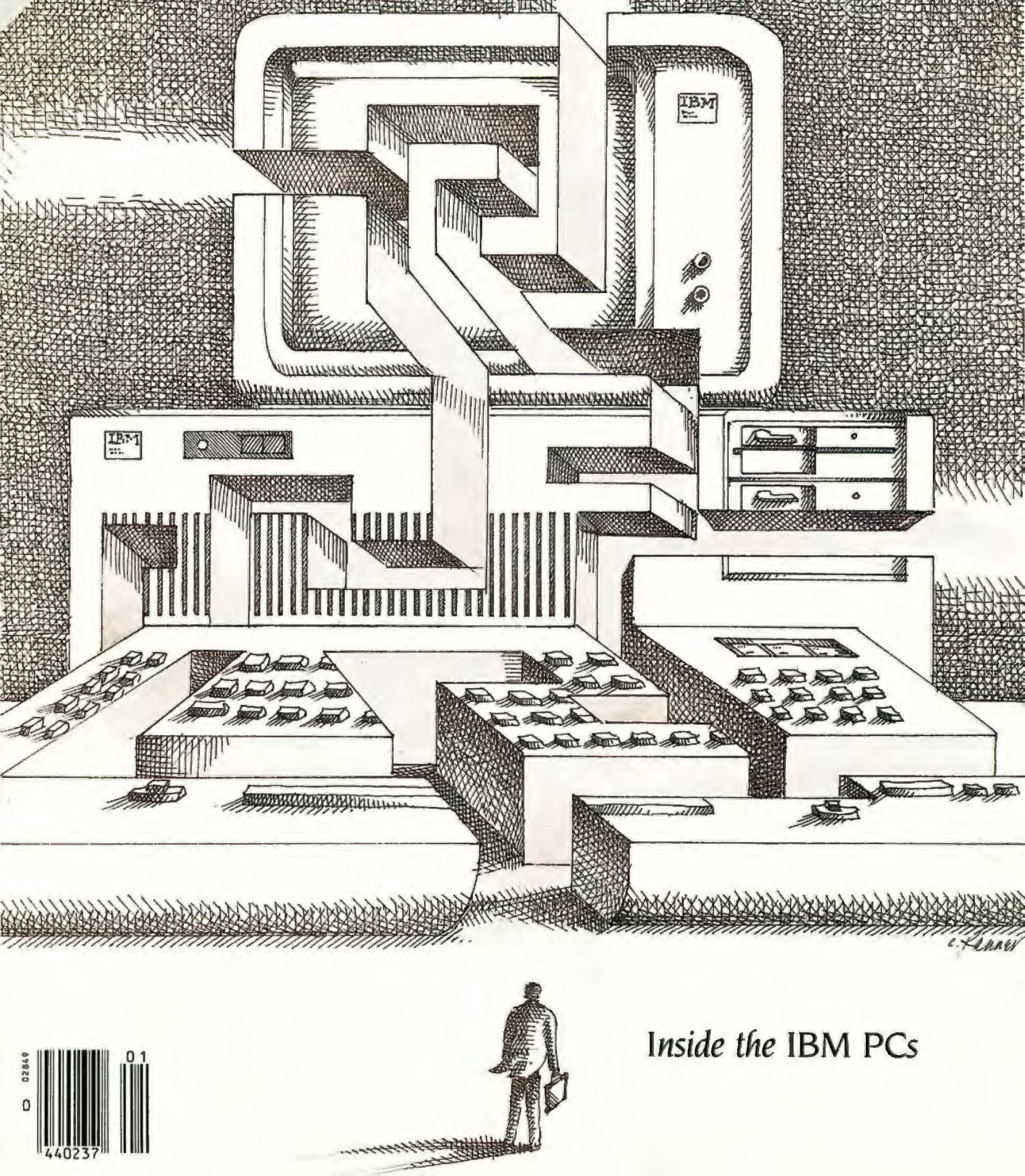
Special IBM Issue

BYTE

THE SMALL SYSTEMS JOURNAL

$3.95
$4.95 in Canada
A McGraw-Hill Publication
0360-5280
Display Until December 15



Inside the IBM PCs

60-DAY MONEY-BACK GUARANTEE

Introducing Borland's New Turbo GameWorks™ $69.95

THE GAMES YOU CAN PLAY, REPLAY, REVISE AND REWRITE BUT CANNOT RESIST. (Turbo Pascal Source Code included!)

We give you the source code, the manual, the diskettes, the 60-day guarantee and the competitive edge. Let the games begin. Chess. Bridge. Go-Moku.

State-of-the art games that let you be player, referee, and rules committee—because you have the Turbo Pascal source code. Which means that you can play a game or create a game, any time and any way you want.

Borland's new Turbo GameWorks lets you combine gamesmanship with craftsmanship. Discover the secret techniques and moves used by the Old Masters. Learn exactly how state-of-the-art computer games are made—so you can go off and make your own. Since you have the source code, you can always change the game. Or rig the game, if no one's looking.

Pure Magic. That's Turbo GameWorks. And part of the "sourcery"—Turbo GameWorks is only $69.95. When combined with our new Turbo Editor Toolbox in the Turbo *New* Pack, it's only $47.50.

SHORT CUTS, SECRETS AND STRATEGIES. The Turbo GameWorks manual takes you step-by-step through all the games. How to play them. How to modify them. How to use the power of Turbo Pascal to write new games.

You'll learn general problem analysis, how to identify all possible moves, "rule of thumb" strategies, procedures for testing strategies, and ways to rate options. You'll also be introduced to "top down" program design, the development of basic algorithms, the use of constants and data structures and ways to design short cuts with incremental updating.

On top of all that, you'll have a lot of fun (if you want to).

So go to play (and work) with Borland's new Turbo GameWorks. It's unique. It's fascinating. And it's brand-new. Be first with the latest and greatest—order your Turbo GameWorks today.

BORLAND'S TURBO GAMEWORKS AND SOME OF ITS MASTER PIECES

Chess, the ultimate strategic game. A game so old that no one knows its exact origins. Turbo GameWorks lets you play chess at six different levels from the beginner to the sophisticated user. And you have many ways of playing with your Turbo GameWorks. Let the computer solve checkmate problems. Set the time limit for each game. And there's more.

Decide whether you or the computer "goes first." Trade places with the computer at any point in the game. It's all possible with GameWorks.

Go-Moku, also known as "Five-in-Line," is a very old Japanese game played on a board of squares. The first player to get five game pieces in a row—either horizontally, vertically, or diagonally—wins the game.

It's an intriguing game. But you're not limited to playing it one way. With Turbo GameWorks, you can modify it your way.

Bridge. Play bridge with a friend or team up against the program—you decide which hands the computer plays. You can even decide to let the program cheat! The program automatically bids and plays its own hands. And, since you can tinker with the source code, you can make "your" Bridge unlike any other.

4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

Inquiry 17 for End-Users. Inquiry 18 for DEALERS ONLY.

With Dazzling Discounts!

TURBO HOLIDAY JUMBO PACK $245.00

Now you get the whole Turbo Pascal family, including its newest members, at an incredible price.

