and documentation will save you hours developing and optimizing your programs!

Routines are organized and documented under the following categories: bit manipulation, file management, MS-DOS support, sorting, string operations, arithmetic calculations, data compression, differential equations, Fourier analysis and synthesis, matrices and vectors, statistics, and much more! All source code is included.

A detailed manual includes a description of the routine, an explanation of the methods used, the calling sequence, and a simple example. For MS/PC-DOS systems.

**Turbo Advantage** **Item #070 $49.95**

## STAT Toolbox for Turbo Pascal

Two statistical packages in one!

**A library disk and reference manual**
Use these powerful statistical routines to build your applications. Routines include: • statistical distribution functions • random-number generation • basic descriptive statistics • parametric and non-parametric statistical testing • bivariate linear regression, multiple and polynomial regression.

**A demonstration disk and manual**
This package incorporates the library of routines into a fully functioning statistical program. Two data management programs are included to facilitate the storage and maintenance of data.

Full source code is included. (For IBM PC's and compatibles. Turbo Pascal version 2.0 or later, and PC-DOS 2.0 or later are required.)

**STAT Toolbox** **Item #050 $69.95**

## TURBO Advantage Display: Form Generator for Turbo Pascal

*TURBO Display* includes a menu driven form processor, 30 Turbo Pascal procedures and functions to facilitate linking created forms to your program, and full source code and documentation. For MS-DOS systems.

Some of the *TURBO Advantage: Source Code Libraries for Turbo Pascal* routines are necessary to compile *TURBO Display.*

**TURBO Display** **Item #072 $69.95**

## The Turbo Pascal Toolbook

### Edited by Namir Clement Shammas

You'll find:

- an extensive library of low-level routines
- external sorting and searching tools, presenting a new database routine that combines the best features of the B-tree, B+ and B++ trees
- window management, to help you create, sort and overlay windows
- artificial intelligence techniques
- mathematical expression parsers, offering two routines that convert mathematical expressions into RPN tokens
- a smart statistical regression model that searches for the best regression model to represent a given set of data.

All routine libraries and sample programs are on disk for MS-DOS systems, and over 800K of Turbo Pascal source code is included!

**Turbo Pascal Toolbook** **Item #080 $25.95**
**Turbo Pascal Toolbook with disk** **Item #081 $45.95**

## TURBO Advantage Complex: Complex Number Routines for Turbo Pascal

**TURBO Complex** provides procedures for performing all the arithmetic operations and necessary real functions with complex numbers. Each procedure is based on predefined constants and types. By using these declarations the size of arrays are easily adapted. Each type declaration is a record with both a real and imaginary part. Use these procedures to build more sophisticated functions in your own programs.

**TURBO Complex** also demonstrates the usage of these procedures in routines for vector and matrix calculation with complex numbers and variables; simultaneous Fourier transforms; calculations of convolution and correlation functions; low-pass, high-pass, band- pass and band-rejection digital filters; and solving linear boundary-value problems.

Source code and documentation is included. For MS-DOS systems. Some of the *TURBO Complex* routines are most effectively used with routines contained in *TURBO Advantage.*

**TURBO Complex** **Item #071 $89.95**

**TO ORDER:** Return this coupon with your payment to: M&T Books, 501 Glaveston Dr., Redwood City, CA 94063. Or, Call TOLL-FREE 800-533-4372 Mon-Fri 8a.m.-5p.m. In CA call 800-356-2002

**YES!** Please send me

- ☐ **TURBO Advantage** # 070 $49.95 ______
- ☐ **TURBO Advantage Display** # 072 $69.95 ______
- ☐ **TURBO Advantage Complex** # 071 $89.95 ______
- ☐ **STAT Toolbox** # 050 $69.95 ______
- ☐ **Turbo Pascal Toolbox** # 080 $25.95 ______
- ☐ **with disk** # 081 $45.95 ______

Subtotal ______
CA residents add salestax ____% ______
Add $2.25 per item for shipping ______
TOTAL ______

Name ______
Address ______
City ______
State ______ Zip ______
☐ Check enclosed. Make payable to M&T Publishing.
Please charge my ☐ VISA ☐ M/C ☐ AMEX
Card no. ______
Expiration Date ______
Signature ______

# Multitasking Between Programs

by Philip R. Burns

The emphasis here is not on multitasking *within* a program, but multitasking *between* programs.

### MS-DOS: A ONE-TASK OPERATING SYSTEM

MS-DOS (or PC-DOS) is designed to handle only one task at a time. DOS does not support concurrency: it lacks good methods for controlling access to machine resources like memory, the display, other peripherals, and even DOS itself. Neither DOS nor the IBM BIOS is reentrant. (*Reentrant* means that more than one task can use the same code at the same time.)

Even when DOS provides standard mechanisms for accessing peripherals, efficiency can be poor. For example, clever programmers early on discovered that writing directly to the display memory of the PC boosted the speed of the display operation by orders of magnitude. Using the standard DOS or BIOS calls to update the display proved to be unacceptably slow.

### CONTEXT SWITCHING

Other clever programmers discovered methods for loading several programs into memory at once, using the *terminate and stay resident* facilities of DOS. Typing a specific key sequence allowed for context switching—the active program stopped running, and the resident program started running. This was not concurrent programming. Only one of the loaded programs actually ran at a time, but often this proved to be extremely convenient. Many popular utilities like Borland's SideKick resulted from this idea. As more and more of these resident utilities cropped up, however, more and more memory space on a typical PC ended up allocated to them. This left less memory for larger applications. Also, the different resident programs frequently conflicted with each other, since each used a different, nonstandard method of doing whatever it was doing.

### GETTING TRUE CONCURRENCY

These problems led programmers to create products that provided true concurrency. A control program was loaded that took over parts of DOS and the BIOS, usually to add the missing reentrancy and to improve performance of common operations. Examples of this kind of concurrent processor include SoftLogic's DoubleDOS and The Software Link's MultiLink. (Other multitaskers like IBM's TopView, Quarterdeck's DesqView, Sunny Hill Software's TaskView, and Microsoft's Windows also basically work this same way, but these multitaskers provide many more facilities. We will talk about them later.)

Such concurrent environments retain the "hot-key" method of context switching as regards the display. The task that "owns" the display and the keyboard is called the *foreground task.* The other programs are called *background tasks.* The foreground task has the entire display to itself; the screen is not split in any way between tasks.

### TIME SLICING

Task switching occurs independently of user action, using the timer interrupt facilities of the PC. Every so often, the currently executing program, whether the foreground program or one in the background, is suspended, its current operating state saved, and another program given control of the processor. (Saving the current state requires that all registers, interrupt vectors, display buffer areas, and so on, be remembered. It is exactly those facilities that are lacking in DOS itself.)

Tasks are scheduled in round-robin order. The amount of time each gets to execute is called its *time slice.* Each of the multitaskers provides different mechanisms for varying the time slice allocated to each task. Some tasks—particularly those that are monitoring real-time events—may need higher priority and longer time slices than others. For example, communications programs may need large time slices to ensure that characters arriving over the serial port are not lost because the task assigned to read the port wasn't active when the character arrived. Some multitaskers boost the priority of communications programs to help in this regard. For example, DesqView provides a high-speed-comm option. The best approach is to write special interrupt handlers for asynchronous I/O. In particular, both an input and output to the serial ports should be buffered and interrupt driven. The standard DOS and BIOS facilities for accessing the serial port are not reliable at high speeds or under multitaskers.

### VIRTUAL SCREEN BUFFERS

The multitaskers intercept the DOS and BIOS functions for the display to prevent the output of background jobs from incorrectly mingling with the output of the foreground jobs. But how then are megalomaniacal programs that write directly to the screen memory buffer handled? The fact that a program is running in the background doesn't prevent it automatically from splattering its output all over the screen if it writes directly to the screen buffer memory.

The solution involves the concept of a virtual screen buffer. Each task gets an area of memory that is the same size as the real screen display buffer. If a task is running in the background, this virtual buffer is updated, rather than the real screen buffer. When a task is switched to the foreground, the contents of this virtual buffer are copied to the real screen buffer so that they appear on the physical display. When a foreground task is switched

*Listing 1*

```
(*--------------------------------------------------------------------*)
(*                  Global 8088/8086/8028x register definitions       *)
(*--------------------------------------------------------------------*)

CONST
                  (* 8086/8088 hardware flags *)

   Carry_Flag     = 1;
   Parity_Flag    = 4;
   Aux_Carry_Flag = 16;
   Zero_Flag      = 64;
   Sign_Flag      = 128;

TYPE

   RegPack  = RECORD        (* 8086/8088 registers                    *)
                CASE INTEGER OF
                   1: ( Ax, Bx, Cx, Dx, Bp, Si, Di, Ds, Es, Flags : INTEGER );
                   2: ( Al, Ah, Bl, Bh, Cl, Ch, Dl, Dh            : BYTE    );
             END;

TYPE
   AnyStr   = STRING[255];

(*--------------------------------------------------------------------*)
(*          Definitions for screen memory reading/writing             *)
(*--------------------------------------------------------------------*)

CONST
   Color_Screen_Address   = $B800    (* Address of color screen       *);
   Mono_Screen_Address    = $B000    (* Address of mono screen        *);
   Screen_Length          = 4000     (* 80 x 25 x 2 = screen area length *);
   Graphics_Screen_Length = 16000    (* Length of graphics screen area *);
   Max_Saved_Screen       = 6        (* Maximum no. of saved screens  *);
   MedRes_GraphMode       = 4        (* Medium resolution graphics    *);
   HiRes_GraphMode        = 6        (* High resolution graphics mode *);
   Mono_TextMode          = 7        (* Monochrome adapter text mode  *);
   CRT_Index              = $03D4    (* CRT index register            *);
   CRT_Data               = $03D5    (* CRT data register             *);
   CRT_Mode               = $03D8    (* CRT mode register             *);
   CRT_Color_Select       = $03D9    (* CRT color select register     *);
   CRT_Status             = $03DA    (* CRT status port               *);
   CRT_Mode_Byte          = $0465    (* CRT mode byte                 *);

TYPE
                                     (* A screen image          *)

   Screen_Type       = ARRAY[ 1 .. Graphics_Screen_Length ] OF BYTE;
   Screen_Ptr        = ^Screen_Type;

VAR
   Write_Screen_Memory : BOOLEAN     (* TRUE to allow direct screen writes *);
   Wait_For_Retrace    : BOOLEAN     (* TRUE to wait for retrace signals   *);
```

*Listing 2*

```
(*--------------------------------------------------------------------*)
(*                      Multitasker definitions                       *)
(*--------------------------------------------------------------------*)

TYPE
   MultiTaskerType     = ( MultiTasker_None, DoubleDos, DesqView, TopView,
                           MSWindows, APXCore, EZDosIt, Concurrent_DOS,
                           TaskView, MultiLink, Other );

VAR
   TimeSharingActive   : BOOLEAN  (* TRUE if multitasker active       *);

                                  (* Which multitasker active         *)
   MultiTasker         : MultiTaskerType;

   Virtual_Screen      : Screen_Ptr (* Alternate display buffer address *);

-----------------------------------------------------------------

(*--------------------------------------------------------------------*)
(*     EGA_Installed  ---  Test if Enhanced Graphics Adapter installed *)
```

to the background, the real screen buffer is copied to that task's virtual buffer. DOS and BIOS calls will reference the real or virtual buffer, depending on whether the current task is the foreground or background task. Instead of writing to the real screen buffer, a program can write directly to the virtual screen buffer. Changes made to the virtual screen buffer will be reflected when the background task next becomes the foreground task. Direct screen writing programs can be made to behave by modifying them—either at the source level or via binary patches—to write to the virtual screen buffer rather than the real screen buffer.

As a plus, some of the multitaskers enhance the performance of the video I/O routines, so that even when only one program is running under the multitasker, screen I/O proceeds more rapidly than under ordinary DOS. This is notable under DoubleDOS and TaskView. The annoying flicker present on the color graphics text screen can also be eliminated on many PC clones.

For example, on the Leading Edge model D, ordinary DOS results in the flickering scroll typical of the the genuine IBM Color Graphics Adapter. The Leading Edge machine, however, has dual-ported display memory, and so TaskView can take advantage of that and completely eliminate flicker when scrolling in the color graphics text mode.

Such performance enhancements and aesthetic improvements are another valuable by-product of the multitaskers.

## DEALING WITH DISK I/O

DOS does not provide any mechanisms for overlapping multiple disk I/O requests. This means that a disk I/O request, once started, essentially proceeds until complete. In fact, while one task is executing within DOS, no other task can even begin a DOS function. This can delay task switching for unacceptable lengths of time. The multitaskers deal with this problem by breaking up long disk I/O requests into smaller, more manageable pieces that can be scheduled more easily. For example, suppose one task requests a read of 10,000 bytes from a disk file, and another task requests a write of 5,000 bytes to a disk file. Both of these requests will be broken up into smaller pieces—say, reading or writing a few hundred bytes at a time—so that a task will not exceed its allotted time slice.

Some multitaskers also emulate some DOS functions themselves in order to obtain more control. TopView does this for

```
(*----------------------------------------------------------------------*)

FUNCTION EGA_Installed : BOOLEAN;

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*     Function:   EGA_Installed                                        *)
(*                                                                      *)
(*     Purpose:    Checks if Enhanced Graphics Adapter is installed.    *)
(*                                                                      *)
(*     Calling Sequence:                                                *)
(*                                                                      *)
(*        EGA_There := EGA_Installed : BOOLEAN;                         *)
(*                                                                      *)
(*           EGA_There --- TRUE if EGA installed                        *)
(*                                                                      *)
(*     Calls:   INTR                                                    *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN  (* EGA_Installed *)
                                       (* Determine if EGA installed      *)
   Regs.AH := $12;
   Regs.BX := $FF10;
   INTR( $10 , Regs );

   IF ( Regs.BH = $FF ) THEN          (* EGA not installed *)
      EGA_Installed := FALSE
   ELSE IF ( Regs.CL = 9 ) THEN
      BEGIN (* EGA present with enhanced display *)
         EGA_Installed := TRUE;
      END
   ELSE IF ( Regs.CL = 13 ) THEN
      BEGIN (* EGA present with monochrome display *)
         EGA_Installed := TRUE;
      END
   ELSE (* EGA present but with old color display  *)
      EGA_Installed := FALSE;

END    (* EGA_Installed *);

----------------------------------------------------------------

(*----------------------------------------------------------------------*)
(*     Current_Video_Mode --- Determine current video mode setting      *)
(*----------------------------------------------------------------------*)
```

*Listing 3*

```
FUNCTION Current_Video_Mode: INTEGER;

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*     Function:   Current_Video_Mode                                   *)
(*                                                                      *)
(*     Purpose:    Gets current video mode setting from system          *)
(*                                                                      *)
(*     Calling Sequence:                                                *)
(*                                                                      *)
(*        Current_Mode := Current_Video_Mode : INTEGER;                 *)
(*                                                                      *)
(*           Current_Mode --- set to integer representing current       *)
(*                            video mode inherited from system.         *)
(*                                                                      *)
(*     Calls:   INTR                                                    *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN  (* Current_Video_Mode *)

   Regs.Ax := 15 SHL 8;

   INTR( $10 , Regs );

   Current_Video_Mode := Regs.Al;
```

many I/O functions.

## MEMORY MANAGEMENT AND TASK SWITCHING

Some of the multitaskers impose fixed sizes on the task partitions when they first start executing. DoubleDOS is an example. (The memory allocation can be changed later, as long as no tasks are running.) Other multitaskers like TopView, TaskView, and DesqView provide for dynamic memory allocation through the use of program information files that tell how much memory a given application will need. These program information files may also indicate other special requirements, such as what interrupts must be saved/restored, whether graphics memory must be saved/restored, whether the program uses the 8087/80287 math coprocessor, whether the program accesses the serial ports, and so on. The more information a multitasker has about a program, the more efficiently it can switch tasks. For example, if a program does not use the 8087, there is no reason to save and restore the state of the 8087. Likewise, if a program doesn't use use graphics, there is no reason to save graphics memory, particularly the possible large memory assigned to the enhanced graphics adapter (EGA).

TaskView also provides the OPEN and SPAWN DOS-level commands that allow the commencement of foreground and background jobs from the DOS command line and even from batch files.

## SWAPPING PROGRAMS TO DISK

A major limitation of multitasking, as described previously, is that all the programs to be executed need to be in memory at the same time. This limits the combined sizes of each task to less than the IBM-imposed limit of 640K. (Actually, the limit can be raised to 704K in many cases, and some multitaskers, such as DesqView, DoubleDOS, and TaskView, allow for that.)

The availability of faster hard disk drives and inexpensive memory expansion products like Intel's Above Board and AST's RamPage have led to the addition of facilities for swapping programs in and out to disk or the expanded/extended memory. This allows many large programs to be executed concurrently. QuarterDeck's DesqView, Microsoft's Windows, and Sunny Hill's Task View provide such swapping facilities. Swapping can consume considerable time if the swapping device is slow; that is why it is preferable to use the fast-access expanded/extended memory rather than disks.

Also, some programs cannot be swapped out. For example, it is not feasible to swap out a communications program or indeed any program designed to deal with real-time events.

## TASK INTEGRATION

The introduction of Apple's Macintosh led to an upsurge in interest in window-based user interfaces. Macintosh applications could be integrated through a common user interface for all programs. In addition, data could be exchanged between applications in a standard way. This led developers to add windowing and at least simple cut-and-paste operations to the concurrent environments. Examples include DesqView, TopView, and—in a much more advanced way—Windows. With these environments, the output of several concurrently running tasks can appear simultaneously on the screen, either in adjacent (*tiled*) windows or in overlapping windows. Data can be exchanged by cut-and-paste operations or sending data from one task to another via message passing.

## MICROSOFT WINDOWS

Microsoft Windows is a special case in many respects. It does not schedule tasks using time slicing. Instead, it waits for a program to make a DOS request before switching to another task. This means that certain types of programs—notably, number crunchers—can take over the processor for extended lengths of time.

It is possible, however, to run Windows itself as a task under one of the other multitaskers and thereby obtain the benefits of the Windows environment and still have time-sliced multitasking. At this time, I have had the best success with running Windows under TaskView.

Future versions of DOS are rumored to include Windows as the primary user interface. True time-sliced task switching will be built into DOS. It will be interesting to see what happens!

## CONCURRENT PC-DOS

Digital Research takes a more ambitious approach to multitasking. Concurrent PC-DOS is an operating system that completely replaces DOS. All the facilities mentioned previously—multiple tasks, menu interface, access to extended memory, swapping, and so on—are provided by Concurrent DOS. In addition to being multitasking, Concurrent PC-DOS is multiuser. Extra terminals can be attached to the PC's serial ports, so up to

```
END     (* Current_Video_Mode *);

------------------------------------------------------------

(*--------------------------------------------------------------------*)
(*     Current_Video_Mode --- Determine current video mode setting     *)
(*--------------------------------------------------------------------*)

FUNCTION Current_Video_Mode: INTEGER;

(*--------------------------------------------------------------------*)
(*                                                                    *)
(*     Function:   Current_Video_Mode                                 *)
(*                                                                    *)
(*     Purpose:    Gets current video mode setting from system        *)
(*                                                                    *)
(*     Calling Sequence:                                              *)
(*                                                                    *)
(*        Current_Mode := Current_Video_Mode : INTEGER;               *)
(*                                                                    *)
(*           Current_Mode --- set to integer representing current     *)
(*                            video mode inherited from system.       *)
(*                                                                    *)
(*     Calls:   INTR                                                  *)
(*                                                                    *)
(*--------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN  (* Current_Video_Mode *)

   Regs.Ax := 15 SHL 8;

   INTR( $10 , Regs );

   Current_Video_Mode := Regs.Al;

END     (* Current_Video_Mode *);
```

*Listing 4*

```
(*--------------------------------------------------------------------*)
(*    Color_Screen_Active --- Determine if color or mono screen       *)
(*--------------------------------------------------------------------*)

FUNCTION Color_Screen_Active : BOOLEAN;

(*--------------------------------------------------------------------*)
(*                                                                    *)
(*     Function:   Color_Screen_Active                                *)
(*                                                                    *)
(*     Purpose:    Determines if color or mono screen active          *)
(*                                                                    *)
(*     Calling Sequence:                                              *)
(*                                                                    *)
(*        Color_Active := Color_Screen_Active : BOOLEAN;              *)
(*                                                                    *)
(*           Color_Active --- set to TRUE if the color screen is      *)
(*                            active, FALSE if the mono screen is     *)
(*                            active.                                 *)
(*                                                                    *)
(*     Calls:   INTR                                                  *)
(*                                                                    *)
(*--------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN  (* Color_Screen_Active *)

   Regs.Ax := 15 SHL 8;

   INTR( $10 , Regs );

   Color_Screen_Active := ( Regs.Al <> 7 );

End     (* Color_Screen_Active *);
```

*Listing 5*

```
(*--------------------------------------------------------------------*)
(*       Get_Screen_Address --- Get address of current screen         *)
(*--------------------------------------------------------------------*)

PROCEDURE Get_Screen_Address( VAR Actual_Screen : Screen_Ptr );

(*--------------------------------------------------------------------*)
(*                                                                    *)
(*     Procedure:  Get_Screen_Address                                 *)
(*                                                                    *)
(*     Purpose:    Gets screen address for current type of display    *)
(*                                                                    *)
(*     Calling Sequence:                                              *)
(*                                                                    *)
(*        Get_Screen_Address( VAR Actual_Screen : Screen_Ptr );       *)
(*                                                                    *)
(*           Actual_Screen --- pointer whose value receives the       *)
(*                             current screen address.                *)
(*                                                                    *)
(*     Calls:   Color_Screen_Active                                   *)
(*              PTR                                                   *)
(*              TimeSharingActive                                     *)
(*                                                                    *)
(*     Remarks:                                                       *)
(*                                                                    *)
(*        This routine assumes that 'IsTimeSharingActive' has already *)
(*        been called so that the value of 'Virtual_Screen' is defined. *)
(*                                                                    *)
(*--------------------------------------------------------------------*)

VAR
   Regs: RegPack;

BEGIN  (* Get_Screen_Address *)

   CASE MultiTasker OF

      DoubleDos:  BEGIN
                     Regs.Ax := $EC00;
                     MsDos( Regs );
                     Actual_Screen := PTR( Regs.Es, 0 );
                  END;
                                 (* For TopView family, if graphics mode, *)
                                 (* we must return actual screen address, *)
                                 (* not virtual buffer address.  The      *)
                                 (* virtual buffer is only for the        *)
                                 (* text modes.                           *)
      TaskView,
      TopView,
      MSWindows,
      DesqView:   IF ( Current_Video_Mode <> HiRes_GraphMode ) THEN
                     Actual_Screen := Virtual_Screen
                  ELSE
                     Actual_Screen := PTR( Color_Screen_Address , 0 );

      ELSE
                  IF Color_Screen_Active THEN
                     Actual_Screen := PTR( Color_Screen_Address , 0 )
                  ELSE
                     Actual_Screen := PTR( Mono_Screen_Address , 0 );

   END  (* CASE *);

END    (* Get_Screen_Address *);
```

*Listing 6*

```
(*--------------------------------------------------------------------------*)
(*             PIBMDOS.PAS --- Multitasker interface routines               *)
(*--------------------------------------------------------------------------*)
(*                                                                          *)
(*  Author:  Philip R. Burns                                                *)
(*                                                                          *)
(*  Date:    Version 1.0: January, 1986.   DoubleDos support.               *)
```

three users can be using one PC at a time (*Some systems support as many as 16 users—Ed.*).

## THE PRODUCTS

I will be discussing interfacing Turbo Pascal programs to the following multitaskers:

| | |
|---|---|
| TopView | (IBM) |
| DesqView | (QuarterDeck Systems) |
| TaskView | (Sunny Hill Software) |
| DoubleDOS | (SoftLogic Solutions) |
| Windows | (Microsoft) |

It turns out that TopView, DesqView, Windows, and TaskView can all use the same basic approach, since DesqView, TaskView, and Windows emulate TopView calls. (You must tell Windows to perform the emulation by setting the appropriate switch in the .PIF file for a program.) TopView extends the functions available through Interrupt $10—the video interrupt—to include facilities for dealing with virtual buffers. TopView also adds Interrupt $15 to provide task-management services of other kinds, such as turning time-sharing on and off.

I do not discuss the more complex interfaces provided by Windows itself.

DoubleDOS does not emulate TopView calls (even though its documentation claims that it does), and so a different mechanism is required for interfacing to DoubleDOS. DoubleDOS has its own set of interrupts for providing task management. This difference is hidden, however, by the higher-level calls, so you do not need to worry about the difference in most instances.

Similar methods should apply to other available multitasking environments, including Digital Research's Concurrent DOS and The Software Link's MultiLink. I do not own MultiLink. I do have Concurrent DOS, but I have not tried interfacing programs to it. Regrettably, it does not provide emulation of the TopView multitasking calls.

## COMPARING FEATURES & EFFICIENCY

Table 1 compares the various multitaskers in terms of features.

The question as to which multitasker is most efficient in terms of processor time is difficult to answer. My experience with a 4.77-MHz processor is that DoubleDOS and TaskView are the most efficient, with no evident loss of speed for single tasks when loaded. Both handle two tasks admirably. DesqView is next, and TopView and Windows bring up the rear. On an 80286 processor, DoubleDOS, TaskView, and DesqView all perform adequately. Windows and TopView are notably slower.

## OVERVIEW OF INTERFACE ROUTINES

The software routines presented in this article provide a very simple common interface to the multitaskers mentioned previously. The facilities include:

1. determining which, if any, multitasker is executing
2. determining the address of the virtual display buffer for a task
3. giving up idle time to other tasks
4. starting and stopping time-sharing to freeze and unfreeze time-critical tasks
5. routines for writing characters and strings directly to the real or virtual display screen memory

These routines are by no means exhaustive; all the multitaskers provide many other facilities not considered here. These routines, however, give you a foundation on which to construct more complicated interfaces if you need them.