- **Turbo Pascal 3.0** combines an integrated programming environment with the fastest Pascal compiler in the galaxy.
- **Turbo Tutor** teaches you step-by-step how to use the Turbo Pascal development environment. With commented source code for all program examples on diskette.
- **Turbo DataBase Toolbox** gives you all the tools you need to sort and search your data and build powerful database applications.
- **Turbo Graphix Toolbox™** gives you a complete library of Pascal procedures to build applications using high resolution graphics and graphics window management.
- NEW! **Turbo GameWorks**—Chess, Bridge, Go-Moku. Secrets and strategies of state-of-the-art computer games are revealed with complete source code.
- NEW! **Turbo Editor Toolbox**—the Pascal modules that let you build your own word processor. A full-featured WordStar-like program, MicroStar™, is included on your disk.

You're getting everything at only $40 each. And if you already own one or several members of the Turbo family, be creative—nothing can stop you from buying the Jumbo Pack, picking out the ones you already have and giving them as holiday gifts for family or friends. At these prices you can afford to give to others and to yourself!

NOT COPY-PROTECTED

Holiday Gift Packs, Turbo GameWorks™ & Turbo Editor Toolbox™

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133

The holiday packs include an upgrade coupon for both options so you get BCD and 8087 support for $39.95 (regularly $55.00)

Carefully describe your computer system!

Mine is: __ 8-bit __ 16-bit

I use: __ PC-DOS __ MS-DOS __ CP/M-80 __ CP/M-86

My computer's name and model is: ______

The disk size I use is: ☐ 3½" ☐ 5¼" ☐ 8"

Name ______

Shipping Address ______

City ______

State ______ Zip ______

Telephone ______

NOTE: Turbo Editor Toolbox and Turbo GameWorks are available for the IBM PC and true-compatibles using Turbo Pascal 3.0 ONLY.

*Gift Pack Offers Last Until March 31, 1986

		Quantity
*Turbo Holiday Jumbo Pack	$245.00	___
*Turbo Holiday Pack	$125.00	___
*Turbo New Pack	$95.00	___
Pascal	$69.95	___
Pascal w/8087	$109.90	___
Pascal w/BCD	$109.90	___
Pascal w/8087 and BCD	$124.95	___
Turbo DataBase	$54.95	___
Turbo Graphix	$54.95	___
Turbo Tutor	$34.95	___
Turbo Editor	$69.95	___
Turbo GameWorks	$69.95	___

These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Amount (CA add 6% tax) ______

Payment: VISA MC Check Bank Draft

Credit Card Expiration Date: ____ / ____

Card #: ______

COD's and Purchase Orders will not be accepted by Borland International. California residents: add 6% sales tax. Outside USA: add $10 and make payment by bank draft, payable in U.S. dollars drawn on a U.S. bank. G15

TURBO "NEW" PACK $95.00

You get both our terrific new Turbos, and pay only $47.50 each. Sold separately, they're $69.95 each.

- Turbo GameWorks™
- Turbo Editor Toolbox™

NEW!

TURBO HOLIDAY PACK $125.00

At about a 30% savings, you get our standard holiday special.

- Turbo Pascal® 3.0
- Turbo Tutor®
- Turbo DataBase Toolbox™

All this for only $125.00. See the order form at right.

SAVE OVER 30% ON OUR GIFT PACKS!

Announcing Borland's New Turbo Editor Toolbox™ $69.95

IT'S ALL YOU NEED TO BUILD YOUR OWN WORD PROCESSOR FOR ONLY $69.95!

You get all the modules you need to build your own word processor.

- You get ready-to-compile source code.
- You get a full-featured word processor that looks and acts like WordStar™; we call it MicroStar™
- You get a 200-page manual that tells you how to integrate the editor procedures and functions into your programs.

You can use Turbo Editor Toolbox "as is" or modify it any way you want.

And you don't get a bill for Royalties. Because Borland doesn't believe you should pay for something more than once.

All this and more for only $69.95.

And until March 1, 1986 you can get Borland's new Turbo Editor Toolbox for even less! (Only $47.50 when you buy the special Turbo New Pack.)

YOU CAN HAVE MANY WINDOWS ON THE WORLD. The new Turbo Editor Toolbox features windowing, a technique that lets you see several documents—or several parts of the same document—at once. You know best what your needs are. Turbo Editor Toolbox lets you open the windows you want. And to make those windows part of your program.

WITH TURBO EDITOR TOOLBOX YOU CAN HAVE THE BEST OF ALL WORD PROCESSORS IN YOUR WORD PROCESSOR. You can make WordStar behave like Multi-Mate. Support windows just like Microsoft's® Word™. And do it as fast as WordPerfect does it. Incorporate your new "hybrids" into your programs to achieve incredible control and power.

Turbo Editor Toolbox. It's the kind of tool that almost everyone needs—and we're the kind of company to give it to you at a reasonable price, without any compromise on quality. (We're so sure you'll be satisfied that we offer a 60-day money-back guarantee—something no one else does.)

THE CRITICS' CHOICE

"Turbo Pascal has got to be the best value in languages on the market today—and Borland International, by delivering excellent products at reasonable costs, is leading the software industry where it has to go. Turbo Pascal is more than just a good program at a low cost. It's also a low-cost, well-conceived programming language making it possible for lots of people to produce good programs."
Jerry Pournelle, BYTE

"This compiler, produced by Borland International, is one of the best programming tools presently available for the PC."
Michael Covington, PC Tech Journal

"Language deal of the century...Turbo Pascal."
Jeff Duntemann, PC Magazine

BORLAND INTERNATIONAL

Three Special Gift Packs

60-DAY MONEY BACK GUARANTEE

Borland's SuperKey And SideKick Work So Well Together, You'll Hardly Work At All.

AN UNBEATABLE TEAM AT AN UNBEATABLE PRICE! We've teamed the best with the best to make the greatest. The best keyboard enhancer, SuperKey®. The best desktop organizer, SideKick®. The dynamic duo working hand in hand to let you do many different things at once. A way that cuts down the keystrokes, so you're working instead of just typing. A way that wasn't possible until we paired the electronic wizardry of SuperKey with the practical efficiency of SideKick. SuperKey brings the magic. SideKick does the details. The "S-Team" works beautifully together because we designed them that way.

GET SUPERKEY AND SIDEKICK TODAY and you'll have an unbeatable team at an unbeatable price — and a $15 rebate back in the mail.

SIDEKICK INCLUDES: * Calculator * Notepad * Auto dialer & phone directory * ASCII table * Perpetual calendar & datebook * Help window * Full-screen editor with word-wrap, paragraph editing and much much more. (Chances are that once you have SuperKey and SideKick working together for you, you'll never need to use a word-processing program again).

SUPERKEY INCLUDES: *Encryption to keep confidential files confidential * Programmable keys that let you turn a thousand keystrokes into one * Keyboard lock * Automatic turn-off of your screen after a pre-set time so the expensive phosphorus in your monitor's screen isn't etched or ruined * Secret Password protection and more.

IF YOU USE SIDEKICK, YOU NEED SUPERKEY. BECAUSE SUPERKEY AND SIDEKICK CAN MAKE YOUR DAY GO SOMETHING LIKE THIS:

8:00 am. You got to work on time, despite the 44-mph turkey ahead of you in the fast lane. It's spreadsheet time. You hit one key. Lotus 1-2-3 (or whatever) is up and running. (One key, because SuperKey has recorded all the CD\123 <ENTER>123< ENTER> <ENTER> / F <ENTER> R <ENTER> SALES <ENTER> <PgDn> foolishness and your one keystroke played all that back instantly. One keystroke instead of a minuet).

8:03 am. You're into the spreadsheet. Phone rings. You kick in SideKick's Notepad—without leaving your spreadsheet. You talk. You listen to Frank. You make notes that tell you that Frank is upping the numbers from yesterday's order and he needs a new price and delivery date. He wants a meeting. Fast, but when? You have SideKick fire up your Calendar. Time agreed and noted—in SideKick's NotePad. Conversation ends. Your spreadsheet is still there.

8:07 am. You're watching the spreadsheet but you're thinking about the new bid you have to figure out. So you have SideKick's Calculator pulled up on the screen—over a small piece of the spreadsheet—which doesn't go away.

8:08 am. SideKick is coming up with new numbers. SuperKey keeps the spreadsheet on a roll. Satisfied with the numbers, you have SideKick auto-dial Frank's number. Talk. Talk. Hang up.

8:09 am. Spreadsheet about done. You're watching it, but thinking about what Frank just said on the phone. He liked your numbers. He ordered. He said, "That was fast. We won't need that meeting. (SideKick cancels it from your Calendar). And he also said, "How did you get all that done so quickly?" And you said, "I've got a couple of new guys working for me."

IF YOU DON'T USE SIDEKICK YET, YOU GET THEM BOTH AND FOR A LIMITED TIME, A $15.00 CASH-BACK! Because SuperKey and SideKick are so compatible, we let them move in together. Into their own blister-pack. With the $15.00 cash-back coupon and the manuals. Which is what you get for $139.90 instead of the usual $154.90. You need to fill-in the cash-back coupon, along with your registration cards and proof of purchase, and mail it back. We'll rush you your $15.00 rebate right away. Rebate offer ends March 31, 1986. (PS: You can still buy SuperKey and SideKick separately. SuperKey **$69.95**, SideKick **$84.95**. Not copy-protected.)

THE CRITIC'S CHOICE

"If you want the ultimate in sophistication, you won't find anything finer on the battlefield right now than Borland's SuperKey and SideKick combo." **LEON A. WORTMAN, InfoWorld**

BORLAND

4585 Scotts Valley Drive, Scotts Valley CA 95066
Phone (408) 438-8400 Telex 172373

NOT COPY-PROTECTED

SuperKey SIDEKICK

SuperKey and SideKick require DOS version 2.0 or later.
Software for your IBM®PC, XT, AT, 3270, PCjr and true compatibles.

Available at better dealers nationwide. Call (800) 556-2283 for the dealer nearest you. To order by Credit Card call (800) 255-8008, CA (800) 742-1133.

Yes, I want the Best!

	Quantity
The SideKick-SuperKey Team **$154.90** $15 cash-back ONLY when purchased together. Both come not copy protected	______
SuperKey **$69.95**	______
SideKick **$54.95** (copy protected)	______
SideKick **$84.95** (not copy-protected)	______

Subtotal ______
(CA res. [illegible] tax ______
Amount Enclosed ______
Payment VISA MC BankDraft Check
Credit Card Exp. Date ____/____
Card # ______

$15.00 Cash Back

SuperKey only—Outside USA and Canada: Due to US Government restrictions of export or technology, purchase of SuperKey (without DES) can only be made through authorized distributors.

SideKick only: PCjr requires the not copy-protected version.

$15 cash-back rebate upon receipt of signed license agreement and cash-back coupon. Your $15 cash-back rebate included tax when applicable. These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered.

Name ______
Shipping Address ______
City ______
State ______ Zip ______
Telephone ______

CODs and Purchase Orders WILL NOT be accepted by Borland. California residents, add 6% sales tax. Outside USA add $10 and make payment by bank draft, payable in US dollars drawn on a US bank.

G15

Inquiry 17 for End-Users. Inquiry 18 for DEALERS ONLY.

Cover illustrated by Catherine Kanner

VOLUME 10, NUMBER 11, 1985

BYTE

Inside the IBM PCs

BYTE (ISSN 0360-5280) is published monthly with one extra issue per year by McGraw-Hill Inc. Founder: James H. McGraw (1860-1948). Executive, editorial, circulation, and advertising offices: 70 Main St., Peterborough NH 03458, phone (603) 924-9281. Office hours: Mon–Thur 8:30 AM – 4:30 PM, Friday 8:30 AM – 1:00 PM, Eastern Time. Address subscriptions to BYTE Subscriptions, POB 590, Martinsville, NJ 08836. Postmaster send address changes, USPS Form 3579, undeliverable copies, and fulfillment questions to BYTE Subscriptions, POB 596, Martinsville, NJ 08836. Second-class postage paid at Peterborough, NH 03458 and additional mailing offices. Postage paid at Winnipeg, Manitoba. Registration number 9321. Subscriptions are $21 for one year, $38 for two years, and $55 for three years in the USA and its possessions. In Canada and Mexico, $23 for one year, $42 for two years, $61 for three years. $69 for one year air delivery to Europe. 17,100 yen for one year surface delivery to Japan. $37 surface delivery elsewhere. Air delivery to selected areas at additional rates upon request. Single copy price is $3.50 in the USA and its possessions, $3.95 in Canada and Mexico, $4.50 in Europe, and $5 elsewhere. Foreign subscriptions and sales should be remitted in United States funds drawn on a U.S. bank. Please allow six to eight weeks for delivery of first issue. Printed in the United States of America.

Address all editorial correspondence to the Editor, BYTE, POB 372, Hancock, NH 03449. Unacceptable manuscripts will be returned if accompanied by sufficient first-class postage. Not responsible for lost manuscripts or photos. Opinions expressed by the authors are not necessarily those of BYTE.

BYTE

Editorial and Business Office: 70 Main Street, Peterborough, New Hampshire 03458, (603) 924-9281
West Coast Offices: McGraw-Hill, 425 Battery St., San Francisco, CA 94111, (415) 362-4600
McGraw-Hill, 1000 Elwell Court, Palo Alto, CA 94303, (415) 964-0624
New York Office: 1221 Avenue of the Americas, New York, NY 10020, (212) 512-2000

Officers of McGraw-Hill Information Systems Company: President, Richard B. Miller. Executive Vice Presidents: Frederick P. Jannott, Construction Information Group; Russell C. White, Computers and Communications Information Group; J. Thomas Ryan, Marketing and International. Senior Vice Presidents: Francis A. Shinal, Controller; Robert C. Violette, Manufacturing and Technology. Senior Vice Presidents and Publishers: Laurence Altman, Electronics Week; Harry L. Brown, BYTE and Popular Computing; David J. McGrath, Construction Publications. Group Vice President: Peter B. McCuen, Communications Information. Vice President: Fred O. Jensen, Planning and Development.