## PROGRAM LISTINGS

Here are the contents of each of the program listings:

| | |
|---|---|
| Listing 1: | file TRYMDOS.GLO —global declarations for multitasker interface |
| Listing 2: | file EGAINST.PAS —determines if enhanced graphics adapter installed |
| Listing 3: | file CURVID.PAS —obtain current video display mode |
| Listing 4: | file COLORSCA.PAS —determine if color screen installed |
| Listing 5: | file GETSCREN.PAS —get address of screen memory |
| Listing 6: | file PIBMDOS.PAS —the interface routines themselves |
| Listing 7: | file WRITESXY.PAS —writes string to display |
| Listing 8: | file WRITECXY.PAS —writes character to display |
| Listing 9: | file READKBD.PAS —reads character from keyboard |
| Listing 10: | file SCROLL.PAS —scrolls screen using BIOS |
| Listing 11: | file MOVETOXY.PAS —move cursor to specified screen location |
| Listing 12: | file TRYMDOS.PAS |

| | DesqView | DoubleDOS | TopView | TaskView | Windows |
|---|---|---|---|---|---|
| Tasks allowed | 9 | 2 | Limited by memory | 10 | Unlimited |
| Memory requirements | 141K | 20-70K | 190K | 57K | 145K |
| Dynamic allocation | Yes | No | Yes | Yes | Yes |
| Swapping to disk allowed | Yes | No | No | Yes | Yes |
| Time-sliced scheduling | Yes | Yes | Yes | Yes | No |
| Removable without reboot | Yes | No | Yes | Yes | Yes |
| Customizable menus | Yes | No | Yes | Yes | Yes |
| DOS commands to start tasks | No | No | No | Yes | No |
| TopView compatibility | Yes | No | Yes | Yes | Yes |
| Multiple window support | Yes | No | Yes | No | Yes |
| Program information files | Yes | No | Yes | Yes | Yes |
| Boost priority of comm. progs | Yes | No | No | No | ? |
| Cut and paste between progs | Yes | No | Yes | Yes | Yes |

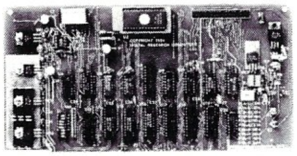
**Table 1. Comparison of Multitasker Features**

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.*

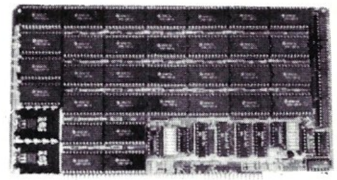


PC BOARD SET, FULL DOCUMENTATION, 8 IN. DISKETTE WITH SOFTWARE.

NEW! $69.95

## 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.

## 128K S100 STATIC RAM/EPROM BOARD

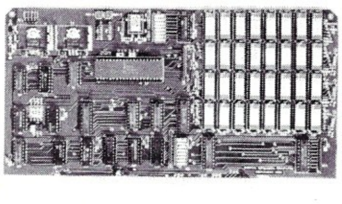
*JUST OUT! USES POPULAR 8K X 8 STATIC RAMS (6264) OR 2764 EPROMS. FOR 8 OR 16 BIT DATA TRANSFERS! IEEE 696 STANDARD. LOW POWER. KITS ARE FULLY SOCKETED. FULL DOC AND SCHEMATICS INCLUDED. 24 BIT ADDRESSING.*

NEW! $59.95 BARE PC BOARD — $219.00 128K RAM KIT — $139.00 128 EPROM KIT

## 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

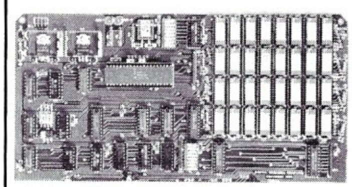
#LS-100

## 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!***

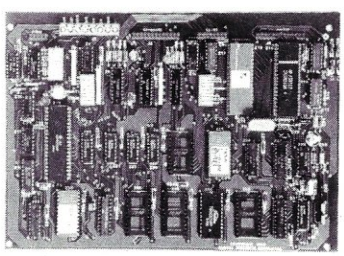
## ZRT-80 CRT TERMINAL BOARD!

*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!

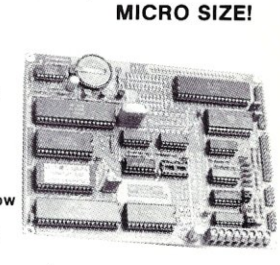
★ FROM LINGER ENTERPRISES ★

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.



$99.95 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

—sample program to demonstrate interface

Some of the routines are written using the Turbo INLINE facility. This allows machine code to be inserted within a Turbo program. The corresponding assembler code always accompanies any INLINE machine code. The code was converted from assembler to INLINE statements using the excellent, public-domain INLINE.COM assembler by Dave Baldwin.

For the most part, you should examine the code and the comments therein to see how the various routines work. Following are a few pointers about what is going on in the code.

## IMPORTANT GLOBAL VARIABLES

There are several important global variables used in the interface code that you should know about. Virtual_Screen contains the address of the real screen buffer if no multitasker is operating. This will be $B800:0000 if a color graphics adapter is connected, or $B000:0000 for a monochrome adapter. If a multitasker is operating, Virtual_Screen contains the address of the virtual screen buffer. The routines WriteSXY and WriteCXY store strings and characters at specified row and column positions in the real/virtual screen buffer.

MultiTasker indicates which multitasker is active. If no multitasker is active, MultiTasker has the value Multitasker_None.

TimeSharingActive is true if a multitasker is active; otherwise, it is false.

Wait_For_Retrace should be set true initially if your color graphics adapter produces snow when the real screen buffer memory is accessed. For a genuine IBM Color Graphics Adapter, Wait_For_Retrace should be set true. Wait_For_Retrace should be set false for many compatibles like the Compaq or Zenith, whose video-display circuitry handles retrace interlocks automatically. Wait_For_Retrace may be set false by the interface code, depending on which multitasker is determined to be active. Initially, Wait_For_Retrace is set false if an enhanced graphics adapter or a monochrome adapter is found.

Write_Screen_Memory should be set true to allow the routines WriteSXY and WriteCXY to access the screen buffer directly. This may be either the real screen buffer or the virtual screen buffer when a multitasker is active. Write_Screen_Memory can be set false to cause WriteSXY and WriteCXY to use the BIOS

```
(*          Version 2.0: April, 1986.     Add DesqView support.            *)
(*          Version 3.0: July, 1986.      Add TopView/Windows support.     *)
(*          Version 3.1: September, 1986. Update for TaskView support.     *)
(*          Version 3.2: December, 1986.  Distinguish TaskView/DesqView.   *)
(*                                                                         *)
(*  Systems: MS DOS or PC DOS with DoubleDos/DesqView/TopView/Windows      *)
(*          installed.                                                     *)
(*                                                                         *)
(*  History: These routines provide a simple interface for PibTerm         *)
(*          with SoftLogic's DoubleDos multitasking executive,             *)
(*          Quarterdeck's DesqView multitasker, IBM's TopView,             *)
(*          MicroSoft's Windows, and Sunny Hill's TaskView.                *)
(*          (Windows is handled as a Topview-emulating product.  This is   *)
(*          also true for TaskView and DesqView, but those programs do     *)
(*          not require the explicit screen updates TopView requires.      *)
(*                                                                         *)
(*          If you have another multitasker, you should be able to         *)
(*          replace these routines fairly easily with similar-acting       *)
(*          ones for your multitasker.  Use the global types defined       *)
(*          for MultiTasker and MultiTaskerType.                           *)
(*                                                                         *)
(*          Note also that the routine Get_Screen_Address in Pibscren.pas  *)
(*          needs to know about multitaskers.                              *)
(*                                                                         *)
(*          With DoubleDos, it is necessary to reobtain the display buffer *)
(*          address every time the screen memory is written to.  With      *)
(*          DesqView, this is unnecessary.  With TopView and Windows,      *)
(*          it is necessary to inform them that the screen has changed.    *)
(*          TaskView works like DesqView.                                  *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*          Please leave messages on Gene Plantz's BBS (312) 882 4145      *)
(*          or Ron Fox's BBS (312) 940 6496.                               *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

(*-------------------------------------------------------------------------*)
(*          IsTimeSharingActive --- Checks if multitasker is active        *)
(*-------------------------------------------------------------------------*)

FUNCTION IsTimeSharingActive : BOOLEAN;

(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*    Function: IsTimeSharingActive                                        *)
(*                                                                         *)
(*    Purpose:  Checks if multitasker is active                            *)
(*                                                                         *)
(*    Calling Sequence:                                                    *)
(*                                                                         *)
(*       Ts_On := IsTimeSharingActive : BOOLEAN;                           *)
(*                                                                         *)
(*          Ts_On --- TRUE if multitasker is active.                       *)
(*                                                                         *)
(*    Calls:  MsDos                                                        *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

VAR
   Regs : RegPack;

(*-------------------------------------------------------------------------*)

FUNCTION Get_TopView_Screen_Address : BOOLEAN;

VAR
   SegS : INTEGER;
   SegO : INTEGER;

BEGIN (* Get_TopView_Screen_Address *)

   Regs.Di := 0;
   Regs.Ax := $FE00;

   IF Color_Screen_Active THEN
      Regs.Es := Color_Screen_Address
   ELSE
      Regs.Es := Mono_Screen_Address;

   SegO    := 0;
```

for screen access.

## VIRTUAL BUFFER USAGE

The routine IsTimeSharingActive determines if one of the supported multitaskers is active. If so, Virtual_Screen points to the virtual screen buffer, and MultiTasker indicates which of the multitaskers is active.

Under TopView, DesqView, TaskView, and Windows, the location of the virtual buffer for a task remains the same throughout the lifetime of the task. This means that the virtual screen buffer can always be updated in preference to updating the real screen buffer. In TopView and Windows, once an update is performed using a direct screen write, it is necessary to inform TopView/Windows that the virtual buffer has been changed. Then TopView/Windows will update the real screen memory (and thereby the physical display) to match the virtual buffer contents. The routines Sync_Screen and Sync_Entire_Screen perform this function.

Note that if standard BIOS and DOS calls are used, it is not necessary to worry about screen synchronization; the multitaskers handle it automatically. Also, even when the virtual buffer is written to directly, TaskView and DesqView figure out themselves whether the virtual buffer has been changed, using a polling algorithm, and they update the real display automatically to match.

With DoubleDOS, the location of the display changes depending on whether the task is the foreground task or the background task. Regrettably, it is necessary to re-obtain the current buffer address every time information is to be written to the display memory. Further, since the real screen buffer address is returned if the task is the foreground task, it is necessary to worry about retraces to prevent snow on color graphics adapters. Also, to prevent a task switch from occurring in the middle of a long screen update—therefore causing an incorrect display—it is necessary to freeze a task before the screen write and then unfreeze it afterwards. Since this is a lot of work, writing single characters to the display using direct screen writes with DoubleDOS tends to be slow. It is simpler to use the BIOS call for a single character. (This is what the routine WriteCXY, included here, does.)

The routines TurnOnTimeSharing and TurnOffTimeSharing demonstrate how to freeze and unfreeze programs. For most programs, these routines need to be called only for DoubleDOS, as described previously.

The routine Get_Screen_Address gets the address of the current screen buffer—either the real screen or the virtual screen.

## DONATING TIME TO OTHER TASKS

Sometimes a task will be sitting in a tight loop waiting for some external event to occur. For example, a communications program may stay in a tight loop waiting for a character to appear over the serial port. Or a program may be sitting and waiting for a character to be typed at the keyboard. Such loops degrade the overall performance of a system running under a multitasker. This is because processor time is wasted on the program executing the busy wait.

One way to increase performance in these situations is for the program in the busy wait condition to give up its time slice to other tasks while it is waiting for the external event. The routine GiveAwayTime implements this strategy. The sample main program TryMDos includes a test of a busy wait keyboard read loop (routine TryGiveAway).

*Continued on page 56*

# Communicating at 9600 bps

by Steve Bosak and John Sojak

## A Review of Two More High-Speed Dial-Up Modems—Part II

In Part I of our article (January/Feburary 1987), we discussed half-duplex, high-speed modems. These half-duplex, or simulated full-duplex modems, are the most prevalent and have been on the shelves for a long time. Part of the reason the V.29 and related modems lead in implementing high-speed data transfers over dial-up lines is the reliance on fax and fax-like hardware engines at the heart of their technologies. Of all the high-speed, half-duplex modems released so far, only the Telebit Trailblazer shows a significantly original design.

Simulated full-duplex modems have their limitations, however. As pointed out in the last issue, the V.29 and related modems switch the lines selectively on demand. If data is suddenly transmitted from a modem that was previously a receiver, the turnaround must occur prior to any transfer. In some modems the turnaround time required when changing from data-file transmission to terminal-to-terminal communication can be irritating. Also, some significant high-speed applications could be seriously impaired or completely barred by delays in the datastream switching.

Unattended file transfers can be extremely tricky on a few of these modems. Although you can anticipate the turnaround times in setting up a macro file for automatic log-on and transfers, the margin for error is very high. A perfectly good unattended log-on can become frustrated by sudden data bursts and delays turning the lines around between queries and answers.

High-speed transfers in both directions are downright impossible, and interactive graphics of fax representations can never be used with such turnaround delays. File transfer and query of the remote is also impossible—for instance, what if recently uploaded data needed to be run on the remote end, with the results in turn being captured by the originating modem? If these scenarios sound farfetched and will seldom be a concern for you or your business, then half-duplex, high-speed modems will probably be more than adequate for your needs. Never underestimate the world of telecommunications, though—half-duplex may be a good first step, but a full-duplex implementation would be more reassuring.

There are currently at least two-and-one-half implementations of full-duplex, 9600-bps modems on the market. Concord Data has a full-duplex implementation, but at this writing, only in synchronous mode. Codex, the Motorola subsidiary and an old-hand modem manufacturer, has released a fully symmetrical 9600-bps modem that incorporates nearly all of the CCITT V.32 recommendations. Finally, a halfway implementation of full-duplex comes from USRobotics—it, too, incorporates most of the V.32 operating principles. For our reviews, we chose to look at the Codex and USR modems, leaving the world of synchronous communications for another time, perhaps.

The operating principles of V.32 call for full-duplex communication. As mentioned previously, this is a difficult task over standard phone lines. Modems operating at 1200 bps were the last to have the luxury of dedicating exactly half of the available phone bandwidth to the originate and answer channels. Modems operating at 2400 bps (V.22bis) have to overlap portions of the originate and answer channels selectively, relying on new modulation techniques to untangle the data. This overlap is nearly total in any complete implementation of full-duplex 9600-bps transmission—there is just no way around it.

Although data is encoded and decoded on either end, through various modulation techniques such as QAM (see Part I), the data path still requires the full, or nearly full, bandwidth. In essence, then, both answering and originating modems are washing data over each other. To further complicate matters, the result codes of EDC are also washed back and forth. This phenomenon is called *far-end echoing* and must be eliminated if the data is going to be passed back and forth with integrity or without endless retransmissions and timeouts. If the echoes were merely a question of timing—that is, if the echo were to occur at a predictable interval—the modem could be trained to look for the echo at a specific time, determined by a series of initial training tones. Unfortunately, the longer the distance, the more unreliable the time intervals—long-distance, dial-up lines frequently switch and reswitch in midconnection, using variable land and satellite links. This requires that the modem retrain dynamically to suit any change in conditions. That's a tough job, adding significantly to the complexity of an already highly intricate design. It's no wonder that full-duplex, 9600-bps modems have been slow coming to market.

Many modem companies, understandably, thought cancellation of this sticky far-end echo was insurmountable or not worth the design effort. Codex, as far as we have been able to determine, is the only modem manufacturer to achieve reliable far-end echo cancellation; both Codex and USRobotics have found some ways around this and other limitations imposed by dial-up phone lines.

The Codex 2260 modem uses true far-end echo cancellation. After initial handshaking, the originating modem sends a series of training tones to the answering modem. The training tone is interpreted for line quality and far-echo timings. Throughout the following transmission the modems continue to monitor and retrain, when necessary, without degradation of the throughput. As mentioned earlier, this dynamic recalibration is a must on long-haul lines that frequently shift paths. Although some released and proposed V.32 full-duplex modems offer some measure of echo cancellation, none offer the dynamic recalibration used in the 2260. Cancellation was a major engineering feat, and a lot of R&D dollars are behind the Codex success. Understandably, Codex has patented its cancellation technology.

USRobotics, on the other hand, has chosen to take a different tack with its Courier HST modem. The HST falls somewhere between V.29s and a full implementation of V.32. The modem carves out the lion's share of the phone data channel for the sending modem and reserves a narrow 300-bps channel for the receiving modem to send ACKs, NAKs, or characters to the sender. This asymmetrical full-duplex approach, as USR calls it, does away with many of the turnaround headaches presented by V.29 modems. Terminal-to-terminal communication is not impaired, and characters echo normally to both originating and answering terminals. Unattended transfers pose no problems here, either, although some of the hypothetical applications we outlined earlier might still cause problems. Because of the low-end 300-bps channel reserved for two-way communication, the USR Courier HST needs no far-end cancellation. Hence, there is simply no significant overlap of channels.

## MODULATION SCHEMES IN V.32 MODEMS

When we first began reviewing high-speed modems (nearly seven months ago), no V.32 implementations existed. V.29 modem manufacturers took V.29 as a guideline, implementing their own EDC protocols on the existing fax data pumps. We hoped V.32 implementations would be more compatible because the recommendations outlined EDC modulation techniques in great detail. Alas, what with USR's half/full-duplex version and Codex's full/full-duplex implementation, we again have compatibility problems. The only hopeful sign is that both modems do adhere to the V.32 modulation schemes for EDC, if not duplex.

Both the modems we reviewed use Trellis Coded Modulation (TCM), and both have fallbacks to QAM modulation, as recommended by the CCITT. In talking about modulation techniques, you get the distinct impression that the normal world has fallen away—no pedestrian terms can quite come close to explaining the feats of voodoo accomplished by these technologies. Analogy is the only reasonable way to view the techniques employed. For this reason, *trellis* is a good term for this modulation scheme.

TCM hangs data on a trellis, or multidimensional signal. This hung code has a real-time shape to it, much as vines spread across a real vine trellis. In addition, redundant code is strung across the trellis, giving it a distinct shape. The decoding modem then checks the shape of the received data. The shapes must match. Remember, in high-speed modulation, the phase shift as well as the amplitude is used to provide more data states and a multidimensional shape to the data. This serves both as EDC and a more efficient use of the bandwidth, offering higher speed. The USR HST, for instance, uses a 2400-baud signaling trellis, each baud or plank in the trellis transmitting four data bits. As per CCITT recommendations, if TCM is not present or is disabled, the modems fall back to QAM modulation techniques; the public-domain protocols from Microcom (known as MNP) are QAM modulation techniques.

TCM shows great promise for further advances in speed as the technique may be able to hang four, five, six, or more data bits on each baud.

## TESTING THE MODEMS

We took care to test the modems over a period of days and at different times of the day. Because of the late arrival of both sets of modems, however, the testing period was somewhat shorter than for those in the V.29 category. Line noise can vary within a given session, not to mention over a period of hours or days. We used the same 196,096-byte file for the Codex 2260 and USR HST transfers as we used in the previous sessions, with bits per second determined by 196,096 × 10 divided by the transfer time in seconds. We measured hard-disk access time and subtracted it from the total transfer time. Retransmission of packets was included in transmission times. The results are shown in Tables 1-3. Note that terminal turnaround times are not included in this evaluation as they were in Part I of our article; because of the duplex schemes of both modems, it was unnecessary to include such figures.

### THE USROBOTICS COURIER HST

As with USRobotics' previous modem designs, the HST does not rely on outside vendors for chip sets. The V.29s, you'll recall, have by and large hung their own buffers, ROMs, and EDC on the Rockwell 96FAX facsimile data pump. Similarly, most commercial 2400- and 1200-bps modems have standard chip sets provided by outside manufacturers. The HST has been designed from the ground up, using off-the-shelf, easily obtained chips—another USR design characteristic.

What appears to be happening with the USR hardware is this: two 8031 processors are used as the primary data pumps and each 8031 is supported by PROMs. A 16-bit signal processor with two support ROMs *may* be sorting the high-speed signals when in 9600-bps mode. As with earlier USR products, it appears that the 8031s are receive and send—one for each—their support ROMs offering the fall-back 2400-, 1200-, and 300-bps emulations (more on the lower speeds later). One ROM, mounted midboard with the USR logo, no doubt holds the proprietary EDC and other registers unique to the high-speed mode. Nonvolatile, battery-backup memory is on-board for register storage of up to four phone numbers.

| Modem | Transmission (bps) |
|---|---|
| USR Courier HST | 8.006K |
| Codex 2260 | 8.136K |

**Table 1. Modem throughput with 196K ASCII file**

| Modem | Turnaround Time |
|---|---|
| USR Courier HST | 385 msec |
| Codex 2260 | N/A (0) |

**Table 2. Turnaround time, transmission mode**

| Modem | Retransmission |
|---|---|
| USR Courier HST | 0 |
| Codex 2260 | 0 |

**Table 3. Retransmissions per 196K transfer**

Of course, the modem handles all standard Hayes AT commands as well as extended S-register and M-registers for special settings and MNP settings. As with its earlier 2400-bps Courier, USR's HST has all the register commands laid out on the underside of the case. Front-panel LEDs indicate high-speed, auto-answer, and ARQ modes—USR's proprietary algorithm EDC added onto the TCM. Carrier Detect, Ready to Send, Terminal Ready, and Off Hook—the standard indicators on most modems—are some of the other LEDs on the panel. One standard female RS-232 interface, two phone jacks (handset and wall), and the DIP switches are on the rear panel. A volume control is under the LED ledge on the front panel.

The HST's emulation modes are impressive—2400-bps employing MNP, 2400-bps (V.22bis) standard (no hardware EDC), 1200-bps V.22, and 300 212A all worked flawlessly. Virtually no special settings of any registers was necessary, the modem dynamically retraining on the answer from the remote modem. We went from 300 bps to 9600 bps, down to 2400 bps, and back to 300 bps with absolutely no problems. Other previously tested modems handled very poorly or not at all in their supposed emulation modes, often dropping carrier or failing to retrain properly for the remote's speed.

The HST seemed effortless to use compared to the register flipping needed with many of the previous modems we tested. The 300-bps asymmetrical channel provided a chance to test an unattended data transfer. The HST logged itself onto the remote HST using Lattice's SideTalk, entered a password, and transferred a text file to the remote (also running unattended) without a hitch.

For the throughput test transfers, we used MEX-PC, SideTalk, and CrossTalk Fast. Any software that allows simple, nonprotocol transfers should work with the HST, including USR's own Telepac. Terminal software should be set to 19.2K bps, 8 bits, no parity, one stop bit. See the tables accompanying this article for transmission throughput.

Other added features of the HST include register settings for speaker on/off, an inactivity timer to disconnect in the event of no data transmission during those unattended operations, call-duration reporting, help screens for the registers, and standard diagnostics.

But the $995 price tag has got to be one of the Courier HST's strongest "features."

### THE CODEX 2260

This is a modem's modem. The features on the 2260 go on for pages. Here are some of the highlights. It is capable of either synchronous or asynchronous communication at either 4800 or 9600 bps. It has leased and normal phone-line jacks, standard EIA RS-232 connection, 20 nonvolatile RAM stored phone numbers, built-in diagnostics, full and extended AT commands, and preset options menus that set a variety of registers and functions at the push of one button.

The front display is rather special. The standard power CTS AA lights are on the front of the modem. Pull back a hinged cover, however, and the modem has an LCD display that allows you to select preset options *packages* or set individual functions with the push of one or two buttons. The LCD walks you through the options and signals the results of your selection—a nice touch. Also under the front panel is a voice/data switch and options for dialing prestored numbers right from the front-panel buttons.

Under the big hood is enough silicon to choke an electronic horse. The main powerhouse is a 68000 CPU with multiple support ROMs and RAM. The CPU motherboard is piggybacked onto the support board. The construction is sturdy. Battery backup for ROMs and what appears to be a second RS-232 port are also inside. Does the 2260 support *two* serial ports? At any rate, the 2260 is a class act inside and out. But for Porsche construction and performance, you pay. The suggested retail is a whopping $3,495.

The modem was easy to set up. Per the user manual's suggestion, we set one modem to Option 2 on the LCD push-button menu and the other to Option 4. We used Crosstalk exclusively for the tests, but again, any nearly hands-off software should do the trick. The modem performed admirably. Speed results are nearly the same as those for the USR HST. See Table 1 for details.

### CONCLUSIONS

Would the V.29 family of modems suit your needs? If you anticipate little terminal-to-terminal communication and no unattended transfers, then they should do fine. The question of compatibility should rear its ugly head into your decision unless you anticipate that you will never have to communicate at high-speeds with any modems other than the ones you've purchased (and may or may not be compatible with the outside standards.) At this writing, the only half-duplex contender for CCITT approval is the Telebit TrailBlazer (reviewed in Part I). The TrailBlazer has a powerful engine under its hood, and the company's claim to turn it into a true front-end data processor for PCs has to be taken seriously.

Meanwhile Codex has the most elegant, fully blown implementation of V.32 available. Its full-duplex and far-end echo cancellation are a marvel.

Also note that we detected no retransmissions of blocks in the V.32 modems. This could be attributable to line conditions, but both modems experienced none in any test—maybe because of the TCM modulation scheme. Both modems classed at the same relative speed as the TrailBlazer.

If pressed to the wall to decide before any true industry standard is reached, however, you'd have to go with USRobotics' HST. This modem is loaded with features for its price. You get the best of the V.32 family (no turnaround headaches of the V.29). Its asymmetrical design has kept the price low. And the HST is the only modem we tested that offers *real* compatibility with other 2400-, 1200-, and 300-bps modems, thereby not sacrificing communications with the large, installed user base operating at those speeds. True, the UPTA 96 has other speeds as snap-in options, but these add-ons also add on to the price tag of the base modem, and the frustrating turnarounds at high-speed are still there in the V.29 UPTA. The HST offers the most bits for the buck, by far.