Officers of McGraw-Hill, Inc.: Harold W. McGraw, Jr., Chairman; Joseph L. Dionne, President and Chief Executive Officer; Robert N. Landes, Executive Vice President and Secretary; Walter D. Serwatka, Executive Vice President, Publishing Services; Ralph R. Schulz, Senior Vice President, Editorial; Shel F. Asen, Vice President, Manufacturing; George R. Elsinger, Vice President, Circulation; Ralph J. Webb, Vice President and Treasurer.

Pure power.

QIC-60 streaming tape backup is more than just fast and simple to operate. It's also the leader in powerful processing.

For example, our unique five-head tape deck offers immediate read-after-write verification. No rewinding necessary. And no re-recording of the entire backup after an error.

Another powerful advantage is automatic adjustment of the head mechanisms to seek the best track alignment every time a tape is inserted. Plus our exclusive power sensor is designed to alert and freeze the write head if PC power is reduced, which eliminates the chance of disk damage. We're so confident in the technical excellence of QIC-60 that we offer an aggressive one-year limited warranty on the entire line, compared to the 90-day warranty provided by other manufacturers.

Best of all are a pair of backup bonuses that turn QIC-60 into a sophisticated data management tool. When you select mirror-image restore, QIC-60 sees bad disk sectors and automatically reroutes data to safe locations for improved data reliability.

Tecmar offers AT owners a special enhancement with an internally-mounted QIC-60 tape drive that costs even less than external systems. And when you add our internal AT Hard Disk to the package, you have a fully upgraded system ready to use today.

Pure speed. Pure power. Pure innovation from Tecmar.

For a free demonstration of the new standard in tape backup, see your Tecmar dealer or call us at **216/349-1009** for the location of the dealer nearest you.

Tecmar

THE POWER BEHIND THE PC

Inquiry 197

6225 Cochran Road Solon, Ohio 44139

INTEL AND FUTURE IBM PCs

The planning stages of this issue started under a cloud—we thought there was nothing else interesting to say about the IBM Personal Computers, a technology that some people thought was obsolete when it was introduced three years ago. But as you'll see, we were wrong—the PC world is richer now than ever before.

This special issue peers into some seldom-seen corners of the IBM PC family of machines. Stephen Fried presents some startling findings concerning the 80287 NDP; Bill Claff and Steve Satchell explore the 80286 from a programmer's point of view; Tom Wadlow explains how to use interrupts to write your own desktop accessories; John Haynes demonstrates how to use Lotus 1-2-3 to design circuits; Rick Grehan looks at sub-$1000 hard disks; Jon Edwards highlights some of the best available PC public-domain technical software; and more.

The artwork accompanying the articles is another special aspect of this issue—each of the 18 pieces of art was commissioned from one of the top female illustrators in the country.

IBM/INTEL IMPACT

The IBM PC changed the face of personal computing in 1982 by shifting the primary use of the machines from hacking to tracking corporate finances, databases, and memos. An ironic 1985 turnabout finds the PCs being blamed for blunting the creativity of personal computing and sapping it of its vitality. Hardware designers gripe about Intel parts, like the 80286 and 80287, that haven't worked correctly in their first few iterations, and programmers complain about the headaches of programming in Intel's segmented-memory universe. (I vote for the following for programmer's buzzwords of the year—address paranoia: constantly checking for 8086 segment boundaries.) Most of the newer and more innovative machines of the last two years, like the Macintosh, Amiga, and Atari ST computers, have been based on the Motorola MC68000 microprocessor rather than the Intel iAPX86 family of processors that power the PCs.

With IBM owning part of Intel and apparently wedded to the Intel architecture, the microcomputer world seems to be polarizing around the two microprocessor giants.

TECHNICAL CULTURE

Some people speculate that definite architectural biases are developing around Motorola and Intel. Motorola seems to be a company with a bias toward single-processor systems and open, flexible architectures. Intel's bias seems to favor complex, multiple processor architectures that are of necessity more rigid and formal. These biases show clearly, for example, in the companies' respective approaches to memory management and in their designs for the VMEbus and the Multibus II: Motorola makes hardware memory management a system designer's option, while Intel includes it as a microprocessor function with the iAPX286 and higher processors; Motorola's bus is nicely suited for single-processor systems, while Intel's design is optimal for multiple boards and processors.

These architectural biases seem to favor Motorola hardware for low-cost single-user machines and Intel's for multiuser computers such as the three-user IBM PC AT. Of course, either company's microprocessor can power a single- or multiuser machine: Many of the high-end supermicros are MC68010-based machines, and most PC ATs are being used presently as single-user machines until a multiuser version of MS-DOS appears to make the vast PC software base available to multiuser ATs.

A close look at these microprocessors reveals that Intel often solves problems by throwing hardware at them, while Motorola usually just hands programmers a toolbox.

THE iAPX286/386

One of the principal design goals of the 80286 and 80386 microprocessors was to facilitate high-performance multiuser and multitasking capability. To provide the protection needed to run separate tasks or multiple users in different parts of memory, Intel built task switching and memory management into the silicon of its two newest microprocessors.

Task switching is done entirely in hardware on the 80286/80386 processors. Operating systems only need to specify which task runs next in the Intel environment, and the hardware handles the switching. Task switching is possible due to the addition of descriptor tables to the 80286/80386 programming model that tell the CPU where to find the instructions and data for individual tasks (or users).

Descriptor tables also facilitate 80286/80386 memory management and virtual memory. On-chip memory management can save 100 to 150 nanoseconds in the memory-access time cycle—this is the typical time to send a calculated logical-memory address to an external MMU (memory-management unit) to find a physical address. On-chip MMUs also save time by permitting the use of special instructions to reduce the virtual-memory delays incurred by the swapping of data from disk to memory. Using virtual memory, the 80286 can address up to 16 megabytes of memory, while the 80386 can address

(continued)

THE ACCENT™ IS ON EXPANDED MEMORY...

WITH MAYNARD'S NEW LOTUS-COMPATIBLE CARD!

Now, Maynard gives you 2MB of Lotus-compatible Expanded Memory with Accent.™ Designed expressly for your personal computer, Accent™ adds memory power to Lotus 1-2-3,* Symphony,* and all other Expanded Memory programs so you can create larger spreadsheets than ever! An optional mouse places speed, access, and customized commands at your fingertips, saving time and eliminating inconvenient, multi-key cursor controls. Accent™ is available at the finest computer retail stores everywhere. Contact your local dealer or write to us today for product information.

*Lotus 1-2-3 and Symphony are trademarks of Lotus Development Corporation.

Accent™

Expanded Memory Card by Maynard Electronics

Maynard Electronics

Shaping tomorrow's technology.

430 E. SEMORAN BLVD., CASSELBERRY, FL 32707 305/331-6402

Inquiry 123

EDITORIAL

1500 bytes for every man, woman, and child in the world (64 terabytes).

These high-level memory-protection and -management capabilities are well suited to computers that rely on multitasking and multiuser applications. In fact, CPUs with this kind of power may well be wasted in machines employed by a single user.

THE MOTOROLA MC68000

Apple Computer's choice of the MC68000 to power the Lisa and the Macintosh computers sparked mainstream interest in that microprocessor even though it was used earlier in a variety of other machines. While the excitement created by these Apple machines had more to do with the software developed to run them than the CPU, the MC68000 nevertheless became suddenly more visible to the microcomputer community.

The MC68000 does not provide on-chip hardware support for memory protection and management. External MMUs are available from Motorola and other sources, or programmers can manage the 68000's linear address space in software. Some companies have developed proprietary hardware to manage memory.

Similarly, virtual memory on 68000 machines is a software function, as is memory protection for multiuser/multitasking operations. The newest 68000, the MC68020, has special instructions to ensure data security in single- and multiprocessor systems (see "The MC68020 32-bit Microprocessor" by Paul F. Groepler and James Kennedy, November 1984 BYTE, page 159).

In fact, the design principles behind the 68000 were stated in BYTE over two years ago ("Design Philosophy Behind Motorola's MC68000" by Thomas W. Starnes, April 1983, page 70): "They would design it [the 68000] for programmers, to make their job easier, by providing functions in a way that most programmers could best use them." The head start that Apple's (and, more recently, Atari's and Amiga's) programmers have developed in creating exciting new software indicates that Motorola met its 68000 design goals. As further evidence, consider that the first microcomputer UNIX ports were to 68000 machines, too, partly due to the 68000's similarity to Digital Equipment Corporation's VAX hardware and partly to its programming accessibility.

THE SILICON BRICK ROAD

So the Motorola versus Intel biases mentioned earlier may boil down to a simple matter of orientation—Intel designs focus on hardware capability, while Motorola's emphasize programmability.

For example, look at the companies' respective approaches to graphics. Intel has designed, and plans to ship in early 1986, a graphics coprocessor (the 82786) with an on-chip CRT interface that provides 640 by 480 resolution with 256 colors, or 1024 by 1024 resolution with 4 colors. Motorola's 68020 adds eight new bit manipulators to its instruction set to assist the programming of bit-mapped graphics. The Intel approach will be faster and more powerful but will require additional programming, hence higher software overhead, and will add substantial hardware costs. Motorola's scheme will be cheaper and well suited to CRT hardware developments.

What this all means is hard to predict, of course, but I'll take a stab at a couple of certainly debatable propositions. First, I think the Intel/IBM machines of the future will be the more powerful number crunchers and will be better suited to multiuser and networking environments. After all, the second word in the name IBM is Business. Second, I expect the innovations in software that will keep our industry vital to originate on 68000 machines. Lastly, I think the classic single-user microcomputer, the machine we all want on our desk or at home to play and tinker with, will be a 68000-family machine.

What brings all this to mind is our current assignment—to bring to BYTE readers a special issue, similar to the one you're holding, on 68000 machines. Look for it next June.

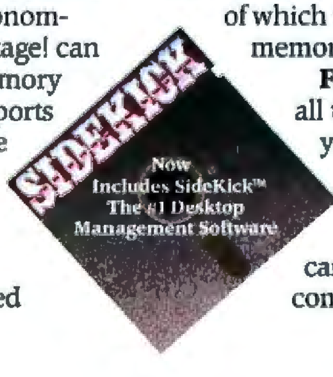
—*G. Michael Vose*
Senior Technical Editor

4 Out Of 5 PC-AT™ Expansion

The overwhelming choice of IBM® PC-AT users, Advantage! from AST sets the standard in high-powered multifunction enhancement. Advantage! was the first multifunction board for the PC-AT, and it remains the leader by providing millions of characters of memory capacity, two serial ports, a parallel port and a game port—in a single expansion slot. Now we've even included SideKick—Borland's multipurpose software selected InfoWorld's 1984 software of the year—an $84.95 value at no extra charge.

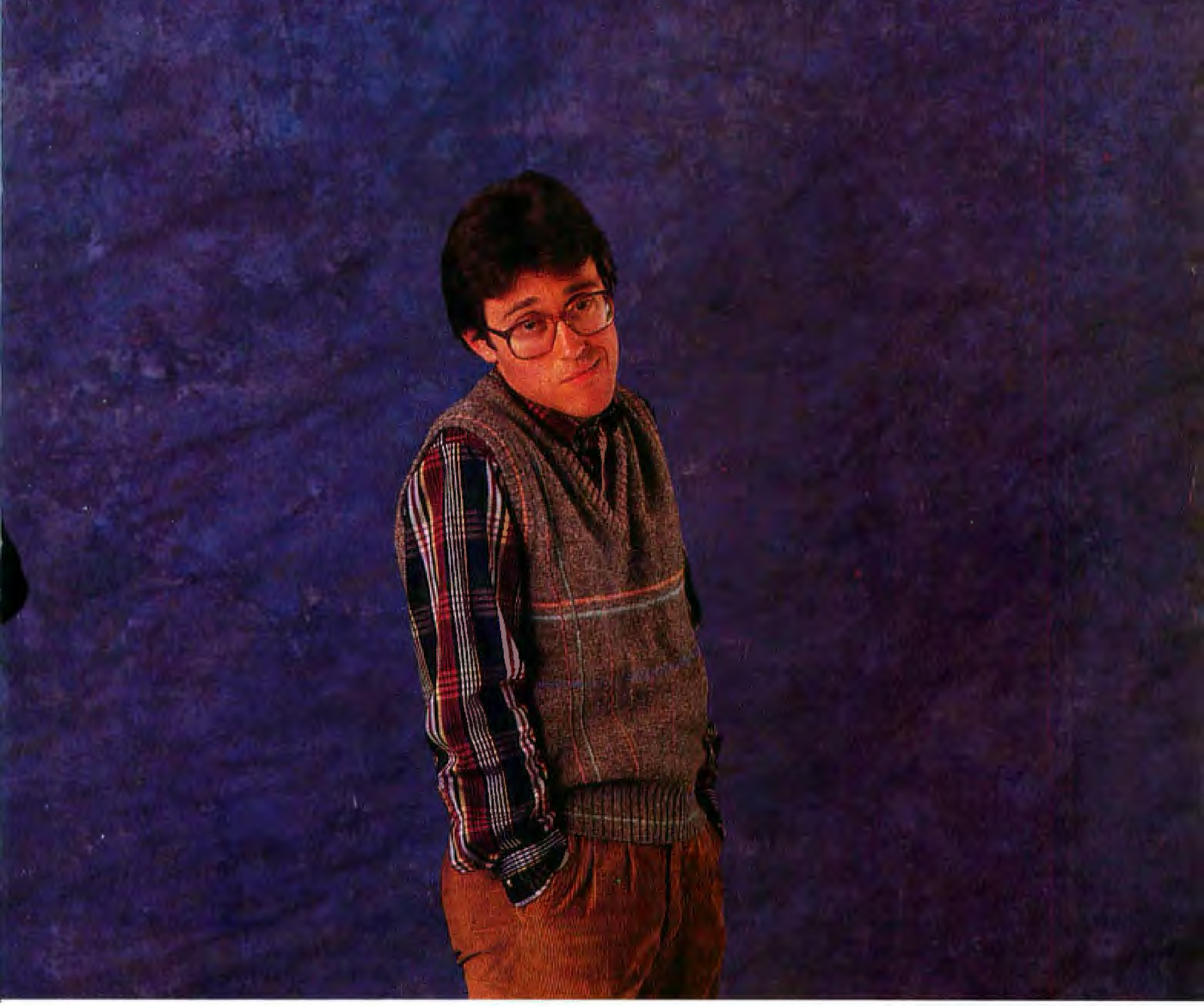
First In Memory. All it takes is Advantage! There's no need to add hard-to-find chips on the system board. Whether you sell 256K or 512K ATs, our unique memory addressing technique lets you add up to 3 Megabytes of parity checked memory efficiently and economically. For flexibility, Advantage! can use either 64K or 256K memory chips. And of course, it supports your AT's high performance 16-bit bus and faster program processing speed.