Most users should take a wait-and-see attitude. What *will* the standard be? For the time being, only the CCITT and the pressures of the marketplace can determine which of these speedy modems will emerge as a clear leader in the high-speed race, thereby setting an industry standard. §

## PRODUCT INFORMATION

**USR Courier HST**
V.32 9600, 2400, 1200, 300 bps
**Price:** $995
**Recommended software:** CrossTalk Fast, SideTalk. Call manufacturer for other PC-to-PC and mini and mainframe software packages.
**USRobotics**
8100 N. McCormick Blvd.
Skokie, IL 60076
(312) 982-5010

**Codex 2260**
V.32 9600 and 4800 bps
**Price:** $3,495
**Recommended software:** CrossTalk Fast. Call manufacturer for other PC-to-PC and mini and mainframe software packages.
**Codex Corp.**
20 Cabot Blvd.
Mansfield, MS 02048
(617) 364-2000

# Program Interfacing to Microsoft Windows

by William Wong

## Part—II How A Windows Application is Configured, Constructed, and Created

Programming for Microsoft Windows is different from programming for DOS. This part of the series is an introduction to how a Windows application is configured, constructed, and created. Unfortunately, Windows requires that you have knowledge of a large part of the system to build a good application. The application presented here is a skeleton on which subsequent articles will be built.

DOS programs follow a single, logical execution thread that calls DOS as necessary. DOS returns when the requested operation is finished. In addition, there is only a single active program. A program can invoke another program, but the invoking program will not continue until the invoked program terminates.

Windows wants its programs to run as cooperative coroutines. Information is exchanged through use of a set of standard and user-defined messages. Message passing is based on the use of windows within Windows. Each application can have one or more windows associated with it. Each window has a message-handling function that is called when a message is passed to the window. The main difference between this message-passing approach and that of the conventional DOS program is that Windows tells an application when things happen, whereas DOS programs poll DOS.

### HOW A WINDOWS APPLICATION WORKS

A Windows application is built using a number of different tools, which will be discussed in detail later in this article. These are included as part of the Windows Development Toolkit, available from Microsoft. Anyone interested in developing Windows applications should obtain a copy of the toolkit from Microsoft.

A Windows application consists of program code, data, and resources. Resources include items such as strings, icons, and bit maps. These items are put together using the program linker and the resource compiler; this adds complexity, but the results are well worth it.

An application uses a single program stack. It starts a little differently from a DOS program. The program is also loaded differently from a DOS program. In particular, the interface to the system is at the function level (not true in DOS), and Windows does not use interrupts. A Windows application is started by calling the WinMain function, which must be defined within the application.

The WinMain function initializes the application and allocates the initial windows used by the application. It then enters a loop that waits for messages. The messages are processed or passed on as necessary. The program should exit from the function when it receives a terminate message.

Note that the WinMain function does not handle the messages for a specific window. Instead, a window has a function that is called when a message is sent to the window. This function can send additional messages, but control will return to the function when the messages have been processed. The stack-like operation of message handling allows an application to use a single stack. Eventually, control returns to WinMain, which gives control back to Windows.

The coroutines are Windows and the applications it runs. Switching between the coroutines is done through the Windows functions. The coroutines must cooperate, because the switch occurs only at the specified interface and not in a time slice or priority fashion as is normally found in conventional multitasking systems. The advantages of the coroutine approach are simplicity, speed, and minimized stack space. The disadvantages are some added complexity and the ability of an application to dominate the use of the system. The latter is often seen while running the clock in Window. The clock will appear to stop and then jump ahead to catch up because some other application did not give up control.

For this reason, all applications must be written to run in a cooperative fashion. Time-consuming operations must relinquish control periodically to allow other applications to run.

Before going into the structure of an application, this article will take a look at some of the utilities used to create the various parts.

### WINDOWS' UTILITIES

The following utilities are included as part of the Windows Development Toolkit. In addition, the toolkit includes support for C, Pascal, and assembler. These compilers and assembler may be obtained as separate items. Version 4 of the C compiler is used for subsequent examples in this series.

| Program | Description |
|---|---|
| MAKE | File-creation utility |
| LINK4 | Linker |
| RC | Resource compiler |
| MAPSYM | Symbol-table generator |

| | |
|---|---|
| SYMDEB | Symbolic debugger |
| | |
| ICONEDIT | Icon editor |
| FONTEDIT | Font editor |
| DLGEDIT | Dialog-box editor |

The MAKE utility is a common program in a programmer's toolkit. It uses a script file to generate programs based upon the existence of files and their associated time stamps. For example, an .EXE file must be linked from an .OBJ file and possibly a library file. These, in turn, were compiled from a source file, which may have used one or more include files. A change to a source file, requires the .OBJ and .EXE files to be regenerated. MAKE is important because Windows applications tend to be modular with many resources in different files. Changing a single icon or header file may invalidate a number of other files. The bottom line, MAKE Window's applications.

LINK4 is a new version of the Microsoft linker that keeps much of the information necessary for loading a Windows application. The output of the linker is processed by RC, the resource compiler. RC actually performs two functions. The other function is to take text-file descriptions of resources such as strings and icon-file references and combine the various files into a binary resource file, which is combined later, by RC, with the output from LINK4.

MAPSYM is used to generate a special symbol file containing the program symbols. The symbol file is used by SYMDEB when debugging the application.

The three editors—ICONEDIT, FONTEDIT, and DLGEDIT—are used to create icon, font, and dialog-box resources respectively. The files generated by these programs are included in the final .EXE file, through the use of RC. The editors run under Windows, so you will need at least one machine already running Windows. The programs have a minimal user interface but are adequate for development purposes. DLGEDIT is the most limited of the three, because it can only generate new files, not edit old ones.

The syntax of the text file for the resource compiler, RC, will be covered later in this article. None of the other programs will be covered, since their use does not impact the construction of a Windows application program.

## SAMPLE PROGRAM

The basis of a Windows application is presented in the TEST program listings at the end of the article. The TEST program does



**Table 1. .RC Command Syntax**

```
id BITMAP [load-option] [memory-option] filename
id CURSOR [load-option] [memory-option] filename
id FONT   [load-option] [memory-option] filename
id ICON   [load-option] [memory-option] filename

id type   [load-option] [memory-option] filename


STRINGTABLE [load-option] [memory-option]
BEGIN
  string_id, "string"
END


id ACCELERATORS
BEGIN
  event, id [, ASCII] [, VIRTKEY] [, NOINVERT] [, SHIFT] [, CONTROL]
END


id MENU [load-option] [memory-option]
BEGIN
  menu-statements
END

id DIALOG [load-option] [memory-option] x, y, width, height
dialog-options
BEGIN
 dialog-statements
END


Options include:
----------------
  id              : integer or symbol
  load-option     : PRELOAD or LOADONCALL
  memory-option   : FIXED or MOVABLE
                    DISCARDABLE
```

nothing more than put up a window. The window can be moved and converted into an icon, but little else can be done with it. Even so, the size of the program indicates the relative complexity of starting a Windows application. This complexity may be deceptive, however, with respect to the overall size of an application. Larger applications tend to benifit more by the facilities provided by Windows. (This will become more apparent as subsequent articles present more of Windows.)

The files involved in building the TEST program are:

| **Program** | **Description** |
|---|---|
| TEST. | script for MAKE |
| TEST.H | header file with common information |
| TEST.RC | script for RC |
| TEST.DEF | script for LINK4 |
| TEST.C | C source file |
| TEST.ICO | icon file from ICONEDIT |
| TEST.RES | resource file from RC |
| TEST.OBJ | object file from C compiler |
| TEST.EXE | executable program from LINK4 |

The last four files have a binary format that is not usually examined and cannot be listed here. The .ICO file was created by the icon editor, ICONEDIT. This is the icon that is displayed on the icon bar when the application is moved there. The form of the icon does not affect the operation of the program. Users can change the icon and regenerate the application to use the new icon.

The first five files will be described and discussed in the order of their preceding presentation and at the end of the article. Some of the additional options for particular functions will be addressed as they are used.

The TEST file is the script used by MAKE to create TEST.EXE. Each section begins with a file name followed by a colon. This indicates the file to be created. The files to the right of the colon are the ones to be examined. If the file to be created does not exist or if its time stamp is earlier than any of the files listed to the right, then the subsequent commands, which are indented, are performed.

The first section is used to create the resource file, TEST.RES. The second section creates the object file, TEST.OBJ, from the C program source, and the final section builds the executable program, TEST.EXE. The TEST.SYM file is also created by the last section for debugging purposes and is optional.

TEST.H is a common header file that uses the C precompiler syntax. This file defines only two constants, which are used to identify two of the strings that are used in the program TEST.C and included in the resource file TEST.RC. The program source can refer to any number of header files, which need not be common to the resource file.

The TEST.RC file is the script used to drive the resource compiler, RC, which creates TEST.RES. The script uses C preprocessor commands, like #include, in addition to special commands that will be addressed here. The script describes icons, menus, and dialog boxes. Common text strings are also defined here. In fact, it is preferable to put all text in the resource file, for two reasons. First, this keeps the application modular, making changes to the program and text more independent than they would be if the application were not modular. Second, it allows applications to be customized in different languages without requiring access to the program source.

The commands for the resource script may be used as many times as necessary and can be placed at any point in the file. This allows menus to be grouped with dialog boxes with which they may be used.

There are four single-line commands that can be used in the script file. The command is the second word on the line and is followed by a file name. The commands are ICON, CURSOR, BITMAP, and FONT, which refer to files of type .ICO, .CUR, .BMP, and .FON. The first three are created using ICONEDIT, while .FON is created using the FONTEDIT program. Objects loaded into the resource file in this fashion can be referenced using the first word on the command line. The word for a font object must be a positive integer. A single icon is used in the sample program.

Two additional options can be placed between the command name and the file name. The first option specifies how the indicated file is to be loaded. If there is no reference, the file is to be made part of the resource file. The other two options are PRELOAD and LOADONCALL. PRELOAD indicates the file is to be loaded when the resource file is created. LOADONCALL indicates the file is included when the resource compiler combines the resource file with the .EXE file to make a complete Windows application. PRELOAD allows a resource to be fixed when the resource file is created. The default is LOADONCALL.

The other option controls what kind of memory allocation procedure is to be used with respect to the object when the application is running. The options are FIXED and MOVABLE plus DISCARDABLE. The defaults for ICON, CURSOR, and FONT are MOVABLE and DISCARDABLE, while bit map is MOVABLE and not discardable (which is not directly specified). The memory used by a discardable object can be reused by Windows. It will be reloaded from the program file if it is required again. An object can be FIXED or MOVABLE. FIXED keeps the object in the same place in memory, while MOVABLE allows Windows to put the object anywhere it likes. An application tells Windows when it needs an object for a MOVABLE and DISCARDABLE object.

User-defined resources can also be included and defined using the same syntax as the predefined Windows objects. The default option is MOVABLE. The object type, which is the second item on the command-line, can be a nonreserved word or a positive integer greater than 255. What is contained in these resources is determined by the application. It could be a database, constants used in computations, or setup information. This is an extremely powerful and easy-to-use feature.

The other predefined objects are four multiple-line commands: STRINGTABLE, ACCELERATORS, MENU, and DIALOG. The last section in the sample file contains a single STRINGTABLE. These objects and their formats are covered next.

A STRINGTABLE is a list of strings, with an identifier associated with each string. This allows an application to access the corresponding string. The strings can be used for any purpose. The TEST program uses two strings. Multiple STRINGTABLE entries can be included. Multiple entries allow different attributes to be associated with an object. For example, some strings may reside in a MOVABLE object, while another must remain FIXED.

An ACCELERATOR is a list of keys that are translated in a special fashion. Normally, menu selections return a value indicating which item was selected, if any. An accelerator entry consists of a key specifier, such as Ctrl-C, and the corresponding value to be sent to a window.

A MENU entry describes a menu that can be associated with a window. The top-level menu items describe the titles found on the menu bar across the top of the window where the menu is used. The pull-down menu selections are specified using the POPUP entry, where the *text* associ-

```
event               : "c"    character
                      "^c"   control character
                       c     character with ASCII option
                       v     virtual key defined in WINDOWS.H
                             with VIRTKEY option

menu-statements     : MENUITEM SEPARATOR
                      MENUITEM "text", integer [, MENUBREAK]
                        [, CHECKED] [, INACTIVE] [, GRAYED]
                      POPUP "text" [, MENUBREAK] [, MENUBARBREAK]
                        [, CHECKED] [, INACTIVE] [, GRAYED]

dialog-options      : STYLE window-style
                      CAPTION "text"
                      MENU menuid
                      CLASS "text"
                      CLASS integer
dialog-statements : dlg-type "text", id, x, y, width, height [, dlg-style]
                      CONTROL  "text", id, dlg-class, dlg-style, x, y, width,
                               height
                      EDITTEXT id, x, y, width, height [, dlg-style]
                      LISTBOX  id, x, y, width, height [, dlg-style]
dlg-type            : LTEXT
                      RTEXT
                      CTEXT
                      CHECKBOX
                      PUSHBUTTON
                      GROUPBOX
                      DEFPUSHBUTTON
                      RADIOBUTTON
                      ICON
dlg-style           : attributes for entry
                      (to be described in a later article)
dlg-class           : BUTTON
                      STATIC
                      EDIT
                      integer
                      "name"

x      is in 1/4th character widths units.
y      is in 1/8th character height units.
width  is in 1/4th character widths units.
height is in 1/8th character height units.

==========================================


   The following files are used to create the TEST.EXE file.

TEST: file used by MAKE to create MAKE.EXE
------------------------------------------

test.res: test.rc test.ico test.h
    rc -r test.rc

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

test.exe: test.obj test.res test.def
    link4 test, test/align:16, test/map, slibw swlibc/NOD, test.def
    mapsym test
    rc test.res

------------------------------------------


TEST.H: header file which defines common items
----------------------------------------------

/* ====  Windows Test Program Header File  12-16-86 WGW  ====    */
/*                                                               */
/* ----  String table constants  ----                            */

#define IDSNAME     100        /* ID for program name string    */
#define IDSTITLE    200        /* ID for window title string    */


/* ====  End of Windows Test Header File  ====                   */

---------------------------------------------


TEST.RC: File defining resources for TEST.EXE
---------------------------------------------
```

ated with the POPUP entry is that which will be shown on the menu bar. The Pop-up menu is displayed when the *text* on the menu bar is selected. Actually, the POPUP menu pulls down under the *text* on the menu bar.

The MENU entry allows you to create and modify a menu easily. The ordering of the items in the MENU entry is critical. Items that come first are positioned to the left of the menu bar. Items in a POPUP entry that come first are located at the top of the pull-down menu. The number associated with an item, which can be selected, is the value that is sent as a message to a window.

A MENU entry can have a number of attributes that normally are initialized within the MENU entry. The attributes can be altered by the application. For example, the CHECKED attribute indicates that a check mark appears before the text. Selecting such an entry normally toggles the attribute on or off. An application may change the state of other entries if they are mutually exclusive.

Finally, there is the DIALOG entry. It defines a single dialog box. The configuration and usage of a dialog box is complicated due to the large number of options it offers. This entry will be discussed in a later article, along with the supporting Windows functions.

A dialog box has a position and size associated with it. You can change this with an application. Each entry within a dialog box also has a position that is relative to the dialog box. The following statements can be included within a dialog box:

CONTROL
EDITTEXT
LISTBOX
GROUPBOX
LTEXT
RTEXT
CTEXT
CHECKBOX
PUSHBUTTON
DEFPUSHBUTTON
RADIOBUTTON
ICON

A CONTROL entry is a user-defined region that can be used for specialized controls. The EDITTEXT entry is a rectangular region in which text can be entered or modified. A LISTBOX can have an optional vertical scroll bar. A LISTBOX can have any number of items, not listed in this part of the definition, which can be selected. The items displayed in a LISTBOX are added using Window functions. GROUPBOX is a rectangular box that can have a title and can contain other items such as RADIOBUTTONS and PUSHBUTTONS.

LTEXT, RTEXT, and CTEXT entries display text that is aligned to the left, aligned right, or centered, respectively. Text is wrapped automatically to the beginning of the next line if the text is too long.

A CHECKBOX is a square box that can be selected. An *X* is placed within the box when the option is enabled. The *text* is displayed to the right of the box. PUSHBUTTON and DEFPUSHBUTTON are rectangles that contain the *text* specified in the entry. A button is set to reverse-video mode when selected. The DEFPUSHBUTTON has a bolder outline around the button than the PUSHBUTTON. A RADIOBUTTON is similar in operation to a CHECKBOX, except that an application typically groups RADIOBUTTONs together and allows only one button to be selected at a time.

The ICON entry specifies that an icon is to be displayed at the indicated position. The *text* is the name of an icon that is defined elsewhere in the resource script using the ICON command. This allows an icon to be used within a number of dialog boxes.

Dialog-box definitions can be created using the DLGEDIT program. Unfortunately, the program cannot modify an existing definition. Also, the program creates one file per dialog box. For this reason, the #include directive often is used to reference these files.

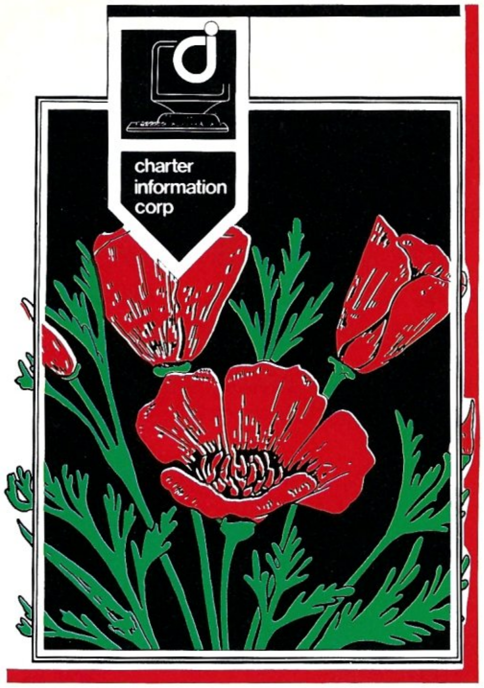
Subsequent articles will address the ACCELERATORs, MENU, and DIALOG definitions in detail. The previous description gives an overview of how a Windows application is built, however, as well as listing some of the major building blocks that can be used in the process.

The TEST.DEF file is the LINK4 script. There are a number of options that can be used that will not be covered here. The NAME and DESCRIPTION essentially describe what is going on. The STUB command indicates which loader file is to be included. It provides a check and an error message if the program is not used within Windows. The CODE and DATA commands are used to indicate what type of segment attributes are to be used when creating the program. MOVABLE allows Windows to move a segment. A small or medium model program can be MOVABLE without any other special programming considerations because all pointers

```
#include "windows.h"
#include "test.h"

test  ICON    test.ico

STRINGTABLE
BEGIN
    IDSNAME,     "Test"
    IDSTITLE,    "Sample Windows Program"
END

----------------------------------------------
```

```
TEST.DEF: Link pecification for TEST.EXE
-----------------------------------------

NAME     Test

DESCRIPTION 'Logic Fusion Inc. Basic Windows Program'

STUB     'WINSTUB.EXE'

CODE     MOVEABLE
DATA     MOVEABLE MULTIPLE

HEAPSIZE    4096
STACKSIZE   4096

EXPORTS
    MyWndProc     @1
```

```
TEST.C: Main C program for TEST.EXE
-----------------------------------


/*  ====  Basic Windows Test Program    12-05-86 WGW  ====                  */

#include "windows.h"            /* common Windows  definitions              */
#include "test.h"               /* common TEST.EXE definitions              */


/* ====  Forward Function Definitions  ====                                 */

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


/* ====  Internal Global Variable Definitions  ====                         */

char    szAppName     [ 10 ] ;
char    szWindowTitle [ 30 ] ;


/* ====  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 )
```




# POPPY

## brings compatibility to full flower.

**Multi-User • PC-DOS • Turbo-DOS • S-100**

**Yes, they can co-exist!**

**And you'll love the way they look!**

This remarkable workstation from Charter Information Corp gives you monochrome, color, and EGA capabilities all on one blooming board which fits into your multi-user Turbo-DOS system. In fact, you can fit up to sixteen POPPY boards inside a box for true multi-user capabilities. POPPY uses PC compatible monitors and keyboards which can be up to 200 feet from the central system.

Run all your favorite PC-style programs, including Lotus 1-2-3, Wordstar, Sidekick, Flight Simulator, Jet, and more under PC-DOS or Turbo-DOS on your S-100 system. And then share your peripheral resources among various users.

And if you're a reseller who hasn't made multi-user machines before, give us a call. We'll make the whole system for you in 3 to 10 days. If you missed seeing us at COMDEX, be sure to write or call for our free brochure. We think you'll agree that POPPY will really add to your computing landscape.

You've planted a seed in my mind.

☐ Please send me more information on the systems that you will integrate for me.

☐ Send information on being a dealer for your product.

Name ____________________

Title ____________________

Company ____________________

Address ____________________

City ____________________

State __________ Zip ______

Telephone (____) ______________

**Charter Information Corp**

*Where good ideas improve.*

2421 Rutland
Austin, Texas 78758
(512) 835-1111

# PC/VI™

UNIX's VI Editor Now Available For Your PC!

Are you being as productive as you can be with your computer? An editor should be a tool, not an obstacle to getting the job done. Increase your productivity today by choosing **PC/VI** — a COMPLETE implementation of UNIX* VI version 3.9 (as provided with System V Release 2).

**PC/VI** is an implementation of the most powerful and most widely used full-screen editor available under the UNIX operating system. The following is only a hint of the power behind **PC/VI:**

- Global search or search and replace using regular expressions
- Full undo capability
- Deletions, changes and cursor positioning on character, word, line, sentence, paragraph, section or global basis
- Editing of files larger than available memory
- Shell escapes to DOS
- Copying and moving text
- Macros and Word abbreviations
- Auto-indent and Showmatch
- MUCH, MUCH MORE!

Don't take it from us. Here's what some of our customers say: "Just what I was looking for!", "It's great!", "Just like the real VI!". "The documentation is so good I have already learned things about VI that I never knew before." — *IEEE Software*, September 1986.

**PC/VI** is available for IBM-PC's and generic MS-DOS† systems for only $149. Included are CTAGS and SPLIT utilities, TERMCAP function library, and an IBM-PC specific version which enhances performance by as much as TEN FOLD!

# PC/TOOLS™

What makes UNIX so powerful? Sleek, Fast, and **POWERFUL** utilities! UNIX gives the user not dozens, but hundreds of tools. These tools were designed and have been continually enhanced over the last fifteen years! Now the most powerful and popular of these are available for your PC! Each is a complete implementation of the UNIX program. Open up our toolbox and find:

| | | | |
|---|---|---|---|
| • BFS | • DIFFH | • OD | • STRINGS |
| • CAL | • DIFF3 | • PR | • TAIL |
| • CUT | • GREP | • SED | • WC |
| • DIFF | • HEAD | • SEE | |

All of these for only $49.00; naturally, extensive documentation is included!

# PC/SPELL™

Why settle for a spelling checker which can only compare words against its limited dictionary database when **PC/SPELL** is now available? **PC/SPELL** is a complete implementation of the UNIX spelling checker, renowned for its understanding of the rules of English! **PC/SPELL** determines if a word is correctly spelled by not only checking its database, but also by testing such transformations as pluralization and the addition and deletion of prefixes and suffixes. For only $49.00, **PC/SPELL** is the first and last spelling checker you will ever need!

Buy **PC/VI** and **PC/TOOLS** *now* and get **PC/SPELL** for only $1.00! Site licenses are available. Dealer inquiries invited. MA residents add 5% sales tax. AMEX, MC and Visa accepted without surcharge. Thirty day money back guarantee if not satisfied! Available in 8", 5¼" and 3½" disk formats. For more information call today!

*UNIX is a trademark of AT&T. †MS DOS is a trademark of Microsoft.

**CUSTOM SOFTWARE SYSTEMS**
*P.O. BOX 678 • NATICK, MA 01760*
*617 • 653 • 2555*

UNIX TOOLS FOR YOUR PC

UNIX TOOLS FOR YOUR PC

*Does this look familiar? What if each change you made to your program was ready to test in seconds instead of minutes?*

***"The SLR tools will change the way you write code. I don't use anything else.", Joe Wright***

RELOCATING MACRO ASSEMBLERS • Z80 • 8085 • HD64180

- Generates COM, Intel HEX, Microsoft REL, or SLR REL
- Intel macro facility
- All M80 pseudo ops
- Multiple assemblies via command line or indirect command file
- Alternate user number search
- ZCPR3 and CP/M Plus error flag support, CP/M 2.2 submit abort
- Over 30 user configurable options
- Descriptive error messages
- XREF and Symbol tables
- 16 significant characters on labels (even externals)
- Time and Date in listing
- Nested conditionals and INCLUDE files
- Supports math on externals

requires Z80 CP/M compatible systems with at least 32K TPA

$49.95

**SLR Systems**
1622 N. Main St., Butler, PA 16001
(412) 282-0864 (800) 833-3061

THE MISSING LINK

PC-925 TURNS YOUR PC INTO A FULL FUNCTION TELEVIDEO-925

• FEATURING •

- 4 VIDEO PAGES
- SUPPORTS 25th LINE
- 5 VIDEO ATTRIBUTES
- PROTECTED FIELDS
- 40 PROGRAMMABLE FUNCTION KEYS
- FILE TRANSFER
- AUTO DIAL/LOG ON
- KEYBOARD MACROS
- "MOVE" FILE EXCH
- TVI/WS SWITCHABLE CURSOR PAD CODES

$79.95 Postage Paid

**MS-DOS & PC-DOS COMPATIBLE**
**PUT YOUR PC'S TO WORK ON YOUR MULTIUSER...**

TELEVIDEO® TELEVIDEO SYSTEMS WORDSTAR® MICROPRO INC.

**DIGITAL DECISIONS SYSTEMS**

CALL OUR BBS FOR CURRENT PRICING (201) 772-4209

229 Lakeview Ave. • Clifton, NJ 07011
**1-800-862-6233** Wait for tone, dial Ext. 0014762 **(201) 772-6749**
**Call/Write for our *FREE* 140 page Color Catalog**

```
  HANDLE hInstance ;
{
  BOOL      result ;
  PWNDCLASS pTypeClass ;

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

  LoadString ( hInstance, IDSNAME,    (LPSTR)szAppName,      10 ) ;
  LoadString ( hInstance, IDSTITLE,   (LPSTR)szWindowTitle, 30 ) ;


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

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

  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      */
}


/* ----  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) */
{
  MSG   msg ;
  HWND  hWnd ;
  HMENU hMenu ;


  /* ----  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) szWindoTitle, 30 ) ;
  }
  else
  if ( MainInit ( hInstance ))                  /* try to initialize    */
    return FALSE ;                              /* exit if error occurs */


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

  hWnd = CreateWindow ( (LPSTR) szAppName       /* application name     */
                      , (LPSTR) szWindowTitle   /* window title         */
                      , WS_TILEDWINDOW          /* window type          */
                      , 0                       /*  x - ignored (tiled) */
                      , 0                       /*  y - ignored (tiled) */
                      , 0                       /* cx - ignored (tiled) */
                      , 0                       /* cy - ignored (tiled) */
                      , (HWND)    NULL          /* no parent window     */
                      , (HMENU)   NULL          /* use class menu       */
                      , (HANDLE)  hInstance     /* handle to window     */
                      , (LPSTR)   NULL          /* no parameters        */
                      ) ;


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

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


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

are with respect to the code and data segment registers that are handled by Windows. Large model programs require special handling if a segment is MOVABLE, because its position may be changed by Windows. The considerations for this configuration will be covered in a later article.

HEAPSIZE and STACKSIZE specify the number of bytes for the local heap and stack used by the program. The EXPORTS command provides a list of functions that will be referenced by Windows. In this case, MyWndProc is the function that handles the messages for the single window class used by TEST. Windows calls this function after the window class has been registered and a message is available for the window.

Finally, there is the TEST.C file, which is the actual program. The general structure of the program will be discussed here; details of the various functions will be reserved for later articles.

The program starts by including common Windows definitions and constants used within this program. In the latter case, the file contains the string ID number definitions. The forward reference is for the C compiler. The global variables are used to keep copies of the strings. This allows the strings to remain resident and also allows the application to modify them.

The MainPaint function is used to paint a region in the system background color. This is generally a window update function that is necessary because Windows does not keep a copy of a Windows contents if it is overwritten by another window. Instead, an application is required to redraw the specified area when an overlapping window is removed. In this case, the only information provided is the color of the system background. The function must erase the area because Windows does not perform an erase operation when the overlapping window is removed. The PAINTSTRUCT is passed by Windows to the application and specifies the area to redraw.

MainInit is the function used to initialize the application. It makes a local copy of the resource strings and then registers the window class. A window class describes how a window works. An application can use more than one window of the same class. Multiple instances of an application can use the same class while sharing the same program code. (Windows will be discussed in more detail in the next article.) An object is allocated from the local heap, because Windows must use FAR pointers, and this application uses a small memory model that uses NEAR pointers.

The WinMain function is the one that Windows calls when starting the application. The Pascal keyword is used to indicate that the Pascal calling sequence is used instead of the C convention. The Pascal version pushes the parameters on the stack in a left-to-right order, and the called function removes the parameters from the stack. C normally pushes the parameters in the opposite order, and the parameters are removed from the stack after the function returns.

The parameters to WinMain indicate command-line options and whether the program is being started for the first time. The last parameter is the suggested method of display—icon, window, or hidden.

The WinMain function tries to initialize the default window class used by the application if necessary. Otherwise, it uses the existing version and copies the resource strings from the previous invocation.

A new tiled window is then created using the window class already defined. The application then goes into a tight loop, which waits for a message, translates it, and then dispatches it to its windows. The application terminates when there are no more messages that occur when a terminate message is sent to the application. The application actually can perform more computation and translation within this loop, but these tasks normally are performed by the function associated with a window.

Finally, there is MYWNDPROC. This is the function that handles messages to the window class defined by the application. The window handle and message are passed as parameters. The optional message parameters are a single 16-bit value and 32-bit value. The function dispatches to the respective message, which is an integer, which processes the message and then returns to Windows. Not all of the messages are listed in this example. The messages normally sent to Windows will be covered in a later article.

The WM_SYSCOMMAND is the message received when an item is selected from the default menu associated with a window. It is the one presented when the square in the upper-left-hand part of the screen is selected with the cursor. WM_DESTROY is received when an application should be terminated. The WM_PAINT message is sent when Windows removes an overlapping window. The WM_ERASEBKGND message is similar to WM_PAINT. Note that all messages must be processed in some fashion. The DEFWINDOWPROC function is Windows default message handler.

That is about it for the program and the files used to create a basic Windows application. The basic structure will grow as more complex applications are developed.

```
    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 ;
  {
    PAINTSTRUCT ps ;
    HBRUSH       hbr, hbrOld ;
    RECT         rect ;

    switch  (message)
    {

     case WM_SYSCOMMAND :                   /* process system command       */
       return DefWindowProc ( hWnd, message, wParam, lParam ) ;

     case WM_DESTROY :                      /* time t exit program         */
       PostQuitMessage ( 0 ) ;
       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) ;
 }

 /* ====  End of Basic Windows Test Program  ====                          */
  , wParam, lParam ) ;
   }

   return (0L) ;
 }

 /* ====  End of Basic Windows Test
```

## SUMMARY

The approach to building a Windows application is very different from the one to building a conventional DOS program. As in most programming projects, things move along more quickly after the basic skeleton is completed.

Subsequent articles will address some of the additional functions that can be built using Windows, as well as the other resources available under Windows. These resources make the application possibilities under Windows much greater than under DOS alone. §

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

*All the source code for articles in this issue is available on a single, MS-DOS disk. To order send $14.95 to Micro/Systems Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600 ext. 216. Please specify the issue number.*

# Classic Technology's 286 Speed Pak

by Charles H. Strom

## Upgrading the PC/XT to an 80286, 1-Megabyte Machine

As the marketplace for IBM PCs and compatibles matures, we see an increasing presence of the PC/AT and its clones. These machines promise high performance as well as compatibility with future operating-system extensions. Microsoft has made it clear that it does not see a big future for 8088-based systems, but there remains a large installed user base of PCs, XTs, and so on. We are therefore likely to see the continued development of software for this vintage of computer for many years. For quite a while, we have seen the proliferation of various accelerator boards for the IBM PC/XT and its clones, promising everything from increased speed to miraculous transformation to an AT work-alike at a fraction of the cost. Many of these promises are exaggerated, but the basic performance improvement is undeniable.

I wish to focus on a particular accelerator product that has recently received quite a bit of attention—the 286 Speed Pak from Classic Technology Corp. The Speed Pak is actually the nucleus of a family of products from Classic. Although I received literature on supplemental hardware and software, I have not had the opportunity to try it. After summarizing features of the Speed Pak itself, I'll discuss the advertised expansion capabilities.

### BRINGING UP THE BOARD

In simple terms, the Speed Pak is an IBM-compatible expansion board designed for installation in a PC or XT machine. Within limits, it operates well on Far East XT clones (see later). The board has an 80286 CPU chip operating at a clock speed of 8 MHz as well as 1 megabyte of RAM (256K chips), an XT-compatible 16-bit BIOS (on two PROMS, as the data bus is 16 bits wide), a socket for an optional 80287 math coprocessor, and all necessary support logic.

Installation on an IBM machine is trivial and is covered in layman's terms in the thin manual. You remove your 8088 chip from its socket and install it in the Speed Pak and then run a ribbon cable from the board to the original 8088 socket. This design allows you to flip a switch on the rear of the board (accessible without opening the computer) to invoke a 100 percent 8088 mode. Unlike other accelerators, this actually switches CPU chips and thus requires a hardware reset. As a plus, it guarantees 100 percent compatibility when in 8088 mode. Finally, the board contains connectors designed to accommodate expansion memory used in multiuser setups (discussed later).

As previously mentioned, installation on IBM machines is straightforward. Installing the Speed Pak on clones is slightly more difficult, and the manual does not address them. You have to limit memory to 256K on the system board to prevent memory conflicts, which may mean juggling chips on some of the 640K system boards that have become the de facto standard among clones.

Other than that, I noted one odd problem on my Beltron clone system board—all seemed perfect when doing floppy-disk operations—COPY with VERIFY from hard disk to floppy proceeded with no reported errors. When I looked on the floppy disk, though, I found it contained no data! The fix was simple; the Classic board has a jumper that is set for XT or PC with no explanation of its purpose. Once I set it for PC (even though it was installed in an XT clone), floppy-disk operations were perfect at both standard and accelerated speeds.

Parenthetically, Classic reports compatibility with IBM PC Model 2, IBM XT, Compaq Portable, Ericsson EPC, Columbia MPC, and the JDR Microdevices XT clone. The company specifically states that the board cannot operate with the IBM PC-1, the Leading Edge Models M or D, or the Zenith 150. In order to ensure compatibility, the host machine must use an 8088 running at 4.77 MHz—ruling out turbo or 8086-based units—and the BIOS must be 100 percent IBM-compatible. Naturally, the CPU chip must be removable.

### BENCHMARKING THE SPEED PAK

I used the Classic board on an everyday basis for several weeks on both the Beltron XT clone and on an IBM PC. I used several benchmarks, including several of the the *PC Magazine* benchmarks (released into the public domain and available on PC-Blue's Volume 135). Also refer to Sol and Don Libes' benchmark table in the May/June 1986 *Micro/Systems Journal* for an explanation of the programs as well as results for several other systems. For completeness I have also included Paul

| Benchmark | Stock 8088 4.77 MHz | Speed Pak 80286, 8 MHz |
|---|---|---|
| 1 | :29 | :09 |
| 5 | :48 | :15 |
| 6 | :38 | :11 |
| 7 | 1:45 | :32 |
| 8 | 1:31 | :28 |
| 9 | 2:07 | :40 |
| 10 | :09 | :02 |
| 11 | 1:04 | :35 |
| SIEVTIME | 2:02 | :31 |
| NORTON SI | 1.0 | 7.1 |

**Table 1. Benchmark results**

Homchick's SIEVTIME program, which calculates the time required to find 1,899 primes using the Sieve of Eratosthenes (available on GEnie's IBM RoundTable), and the Norton SYSINFO (SI). Results are shown in Table 1.

To confirm that the Speed Pak's 8088 mode is truly equivalent to that of a vanilla XT, I repeated all measurements with the board in standard slow mode. All results are identical to those of the stock XT. It is clear from inspection that operations are substantially increased, even more than those of either an IBM AT or an XT running the Microway 8086 Number Smasher board at 9.5 MHz.

### COMPATIBILITY

I used the Speed Pak with a variety of add-on boards, including the AST Rampage EEMS memory board, several varieties of hard- and floppy-disk controllers, video boards, I/O boards, and so on and experienced no incompatibilities whatsoever. The system behaves just like an XT does, only faster. Software was similarly well behaved. The only slight kink I encountered was that a utility called SPEEDUP.COM, supplied by Classic, did not operate on non-IBM systems. The manufacturer supplies this program effectively to disable the lower RAM that is present on the system board. As this memory uses an 8-bit data path, it is, of course, slower than the Speed Pak's on-board 16-bit-wide memory. Disabling the slower memory results in faster operation of all memory-dependent functions. Classic advises that SPEEDUP can operate only with a true IBM ROM.

One last note about the issue of compatibility. Several Speed Pak dealers tout the board as miraculously transforming your PC or XT into an AT. Such is not the case. In order for this to be possible, it would be necessary to rework totally the interrupt structure of the basic machine, add another 8259A interrupt controller (the PC/XT has one, whereas the AT has two), and so on. Classic is straightforward when queried on the subject and in no way misrepresents the board's capabilities.

### ENHANCEMENTS

The manufacturer offers a multiuser system based on an IBM PC or XT, a Speed Pak board, 2.5-megabyte memory-expansion modules that piggyback on the accelerator, and workstation adapter boards. One workstation board can support up to four stand-alone terminals by the addition of piggyback modules, each of which includes a dedicated 6845 CRT controller. Data I/O is over shielded 25 conductor cable through a modified RS-422 link. Each user has his or her own dedicated DOS 3.x in a separate 640K memory partition. I did not have the opportunity to test this configuration.

Another feature touted by Classic is future compatibility with DOS-286. Because the board can directly address up to 6 megabytes of RAM in 80286 protected-mode operation, the manufacturer assures me that it will support Microsoft's long-awaited 286 operating system.

### CONCLUSION

At a street price of $700-$800, Classic Technology's Speed Pak is a reasonable alternative for users who wish to upgrade their PC, XT, or compatible while not taking a substantial loss on their hardware or software investment. Although the board does not convert the system magically into an AT, it does offer respectable performance, excellent compatibility, and a future upgrade path to the new 80286 DOS or multiuser operation. §

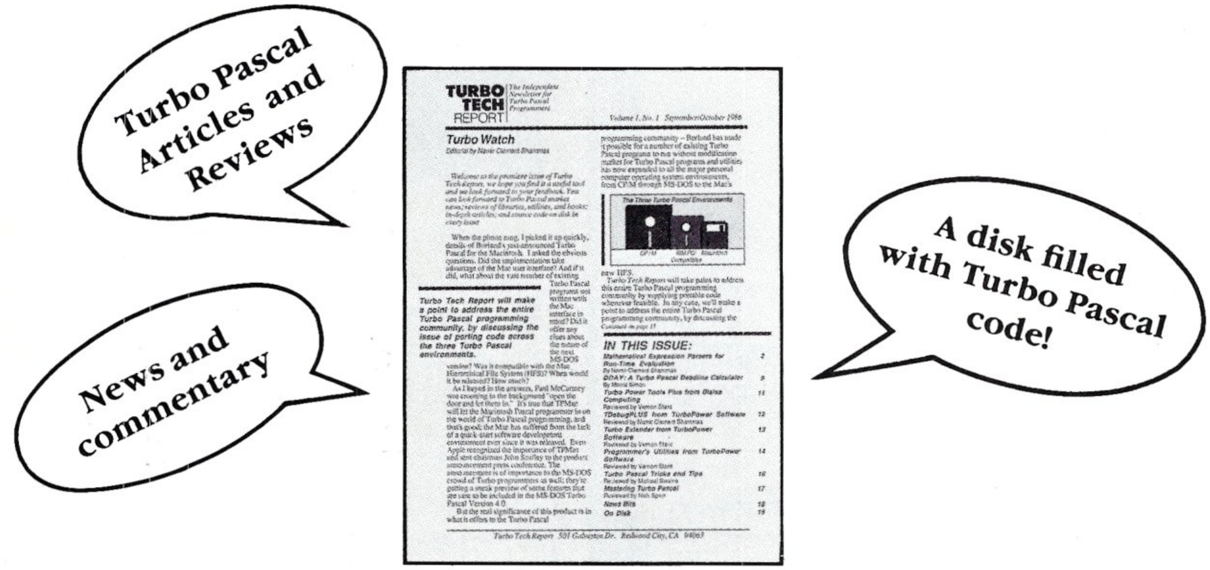
### PRODUCT INFORMATION

| | |
|---|---|
| 286 Speed Pak (8 MHz) | $995 |
| 286 Speed Pak (10 MHz) | $1,095 |
| Workstation adapter* | $495 |
| 2.5-megabyte memory module* | $895 |

*both components required for multiuser operation

**Classic Technology Corp.**
2090 Concourse Dr.
San Jose, CA 95131
(408) 434-9333

# Turbo Tech Report Speaks Your Language.




**The newsletter/disk publication for Turbo Pascal® users**

Are you a devoted Turbo Pascal programmer, tired of reading about other languages? Are you looking for powerful utilities written in Turbo Pascal that you can use to develop software or incorporate into your programs? Are you interested in improving and expanding your Turbo Pascal programming skills?

Then you deserve a subscription to *Turbo Tech Report*, the bimonthly newsletter/disk publication from the publishers of *Dr. Dobb's Journal* and *Micro/Systems Journal*. Each issue delivers more than 250K of Turbo Pascal source code programs on disk, and 20+ pages of articles, Turbo Pascal software and book reviews, and analysis and commentary. It's the only publication delivering such focused technical articles with code on disk—and it doesn't waste your time with information about other programming languages. Each valuable issue contains:

- ***Articles*** on topics like speedy 3D graphics, mathematical expression parsers, creating global gotos, memory resident and AI applications and more—all written by Turbo experts.
- ***Reviews*** the latest Turbo Pascal software programs from companies like Borland International, Blaise Computing, Media Cybernetics, Nostradamus, TurboPower Software, and more!
- ***News and commentary*** detailing the latest products and developments in the Turbo Pascal programming community.
- ***A disk filled with Turbo Pascal code!*** You'll get the Turbo Pascal utilities and routines discussed in the newsletter's articles, as well as applications developed by Turbo users from around the world. You'll receive programs that make labels, generate menus, provide faster screen access, transfer files between CP/M and MS-DOS computers, and more!

If you're an expert Turbo Pascal programmer or a novice interested in expanding your Turbo skills, you need a publication that speaks your language: *Turbo Tech Report*. Subscribe today at the special price of just $99—that's 33% off the regular price of $150. To order by credit card, call toll-free 1-800-528-6050 ext. 4001 and ask for item 300. Or mail the attached coupon with your payment to *Turbo Tech Report*, 501 Galveston Drive, Redwood City, CA 94063.

# The Scientific Computer User

by A.G.W. Cameron

## A Comparison of TeX With Manuscript

### INTRODUCING MANUSCRIPT

For some time now rumors have been seeping out of Lotus Development Corp. about the development of a scientific word processor, centered around the programming efforts of Jonathan Sachs. Now it is official: the product is Manuscript ($495). At the announcement, held at a newly opened Marriott Hotel in Cambridge, Massachusettes, I got some feeling for the way the product was developed and what the objectives were.

Jonathan Sachs has long been my favorite programmer—not because of 1-2-3, of which he was the principal author, because I do not use spreadsheets, but because of a lot of highly useful software that he wrote several years earlier for use on Nova minicomputers, of which I had two until recently. This software comprised a variety of utilities, and because I had the source code for all of it, I was able to add several enhancements of my own. I learned assembly-language programming by playing with Sachs' source code, and I learned to appreciate some of the nice techniques he used. Thus the advent of a Sachs word processor for scientists and engineers was something that I looked forward to.

After the announcement, my verdict is this: Manuscript is an excellent word processor for anyone *but* scientists and engineers—unless the scientists and engineers don't want to use equations.

To understand why this is so, let's examine the motive that led Sachs to the development of Manuscript. At the introduction Lotus showed a nice film that went into this history in a little detail. Sachs was trying to prepare a report using an outline program, one that allowed the major headings in the report to be seen on the screen together, suppressing the display of the lower-level text material. He became very dissatisfied with the outline program and decided that he should be able to throw together an improved utility relatively quickly. While he was starting to do this, Lotus established its Engineering and Scientific Products Division, which was intended to help broaden its product base. This division heard about Sachs' project and got together with him to transform this project into a full-featured product. Thus, as Lotus put it, "a 3-month hack turned into an 18-month development cycle." Sachs remained the principal author, and apparently most of the time the effort did not involve more than about four programmers.

Thus Manuscript has as its focal point the outline capability. Lotus is correct in thinking that any kind of professional people, not just scientists and engineers, can benefit from a word processor built around this capability. Manuscript supports a large number of printers, and thus a lot of the effort in formatting documents depends upon the capabilities of different printers. It should be noted that most attempts at establishing democracy give the aristocracy short shrift. In this case, the aristocracy consists of printers such as the Apple LaserWriter, whose capabilities are not properly exploited because most printers are incapable of doing many of the things that can be done with a good page-description language such as PostScript. Manuscript also provides a full-page preview, which approximates the way an output page will look and also provides a magnifying window that allows the text to be read. It appears that equations (which are treated like pieces of graphics) were added toward the end of the project, almost as an afterthought. They are by far the weakest aspect of Manuscript.

### OBJECTIVES OF TeX

The principal objective of the TeX formatting program, on the other hand, was to allow mathematicians to typeset equations "beautifully." It has become the principal method for formatting documents of all kinds at my own institution, the Center for Astrophysics. It is well suited to the work of scientists and engineers—the target for Manuscript—which is my primary motivation for comparing these two programs in this review. In some ways the two programs are complementary and can feed on each other's strengths.

The author of TeX is Donald E. Knuth of the Computer Science Department at Stanford University. His motivation was to produce a scheme that would allow him to typeset the mathematics in his well-known books of computational algorithms in a manner that met his high standards of aesthetics. The program he produced now runs on a wide variety of mainframes, minicomputers, and recently on both PCs and Macs. On all these machines, a manuscript that has been passed through the TeX processor is converted into a standard ASCII device-independent (.DVI) output file. The name and particular logo TeX have been copyrighted by the American Mathematical Society, and any program that wishes to call itself TeX must produce a standard .DVI file from a complicated input file that has variously been called a "torture test" or a "trip test." Donald Knuth himself passes judgment on whether this has been done correctly. Thus TeX is intended to be quite machine-independent, to give the same output on a given printer driven by different computers, and to give output on any other printer positioned the same way on the page. Even the fonts have been standardized and are intended to look the same apart from differences in resolution on different machines.

Thus TeX as a standard program is not customized for any particular computer (or at least it was not until recently). The TeX authentication process does not prevent the addition of features that will help users, particularly visually oriented aids. This is what is now happening on the microcomputer implementations of TeX. There are two vendors of TeX for microcomputers—Addison-Wesley and Personal TeX—and each sells both a PC version and a Mac version. I think that all these implementations could benefit from borrowing some techniques from Manuscript, and Manuscript could benefit by importing some aspects of the TeX program.

You can use any editor capable of producing ASCII output to prepare a TeX source file. TeX is the ultimate generalized markup language; nearly all of its control sequences entered into a source file are intended to deal with the types of fonts and the positions of things on a page. TeX is verbose and needs to be because it contains many hundreds of control sequences and you are likely to add some of your own. Thus abbreviations may be hard to remember, but of course you can always make your own abbreviations by defining verbose control sequences as brief ones if you prefer.

Mathematics and the physical sciences make use of large numbers of special symbols that you cannot find on standard keyboards or in the ASCII character sequence. This is why TeX is usually printed on machines—usually dot-matrix and laser printers—that reproduce graphic images using the raster principle. A large number of fonts has been especially prepared for use with TeX, and some or most of these fonts are supposed to be available for use with all TeX-driven printers. When TeX determines where characters are to be placed on a page, it needs to know the height (above a baseline), depth (below a baseline), and width of each character that is used. This information is stored in TeX Font Metric (TFM) files, and TeX and all printer drivers must have access to the TFM files for the fonts that are used in a source document. You can use other fonts not in the standard set with TeX, provided TFM files exist for these fonts. This is the case for internal Apple LaserWriter fonts, which I will use in one of the illustrations presented later.

I have given a more extensive review of TeX in the September/October 1985 issue of *Micro/Systems Journal*, which you should consult for a review of the language in greater depth.

## USING MANUSCRIPT

Manuscript makes use of a tremendous number of menus. You can select items by using the cursor motion keys to highlight an item and then using Enter, or you can just type the first letter of an item. For the more common items, you can use "accelerator keys," which are dedicated control keys and which accomplish an action directly without requiring you to go through about four steps using menus. The richness of this list of selections is such that it is difficult to remember all the menus, let alone what is available on each one. The help facility is quite good for reminding you what to do, though.

Documents are considered to be either structured or unstructured. You must structure a document in order to use the outline facilities. In a structured document there will be a certain number of headings at level 1, then under each of these headings there may be subheadings at level 2, subsubheadings at level 3, and so on. The last level is usually the main text of the document, with particular parts of the text associated with a preceding level of subsubheading. The higher-numbered levels can be suppressed from display. Thus you can see the outline of the whole document on the screen by looking only at level 1 headings, and you can expand the substructure under each of the principal headings at will in order to look at the outline of a particular part of the document. None of this is possible with an unstructured document. Only unstructured documents can be exported as ASCII files, however, and any ASCII file imported is unstructured.

Even unstructured documents are divided into blocks. TeX requires a blank line to be left between paragraphs, and such paragraphs become individual blocks when imported into Manuscript. The screen displays a solid horizontal line between blocks. When a given block is imported into Manuscript, it is reformatted, with carriage returns being replaced by spaces.

In a structured document it is very easy to rearrange headings and subheadings. Simply cut out such a heading into the clipboard and insert it elsewhere. Everything subordinate to a given heading moves with it. The process appears to be instantaneous because the text is not actually moved around in memory. The individual blocks within Manuscript are located in memory with pointers, and cutting and pasting consists simply of changing the pointers in a linked list. I have learned that this is a favorite Sachs approach to programming, and it is one of the strongest features of the program.

A large part of Manuscript deals with formatting. You can format an unstructured document globally and also locally, down to specific words if desired. Four accelerator keys are assigned to changing the typestyle: normal, italic, bold, and underlined. Manuscript only recognizes these four styles, but it allows you to vary the fonts that correspond to these styles locally. If you want to make many such font changes (and if your printer will allow you to do so), then the large amount of navigation through menus that you must endure makes this process much clumsier than the TeX practice of letting you change font and type size with a simple declaration.

A structured document allows separate formats to be established at every structural level, as well as individual treatment of the title page, of the first page, and of odd- and even-numbered pages if desired. Because there are accelerator keys that allow the headings at a given level to be promoted or demoted a level, together with attached text, the formats will automatically be reassigned at the same time. This is a nice touch.

Now I return to a theme I mentioned earlier—the support of a large number of printers. Not counting minor variations in a given line of printers, such as different carriage widths, I count that Manuscript currently supports 22 printers. These cover all kinds of printers, from laser to daisywheel printers, with a variety of dot-matrix printers in between. The most capable printer of the lot is the Apple LaserWriter (with or without the Plus upgrade), and this is so much more capable than the others that Lotus apparently did not consider it worthwhile to build into Manuscript the capabilities needed to really exploit the virtues of the PostScript page-description language. Most of the cartridges for the Hewlett-Packard LaserJet series are supported, but HP's stinginess in providing memory for those printers means that you may have to play some fancy trade-offs in how you use the available resolution. On a dot-matrix printer such as the Epson LQ1500, only some of the native fonts are supported, and no use is made of the ability to download a character set to this printer to enable a wider variety of choices. On a daisywheel printer such as the Diablo 630, only a few printwheels are supported, and changing the printwheels to enable special fonts to be used in the midst of text is not allowed. Perhaps more imaginative use of printer capabilities will be implemented in later versions of Manuscript, but for now I must give this situation rather low marks.

Manuscript does allow the importation of graphics and tabular material into itself, and the merging of this graphics with text in the output process, provided the printer will support it. The sources of this input material are very limited, however. You can input material from 1-2-3 and Symphony (of course!). You can import and export Document Content Architecture (DCA) files. And you will be able to import graphics files from Lotus' Freelance Plus graphics program as soon as it is released, which is expected momentarily as of the time of writing. These are capabilities that TeX lacks, although my TeX driver for the Apple LaserWriter does allow me to merge graphics (which I write in PostScript) with text (see my article on TeX and PostScript in the March/April 1986 issue of *Micro/Systems Journal*).

Tables are notoriously hard to prepare in TeX. Perhaps as partial compensation, using TeX you can obtain with some precision a variety of effects in tables that are hard to do with other text-formatting programs. In contrast, making tables with Manuscript is very easy, and there is a lot of flexibility in how they are laid out. Column dividers can be doubled and/or made thick. Columns can be treated as units and interchanged within a table. For most purposes the Manuscript method of making tables is thus preferable, although some of the fine control available with TeX is missing.

Other major features of Manuscript include automatic hyphenation with manual overrides if desired; spelling checking; merging of boilerplate material; and insertion of ASCII text picked up from DOS calls, such as the insertion of the date. Two documents can be edited using a pair of windows. Then there are the equation and preview features, which merit some examination in the following section.

## EQUATIONS AND PREVIEW

With TeX you can present equations in two modes—an in-line or equation mode and a display mode. TeX continues to use the same point size as in the text within which the equation is embedded. Mathematical functions such as cos, tan, or log use the normal style of the surrounding text, usually Roman but perhaps a form of sans serif font. Mathematical variables appear in a special TeX math italic font, or a Greek font, or a special symbols font, or whatever is appropriate (the TeX fonts provide a large number of possibilities). The spacing employed for equations is quite different from that for ordinary text, and allows you to make changes in this spacing, even rather miniscule ones, as may be appropriate.

To compare the behaviour of TeX and Manuscript, I prepared a short paragraph containing several rather long words as well as in-line equations and a display equation. This was intended not only to show the way in which the equations appear but also to show how well these two programs do in breaking paragraphs into lines when the column width is narrow, as is the case in this magazine.

Figure 1 shows this paragraph as output by TeX using 9-point Times Roman type with the Apple LaserWriter. The paragraph is reasonably appealing to the eye, and the program has broken the paragraph into lines in such a way that no crowded line appears next to a very loose line. The display equation was produced from the input shown in Figure 1a.

In Manuscript equations are treated as graphic inserts. The manual states: "Manuscript automatically sizes every equation to best fit the amount of space on the page." *I cannot emphasize too much that that is absolutely the wrong thing to do* and I'll explain why. Figure 2 shows the same paragraph as that used for Figure 1 as output from Manuscript in 9-point Times Roman using the Apple LaserWriter. Note how the intercharacter spacing in this paragraph is less than that used by TeX, so the font metrics used for the same font are clearly different. Note also that the division of the paragraph into lines is quite pleasing. But what a mess the equations have made of things! First, although the display equation is much too large, the infinity symbol is much too small. The major error, though, is that the in-line equations have been made into display equations in the middle of a sentence. Manuscript uses an equation notation that seems to have been borrowed from the troff facility in Unix, which is much inferior to TeX. The last equation in Figure 2 was output by the input in Figure 2a.

Manuscript allows you to modify the appearance of equations in a variety of ways. For example, adding "type=text" outside the double quotes in Figure 2a "treats the formula as text and not as a separate graphic image; the equation is

---

**The only thing special about a special function is that it has arbitrarily been so named. Special functions are also called higher transcendental functions, or functions of mathematical physics, or functions which satisfy certain frequently-occurring second-order differential equations. One special function is the error function erf($x$); this and the complementary error function erfc($x$) = 1 − erf($x$) are special cases of the incomplete gamma function. The definition of the complementary error function is**

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$$

**Figure 1**