Now you can have the full complement of 640K memory to run sophisticated business software such as Lotus® 1-2-3™ and dBASE III.™ To utilize multitasking programs like IBM's TopView.™ Or for operating systems such as XENIX,™ and RAM disks such as IBM's VDISK utility—both of which use the AT's extended memory capacity.



First In I/O. Here's all the I/O capability you need now, even if you're starting with a base model AT. Every Advantage! card includes an AT compatible serial port

Board Buyers Own Advantage!™



and a parallel port so you can connect printers, plotters, mice and modems. Or with the appropriate software, you can connect other terminals to create multiuser environments.

With our optional second serial port you can attach even more peripherals, while our optional game port lets you plug in joysticks and other cursor-control devices for business or just for fun.

First In Quality. AST's reputation is built on quality products, quality support and quality service, including a return rate we're very proud of, averaging 0.2 percent. Our complete documentation means Advantage! is exceptionally easy to install and use, but if it's not enough we're always here to help.

Four out of five buyers agree, the choice is Advantage!—only from AST. Ask your dealer, or call our Customer Information Center (714) 863-1333 for more information. AST Research, Inc., 2121 Alton Avenue, Irvine, CA 92714 TWX: 753699ASTR UR

Advantage! trademark of AST Research, Inc. IBM, PC-AT and TopView trademarks of International Business Machines Corp. SideKick trademark of Borland International. dBASE III trademark of Ashton-Tate. Lotus registered trademark and 1-2-3 trademark of Lotus Development Corp. XENIX trademark of Microsoft Corp.

FEATURES

Memory Expansion
- 128Kb to 3.0Mb in a single slot
- User Upgradeable with either 64K or 256K memory chips
- Split Memory Addressing rounds out AT's system memory to 640K and continues memory expansion at 1MB

Advantage! Supports AT's Full Program Processing Speed

I/O Expansion
- Up to 2 Serial Ports (1 optional)
- Parallel Printer Port
- Optional Game Port

AST RESEARCH INC.

Inquiry 221 for End-Users. Inquiry 222 for DEALERS ONLY.

AN ANNOTATED BIBLIOGRAPHY OF RECENT BOOKS

Technical topics for the IBM PC *family*

BY DONALD EVAN CRABB

This select annotated bibliography is an attempt to cover some of the more noteworthy and recent books published on technical topics dealing with the IBM Personal Computer (PC, XT, PCjr, and AT) and compatible computers. Because of the large number of books written in the past couple of years that fall into this category (Bowker's Paperback Books in Print *lists more than 1000 titles that a comprehensive bibliography would include), a number of compromises have been made in compiling this list.*

This bibliography reflects, therefore, my personal bias as well as many logistical considerations, including the availabililty of the books listed. I have attempted to cover a variety of technical topics, while concentrating on the areas of hardware and microprocessor technology of the PC.

The books listed are divided into seven sections: (1) Hardware Architecture, Expansion, and Maintenance, (2) Intel PC Microprocessors and Assembly Language, (3) High-Level Languages, (4) Graphics, (5) Data Communications, (6) Miscellaneous, and (7) References. Each section (except Miscellaneous) is introduced with a brief description of the category. As is the case with most annotated bibliographies, not every entry carries the same detailed level of annotation. The annotations were written to reflect the content, intended audience, writing style, and value of the books covered.

One striking fact that I discovered while compiling this bibliography and reviewing the books listed was the lack of a complete topical guide to books about the PC (or microcomputers in general) other than the referential listings in Bowker's Books in Print *and* Computer Books and Serials in Print 1984.

Another fact I discovered while researching this article (in April and May of 1985) was the current lack of good technical books about the IBM AT. Some of the books discussed below cover the AT, but none are devoted primarily to that machine.

[*Editor's note:* BYTE *has commissioned a review, for later publication, of* The Practical Guide to the IBM Personal Computer AT *by* Dennis L. Foster *(Reading, MA: Addison-Wesley, 1985.)*]

HARDWARE ARCHITECTURE, EXPANSION, AND MAINTENANCE

This topic is a catchall for books that discuss, primarily, the hardware aspects of the PC. The books cover the hardware design and architecture of the PC (including how the microprocessor is integrated into the overall design of the machine and how support components work with it); how to expand your PC with additional memory, coprocessors, hard disks, modems, and external devices; how to repair and maintain your PC, how to connect your PC to a variety of external devices (and design the hardware and software interfaces necessary to accomplish this); and overall discussions of the hardware and software that comprise the PC.

The books in this section all have a technical flavor to them, but most are written so that nontechnical readers can gain valuable information from them.

Alvernaz, Bil. *Expanding Your IBM PC.* Bowie, MD: Brady Communications, 1984, $16.95, 256 pages.

This practical handbook should be on the shelf of every PC owner who plans on adding functions or features to the system. The book covers how to install boards (memory, multifunction, graphics, etc.) in the expansion slots, how to add memory chips to the PC motherboard, how to install floppy- and hard-disk drives and controllers, and how to install a system-expansion box. The discussion of power-supply requirements is particularly informative. The text provides enough detail to whet the appetite of technical users, while remaining easily readable for the nontechnical audience. The overall focus is toward the beginning computer enthusiast, although the writing is well done, without a reliance on jargon.

Coffron, James W. *The IBM PC Connection.* Berkeley, CA: Sybex, 1984, $17.95, 264 pages.

While the content of Coffron's book is much more technically oriented than the Alvernaz book, the style is

(continued)

Donald Evan Crabb is director of the instructional undergraduate laboratories at the University of Chicago (1100 East 58th St., Chicago, IL 60637.) His articles have appeared in several computer magazines.

still fairly easy to read. The book covers a wide range of subtopics falling within the rubric of computer control of external devices. Areas covered include interfacing software for IBM Personal Computer I/O (input/output), incorporating input data from external devices, I/O hardware for the PC, a sample computer-interfacing application (a home-security system controlled by the PC), adding voice-synthesis capabilities to the PC, analog-to-digital conversion and its converse, a useful appendix that includes a glossary, and data sheets from manufacturers of external devices to interface with the PC.

Friedman, Herb. *The Complete Guide to Care and Maintenance for the IBM PC, XT, AT, and Portable PC*. Englewood Cliffs, NJ: Prentice-Hall, 1985. $15.95. 198 pages.

Herb Friedman, a well-known author of articles on a variety of electronics subjects, has produced an easy-to-read, accessible little book on the subject of maintaining your IBM PC. Although the book does not go into copious detail about the hardware and how it functions, Friedman covers all the salient points of PC maintenance, including maintaining disk drives and adjusting them, keeping dot-matrix and daisy-wheel printers running smoothly with your PC, installing and using various accessories to enhance your PC computing environment (including the use of power-conditioning equipment, surge suppressors and RF filters, and uninterruptible power supplies), protecting your system from environmental problems (e.g., temperature, dust, and humidity), and obtaining (or making your own) good-quality serial and parallel cables and connectors.

The book contains more than 60 illustrations and is written in a clear, nontechnical style. This book is both a good introduction to the subject of PC maintenance for computer novices and a good reference work for technically oriented PC owners and users.

Markowsky, George. A *Comprehensive Guide to the IBM Personal Computer*. Englewood Cliffs, NJ: Prentice-Hall, 1984. $19.95. 516 pages.

Written for the PC user who wants to get a firm overall understanding of the hardware and systems software of the PC, this book does not assume the user has a technical orientation. Markowsky has combined material from the IBM PC manuals (the *Guide to Operations*, *DOS*, *BASIC Compiler*, and *BASIC Intepreter* manuals) with information from the IBM PC *Technical Reference* and the *Macro Assembler* manuals. The result is an easy-to-read and very informative book on the PC and how it works.

The book also contains important discussions on the PC's components and programs available, as well as how to select a system configuration. The technical discussions move from how to set up the PC to how to run it. The sections on BASIC are useful adjuncts to the information in the *IBM BASIC Manual*. Finally, Markowsky explains 8088 assembly language in a way useful to novice programmers and gives you enough information to get started with assembly language on the PC.

The book covers just about all the aspects of the PC's hardware and software, including peripherals such as printers. A brief introduction to the 8087 coprocessor is also provided. This book is comprehensive enough to have been listed in many of the other topical categories of this bibliography. This book should be very useful to new PC owners as well as seasoned users, as much for its good writing style as for its informational content.

McGlynn, Daniel R. *Modern Microprocessor System Design: Sixteen Bit and Bit-Slice Architecture*. New York: John Wiley and Sons, 1984. $29.95. 295 pages. Hardcover.

If you think you might want to do some major surgery on your PC by modifying its design or structure, this could be the book for you. While it is more of a general monograph on 16-bit microprocessors and special bit-slice components, it provides the basic technical information that you need to undertake such a project.

McGlynn has further provided information on modern peripheral components, such as bubble memories, charge-coupled devices, and CRT (cathode-ray tube) interfacing devices. This book is strictly for the technical reader. The style is a bit dry, and the book reads like the academic monograph that it is. If you want to learn about the design of 16-bit microprocessors and how they are incorporated into PCs, this book will get you started.

Norton, Peter. *Inside the IBM PC: Access to Advanced Features and Programming*. Bowie, MD: Robert J. Brady Co., 1983. $19.95. 262 pages.

This book is a classic in the field of IBM PC technical literature. Norton writes well and matter-of-factly while still communicating the inner secrets of the PC. He treats several areas in depth, including 8088 segment register addressing, disk copy-protection methods, keyboard operation and keyboard codes, the PC's memory map, and much more. While the writing is clear, it is aimed at the technical user.

A good glossary of computing jargon is included, as are suggestions for including assembly-language routines into BASIC and Pascal programs. Unfortunately, what's not included is a disk of the programming access tools (things like ROM BIOS access service routines) that are described in the book. For those, you have to pay an additional $65 (plus tax), and the overall utility of this work is substantially diluted when they are omitted.

Novogrodsky, Seth, Frederic E. Davis, and the editors of *PC World*. *The Complete IBM Personal Computer: The Authoritative Guide to Hardware for Expanding the IBM PC, XT, AT and Compatibles*. New York: Simon & Schuster, 1985. $16.95. 281 pages.

(continued)

HAUPPAUGE

Our 287 FAST/5 Gives your IBM PC/AT A Lesson in Speed.

Now, When the Subject Turns to Fast Math, Your AT Will Get an A+.
Right now, the 80287 math Coprocessor in your PC/AT runs at 4MHz. Which means there's room for improvement. That's why Hauppauge has developed the unique 287 FAST/5—a module that gives your AT an unmatched 5MHz capability. Speeding up your math computations by 25%. Now that's an A+ performance! To insure easy installation, we've created a carrier mount for our 287 FAST/5. Just plug the module into your PC/AT's 80287 Numeric Coprocessor socket. No expansion slot required.

1·2·3 GETS A LESSON IN SPEED, TOO!

Introducing Hauppauge's New Lotus Support
In addition to our 287 FAST/5, Hauppauge has recently developed a Lotus 1-2-3 support package that can accelerate your calculations up to 30 times. The package? Our 8087 coprocessor chip and Recalc™ software. Recalc links your Lotus 1-2-3 program to the fast math capability of our 8087 chip. By sending numeric operations to the 8087, computation time can be reduced from 60 seconds to as little as 2 seconds. And the more complex the calculation, the more time you save!

Affordability
We've raised the speed limit for Lotus 1-2-3 and lowered the cost of Recalc software. When you buy any of the Hauppauge products listed below, you can get Recalc for only $49 (suggested retail price is $95).

87 CHIP **$129**
Our 8087 Math Coprocessor for the IBM PC, PC/XT and PC compatibles.

87-2 CHIP **$229**
8 MHz Math Coprocessor for Compaq Desk-Pro and Olivetti PC's.

287 CHIP **$249**
80287 Math Coprocessor chip that runs at 4MHz in the IBM PC/AT.

287 FAST/5 .... **$249**
Our 80287 Math Coprocessor enhancement module that boosts speed in the IBM PC/AT to 5MHz.

Now Available! The 287 FAST/8 .... **$379**
In addition to our 1-2-3 support, we're introducing our new 287 FAST/8—a module that gives your IBM PC/AT an unprecedented 8 MHz capability. Doubling the speed of your math calculations! (module pictured above)

Hauppauge's 287 FAST/5 and Lotus support package are teaching the IBM PC, PC/XT and PC/AT a thing or two about fast math. Learn more by calling us today—or contact your local computer dealer.

Hauppauge Computer Works
358 Veterans Memorial Highway, Suite MSI
Commack, New York USA 11725 (516) 360-3827
Available from your local computer dealer

Advertising: Scali Communications, San Francisco

Hauppauge
(Pronounced: "Ha-pog")

Novogrodsky et al. have provided a very readable introduction to the hardware of the PC and how to add to it. The book covers the PC, XT, AT, and compatibles. It is part of a helpful set of books on the PC published by Simon & Schuster and PC World Books and provides all sorts of useful discussions of replacement keyboards and monitors, multifunction boards, voice-recognition devices, plotters and printers, modems, bubble memories, optical and mechanical mice, joysticks and trackballs, coprocessors, local-area networking hardware and software, and, most important, how to hook all of this stuff up to the PC.

This is a very useful reference book to have around, whether you're a new PC user or an old hand. The writing style is clear and aimed at the nontechnical user (although technical details are provided). The authors have also included an appendix that lists the names and addresses of the hardware manufacturers discussed in the book. I'd like to see four-color pictures replace the "black-on-blue" halftones used to illustrate hardware examples, but that's a minor complaint.