```
$${\rm erfc(}x{\rm)} = {2 \over
\sqrt{\pi}}\int_x^\infty
e^{-t^2}\/dt$$.
```

**Figure 1a**

**The only thing special about a special function is that it has arbitrarily been so named. Special functions are also called higher transcendental functions, or functions of mathematical physics, or functions which satisfy certain frequently-occurring second-order differential equations. One special function is the error function** $\mathrm{erf}(x)$

**; this and the complementary error function** $\mathrm{erfc}(x) = 1 - \mathrm{erf}(x)$

**are special cases of the incomplete gamma function. The definition of the complementary error function is**

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^\infty e^{-t^2} dt$$

**Figure 2**

```
\equation "center 'erfc'(x) = [2
over root pi] int super infinity
sub x [e super [-t super 2]] dt
"\.
```

**Figure 2a**

sized to fit on a single line and between surrounding text." That seemed promising, so I tried it in each of the three places where an equation or mathematical expression appeared in my sample paragraph. The result is shown in Figure 3. It is still not at all satisfactory—the in-line equations are still treated as graphic images and the character sizes are all too small compared to those in the surrounding text. The display equation is practically unreadable.

Evidently one more refinement was required. Manuscript also lets you insert the statement "scale=*scale*" in the equation expression, where *scale* is a number that multiplies the dimensions. So, by playing around with this technique, you can size individual equations so that their text size approximately matches that of the surrounding text. An in-line equation will not be divided between lines, however, so you may have to fiddle around with dividing the in-line equation into two parts to make things look right. I find all this totally unacceptable.

Let me show you what happens with a dot-matrix printer. Figure 4 shows my paragraph output on the Epson LQ1500, with "type=text" used for the in-line equations but not for the display equation. Notice that the type size is a lot larger, approximately 12 point. There has been a lot more trouble dividing the paragraph into lines at this point size. Notice also that the typestyle of the equations is different from that of the surrounding text, showing that Manuscript *can* print things in graphics mode on dot-matrix printers when it wants to.

For comparison, I'll now show you what TEX can do on the LQ1500. The first thing to note is that I had to use a different font from that used in Figure 1 because Times Roman is not a TEX font, although if I specified Computer Modern Roman at 9 points the output would look quite similar to that in Figure 1, except fuzzier. To provide a better comparison with Figure 4, though, I used 12-point type. The result is shown in Figure 5. Notice that TEX had about the same trouble breaking the paragraph into lines, and it chose to do it in different places. The equations look much better than those produced by Manuscript, and they were obtained quite straightforwardly.

Manuscript cannot really claim to be a document-preparation system for scientists and engineers until its equation feature is thoroughly overhauled. Because Manuscript produces its equations using

The only thing special about a special function is that it has arbitrarily been so named. Special functions are also called higher transcendental functions, or functions of mathematical physics, or functions which satisfy certain frequently-occurring second-order differential equations. One special function is the error function erf($x$); this and the complementary error function erfc($x$) = 1 − erf($x$) are special cases of the incomplete gamma function. The definition of the complementary error function is

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}\,dt$$

**Figure 3**

The only thing special about a special function is that it has arbitrarily been so named. Special functions are also called higher transcendental functions, or functions of mathematical physics, or functions which satisfy certain frequently-occurring second-order differential equations. One special function is the error function erf($x$); this and the complementary error function erfc($x$)=1−erf($x$) are special cases of the incomplete gamma function. The definition of the complementary error function is

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}\,dt$$

**Figure 4**

The only thing special about a special function is that it has arbitrarily been so named. Special functions are also called higher transcendental functions, or functions of mathematical physics, or functions which satisfy certain frequently-occurring second-order differential equations. One special function is the error function erf($x$); this and the complementary error function erfc($x$) = 1 − erf($x$) are special cases of the incomplete gamma function. The definition of the complementary error function is

$$\mathrm{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}\,dt$$

**Figure 5**

# Quality S-100 Products

## Now Available! FULCRUM'S New...

# 256K STATIC RAM

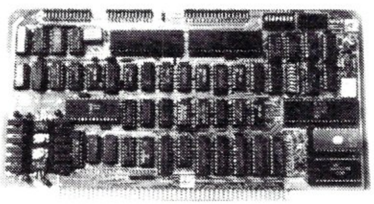
- 8/16 Bit Transfers
- 24 Bit Extended Addressing
- Battery Back-Up

**$295**

---

*Designed For Speed & Flexibility.*

## MPUZ CPU

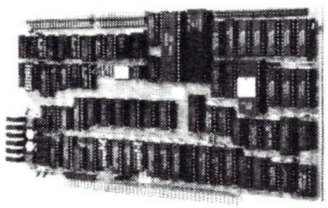


This NEW MPUZ CPU utilizes the Z-80 8MHz uP as a basis for its 8MHz CPU for S-100 systems, and has been carefully designed to meet the requirements of the IEEE - 696 standard. The quality and performance this CPU provides is rarely found in S-100 products, and you can see why... ***only $299***

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

*Best Value In Disk Controllers.*

## OMNIDISK



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

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

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

*CP/M *2 2 configured for OMNIDISK $60 *Trade mark of Digital Research FREE U.P.S ground shipping on prepaid orders Shipping is added to VISA, M/C, and C.O.D orders CA residents, please add sales tax

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

*Also in FULCRUM'S Family* OMNIRAM 64K memory board Serial I/O 2:2 & Video I/O Interface boards. Relay board. I-8080. 8015 and 8035 main frames with 21 slot mother boards. CPA front panel and A/D board graphics output anyway, there is no reason why a much larger range of special symbols should not be made available when this is done. That is, there needs to be a mechanism for importing fonts into Manuscript just as TeX has. And that ridiculous sizing mechanism must be totally abolished!

I am happy to report that the Manuscript preview feature works much better. This requires a Hercules graphics card (or the Plus upgrade) or an EGA card. Manuscript presents you with a screen divided into a left half and a right half. On the left appears an approximation of an output page. I say approximation because Manuscript substitutes its own vectorized preview fonts for the ones that the printer will actually use. This page outline is unreadable because of the smallness of the type under ordinary circumstances, but superimposed upon it is a rectangular window that you can move around with the cursor keys. This corresponds to the magnification window that appears on the upper right of the screen. You can shrink the rectangle on the left, and the text will grow correspondingly larger on the right. There are nine of these magnification steps. This is very effective and has good interaction with the user.

When you look at the page outline, an equation appears as a series of small rectangular blobs. If you center the rectangular window on such an equation and invoke the graphic zoom, a large rectangle appears in the middle of the screen, and the equation is blown up to fill as much of the rectangle as it can. This is fine for examining the details of equations, but it is poor for comparing the relative sizes of different equations. It would have been so much better, if somewhat slower, to have used graphics mode for the text as well, so that you could have examined the equation in context with the accompanying text.

This is what happens with various TeX preview techniques. The first TeX previewer was introduced by Textset (now Arbortext). It uses three sets of TeX font files at different resolutions to show how the actual TeX output will look at three different magnifications. It operates rather slowly but has the advantage that you can change magnification while examining the text. Personal TeX has just introduced a previewer called Maxview ($125), which I prefer to the Arbortext preview program. Maxview uses the actual font files that will be used to print the TeX file. It projects these on the screen and combines some number of pixels in the output into a single pixel on the screen. Typically you would combine four pixels into one for 180-dot-per-inch fonts, such as are used with the Epson LQ1500, and nine pixels into one for 300-dot-per-inch fonts, such as are used with laser printers. I find this kind of preview more legible than the other kind. It is unfortunate that you must choose your magnification at the start of a viewing session and cannot change it in the middle of the session. At the TeX users' group meeting this summer, Addison-Wesley showed a previewer that was under development for incorporation into its MicroTeX, but which will not be available for many months to come. This allowed you to see the preview of the output as the TeX program processed the document. It also depended on the use of special low-resolution TeX fonts. This is an example of the kind of customization of the basic TeX program that is going on for microcomputer users. I understand that the Addison-Wesley version of TeX for Macs displays a magnification window somewhat like that in Manuscript. Thus Manuscript and microcomputer versions of TeX are running along parallel paths as far as preview is concerned.

This comparison review is intended to help you decide if Manuscript will be useful for your purposes. Manuscript is very good for a lot of things, but *don't* use it for equations. §

*A.G.W. Cameron is professor of astronomy at the Harvard-Smithsonian Center for Astrophysics, 60 Garden St., Cambridge, MA 02138.*

### ADDRESSES

**Addison-Wesley Publishing Co.**, Reading, MA 01867; (617) 944-3700.

**Arbortext Inc.**, 416 Fourth St., P.O. Box 7993, Ann Arbor, MI 48107; (313) 996-3566.

**Lotus Development Corp.**, 55 Cambridge Pkwy., Cambridge, MA 02142; (617) 577-8500.

**Personal TeX Inc.**, 20 Sunnyside, Ste. H, Mill Valley, CA 94941; (415) 388-8853.

**Multitasking Between Programs**
*Continued from page 31*

Incidentally, some of the multitaskers are quite adroit at figuring out if a program is waiting for keyboard input and will donate that program's time slice automatically to other programs. DesqView seems to do this best. Under TaskView, a program waiting for keyboard input is suspended automatically and its time donated to other tasks.

### WriteSXY and WriteCXY

These routines allow for writing strings (WriteSXY) and single characters (WriteCXY) to the display. It may be either the real screen buffer or the virtual screen buffer, if Write_Screen_Memory is true. If Write_Screen_Memory is false, then the standard BIOS calls using video interrupt $10 are used instead.

Note that the (X,Y) column positions do not take account of any windows set using the Turbo WINDOW procedure.

WriteSXY demonstrates how a task can be frozen/unfrozen in DoubleDOS.

WriteCXY and WriteSXY are based in part on routines written by Brian Foley.

### CLEARING THE SCREEN

I have run into trouble when trying to clear the screen using the built-in Turbo procedure ClrScr under several of the multitaskers. What happens is that the screen is cleared even when a task is running in the background, or sometimes that entire screen is cleared when only a small portion should be (because a Turbo window has been defined using WINDOW).

To avoid this problem, I call the BIOS scroll routine and indicate that I want to scroll zero lines. This, by definition, clears the indicated area of the screen. I find that this works correctly under all of the multitaskers.

The routine Scroll performs the BIOS call, and the routine Clear_Screen in the sample program invokes Scroll to clear the screen. Clear_Screen also invokes MoveToXY, which calls the BIOS to move the cursor to a specified screen position. The BIOS function to move the cursor is quite efficient.

Why use MoveToXY rather than Turbo's GoToXY? The Turbo routine takes notice of Turbo windows, while MoveToXY does not.

### THE SAMPLE PROGRAM TRYMDOS

The sample program TRYMDOS.PAS demonstrates the use of the multitasker in-

```
   SegS    := Regs.Es;

   INTR( $10 , Regs );

   Virtual_Screen := PTR( Regs.Es , Regs.Di );

   Get_TopView_Screen_Address := ( ( Regs.Es <> SegS ) OR ( Regs.Di <> SegO ) );

END   (* Get_TopView_Screen_Address *);

(*---------------------------------------------------------------------------*)

BEGIN (* IsTimeSharingActive *)
                                    (* Assume timesharing not active *)
   IsTimeSharingActive := FALSE;
   MultiTasker         := MultiTasker_None;

                                    (* Get initial screen address    *)
   IF Color_Screen_Active THEN
      Virtual_Screen := PTR( Color_Screen_Address , 0 )
   ELSE
      Virtual_Screen := PTR( Mono_Screen_Address  , 0 );

                                    (* If DDos is active, $E4 should *)
                                    (* return a non-zero value in Al *)
   Regs.Ax := $E400;
   MsDos( Regs );

   IF ( Regs.Al <> 0 ) THEN
      BEGIN
         IsTimeSharingActive := TRUE;
         MultiTasker         := DoubleDos;
         EXIT;
      END;
                                    (* See if DesqView is active.       *)
                                    (* We do a time/date call with      *)
                                    (* DESQ as date.  If DesqView is    *)
                                    (* active, this will be accepted.   *)
                                    (* If not, it returns as invalid.   *)
                                    (* While we're at it, get the       *)
                                    (* display buffer address, which    *)
                                    (* never changes.                   *)
                                    (*                                  *)
                                    (* NOTE:  Newer versions of TaskView *)
                                    (*        also accept this DesqView  *)
                                    (*        call, so we must check the *)
                                    (*        TopView number to differ-  *)
                                    (*        entiate them.              *)
   Regs.Ax := $2B01;
   Regs.Cx := $4445;   (*'DE'*)
   Regs.Dx := $5351;   (*'SQ'*)
   MsDos( Regs );

   IF ( Regs.Al <> $FF ) THEN
      IF Get_TopView_Screen_Address THEN
         BEGIN

            IsTimeSharingActive := TRUE;
                                    (* Distinguish TaskView from TopView *)
            Regs.Ax := $1022;
            Regs.Bx := 0;
            INTR( $15 , Regs );

            IF ( Regs.Bx = 1 ) THEN
               MultiTasker         := TaskView
            ELSE
               MultiTasker         := DesqView;

            EXIT;

         END;
                                    (* Check for TaskView or TopView.  We do   *)
                                    (* a request for a TopView version number. *)
                                    (* If BX comes back $0001, this must be    *)
                                    (* TaskView.  Anything non-zero indicates  *)
                                    (* TopView or a compatible program.        *)
                                    (* Note:  This catches older TaskView      *)
                                    (*        versions which don't understand  *)
                                    (*        the DesqView call.               *)
   Regs.Ax := $1022;
```

```
   Regs.Bx := 0;
   INTR( $15 , Regs );

   IF ( Regs.Bx <> 0 ) THEN
      BEGIN

         IF ( Regs.Bx = 1 ) THEN
            MultiTasker        := TaskView
         ELSE
            MultiTasker        := TopView;

         IF ( NOT Get_TopView_Screen_Address ) THEN
            MultiTasker := Multitasker_None
         ELSE
            IsTimeSharingActive := TRUE;

      END;

END   (* IsTimeSharingActive *);

(*--------------------------------------------------------------------------*)
(*    TurnOnTimeSharing --- allow timesharing to proceed                    *)
(*--------------------------------------------------------------------------*)

PROCEDURE TurnOnTimeSharing;

(*--------------------------------------------------------------------------*)
(*                                                                          *)
(*    Procedure:  TurnOnTimeSharing;                                        *)
(*                                                                          *)
(*    Purpose:    Activates timesharing                                     *)
(*                                                                          *)
(*    Calling Sequence:                                                     *)
(*                                                                          *)
(*       TurnOnTimeSharing;                                                 *)
(*                                                                          *)
(*    Calls:  MsDos                                                         *)
(*                                                                          *)
(*--------------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN (* TurnOnTimeSharing *)

   CASE MultiTasker OF
                                   (* If DDos is active, $EB turns  *)
                                   (* on timesharing                *)
      DoubleDos:   BEGIN
                      Regs.Ax := $EB00;
                      MsDos( Regs );
                   END;
                                   (* Int 15H for TopView family products *)
      DesqView,
      TopView,
      MSWindows,
      TaskView:    BEGIN
                      Regs.Ax := $101C;
                      INTR( $15 , Regs );
                   END;

      ELSE;

   END (* CASE *);

END   (* TurnOnTimeSharing *);

(*--------------------------------------------------------------------------*)
(*       TurnOffTimeSharing --- suspend timesharing under DoubleDos         *)
(*--------------------------------------------------------------------------*)

PROCEDURE TurnOffTimeSharing;

(*--------------------------------------------------------------------------*)
(*                                                                          *)
(*    Procedure:  TurnOffTimeSharing;                                       *)
(*                                                                          *)
(*    Purpose:    Suspends timesharing                                      *)
(*                                                                          *)
(*    Calling Sequence:                                                     *)
```

terface routines in a simple way.

After determining which multitasker is active, TRYMDOS begins writing a series of lines directly to the screen buffer memory (real or virtual). This continues until you press a key.

Then a test of donating time to other partitions is performed. Reference is made in the output of TRYMDOS to the public-domain program CPU.COM, which provides a measure of the effective clock speed for a given hardware and software configuration. That program is extremely handy for comparing the throughput of the various multitaskers under different program loads. CPU.COM is generally available on many BBS systems. You can also use other programs, like Norton's SI.

If you do not run TRYMDOS under a multitasker, you will be prompted as to whether to use direct screen writes and also whether to wait for the retrace signals. This lets you compare the relative speeds of direct screen writing versus BIOS writes and also whether your machine produces snow in color graphics mode. §

*Philip R. Burns is a statistical analyst in the Vogelback Computing Center of Northwestern University in Evanston, Illinois. He has 18 years of experience with a variety of mainframe, mini, and micro systems. He is the author of the PIB*.* Turbo Pascal programs available on many bulletin-board systems.*

---

*All the source code for articles in this issue is available on a single MS-DOS disk. To order send $14.95 to Micro/Systems Journal, 501 Galveston Dr., Redwood City, CA 94063 or call (415) 366-3600, ext. 216. Please specify the issue number.*

```
(*                                                                         *)
(*      TurnOffTimeSharing;                                                *)
(*                                                                         *)
(*    Calls:  MsDos                                                        *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN (* TurnOffTimeSharing *)

   CASE MultiTasker OF
                                       (* If DDos is active, $EA suspends *)
                                       (* timesharing                     *)
      DoubleDos:   BEGIN
                      Regs.Ax := $EA00;
                      MsDos( Regs );
                   END;
                                       (* Int 15H for TopView family products *)
      DesqView,
      TopView,
      MSWindows,
      TaskView:    BEGIN
                      Regs.Ax := $101B;
                      INTR( $15 , Regs );
                   END;

     ELSE;

   END (* CASE *);

END   (* TurnOffTimeSharing *);

(*-------------------------------------------------------------------------*)
(*             GiveAwayTime --- gives away time slices to other task       *)
(*-------------------------------------------------------------------------*)

PROCEDURE GiveAwayTime( NSlices : INTEGER );

(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*    Procedure:  GiveAwayTime;                                            *)
(*                                                                         *)
(*    Purpose:    Gives away time slices to other tasks                    *)
(*                                                                         *)
(*    Calling Sequence:                                                    *)
(*                                                                         *)
(*       GiveAwayTime( NSlices :  INTEGER );                               *)
(*                                                                         *)
(*          NSlices --- # of slices (55 ms) to give away, if DoubleDos.    *)
(*                      For other multitaskers, the entire remaining       *)
(*                      time-slice is given up.                            *)
(*                                                                         *)
(*    Calls:  MsDos                                                        *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN (* GiveAwayTime *)

   CASE MultiTasker OF
                                 (* Function EE gives time to other part. *)
      DoubleDos:   BEGIN
                      Regs.Ah := $EE;
                      Regs.Al := NSlices;
                      MsDos( Regs );
                   END;
                                 (* Int 15H for TopView family products *)
      DesqView,
      TopView,
      MSWindows,
      TaskView:    BEGIN
                      Regs.Ax := $1000;
                      INTR( $15 , Regs );
                   END;

      ELSE;

   END  (* CASE *);

END   (* GiveAwayTime *);

(*-------------------------------------------------------------------------*)
(*    Sync_Screen --- Synchronizes multitasker screen with hardware screen *)
(*-------------------------------------------------------------------------*)

PROCEDURE Sync_Screen( S_Pos: INTEGER; NChars : INTEGER );

(*-------------------------------------------------------------------------*)
(*                                                                         *)
(*    Procedure:  Sync_Screen;                                             *)
(*                                                                         *)
(*    Purpose:    Synchronizes multitasker and hardware screens            *)
(*                                                                         *)
(*    Calling Sequence:                                                    *)
(*                                                                         *)
(*       Sync_Screen( S_Pos : INTEGER; NChars: INTEGER );                  *)
(*                                                                         *)
(*    Calls:  INTR                                                         *)
(*                                                                         *)
(*    Remarks:                                                             *)
(*                                                                         *)
(*       This facility is required by the TopView-family products.         *)
(*                                                                         *)
(*-------------------------------------------------------------------------*)

VAR
   Regs  : RegPack;
   Daddr : Screen_Ptr;

BEGIN (* Sync_Screen *)

   IF ( MultiTasker IN [TopView,MSWindows] ) THEN
      WITH Regs DO
         BEGIN

            Regs.Es := SEG( Virtual_Screen^ );
            Regs.Di := OFS( Virtual_Screen^ ) + S_Pos - 1;
            Regs.Cx := NChars SHL 1;
```

```
            Regs.Ah := $FF;

            INTR( $10 , Regs );

         END;

END   (* Sync_Screen *);

(*----------------------------------------------------------------------*)
(* Sync_Entire_Screen --- Synchronizes multitasker screen with hardware  *)
(*----------------------------------------------------------------------*)

PROCEDURE Sync_Entire_Screen;

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*    Procedure:  Sync_Entire_Screen;                                   *)
(*                                                                      *)
(*    Purpose:    Synchronizes multitasker and hardware screens         *)
(*                                                                      *)
(*    Calling Sequence:                                                 *)
(*                                                                      *)
(*       Sync_Entire_Screen;                                            *)
(*                                                                      *)
(*    Calls:  INTR                                                      *)
(*                                                                      *)
(*    Remarks:                                                          *)
(*                                                                      *)
(*       This facility is used by the TopView-family products when the  *)
(*       entire screen has been updated.                                *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

VAR
   Regs : RegPack;

BEGIN (* Sync_Entire_Screen *)

   IF ( MultiTasker IN [TopView,MSWindows] ) THEN
      WITH Regs DO

         BEGIN

            Regs.Es := SEG( Virtual_Screen^ );
            Regs.Di := OFS( Virtual_Screen^ );
            Regs.Cx := Screen_Length SHR 1;
            Regs.Ah := $FF;

            INTR( $10 , Regs );

         END;

END   (* Sync_Entire_Screen *);
```

### *Listing 7*

```
(*----------------------------------------------------------------------*)
(*         WriteSXY --- Write text string to specified row/column       *)
(*----------------------------------------------------------------------*)

PROCEDURE WriteSXY( S: AnyStr; X: INTEGER; Y: INTEGER; Color: INTEGER );

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*    Procedure:  WriteSXY                                              *)
(*                                                                      *)
(*    Purpose:    Writes text string at specified row and column        *)
(*                position on screen.                                   *)
(*                                                                      *)
(*    Calling Sequence:                                                 *)
(*                                                                      *)
(*       WriteSXY( S: AnyStr; X: INTEGER; Y: INTEGER; Color: INTEGER );*)
(*                                                                      *)
(*          S      --- String to be written                             *)
(*          X      --- Column position to write string                  *)
(*          Y      --- Column position to write string                  *)
(*          Color  --- Color in which to write string                   *)
(*                                                                      *)
(*    Calls:   None                                                     *)
(*                                                                      *)
(*    Remarks:                                                          *)
(*                                                                      *)
(*       This routine is based in part upon one by Brian Foley.         *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

BEGIN (* WriteSXY *)
                                         (* Freeze screen for DoubleDos *)

   IF ( MultiTasker = DoubleDos ) THEN
      BEGIN
         TurnOffTimeSharing;
         Get_Screen_Address( Virtual_Screen );
      END;

  INLINE(
    $1E                  {       PUSH  DS                    ;Save DS}
                         {;}
                         {;  Check if we're going to use BIOS}
                         {;}
    /$A0/>WRITE_SCREEN_MEMORY{   MOV   AL,[<Write_Screen_Memory]  ;See if we're writing to screen memory}
    /$D0/$D8             {       RCR   AL,1                  ;}
    /$73/$5D             {       JNC   BIOS                  ;No -- skip to BIOS code}
                         {;}
                         {;  Set up for direct screen write.}
                         {;  Get row position and column positions, and offset in screen buffer.}
                         {;}
```

```
/$C4/$3E/>VIRTUAL_SCREEN{          LES   DI,[>Virtual_Screen]   ;Get base address of screen}
/$8B/$46/<Y               {        MOV   AX,[BP+<Y]             ;AX = Row}
/$48                      {        DEC   AX                     ;Row to 0..24 range}
/$B9/$04/$00              {        MOV   CX,$0004               ;CL = 4; CH = 0}
/$D3/$E0                  {        SHL   AX,CL                  ;AX = Row * 16}
/$89/$C3                  {        MOV   BX,AX                  ;Store in BX}
/$D1/$E0                  {        SHL   AX,1                   ;AX = Row * 32}
/$D1/$E0                  {        SHL   AX,1                   ;AX = Row * 64}
/$01/$D8                  {        ADD   AX,BX                  ;AX = (Row * 64) + (Row * 16)}
                          {                                     ;    = Row * 80}
/$8B/$5E/<X               {        MOV   BX,[BP+<X]             ;BX = Column}
/$4B                      {        DEC   BX                     ;Col to 0..79 range}
/$01/$D8                  {        ADD   AX,BX                  ;AX = (Row * 80) + Col}
/$D1/$E0                  {        SHL   AX,1                   ;Account for attribute bytes}
/$89/$FB                  {        MOV   BX,DI                  ;Get base offset of screen}
/$01/$C3                  {        ADD   BX,AX                  ;Add computed offset}
/$89/$DF                  {        MOV   DI,BX                  ;Move result into DI}
/$8D/$76/<S               {        LEA   SI,[BP+<S]             ;DS:SI will point to S[0]}
/$A0/>WAIT_FOR_RETRACE {           MOV   AL,[<Wait_For_Retrace] ;Grab this before changing DS}
/$8C/$D2                  {        MOV   DX,SS                  ;Move SS...}
/$8E/$DA                  {        MOV   DS,DX                  ; into DS}
/$8A/$0C                  {        MOV   CL,[SI]                ;CL = Length(S)}
/$E3/$72                  {        JCXZ  Exit                   ;If string empty, Exit}
/$46                      {        INC   SI                     ;DS:SI points to S[1]}
/$8A/$66/<COLOR           {        MOV   AH,[BP+<Color]         ;AH = Attribute}
/$FC                      {        CLD                          ;Set direction to forward}
/$D0/$D8                  {        RCR   AL,1                   ;If Snow is False...}
/$73/$1C                  {        JNC   Mono                   ; use "Mono" routine}
                          {;}
                          {;  Color routine (used only when Wait_For_Retrace is True) **}
                          {;}
/$BA/>CRT_STATUS          {        MOV   DX,>CRT_Status         ;Point DX to CGA status port}
/$AC                      {GetNext: LODSB                       ;Load next character into AL}
                          {                                     ; AH already has Attr}
/$89/$C3                  {        MOV   BX,AX                  ;Store video word in BX}
/$B4/$09                  {        MOV   AH,$09                 ;Move horizontal & vertical}
                          {                                     ; retrace mask into AH}
/$FA                      {        CLI                          ;No interrupts now}
/$EC                      {WaitH:  IN    AL,DX                  ;Get 6845 status}
/$D0/$D8                  {        RCR   AL,1                   ;Wait for horizontal}
/$72/$FB                  {        JC    WaitH                  ; retrace}
/$EC                      {WaitV:  IN    AL,DX                  ;Get 6845 status again}
/$20/$E0                  {        AND   AL,AH                  ;Wait for vertical}
/$74/$FB                  {        JZ    WaitV                  ; retrace}
/$89/$D8                  {        MOV   AX,BX                  ;Move word back to AX...}
/$AB                      {        STOSW                        ; and then to screen}
/$FB                      {        STI                          ;Allow interrupts}
/$E2/$EA                  {        LOOP  GetNext                ;Get next character}
/$E9/$4D/$00              {        JMP   Exit                   ;Done}
                          {;}
                          {;  Mono routine (used whenever Wait_For_Retrace is False) **}
                          {;}
/$AC                      {Mono:   LODSB                        ;Load next character into AL}
                          {                                     ; AH already has Attr}
/$AB                      {        STOSW                        ;Move video word into place}
/$E2/$FC                  {        LOOP  Mono                   ;Get next character}
/$E9/$46/$00              {        JMP   Exit                   ;Done}
                          {;}
                          {;  Use BIOS to display string (if Write_Screen is False) **}
                          {;}
/$8A/$76/<Y               {Bios:   MOV   DH,[BP+<Y]             ;Get starting row}
/$FE/$CE                  {        DEC   DH                     ;Drop by one for BIOS}
/$8A/$56/<X               {        MOV   DL,[BP+<X]             ;Get starting column}
/$FE/$CA                  {        DEC   DL                     ;Drop for indexing}
/$FE/$CA                  {        DEC   DL                     ;}
/$8D/$76/<S               {        LEA   SI,[BP+<S]             ;DS:SI will point to S[0]}
/$8C/$D0                  {        MOV   AX,SS                  ;Move SS...}
/$8E/$D8                  {        MOV   DS,AX                  ; into DS}
/$31/$C9                  {        XOR   CX,CX                  ;Clear out CX}
/$8A/$0C                  {        MOV   CL,[SI]                ;CL = Length(S)}
/$E3/$2D                  {        JCXZ  Exit                   ;If string empty, Exit}
/$46                      {        INC   SI                     ;DS:SI points to S[1]}
/$52                      {        PUSH  DX                     ;Save X and Y}
/$1E                      {        PUSH  DS                     ;Save string address}
/$56                      {        PUSH  SI                     ;}
/$FC                      {        CLD                          ;Forward direction}
                          {;}
/$B4/$02                  {Bios1:  MOV   AH,2                   ;BIOS Position cursor}
/$B7/$00                  {        MOV   BH,0                   ;Page zero}
/$5E                      {        POP   SI                     ;Get S address}
/$1F                      {        POP   DS                     ;}
/$5A                      {        POP   DX                     ;X and Y}
/$FE/$C2                  {        INC   DL                     ;X + 1}
/$52                      {        PUSH  DX                     ;Save X and Y}
/$1E                      {        PUSH  DS                     ;Save strin address}
/$56                      {        PUSH  SI}
/$51                      {        PUSH  CX                     ;Push length}
/$CD/$10                  {        INT   $10                    ;Call BIOS to move to (X,Y)}
```

```
    /$59                   {         POP   CX                          ;Get back length}
    /$5E                   {         POP   SI                          ;Get String address}
    /$1F                   {         POP   DS                          ;}
    /$AC                   {         LODSB                             ;Next character into AL}
    /$1E                   {         PUSH  DS                          ;Save String address}
    /$56                   {         PUSH  SI                          ;}
    /$51                   {         PUSH  CX                          ;Length left to do}
    /$B4/$09               {         MOV   AH,9                        ;BIOS Display character}
    /$B7/$00               {         MOV   BH,0                        ;Display page zero}
    /$8A/$5E/<COLOR        {         MOV   BL,[BP+<Color]              ;BL = Attribute}
    /$B9/$01/$00           {         MOV   CX,1                        ;One character}
    /$CD/$10               {         INT   $10                         ;Call BIOS}
    /$59                   {         POP   CX                          ;Get back length}
    /$E2/$DB               {         LOOP  Bios1}
                           {;                                          ;Remove stuff left on stack}
    /$5E                   {         POP   SI}
    /$1F                   {         POP   DS}
    /$5A                   {         POP   DX}
                           {;}
    /$1F                   {Exit:    POP   DS                          ;Restore DS}
  );
                                     (* Unfreeze screen in DoubleDos *)

     IF ( MultiTasker = DoubleDos ) THEN
        TurnOnTimeSharing
                                     (* Synchronize screen for TopView *)

     ELSE IF ( MultiTasker = TopView ) THEN
        Sync_Screen( ( ( Y - 1 ) * 80 + X ) SHL 1 - 1 , ORD( S[0] ) );

  END   (* WriteSXY *);
```

### *Listing 8*

```
(*----------------------------------------------------------------------*)
(*   WriteCXY --- Write character to screen  at specified row/column    *)
(*----------------------------------------------------------------------*)

PROCEDURE WriteCXY( C: CHAR; X: INTEGER; Y: INTEGER; Color: INTEGER );

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*     Procedure:  WriteCXY                                             *)
(*                                                                      *)
(*     Purpose:    Writes a character at specified row and column       *)
(*                 position on screen.                                  *)
(*                                                                      *)
(*     Calling Sequence:                                                *)
(*                                                                      *)
(*        WriteCXY( C: CHAR; X: INTEGER; Y: INTEGER; Color: INTEGER );  *)
(*                                                                      *)
(*           C       --- Character to be written                        *)
(*           X       --- Column position to write character             *)
(*           Y       --- Column position to write character             *)
(*           Color   --- Color in which to write character              *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

VAR
   Save_WTS : BOOLEAN;

BEGIN (* WriteCXY *)
                                     (* Use BIOS for DoubleDos *)

   IF ( MultiTasker = DoubleDos ) THEN
      BEGIN
         Save_WTS            := Write_Screen_Memory;
         Write_Screen_Memory := FALSE;
      END;

INLINE(
  $1E                               {         PUSH  DS                              ;Save DS}
                                    {;}
                                    {;  Check if we're going to use BIOS}
                                    {;}
  /$F6/$06/>WRITE_SCREEN_MEMORY/$01{         TEST  BYTE [<Write_Screen_Memory],1  ;See if we're writing to
screen memory}
  /$74/$4D                          {         JZ    BIOS                            ;No -- skip to BIOS code}
                                    {;}
                                    {;  Set up for direct screen write.}
                                    {;  Get row position and column positions, and offset in screen buffer.}
                                    {;}
  /$C4/$3E/>VIRTUAL_SCREEN          {         LES   DI,[>Virtual_Screen]           ;Get base address of
screen}
```

```
  /$8B/$46/<Y                  {        MOV   AX,[BP+<Y]                 ;AX = Row}
  /$48                         {        DEC   AX                         ;Row to 0..24 range}
  /$B9/$04/$00                 {        MOV   CX,$0004                   ;CL = 4  ; CH = 0}
  /$D3/$E0                     {        SHL   AX,CL                      ;AX = Row * 16}
  /$89/$C3                     {        MOV   BX,AX                      ;Store in BX}
  /$D1/$E0                     {        SHL   AX,1                       ;AX = Row * 32}
  /$D1/$E0                     {        SHL   AX,1                       ;AX = Row * 64}
  /$01/$D8                     {        ADD   AX,BX                      ;AX = (Row * 64) + (Row *
16)}
                               {                                         ;   = Row * 80}
  /$8B/$5E/<X                  {        MOV   BX,[BP+<X]                 ;BX = Column}
  /$4B                         {        DEC   BX                         ;Col to 0..79 range}
  /$01/$D8                     {        ADD   AX,BX                      ;AX = (Row * 80) + Col}
  /$D1/$E0                     {        SHL   AX,1                       ;Account for attribute
bytes}
  /$89/$FB                     {        MOV   BX,DI                      ;Get base offset of
screen}
  /$01/$C3                     {        ADD   BX,AX                      ;Add computed offset}
  /$89/$DF                     {        MOV   DI,BX                      ;Move result into DI}
  /$8A/$5E/<C                  {        MOV   BL,[BP+<C]                 ;BL = character}
  /$8A/$7E/<COLOR              {        MOV   BH,[BP+<Color]             ;BH = Attribute}
  /$F6/$06/>WAIT_FOR_RETRACE/$01 {      TEST  BYTE [<Wait_For_Retrace],1 ;Check retrace wait flag}
  /$74/$17                     {        JZ    Mono                       ; use "Mono" routine}
                               {;}
                               {;  Color routine (used only when Wait_For_Retrace is True) **}
                               {;}
  /$BA/>CRT_STATUS             {        MOV   DX,>CRT_Status             ;Point DX to CGA status
port}
  /$B4/$09                     {        MOV   AH,$09                     ;Move horizontal &
vertical}
                               {                                         ; retrace mask into AH}
  /$FA                         {        CLI                              ;No interrupts now}
  /$EC                         {WaitH:  IN    AL,DX                      ;Get 6845 status}
  /$D0/$D8                     {        RCR   AL,1                       ;Wait for horizontal}
  /$72/$FB                     {        JC    WaitH                      ; retrace}
  /$EC                         {WaitV:  IN    AL,DX                      ;Get 6845 status again}
  /$20/$E0                     {        AND   AL,AH                      ;Wait for vertical}
  /$74/$FB                     {        JZ    WaitV                      ; retrace}
  /$89/$D8                     {        MOV   AX,BX                      ;Move word back to AX...}
  /$AB                         {        STOSW                            ; and then to screen}
  /$FB                         {        STI                              ;Allow interrupts}
  /$E9/$25/$00                 {        JMP   Exit                       ;Done}
                               {;}
                               {;  Mono routine (used whenever Wait_For_Retrace is False) **}
                               {;}
  /$89/$D8                     {Mono:   MOV   AX,BX                      ;Get character +
attribute}
  /$AB                         {        STOSW                            ;Move video word into
place}
  /$E9/$1F/$00                 {        JMP   Exit                       ;Done}
                               {;}
                               {;  Use BIOS to display string (if Write_Screen is False) **}
                               {;}
  /$B4/$02                     {Bios:   MOV   AH,2                       ;BIOS positioning}
  /$B7/$00                     {        MOV   BH,0                       ;Text page 0}
  /$8A/$76/<Y                  {        MOV   DH,[BP+<Y]                 ;Y}
  /$FE/$CE                     {        DEC   DH                         ;Y - 1}
  /$8A/$56/<X                  {        MOV   DL,[BP+<X]                 ;X}
  /$FE/$CA                     {        DEC   DL                         ;X - 1}
  /$CD/$10                     {        INT   $10                        ;Call BIOS to position
cursor}
  /$B4/$09                     {        MOV   AH,9                       ;BIOS display character}
  /$8A/$46/<C                  {        MOV   AL,[BP+<C]                 ;Character to display}
  /$B7/$00                     {        MOV   BH,0                       ;Text page 0}
  /$8A/$5E/<COLOR              {        MOV   BL,[BP+<Color]             ;Color}
  /$B9/$01/$00                 {        MOV   CX,1                       ;One character}
  /$CD/$10                     {        INT   $10                        ;Call BIOS to display
character}
                               {;}
  /$1F                         {Exit:   POP   DS                         ;Restore DS}
);
                               (* Return BIOS write to previous state *)

   IF ( MultiTasker = DoubleDos ) THEN
      Write_Screen_Memory := Save_WTS

                               (* Synchronize screen for TopView *)

   ELSE IF ( MultiTasker = TopView ) THEN
      Sync_Screen( ( ( Y - 1 ) * 80 + X ) SHL 1 - 1 , 1 );

   END   (* WriteCXY *);
```

```
(*---------------------------------------------------------------------*)
(*           Read_Kbd --- Read one character from keyboard             *)
(*---------------------------------------------------------------------*)

PROCEDURE Read_Kbd( VAR Ch: CHAR );

(*---------------------------------------------------------------------*)
(*                                                                     *)
(*     Procedure:  Read_Kbd                                            *)
(*                                                                     *)
(*     Purpose:    Reads one character from the keyboard              *)
(*                                                                     *)
(*     Calling Sequence:                                               *)
(*                                                                     *)
(*        Read_Kbd( VAR Ch: CHAR );                                    *)
(*                                                                     *)
(*           Ch  --- Character read                                    *)
(*                                                                     *)
(*     Remarks:                                                        *)
(*                                                                     *)
(*        This routine centralizes single character keyboard reads so  *)
(*        that time-slicing control for multitaskers is more easily    *)
(*        centralized.  In this particular implementation, the time    *)
(*        spent waiting for a keyboard entry is donated to the other   *)
(*        partitions.                                                  *)
(*                                                                     *)
(*---------------------------------------------------------------------*)

BEGIN (* Read_Kbd *)

   WHILE ( NOT KeyPressed ) DO
      GiveAwayTime( 2 );

   READ( Kbd, Ch );

END   (* Read_Kbd *);
```

## *Listing 10*

```
(*---------------------------------------------------------------------*)
(*              Scroll --- Scroll section of screen                    *)
(*---------------------------------------------------------------------*)

PROCEDURE Scroll( Y1, Y2, X1, X2, Nlines, FG, BG : INTEGER );

(*---------------------------------------------------------------------*)
(*                                                                     *)
(*   Procedure: Scroll                                                 *)
(*                                                                     *)
(*   Purpose:   Scrolls portion of screen.                             *)
(*                                                                     *)
(*   Calling sequence:                                                 *)
(*                                                                     *)
(*      Scroll( Y1, Y2, X1, X2, Nlines, FG, BG : INTEGER );            *)
(*                                                                     *)
(*         (X1,Y1); (X2,Y2) --- corners of region to scroll            *)
(*         Nlines           --- number of lines to scroll              *)
(*         (FG,BG)          --- foreground and background colors       *)
(*                                                                     *)
(*   Calls:  INTR                                                      *)
(*                                                                     *)
(*   Remarks:                                                          *)
(*                                                                     *)
(*      The indicated portion of the screen is scrolled up or down.    *)
(*      If Nlines > 0, then the screen is scrolled up.  If Nlines < 0, *)
(*      the screen is scrolled down.  Setting Nlines to zero blanks    *)
(*      the entire region.                                             *)
(*                                                                     *)
(*                                                                     *)
(*---------------------------------------------------------------------*)

VAR
  Reg: Regpack;

BEGIN (* Scroll *)
  Reg.Cl := X1 - 1;
  Reg.Ch := Y1 - 1;

  Reg.Dl := X2 - 1;
  Reg.Dh := Y2 - 1;

  Reg.Bh := ( BG AND 7 ) SHL 4 + FG;

  IF Nlines >= 0 THEN
     Reg.Ax := $0600 OR Nlines
  ELSE
     Reg.Ax := $0700 OR ABS( Nlines );

  INTR( $10 , Reg );

END  (* Scroll *);
```

## *Listing 11*

```
(*---------------------------------------------------------------------*)
(*        MoveToXY --- Move to (X,Y) on screen using BIOS call         *)
(*---------------------------------------------------------------------*)

PROCEDURE MoveToXY( X: INTEGER; Y: INTEGER );

(*---------------------------------------------------------------------*)
(*                                                                     *)
(*     Procedure:  MoveToXY                                            *)
(*                                                                     *)
(*     Purpose:    Moves to specified (X,Y) position on screen         *)
(*                                                                     *)
(*     Calling Sequence:                                               *)
(*                                                                     *)
(*        MoveToXY( X: INTEGER; Y: INTEGER );                          *)
(*                                                                     *)
(*           (X,Y)  --- Where to move to                               *)
(*                                                                     *)
(*---------------------------------------------------------------------*)

BEGIN (* MoveToXY *)

INLINE(
    $B4/$02       {   MOV    Ah,2           ;BIOS position cursor function}
   /$B7/$00       {   MOV    Bh,0           ;Page 0}
   /$8A/$B6/>Y    {   MOV    Dh,[BP+>Y]     ;Y coordinate}
   /$FE/$CE       {   DEC    Dh             ;Drop by 1}
   /$8A/$96/>X    {   MOV    Dl,[BP+>X]     ;X coordinate}
   /$FE/$CA       {   DEC    Dl             ;Drop by 1}
   /$CD/$10       {   INT    $10            ;BIOS video interrupt}
      );

END   (* MoveToXY *);
```

*Listing 12*

```
(*$C-,U-,R-,V-,K-*)
PROGRAM TryMDos;

(*----------------------------------------------------------------------*)
(*             TryMDos --- Try multitaskers for DOS                      *)
(*----------------------------------------------------------------------*)
(*                                                                       *)
(* Author:    Philip R. Burns                                           *)
(*                                                                       *)
(* Date:      December, 1986                                            *)
(* Version:   1.2                                                       *)
(*                                                                       *)
(* Overview: This program demonstrates the routines contained in the    *)
(*           file PIBMDOS.PAS, which provide Turbo Pascal interfaces    *)
(*           to several popular multitasking programs for DOS:          *)
(*                                                                       *)
(*              - DesqView from QuarterDeck                             *)
(*              - TaskView from Sunny Hill Software                     *)
(*              - TopView from IBM                                      *)
(*              - DoubleDos from SoftLogic                              *)
(*                                                                       *)
(*           Because MicroSoft Windows will emulate TopView, these      *)
(*           routines are also useful for Windows.                      *)
(*                                                                       *)
(* Remarks:  This sample program simply writes lines of text to the     *)
(*           screen until a key is depressed.  Then, it loops waiting   *)
(*           for input to show the difference between giving up and    *)
(*           not giving up unused time to other partitions.             *)
(*                                                                       *)
(*           The include file PIBMDOS.PAS contains the actual           *)
(*           multitasker interface routines.  In addition, a general    *)
(*           purpose routine to write a string is included.  The        *)
(*           string is written directly to screen memory if no          *)
(*           multitasker is active, or to the virtual buffer if a       *)
(*           multitasker is active.  It can also write the string       *)
(*           using standard BIOS calls if desired.                      *)
(*                                                                       *)
(*           Global variables of importance:                            *)
(*                                                                       *)
(*              Virtual_Screen      --- address of real screen if no    *)
(*                                      multitasker running, else       *)
(*                                      address of virtual screen.      *)
(*              MultiTasker         --- which multitasker is active.    *)
(*              TimeSharingActive   --- TRUE if multitasker is active.  *)
(*              Wait_For_Retrace    --- Set this TRUE if your color     *)
(*                                      graphics card produces snow     *)
(*                                      when screen memory is accessed  *)
(*                                      (e.g., set TRUE for IBM's CGA   *)
(*                                      but FALSE for Zeniths and       *)
(*                                      Compaqs, among others).         *)
(*              Write_Screen_Memory --- TRUE to write directly to       *)
(*                                      screen memory -- either the     *)
(*                                      genuine screen memory or the    *)
(*                                      virtual buffer for a multi-     *)
(*                                      tasker.  FALSE to use the BIOS  *)
(*                                      instead.                        *)
(*                                                                       *)
(*----------------------------------------------------------------------*)

(*$I TRYMDOS.GLO  *)
(*$I EGAINST.PAS  *)
(*$I CURVID.PAS   *)
(*$I COLORSCA.PAS *)
(*$I GETSCREN.PAS *)
(*$I PIBMDOS.PAS  *)
(*$I WRITESXY.PAS *)
(*$I WRITECXY.PAS *)
(*$I READKBD.PAS  *)
(*$I SCROLL.PAS   *)
(*$I MOVETOXY.PAS *)

VAR
   LineCount : REAL;
   Ch        : CHAR;
   S         : STRING[8];
   Y         : INTEGER;

(*----------------------------------------------------------------------*)
(*         YesNo --- Prompt for and read Yes/No question/answer         *)
(*----------------------------------------------------------------------*)

FUNCTION YesNo( YesNo_Question : AnyStr ) : BOOLEAN;

(*----------------------------------------------------------------------*)
(*                                                                       *)
(*     Function:   YesNo                                                *)
(*                                                                       *)
(*     Purpose:    Prompt for and read Yes/No question/answer           *)
(*                                                                       *)
(*     Calling Sequence:                                                *)
(*                                                                       *)
(*        Yes_Answer := YesNo( YesNo_Question : AnyStr ) : BOOLEAN;     *)
(*                                                                       *)
(*           YesNo_Question --- Text of Yes/No question to be asked     *)
(*           Yes_Answer     --- Set TRUE if "Yes" answer given, else    *)
(*                              set FALSE.                              *)
(*                                                                       *)
(*     Calls:  GiveAwayTime                                             *)
(*                                                                       *)
(*----------------------------------------------------------------------*)

VAR
   Ans_Ch: CHAR;
   Ch    : CHAR;
   X     : INTEGER;
   Y     : INTEGER;

BEGIN (* YesNo *)

   WRITELN;

   X      := WhereX;
   Y      := WhereY;

   Ans_Ch := ' ';

   REPEAT

      GoToXY( X , Y );
```

```
      ClrEol;

      WRITE( YesNo_Question, ' (y/n)? ' );

      Read_Kbd( Ans_Ch );

      Ans_Ch := UpCase( Ans_Ch );

   UNTIL( Ans_Ch IN ['Y','N'] );

   IF Ans_Ch = 'Y' THEN
      BEGIN
         YesNo := TRUE;
         WRITE('Yes');
      END
   ELSE
      BEGIN
         YesNo := FALSE;
         WRITE('No');
      END;

END   (* YesNo *);

(*----------------------------------------------------------------------*)
(*     Flush_Keyboard --- Flush pending keystrokes in keyboard          *)
(*----------------------------------------------------------------------*)

PROCEDURE Flush_Keyboard;

VAR
   Ch: CHAR;

BEGIN (* Flush_Keyboard    *)

   WHILE( KeyPressed ) DO
      READ( Kbd, Ch );

END   (* Flush_Keyboard    *);

(*----------------------------------------------------------------------*)
(*     Clear_Screen  --- Clear screen using BIOS function               *)
(*----------------------------------------------------------------------*)

PROCEDURE Clear_Screen;

(*----------------------------------------------------------------------*)
(*                                                                      *)
(*     Remark --- The reason for using this is that the built-in       *)
(*                Turbo procedure ClrScr acts strangely with some       *)
(*                hardware configurations under some multitaskers.      *)
(*                This BIOS call appears to work in all cases.          *)
(*                                                                      *)
(*----------------------------------------------------------------------*)

BEGIN (* Clear_Screen *)

   Scroll( 1, 25, 1, 80, 0, White, Black);
   MoveToXY( 1 , 1 );

END   (* Clear_Screen *);

(*----------------------------------------------------------------------*)
(*     Initialize --- Initialize multitasker demonstration              *)
(*----------------------------------------------------------------------*)

PROCEDURE Initialize;

BEGIN (* Initialize *)
                                    (* Clear screen *)
   Clear_Screen;
                                    (* Allow direct memory writes  *)
   Write_Screen_Memory := TRUE;
                                    (* Don't wait for retrace if mono *)
                                    (* adapter or EGA installed.      *)

   Wait_For_Retrace := ( NOT ( EGA_Installed OR ( Current_Video_Mode = 7 ) ) );

                                    (* See which multitasker active *)

   TimeSharingActive := IsTimeSharingActive;

                                    (* Indicate which is active      *)
   CASE MultiTasker OF
                                    (* No multitasker -- ask if we should *)
                                    (* wait for retraces in direct screen *)
                                    (* writes, or use BIOS.               *)
      MultiTasker_None : BEGIN
                            WRITELN('No multitasker active.');
                            Wait_For_Retrace    := YesNo('Wait for retrace in direct screen writes?');
                            Write_Screen_Memory := NOT YesNo('Use BIOS instead of direct screen writes?');
                            WRITELN;
                         END;
```

```
      TopView          : WRITELN('TopView active.');
      DesqView         : WRITELN('DesqView active.');
      TaskView         : WRITELN('TaskView active.');
      DoubleDos        : WRITELN('DoubleDos active.');
      ELSE;
   END (* CASE *);

END   (* Initialize *);

(*----------------------------------------------------------------------*)
(*      Try_Screen_Writes --- Try writing strings to screen memory      *)
(*----------------------------------------------------------------------*)

PROCEDURE Try_Screen_Writes;

BEGIN (* Try_Screen_Writes *)
                                   (* Write lines to screen until a key *)
                                   (* is hit.                           *)

   WRITELN('First test is writing text to virtual buffer.');
   WRITELN('A continuous series of lines will be written until');
   WRITELN('a key is pressed.  You can switch to another partition');
   WRITELN('and see that no bleed though occurs.');
   WRITE  ('Hit a key to start test: ');

   Read_Kbd( Ch );

   Flush_Keyboard;

   LineCount := 0.0;
   Y         := WhereY;

   WHILE( NOT KeyPressed ) DO
      BEGIN
         LineCount := LineCount + 1.0;
         STR( LineCount:8:0 , S );
         WriteSXY( 'Line ' + S + ' is now being displayed on the screen.', 1, Y, White );
         Y         := Y + 1;
         IF ( Y > 25 ) THEN
            Y := 1;
      END;
                                   (* Flush keyboard buffer *)
   Flush_Keyboard;

END   (* Try_Screen_Writes *);

(*----------------------------------------------------------------------*)
(*      Try_GiveAway --- Try giving up time to other partitions         *)
(*----------------------------------------------------------------------*)

PROCEDURE Try_GiveAway;

BEGIN (* Try_GiveAway *)
```

# The UNIX File

by Ian F. Darwin

*This column discusses the UNIX operating system. If you have comments or questions about UNIX or this column, please write to Ian Darwin at Box 603, Station F, Toronto, Ontario, Canada M4Y 2L8. If you have UNIX mail access to the uucp network, mail "ihnp4!-darwin!ian." I can't always answer immediately, but I will get back to you; electronic mail gets answered first!*

This time out your wandering UNIX columnist considers a book on system internals with IP-style networking, looks at UNIX' effects on DOS, and shows a way to write device drivers without having system source code.

## XINU AND INTERNET— A BOOK PREVIEW

Doug Comer's excellent *Operating System Design: The XINU Approach* (reviewed in "The UNIX File" in *Micro/ Systems Journal*, November/December 1985) set a new standard in UNIX operating system books by providing the complete source code for a UNIX-like operating system for educational purposes. Now Comer has taken on the challenge of instructing us in how to add "internet" code to a UNIX-like system kernel. His new book is entitled *Operating System Design Vol. 2: Internetworking with XINU* and is published by the Computer Science Division of Prentice-Hall.

The Internet is the follow-on to the old ARPAnet. The best-known example of internetworking is the TCP/IP networking facility included in 4BSD and some other UNIX systems. Internetworking is based on the idea that no single network can be big enough to serve all purposes, so big networks should be constructed out of smaller ones. The best model of this is small, high-speed local area networks (LANs) that are interconnected with lower-speed, point-to-point links. Users on a "node" or computer on one LAN can talk to users on a node of the other LAN without having to worry about network topology.

Multiple networks can be joined together; at present, most of the university computer centers in North America have multiple local nets that are joined into one of several larger networks, and it has become possible to send electronic mail to colleagues across the continent in a matter of minutes. It is not yet trivial, because not all the problems of internetwork mail routing have been solved, and not all the solutions that have been found are applied universally. But it is possible.

Comer's new book presents version 7 of XINU, his UNIX-like system (the name XINU stands, somewhat recursively, for "XINU Is Not UNIX"). Version 7 of UNIX is regarded by some as the most important release, as it introduced many significant components of UNIX (the UNIX file system used in System V, the world's first f77 compiler, the current shell "sh," struct, refer, dbm, ... ). V7 UNIX was ported widely to microcomputers, and a VAX port of V7 from Bell Labs, called "32/V," was the basis of the first VAX Berkeley distributions. Comer is certainly aware of all this as he releases V7 XINU.

The text of the new book describes internetworking using the internet protocols first developed by DARPA for the ARPAnet. There's a remote file system and Unix-like shell. There is some detail on the ideas behind IP, and a simpler notion than sockets. Sockets are the Berkeley method of accessing the IP protocols. Several observers have criticized the socket methodology as being too complex. One alternative to sockets is Dennis Ritchie's "streams" facility, first publicized in the *Bell Labs Technical Journal* in October 1984 and now a standard part of System V, Release 3. Another simple alternative is presented by Comer in this book.

There will be a new source tape for the new book, including XINU V7 and the internetworking code. The source tapes are distributed by Prentice-Hall. Remember that XINU is not a full system; like earlier versions, this XINU is meant to be compiled on a VAX and downloaded to a small PDP-11. There is also a Microvax version 7 system, but the tape is not ready yet. Both versions have an (optional) Ethernet downloader; this should make downloading faster than it would be with a serial link.

If you want to learn about the internals of a UNIX-like system including a networking scheme that is interconnectable with 4BSD and TCP/IP systems, get this book.

## UNIX TO DOS

Ever since Microsoft bought Seattle Computer's SC-DOS and renamed it MS-DOS, DOS has been feeling the influence of UNIX's more comprehensive facilities. Later releases of DOS have successively incorporated UNIX-like features, most of which are well known to readers of this magazine. Many well-known UNIX programs have been cloned for DOS. These include a complete tools package, the *vi* editor, *uucp* and a port of Documentor's WorkBench. These products deserve a full review, which I hope to provide for *Micro/ Systems Journal* readers in upcoming issues.

The MKS toolkit, written by Mortice Kern Systems, is a series of UNIX-like tools for DOS. It includes almost a hundred of the standard UNIX tools, from *grep* and *tr* to the *vi* screen editor and the Korn shell.

Another version of the *vi* editor, called PC/VI, was written by Custom Software Systems. It advertises a complete UNIX implementation and a 140-plus-page user manual that "is so complete many veteran VI users have learned ... commands they never knew existed." PC/VI sells for $149. The FANSI-CONSOLE terminal driver is recommended but not required.

Vortex Technology has written *UULINK*, a full-function package that provides a variety of features allowing an MS-DOS user to access the *uucp* inter-computer network (which is otherwise available only to UNIX systems). The *UULINK* package features compatibility with UNIX *uucp* systems, comprehensive mail sending and receiving subsystems with capabilities for automated routing to other networks, and support for submitting and receiving USENET "netnews" articles.

One of the more comprehensive packages for UNIX is the documentor's workbench package (DWB). This product includes device-independant *troff* as well as *pic*, *eqn*, and *tble*. Independent software houses have created many device-specific output filters (postprocessors) for DWB, but few companies have done much work on the product itself. SoftQuad is one exception; it has rewritten the intermediate language, added kerning and comprehensive hyphenation-exception facilities, and cleaned up the code and the error messages. But it is not currently selling in the DOS market. Another company, Elan, has ported the full AT&T source for DWB, including *ditroff*, to the MS-DOS system. Your IBM PC compatible can now be used to prepare and proof the same files you work with on a UNIX system with *troff*. I haven't seen a copy of this product yet; if it is really shippable, it should be worth the $495 Elan is asking for it.

These add-ons will make MS-DOS a little more like UNIX. This trend will undoubtedly continue, as will the general influence of UNIX on most other operating systems. System designers are not, and should not be, ashamed to learn from the best.

## NO KERNEL SOURCE? ADD DEVICE DRIVERS ANYWAY

Most small UNIX systems come with a "binary license" for UNIX itself. This allows you to run the system, but not to examine or modify the system source code. This is hardly surprising, since AT&T charges tens of thousands of dollars to give out the source code. Some binary-only systems include, and others exclude, the "reconfiguration" files. These files, if present, allow you to add or change the device drivers present on the system (device drivers are to UNIX what BIOS routines are to CP/M and DOS). These files consist of an "object" or compiled form of the operating system kernel, and the device drivers, to be link-edited with ld(1) to make a new kernel. You use this reconfiguration procedure anytime you add (or find bugs in) the code that knows how to control your new I/O devices. The process is analogous to rebuilding CP/M or DOS with a new BIOS.

Many systems do not include the reconfiguration files or charge extra for them. Current AT&T binary systems do include them, so clearly this is "the way it's supposed to be." At first it sounds outrageous to charge extra for reconfiguration, but it may be a way of making sure that the innocent do not rush in where angels fear to tread. Writing device drivers, like writing BIOS code or interrupt handlers, is not for the faint of heart. At any rate, there's a way around it, at least if you have System III or V UNIX. That way is called "named pipes."

Named pipes, also called FIFOs, provide a simple method of interprocess communication between unrelated processes. One process opens the FIFO for output by name as if it were a file, and another process opens it by the same name for reading. Presto! One process' output is now readable by the other. They differ from ordinary pipes in that the processes do not have to be related and that they generally appear and behave as ordinary files in the file system.

Named pipes can be used to fool your application into thinking that you have a device driver for a parallel line printer or a graphics board, for example, when all you have is the hardware and a bit of ingenuity. This is, of course, most useful on expandable systems (S100, Multibus, VMEbus, and so on) for which a wide range of peripherals exist. The three basic steps are creating the FIFO, building the "daemon" program to read from the FIFO and write to the device, and arranging for the daemon to be run automatically. Once these steps have been completed, you can run your application and write to what looks like a normal UNIX device.

Creating a named pipe is simple. If you are the super user on the system, you can install it in the */dev* directory, and it will look just like a normal output device except for the lack of IOCTLs. Pronounced "eye-OC-tal," these are control requests for I/O. IOCTLs change the behavior of an I/O device in device-dependent ways. Most of the parameters of STTY, for example, are just TTY IOCTLs.

Most kinds of special files are created with a program called *mknod*. The System III/System V version of *mknod(1)* can also make named pipes. For example:

```
mknod /dev/mydevice p
```

makes a named pipe called mydevice in the "dev" directory. You can then write to it just as if there actually were a device. Until you read from the pipe, however, your writing program will hang, waiting for the data to be read. This is done by a "daemon" program that you also must supply.

The daemon program is pretending to be a device driver. Drivers are specific to the operating system, the output device, the overall hardware configuration, the particular device's address, and other factors. They are thus not universal programs. A daemon designed to emulate a driver will likewise be very specific. The overall design of the program is simply to read bytes from the FIFO or named pipe and write them to the physical I/O device.

The daemon program will open and read the named pipe using standard open and read routines; either the system calls or the stdio routines should work. The way in which you write the bytes to the device varies from one UNIX to another. Consult your manufacturer's documentation. Some UNIXes offer a *phys* system call; on others, you must map the I/O device or use special instructions such as PORTOUT. You will have to find this out from your vendor. I suggest that you encapsulate it in a routine with a name like *sendbyte* or *readbyte* to distinguish it from system-provided routines.

If you wanted both read and write access to the device, the code would be more complex; it should probably fork and have one copy read and the other write to avoid deadlock problems. The program, once compiled into binary-executable format, normally would be installed in a public place such as */etc/mydaemon* if you are the superuser or your own bin directory if not.

To arrange for this to be run automatically at all times, the superuser need only add a line to the */etc/rc* file or */etc/inittab* file as appropriate. The normal user similarly can add a line to the *.profile .login* file. Add the line:

```
/etc/mydaemon&
```

The & specifies that the daemon runs detached; this is necessary for the shell file to finish. To start your own copy from .profile, you would say:

```
$HOME/bin/mydaemon&
```

in your .profile.

Once the daemon task has been installed and started, you should be able to write to the special device just as though it were a real device driver. Only you and your system vendor will know for sure.

*Continued on page 72*

# The Micro/Systems Journal Bookshelf

### *Dr. Dobb's Toolbook of 68000 Programming*

This complete collection of practical programming tips and techniques for the 6800 family includes the best articles on 68000 programming ever published in Dr. Dobb's, along with new material. You'll find a set of development tools, routines, useful applications and examples. All programs are available on disk.

**68000 Toolbook**
**Item #040 — $29.95**
**68000 Toolbook with disk**
**Item #041 — $49.95**

Specify MS-DOS, CP/M 8-inch, Osborne, Macintosh, Amiga, Atari 520st.

### *68000 Cross-Assembler*

An executable version of the 68000 Cross-Assembler discussed in the book is also available, complete eithsource code and documentation. Requires CP/M 2.2 with 64K or MS-DOS with 128K.

**68000 Cross-Assembler**
**Item #042 — $25.00**

Specify 8" SS/SD, Osborne, MS-DOS.

### *Dr. Dobb's Toolbook of C*

This authoritative reference contains over 700 pages, including the best C articles for Dr. Dobb's Journal, along with new material. You'll find hundreds of pages of valuable C source code, including a complete compiler, an assembler, and text processing programs, also available on disk.

**Toolbook of C**
**Item #005 — $29.95**

### *Programmer's Guide to CP/M*

*Ed. by Sol Libes*

A valuable reference for anyone who works with CP/M-compatible software and CP/M-based systems, this book provides the sophisticated programmer or software designer with a comprehensive look at CP/M and its capabilities. You'll find in-depth coverage of: structure and format, interfacing to operating systems, utilities and enhancements, assemblers, applications programs, database systems, interpreters, encryption, sort programs andmore. The book also includes software reviews, utilities and enhancements, and a software directory.

**Programmer's Guide to CP/M**
**Item #112 — $12.95**

### *On Command: Writing a Unix-Like Shell for MS-DOS*

*by Allen Holub.*

This book and ready-to-use program demonstrate how to write a UNIX-like shell for MS-DOS. On Command includes a full description and enhanced version of Holub's popular UNIX-like shell, complete C source code, a thorough discussion of low-level DOS interfacing and significant examples of C programming at the system level. You'll learn techniques applicable to MS-DOS, as well as to most other programming environments. Supported features include: Read, aliases, history, and C-Shell-based shell scripts. The UNIX-like control flow includes: if/then/else; while; foreach; switch/case; break; continue. The Shell works on IBM PC's and compatibles. The working program and all source code are included on disk.

**On Command**
**Item #163 — $39.95**

### *\Util*

When used with the shell described in *On Command*, this collection of utility programs and subroutines provides you a fully functional Unix-like environment. Utilities include: at; cp; date; du; echo; grep; ls; mkdir; mv; p; pause; printevn; rm; rmdir; sub; and chmod. Complete source code and manual are included.

**\Util**
**Item #161 — $29.95**

## To Order:

Return this Order Form with your payment to M & T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063. *Or*, call **TOLL-FREE (800) 533-4372** Mon-Fri 8AM-5PM. In **CA call (800) 356-2002**

| Item # | Decription | Format | Price |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

**Subtotal**________

CA residents add applicable **sales tax**____%____

Add $2.25 per item for **shipping**________

**TOTAL**________

Name________

Address________

City________ State________ Zip________

☐ Check Enclosed.
**Make Payable to M & T Publishing Inc.**
Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card No.________ Exp.Date________

Signature________

32A

**Unix File**
*Continued from page 69*

That's all for this month. I welcome letters and electronic mail on these and other topics, especially suggestions for future columns. Cheers! §

*Ian Darwin is director of Research and Development for SoftQuad Inc., a company providing supported troff publishing software for UNIX. He and his wife live in a rural setting, north of Toronto.*

### VENDORS

**Custom Software Systems**
Box 678
Natick, MA 01760
(617) 653-2555

**Elan Computer Group**
260 Alta Vista Ave.
Los Altos, CA 94022
(415) 948-8193

**Mortice Kern Systems**
43 Bridgeport Rd. East
Waterloo, Ontario, Canada N2J 2J4
(519) 884-2251

**Prentice-Hall**
James Fegen, Jr.
Executive Editor
Computer Science & Engineering
Prentice-Hall, Inc.
Englewood Cliffs, NJ 07632
(201) 592-3122

**Vortex Technology**
(213) 390-3920

# The Public Domain Software Forum

by Charles H. Strom

I was recently asked by Sol Libes to write a column dealing with shareware and public-domain software. Never one to pass up an opportunity to espouse opinions on microcomputer hardware and software (solicited or not), I naturally agreed to appear in *Micro/Systems Journal* regularly. My activity on CompuServe and GEnie places me in the midst of today's microcomputer software scene. I hope readers will benefit from my experiences.

I intend to concentrate on two major areas—public-domain software and shareware. For better or worse, shareware is here to stay. I cannot agree with the nay-sayers who criticize all shareware based on their problems with the concept. I have seen dozens of high-caliber packages that put expensive commercial software to shame. In each column, I plan to highlight one of the "ten best" shareware efforts (don't count—there may be more than ten), as well as address the issue's theme: in this case, multitasking. I occasionally will cover commercial software, if the product is worthy of special consideration, either positive or negative.

My interest lies primarily in IBM compatible software, but I still have my trusty Z80 running CP/M and will not forsake that arena. I hope also to cajole my friends to author guest columns on CP/M, Atari, Macintosh, and so forth. Hmm, why are they all running for cover?

As I said, this issue's theme is multitasking. I racked my brains looking for noncommercial software and came up with two programs, MTS.ARC and HOTDOS.ARC. Both of these programs are available widely on various remote systems, GEnie and CIS.

## MTS.ARC

MTS.ARC de-ARCs into two files: HKE.COM and HKE.DOC. I cannot fathom the meaning of MTS (multitasking software?) or HKE, and, in fact, there is no copyright notice or any author's attribution. (If any reader is able to shed some light on this matter, please contact me.) HKE is particular in that it will execute only under DOS 2.00-3.10. My attempt with DOS 3.20 failed. MTS allows partitioning of memory into two tasks that run simultaneously using time slicing. Options include specification of sizes of the partitions, support for dual displays (CGA and mono), CGA graphics support, and so on. Interrupt-driven applications can be run only in partition 1 and BASIC programs in partition 2. Unfortunately, RS-232 communications are not supported, so modem applications will not run. MTS is an interesting program, however, that might be a useful adjunct to your toolkit.

## HOTDOS.ARC

HOTDOS consists of a documentation file and a .COM file that is a fully functional program. It is shareware, and the author requests a $20 donation. Similar in concept to MTS, HOTDOS does not offer true multitasking. It allows partitioning into two memory segments, simultaneous use of two video displays, and CGA graphics support. HOTDOS is more flexible than MTS as far as to which DOS versions it will accept and, unlike what you can do with MTS, you can exit from it without rebooting. My evaluation is that HOTDOS is somewhat more versatile and easier to use that MTS, at the expense of not offering true time slicing. It can, however, make virtually any program into a TSR (terminate-and-stay-resident) program, which can in turn be removed without rebooting.

## TOP TEN: #1

As I promised, I will present a selection for entry to the shareware "hall of fame." This issue's award goes to Vernon T. Buerg for his LIST.COM program. As of press time, the current incarnation is LIST60J, available on GEnie, CIS, and a myriad of private remote systems. Simply stated, LIST is a replacement for the TYPE command of MS-DOS. The program presents a paged display at high scrolling speed, but that is just the beginning. Packed into less than 7 b of program are features such as user-defined colors, support for piped and redirected files, display of line-drawing characters, tab expansion, control-character filtering, hexadecimal display, and wild-card support. LIST will even switch optionally to 43-line mode when running an EGA-compatible display board—a handy feature. Another significant LIST feature is pattern searching within the displayed file. I find that with LIST, I have not used TYPE in months. Why bother when Vern's program can replace it and do much more at higher speed?

This is one of the few programs that takes no time to master and yet quickly becomes indispensable. Vern suggests a contribution of $15, a small price indeed. Vern Buerg's skill at assembly language will undoubtedly come up again in future columns that deal with ARChive files and related utilities.

## CONCLUSION

There you have it for this issue. I welcome comments, suggestions for programs to discuss (if you are a public-domain software or shareware author, feel free to send me a disk of your pride and joy), and, of course, criticisms, too.§

## NEW PC/BLUE RELEASES

The following are recent releases in the PC/Blue library of public-domain MS-DOS software.

**Volumes 251-252**

PAYROLL USA System v1.0 (up to 300 employees)

**Volume 253**

PC Accounting Systems; PC-GL v2.9, PC-

AR v3.6, and PC-PR v1.3

**Volumes 254-255**
PC-Write v2.7

**Volume 256**
Laboratory Benchmark Series Release 4 from *PC Magazine* (PC Labs)

**Volume 257**
DBS-KAT v2.0—super-capacity disk cataloger

**Volumes 258—259**
FANSI-Console v2.0

**Volumes 260-262**
Turbo Pascal Programming

**Volume 263**
LQ v2.24—near-letter-quality for dot-matrix printers

**Volumes 264-265**
Kwikstat—statistical analysis

**Volumes 266-267**
DROEGE: Design Robot for Origination of Exacting Graphic Engineering v1.81—designed for printed-circuit layout

**Volume 268**
COLOR Utility—controls foreground/background colors
CRAYZ15—Turbo Pascal utilities
ML22—Mailing Label System v2.2
PFROI209—Portfolio Management Program v2.09

**Volume 269**
ProComm v2.42—modem communications

**Volume 270**
V20/80386 Disassembler
LaPlace Equation in 2 Dimensions
CONTOUR, by James R. Van Zandt

**Volume 271**
Horse Race Handicapping (R:Base 5000)
Mailing List Management (R:Base 5000)
Morse Code, by Cass H. Lewart
Space War v1.71, by Walter W. Elbers

**Volume 272**
NYWord v2.1—word processor

**Volume 273**
QModem v2.3—modem communications
ScreenWright Professional v2.0

**Volumes 274-276**
DREAM: Data Retrieval, Entry, & Management

The disks are available from the New York Amateur Computer Club, Inc., Box 106, Church Street Station, New York, NY 10008, for $7 per volume (includes media, and shipping and handling); foreign orders, add $2 per volume. A printed catalog is also available for $5 (U.S., Canada and Mexico; $7 foreign).

*Charles Strom's first exposure to computers was in high school, running a then-state-of-the-art IBM 1620 with 20K of core and read/punch cards. He built an IMSAI 8080-based microcomputer in 1977 and was bitten by the micro bug. Although a chemist by training, Charles is a computerist by choice. Current interests include scientific/technical word processing, communications software, and desktop publishing. Charles is the author of numerous review articles and a sysop on the GEnie national time-sharing service.*

# New Products

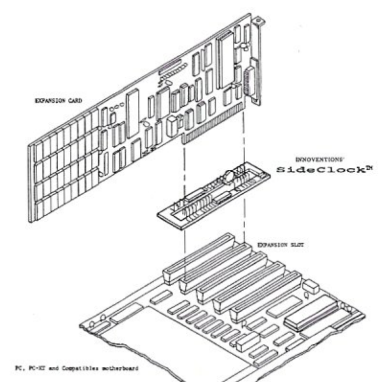
*When contacting vendors, please mention that you read about their products in* Micro/Systems Journal.

## PC COMPATIBLE HARDWARE

### CLOCK/CALENDAR WITHOUT A CARD

Innoventions' SideClock provides clock/calendar functions without requiring a plug-in card or piggy-backing onto other chips. Rather, it uses a special set of contacts that snaps into any bus slot that already carries a plug-in card. It includes a 5-year lithium battery and software for setting up the clock/calendar. The retail price is $59, and the unit comes with a 30-day money-back offer and 2-year limited warranty.

The unit is available from Aristo Computers, 16811 El Camino Real, Ste. 213, Houston, TX 77058; (713) 480-6288.




### LOW-COST LOCAL AREA NETWORK

Simplenet is a low-cost local area network providing medium performance to small- and mid-size organizations. It costs $695 for a 4-user base system. Four more users can be added for $99 each; the same can be done six more times, up to a maximum of 32 users. The LAN operates at 110 kilobaud and is upgradeable to 4 megabaud. The longest cable length available is 1.2Km. It is software compatible with the IBM PC network. Simplenet plugs into asynchronous communication ports. For information contact BC Associates, Fullerton, CA; (714) 526-5151.

### 160- AND 320-MEGABYTE INTERNAL AT HARD DISK SYSTEMS

Storage Dimension has introduced 160- and 320-megabyte internal hard disk systems for the IBM PC/AT and compatibles. The two versions of the AT160F have an average 28-millisecond access time and are compatible with the Western Digital controller card. The package includes a Maxtor XT2190 28-ms full-height drive, a custom ROM BIOS chip-set from Award Software (to be installed on the AT motherboard), and the SpeedStor hard disk integration program. The software allows the disk to be configured as a single logical unit. The drives are preformatted for DOS.

The units are warranted for one year. The AT160F-2, the 320Mb system, retails for $8990. The AT160F-1, the 160Mb system, retails for $4495. A Western Digital floppy/hard disk controller is an additional $200. For more information contact Storage Dimensions, 981 University Ave., Los Gatos, CA 95030; (408) 395-2688.

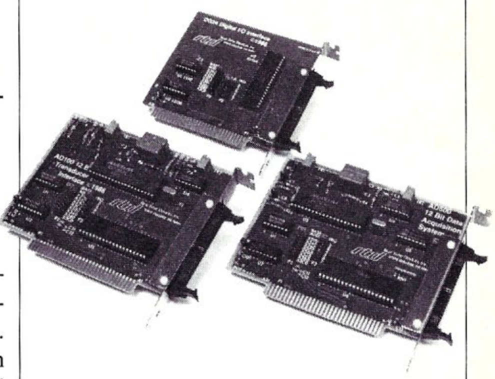
### LOW COST A/D AND D/A BOARDS

Real Time Devices has announced three new analog-input and digital-I/O boards. The AD500 has 8-12 bit analog channels and a programmable amplifier (gains of 1, 10, and 100) and will perform 7.7 conversions per second. Software-selectable analog input ranges are −5 to +5 volts, −500 to +500 millivolts, and −50 to +50 millivolts. Inputs are single-ended and protected to 35 volts. The AD500 also has seven TTL I/O lines. The unit is $239.

The AD100 is a single-channel version of the AD500 and has three additional digital I/O lines. The price is $149.

The DG24 is a digital I/O board using the 8255 peripheral interface chip with 24 TTL-compatible I/O lines. It supports handshaking, and interrupts are jumper selectable. The price is $95.

Connections to the boards are made via a 40-pin connector. A terminal strip connector board and cable assembly (model XB40) is $49. All the products include documentation and a programming example disk. For more information contact Real Time Devices, Inc., 1930 Park Forest Ave., Box 906, State College, PA 16804; (814) 234-8087.



## OTHER HARDWARE PRODUCTS

### MULTIUSER SYSTEM IN SMALL CUBE

L/F Technologies has released CUBIX2, a System V computer measuring only 11 × 11 × 11-inches and supporting eight users, 8Mb of RAM, 220Mb of hard disk storage, a 1.2Mb floppy drive, an 80287 math coprocessor, and 60Mb tape drive. Further, it is AT&T 6300-Plus software compatible.

A floor model, the CUBIX3, supports 16 users and contains 660Mb of hard disk storage and two 1.2Mb floppy disks and an internal, uninterruptible power supply, as well as tape drive and math coprocessor.

A CUBIX2 base system is $4,795, and the CUBIX3 base system is $5,995. For information contact L/F Technologies, 2800 Lockheed Way, Carson City, NV 89701; (702) 883-7611.

### DISK COPIER SYSTEM

The Recortec FDC-502 copies PC disks in 55 seconds. You simply insert the source and destination disks and press the COPY button. A ROM-based processor then automatically executes the copying function. The unit is $995 from Recortec, Inc., 275 Santa Ana Ct., Sunnyvale, CA 94086; (408) 737-8441.

# The LANScape

by B.J. Hall and Michael Cherry

## Part I—An Introduction to Engineering a LAN

*This column will be a regular feature in* M/SJ. *It will feature tips and techniques for system integrators installing local area network systems. It will discuss typical problems and their solutions. Readers' suggestions, comments, and questions are encouraged. They can be addressed either to the editor or to the authors directly (see end of article).*

An introduction to the technical side of local area networks and data communications for the microcomputer must include definitions of some of the ground rules, guidelines, and terminology. In order to understand these basic elements, let's look at the components of LANs. These must include not only hardware and software, but the purpose in networking and goals for the local network. Although it is by no means complete, this article is a foundation for future discussions on all the complex factors involved with networks and networking.

### WHAT ARE LANS?

A local area network (LAN), using microcomputers, is a system designed to tie together personal computers of different types and dissimilar operating systems to provide multiuser, multitasking performance capabilities to the networked PCs.

Local area networks are for sharing. LANs provide a communications path between computers that allows multiple users to access shared resources. Functions include sharing common databases stored on high-capacity disk subsystems, and the ability to route print requirements from any work station to a group of shared printers. The latter ability allows users a choice of printers that are attached to the network.

In a LAN, processing is distributed to the individual microcomputer work stations, rather than using a minicomputer centralized multiprocessor system. There are immediate performance benefits when the processing is distributed in this manner. By utilizing the memory and CPU of each work station, with a central server controlling the movement of information between these work stations, a local network has many times the processing power of a single microcomputer. By relieving the file server of memory and processing responsibility, the LAN allows the work station to be configured to the individual requirements of the users' applications without requiring the reconfiguration of the entire network. This makes the file server available for maximized utilization as a file-transfer and centralized-storage processor, rather than an application processor.

### PLANNING—THE KEY TO SUCCESSFUL INSTALLATION

LANs must provide compatibility among dissimilar PCs and operating systems, simultaneous data access and manipulation, and compatibility with user-purchased and user-developed software packages. These can be difficult tasks for the designer, but they are necessary capabilities in this generation of microcomputing.

Bringing the PC out of the single-user, single-tasking place it has occupied in the workplace raises concerns that must be understood and overcome. In our experience of assisting many companies in the design and implementation of local area networks, we have developed a network design analysis questionnaire that asks most of the important questions about hardware and software. We see it as a guide to a successful network installation. That questionnaire follows in its basic form, but a system engineer should adapt the questions to the specific application being designed. An important question to ask is what the user expects from the completed network. This combination wish list and needs analysis can provide insight into project-planning requirements. As a note of caution, we have found that many companies have unreal expectations about network cost, performance, and training, usually in that order. It is necessary that the system engineer clarify the proposed system capabilities and uncover the hidden costs of the system.

### A NETWORK DESIGN ANALYSIS TOOL

**Hardware**

1. What type of PCs are going to be attached to the network?
2. How many PCs will be:
   a. attached to network immediately?
   b. attached to network at maximum?
3. What types and how many peripherals are going to be on the LAN?
   a. printers, plotters, optical scanners?
   b. hard disk subsystems (local work station or server)?
   c. ¼-inch and ½-inch tape systems?
   d. other?
4. Has the equipment already been purchased? If so, inventory and document.
5. Where are the PCs and devices going to be located?
   a. same of different buildings?
   b. on different floors?
   c. across streets?
6. What are the maximum expected cable distances?
   a. between two PCs?
   b. total network?
7. For cabling purposes, is a building blueprint available? Site diagrams can define any design limitations affecting the choice of topology.
8. Are local building and fire regulations known? This can greatly affect the cost of cabling the network.

**Software**

1. What type of application software will be used over the local area network?
   a. single-user/multiuser
   b. off-the-shelf software packages
   c. custom-developed programs
   d. programming languages and

utilities

2. What do you expect and want to do with the software? This is the user's wish list and needs analysis.
3. Have you estimated the training requirements for both end user and system administrators? This training should include training on both the network software and application programs.

**General**

1. What are the security requirements? This should include a detailed layout of planned user directories and the access needed by each employee.
2. Is it possible to define additional communications planned for the system? Do remote users need dial-in modem access?
3. What is the time frame for installation of the LAN? Take the time to do the planning, design, and implementation correctly.

## PHYSICAL CONSIDERATIONS AND STANDARDS

Local network communications standards at the physical or hardware level address the following areas:

Media—primarily coaxial or twisted-pair cable
Modulation—either broadband or baseband
Access Control—either CSMA or token ring
Topology—star or linear bus cabling scheme

## SOFTWARE CONCERNS

When PCs worked alone, minding their own business, confrontation over who had access to data files did not occur. The potential in a network for file corruption should cause an application designer to become knowledgeable about the file- and record-locking and -unlocking function calls supported by the network operating system software. With the developing standards of IBM's NetBIOS, DOS 3.1, and Novell's NetWare, it is now possible for a developer to write applications that can run correctly on a number of different LAN types.

## DISK SUBSYSTEMS

Virtually all networks offer common high-capacity, hard-disk storage, in the 30-megabyte to 3-gigabyte range. It doesn't take long before the necesssary information outgrows the storage area originally planned for the specific application. Properly designed mass storage can offer not only increased storage space but the opportunity for many users to have simultaneous access to the data. The benefits of mass storage can be realized quickly if the redundancy of multiple copies of programs is eliminated. It makes more sense to let ten users share one program than to buy ten copies of the program. In a local network, considerable cost savings are possible through purchase of network versions or site licenses of the planned software.

## NETWORK OPERATING SYSTEM TYPES

There are two types of software control methods used in micro-based network operating systems: the disk server and the file server.

### The Disk Server Method

**VENDORS**
**Microsoft**—MS-Networks, DOS 3.1
**IBM**—PC LAN Program, NetBIOS
**Vianetix**—Vianet
**Waterloo Microsystems**—PORT
**AT&T**—StarLAN
**3COM**—3+, EtherShare

These are popularly called *peer-to-peer* networks, for their ability to let users link work stations directly to other users' work stations and share printers, modems, and hard disks. On a disk server, the hard disk is partitioned and multiple volumes are created. This permits access to data at the volume level. Each volume is designated by a DOS or network prompt—for example, G>. Once a volume is opened, the individual work station's operating system performs the task of opening the files for access to programs and data. This type of network software offers limited performance due to software overhead inherent in adapting single-user MS-DOS to networking. This is done most commonly by utilizing NetBIOS, a specification standardized by IBM. The function of NetBIOS is to provide the basis for all program control of the network.

### The File Server Method

**VENDOR**
**Novell**—Advanced NetWare

This method uses a proprietary multiuser, multitasking operating system and disk format. This type of network software offers extremely high performance. In file-server networks, the work-station microcomputer executes a shell program that controls access to the network hardware as well as the translation of DOS commands to the network operating system. Extensive security is available. Published specifications of programming tools, at the operating-system level, allow developers to integrate networking capabilities into their applications. Future compatibility with IBM local area network standards is assured by the NetWare support of the IBM network hardware standard, NetBIOS.

## SUMMARY

Local area networks have appeared in response to problems that have developed in the area of data communications in which intelligent devices must share information. Like the computers that attach to them, LANs have different characteristics, specifications, and applications. The selection of a network requires considerable planning and design. To determine a network's superiority, it is necessary to pay careful attention to four key factors in the architecture of the net. These factors are:

1. reliability
2. ease of maintenance
3. ease of expansion
4. performance

To ease the transition from multiple single-user computers to a shared, networked environment, we recommend installation of a pilot, or test, local network. A pilot network is composed of two to four work stations connected to a microcomputer file server that will allow an organization the capability for learning about a LAN. The benefits of a pilot network are:

1. testing the capability of a new technology before implementation of large-scale systems
2. developing uses for high-capacity mass storage
3. finding limitations and network conflicts in current application software
4. planning for next-generation software and systems application development. §

*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 Software Directory

*When contacting software publishers, please mention that you read about their product in* Micro/Systems Journal.

---

**Program Name:** Poor Man's Network
**Requirements:** Two CP/M systems with RS-232 ports
**Description:** Allows two computer systems to read and write files located on local and remote systems. Data from one system can be printed on the other. Works with CP/M 2.2, Micro Methods RP/M, and Echelon's Z-System (ZCPR3 and ZRDOS).
**Price:** $69
**Publisher:** Anderson Techno-Products, Inc., 947 Richmond Rd., Ottawa, Ontario, Canada K2B 6R1; (613) 722-0690.

---

**Program Name:** UNDER-C Library
**Requirements:** PC using either Microsoft C v4.0, Lattice C v3.10 or v2.15, Computer Innovations C-86, or Instant-C
**Description:** A C library with 122 functions, including support for BIOS and DOS calls. Functions are included for manipulating character strings and building screens. Also included are the following utilities: MAKE; CPRINT, a general-purpose printing program; and PFS2TXT, which translates pfs:Write and IBM Writing Assistant files to text format. Complete source code is provided, with no royalties and no copy protection.
**Price:** $95 (includes postage and handling.)
**Publisher:** Quayle Research, Inc., 6548 Edgerton Rd., N. Royalton, OH 44133; (216) 237-4395.

---

**Program Name:** XFER
**Requirements:** PC-DOS or MS-DOS
**Description:** A transfer-function analysis and synthesis program using short circuit transfer impedance functions around an op-amp to compute circuit element values and circuit configurations that will synthesize a desired transfer function. Conversely, given a circuit configuration and element values, will compute a circuit's transfer function. Multiple stages of short circuit transfer impedance functions using forward and feedback elements in op-amp configurations enable synthesis and analysis of most transfer functions having real roots.

Can compute magnitude and phase response and perform sensitivity and Monte Carlo analysis. Circuit configurations can be viewed via a circuit and transfer function editor. Can be used with plotting, transient, and time-domain analysis programs.
**Price:** $72.95
**Publisher:** BV Engineering, 2200 Business Way, Ste. 207, Riverside, CA 92501; (714) 781-0252.

---

**Program Name:** Bastoc
**Requirements:** PC compatible, AT&T 3B2, PC/3B1, Radio Shack 6000/16B or one of several other UNIX and UNIX-like systems
**Description:** A BASIC-to-C translator and BASIC/BASIC-A compiler for Microsoft C v4.0 and Lattice C v3.10. Versions are also available for UNIX and XENIX. Translates Microsoft BASIC, Digital Research CBASIC, and TRS-XENIX MBASIC.
**Price:** $495
**Publisher:** JMI Software Consultants, Inc., 904 Sheble Ln., Box 481, Spring House, PA 19477; (215) 628-0846.

---

**Program Name:** R-Doc/X
**Requirements:** MS-DOS or PC-DOS and CP/M
**Description:** Converts text files from one word-processor format to another, including print enhancements and format-control codes. Multiple files may be translated. Translates between following word processors: WordPerfect, Displaywrite-3, WordStar, Multimate, Microsoft Word, Spellbinder, Leading Edge, Volkswriter Deluxe, PC-Write and standard ASCII.
**Price:** $149
**Publisher:** Advanced Computer Innovations, 1227 Goler House, Rochester, NY 14620; (716) 454-3188.

---

**Program Name:** LANlink
**Requirements:** PC or PC compatible
**Description:** Lets users share files and peripherals among server and satellite PCs via RS-232 connections. Up to 16 PCs can be operated from one server. Includes multitasking at the server PC. Is compatible with TSL's shared-processor system, MultiLink Advanced. Both support record lock and file lock and are compatible with multiuser PC-DOS applications. When the two systems are combined, users can attach dumb terminals to either server or satellite work station.

LANlink features include remote modem access, print spooling, volume/subdirectory security, server status screen, automatic dial-up, and support of DOS 3.2.
**Price:** Starter kit (one server, one satellite, and cabling), is $495; additional satellite $99; additional server $400
**Publisher:** The Software Link, Inc., 8601 Dunwoody Place, N.E., Ste. 632, Atlanta, GA 30338; (404) 998-0700.

---

**Program Name:** SSP/PC Library
**Requirements:** PC or PC compatible with 8087 or 80287 math coprocessor.
**Description:** A library of 145 mathematical routines for scientific, engineering, and statistical applications. Routines can be called from Lattice C, IBM, Lahey, Microsoft, Prospero, and Ryan-McFarland FORTRAN, IBM and Microsoft BASIC, Microsoft QuickBASIC, Microsoft Pascal, Lattice, Computer Innovations, and Microsoft C. User guide contains sample programs for every subroutine. A 30-day money-back guarantee and support are included.
**Price:** $350
**Publisher:** Lattice, Inc., Box 30772, Glen Ellyn, IL 60138; (312) 858-7950.

---

# *Taming MS-DOS*

*by Thom Hogan*

This book is an advanced user's guide to enhancing the MS-DOS environment. Picking up where you DOS manual leaves off, **Taming MS-DOS** takes you beyond the basics. Every chapter is full of tips and techniques that extend the power of DOS so you can work more efficiently.

- You'll learn to customize CONFIG.SYS to maximize the performance of your system and how to use ANSI.SYS to tailor your system prompt and monitor attributes to fit your needs.
- Learn to maximize your batch files with routines using redirection, filters and pipes. You'll find routines that prevent accidental reformatting of your hard disk, redefine function keys and locate files within subdirectories.
- You'll also find batch files that implement a DOS help system, including help text files, a menu system that interprets keyboard input, and a routine for quick redefinition of function keys.
- Taming MS-DOS includes nearly 50 ready-to-use programs that increase DOS's functionality. Now you can easily rename directories and disk volumes, change file attributes, check available RAM and disk memory, display a memory-resident clock, and assign DOS commands to ALT keys.
- Quick reference charts provide easy access to batch command syntax, CONFIG.SYS syntax and ANSI.SYS strings.

The programs, including batch files and DOS enhancements, are available on disk along with full source code.

**Taming DOS** **Item #060 $19.95**
**Taming MS-DOS with disk** **Item #061 $34.95**

**FULL SOURCE CODE INCLUDED!**

# *Nr: A Nroff-like Text Processor for MS-DOS*

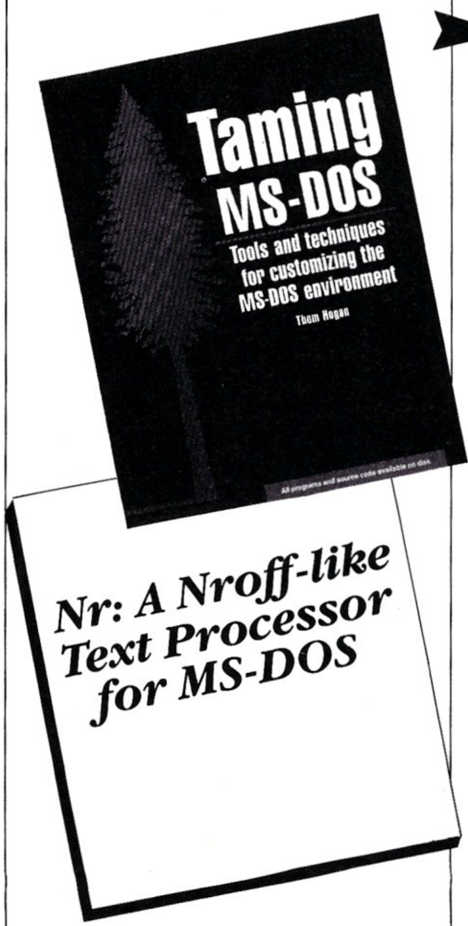
**Nr** is a text processor that is written in C, and is compatible with Unix Nroff. It includes complete implementation of the -MS macro package and an in-depth description of how -MS works.

**Nr** will help you do hyphenation and simple proportional spacing. It supports automatic Table of Contents and Index, automatic footnotes and endnotes, italics, boldface, overstricking and understricking, and left and right margin adjustment. **Nr** also contains:

- extensive macro capability
- number registers in various formats including roman numerals and arabic, spelled out and in outline form.
- strings
- diversions and diversion traps
- input- and output-line traps

**Nr** comes configured for any Diablo-compatible printer, and Hewlett Packard's Thinkjet and Laserjet. It is easily configurable for most other printers and comes with full source code so that you can make it work with your system! Documentation is included. For PC compatibles.

**Nr** **Item #165 $29.95**




**To Order:** Return this Order Form with your payment to M & T Publishing Inc., 501 Galveston Dr., Redwood City, CA 94063. *Or*, call **TOLL-FREE (800) 533-4372** Mon-Fri 8AM-5PM. In **CA call (800) 356-2002**

| **Item #** | **Description** | **Price** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

**Subtotal**__________

CA residents add applicable **sales tax**____ %____

Add $2.25 per item for **shipping**__________

**TOTAL**__________

Name__________

Address__________

City__________ State__________ Zip__________

☐ Check Enclosed.

**Make Payable to M & T Publishing Inc.**

Charge my ☐ VISA ☐ M/C ☐ Amer. Exp.

Card No.__________ Exp.Date__________

Signature__________

32B

# Classifieds

**Micro/Systems Journal** *accepts Classified Ads. The charge is $6/line (3 lines minimum); 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.*

**MS-DOS Shareware**
$3.00 each disk postpaid. 000 catalog on disk, 001 expert system, 007 X-lisp, 014 B&W games, 015 color games. **SMUG**, 39 Hanover St., Asheville, NC 28806-4158.

---

**FOR SALE:** COMPUPRO dual 1.2 Mb floppy disk subsys, DISK 1, DT-8 Qume's, cables, encl, CP/M-80. Like new, orig $1750, ask $900/offer. IDS, IP-225 dot matrix printer w/graphics, parallel port, w/3P + S board, 160 cps, $250/offer. Bob Miller (408) 371-2677.

---

**"DISK SERVICE MANUAL" ($20)**
"Printer/Plotter Manual" ($15); "Super Re-Inking Method" ($7); "Computer Phreaking" ($15); "Absolute Computer Security" ($15). 40+ computer & electroics manuals & software, Catalog: $1. By former NMSU CS Professor. CONSUMERTRONICS, 2011 Crescent, Alamogordo, NM 88310.

---

**COMPUTER PROGRAMMERS**
**FORTRAN To DOS Interface:** Over 85 routines that provide complete control over the DOS and ROM-BIOS services. Command line processing support is provided. $59.95. **C To DOS Interface:** Over 70 routines that provide complete control over the DOS and ROM-BIOS services. $59.95. **The C Library:** Over 140 programming tools. $29.95. **Turbo Pascal Programmer's Library:** Over 170 routines that provide DOS and ROM-BIOS control, file, string, and array manipulation. Math, I/O, and menu routines. Expansion of Borland's Turbo Toolbox. $59.95.
Add $4.00 shipping for each package.
**Kris Jamsa Software, Inc., P.O. Box 26031, Las Vegas, NV 89128.**

---

**LOMAS S100PC FOR SALE:** Home System, 8086, 256K, 2-5″Flpy, ColorMagic32k, Amdek600, Keytronic KB5151, EspritIIIColor Term, $2500 + UPS, o b o (203) 275-3154.

---

**COMPUTER PROFESSIONALS**
with good writing skills to evaluate & review latest computer software products for national sydication. Part-time, contract stringer. Must have access to DEC, IBM PC, MAC or compatible clones. Send resume: PSPA, 830 Bay Ave., Dept G, Capitola, CA 95010.

---

**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.

---

**Run CP/M Programs on IBM PC With EZ80!**
No extra hardware required! Only $39.95.
HG Software Co.,P.O. Box 10151, Austin, TX 78766-0151; (512) 836-3188.

---

**CP/M-80 IMITATES PC-DOS!** Rename drives, autofind overlays, run BASIC-like batch jobs, much more. LOGIC ASSOCIATES, 1433 Thom, Chicago, IL 60660; (312) 274-0531.

---

**ENGINEERING SOFTWARE**
Circuit Design and Analysis Programs. FREE CATALOG and TUTORIAL GUIDE. BV Engineering, 2200 Business Way, #207, Rverside, CA 92501; (714) 781-0252.

---

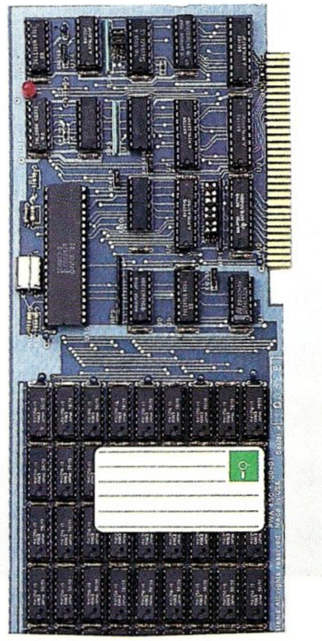
# Advertiser Index

# T ick the better fit...

Our 5th Year
Mention this ad w
and get your choice o
(replaces 8088) or $20 off
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