Sargent, Murray, III, and Richard L. Shoemaker. *The IBM Personal Computer from the Inside Out*. Reading, MA: Addison-Wesley, 1983. $16.95. 488 pages.

Sargent and Shoemaker have written an excellent technical discussion of the IBM PC, its hardware architecture, microprocessor, instruction set, and assembly language and how the unit works as a whole. They further cover how to write hardware device drivers and how to debug interfacing software and hardware.

The text also delves into how the PC controls disk drives, how DMA (direct memory access) works, and how to control I/O interrupts. A useful survey of existing software of the PC is also included. This book is a technician's delight, but nontechnical users also will find it a useful reference work. It accommodates both of these audiences without resorting to the dry prose often found in such volumes.

Sargent, Murray, III, and Richard L. Shoemaker. *Interfacing the IBM Personal Computer to the Real World.* Reading, MA: Addison-Wesley, 1983. $16.30. 288 pages.

If you're thinking about hooking your PC or compatible to a laboratory instrument (to read telemetry data, for example) or to another external device at home, in the office, or in the lab, this is the book for you. The authors give detailed instructions on how to accomplish all of this. Step-by-step instructions on connecting your PC to lights, relays, switches, thermostats, sensors, motors, laboratory displays, and other devices are included.

The book is written more for technical PC users than nontechnical ones, but the style is sufficiently clear to be usable by both audiences. The emphasis in the discussions is on how to design and build the interfaces necessary.

Schweider, Pete H. *How To Repair and Maintain Your Own IBM PC XT*. Carson City, NV: Personal System Publications, 1984. $29.95. 192 pages. Hardcover.

This expensive yet well-written little tome contains some useful information about how to troubleshoot hardware problems with the XT (especially hard-disk problems). Sections of the book will be tough going for nontechnical XT users, while other parts have been clearly written for them. Technical users will find a decent amount of detailed information about the nature of hardware problems with the XT and how to solve them. The emphasis in this book is to provide straightforward information to get your broken XT back in running order.

Stone, Harold S. *Microcomputer Interfacing*. Reading, MA: Addison-Wesley, 1982. $31.95. 383 pages. Hardcover.

This is a clearly written textbook that teaches you how to interface micro-

(continued)

PROUD.

To build a workstation good enough to put your name on, use the modular building blocks we put our name on. ACS, a company dedicated to the support of OEM's in pursuit of excellence.

THE POWER . . . a single board SuperComputer that is **100% IBM XT compatible** from its mounting holes to its memory mapping, but with more speed: 4.7 and **8 MHz.** Floppy controllers and network communications are built right onto the motherboard along with a SASI hard disk interface and as much as **1 Meg** of memory.

THE GLORY . . . a graphics card (one card) with more than **20,000,000 pixels** of video memory powerfully arrayed in 5 planes **(32 Colors).** Driven by an on board graphics processor, this **2K x 2K** color card is compatible with most of the popular CAD/CAM packages . . . like AUTOCAD, PC Draft, GKS, and Tektronics Emulators . . . packages that run up to 5 times faster and virtually eliminate redraw by **zooming and panning** in memory instead of constantly recalculating.

AN IMPRESSIVE DISPLAY
The Peacock high resolution monitor will put all this technology up front where it counts. With a horizontal scanning frequency of **33.5 KHz** and a bandwidth of **40 MHz,** this RGB monitor gives you a **1K x 1K** window on the world with profitable OEM pricing.

If you're going to build a workstation, you might as well build the best. The people at ACS are ready to help. For more information, call or write:

ACS International
2105 Luna Rd. Suite 330
Carrollton, Texas 75006
214-247-5151
TLX 709748 ACS UD

In Canada:
Soltech Industries
9274 194th St.
Surrey, B.C. V2T4W2
604-888-2606

ACS

Inquiry 4

processors to memory, to I/O devices, and to other microcomputers. While this text is geared toward college juniors and seniors studying computer science or electrical engineering, the information it contains is well suited to the technical PC user who wants to know more about how the PC's components interface with each other.

One area emphasized is communications: connecting PCs to external networks, modems, and other data-communications devices. The text also covers the microprogramming necessary to enable the microprocessor to control external devices (real-time programming). The coverage extends to 8-bit microprocessors and the Intel 8086 16-bit processor. Despite the book's intended audience and technical bent, the author writes in an easy-to-read (albeit a bit dry) style. Although its price is on the high side, the information will be of interest to the technical user. The book also functions as a reference to the subject.

Williams, Gene B. *How To Repair and Maintain Your IBM PC*. Radnor, PA: Chilton Book Co., 1984. $12.95. 220 pages.

This is considerably more affordable than the Schweider book, and the writing is a bit more accessible. Williams provides essentially the same information (the focus is the PC, rather than the XT). The audience is the nontechnical user who has a basic understanding of how the PC works. Technical detail is provided for those users who need it.

Zaks, Rodnay. *From Chips to Systems: An Introduction to Microprocessors*. Berkeley, CA: Sybex, 1981. $17.95. 552 pages.

This book serves as a technical authority about how microprocessors are designed and work. Although the book covers more than the Intel 8088/8086 processor, the detailed information it provides makes it an important reference work for technical users of the PC. The discussion includes the PC

(continued)

microprocessor family's role in the microprocessor development cycle, moving from a history of microprocessors to a consideration of how microprocessors are used as the building blocks of microcomputers and how the microprocessor support components fit into the overall final product. If you want to know more about how your PC works from a basic electronic level, start with this book and move on to some of the more specific listings below.

INTEL PC MICROPROCESSORS AND ASSEMBLY LANGUAGE

The IBM PC family features Intel microprocessors throughout the line. The oldest member of the family, the PC, uses the Intel 8088 microprocessor chip, running at a clock speed of 4.77 MHz. This chip features an 8-bit external data path and an internal 16-bit architecture. The IBM PC XT uses the same Intel processor as does the ill-fated PCjr.

The newest member of the IBM PC line, the PC AT, uses a different microprocessor, the Intel 80286, to improve overall system performance (the 80286 is a true 16-bit microprocessor and is clocked in the AT at 6 MHz). The 8087 and 80287 floating-point coprocessor chips can also be used in the PC and AT, respectively, to improve the speed of numeric calculations (the software must be especially designed to use the instruction set of the coprocessors for numerically intensive operations).

Several manufacturers (AT&T, Compaq, etc.) of IBM PC-compatible microcomputers have chosen to use still other microprocessors from the Intel 80000 family to help ensure software compatibility with PC-DOS and MS-DOS operating systems, while improving overall performance. The two most prominent microprocessors in use are the Intel 8086 and the Intel 80186, both true 16-bit microprocessors. The following selection of books represents a sampling of the recent volumes devoted to the Intel 80000 microprocessors, their instruction sets, and their assembly languages.

Bradley, David J. *Assembly Language Programming for the IBM Personal Computer.* Englewood Cliffs, NJ: Prentice-Hall, 1984. $21.95, 352 pages.

If you want to learn how to program in assembly language on your PC, you can start with this book. Bradley ties the concepts of assembly-language programming into those of high-level languages, including BASIC and Pascal, and assumes you have BASIC or Pascal programming experience. Novices may find the book tough going but not impossible.

Bradley presents his material in four sections, covering the fundamentals of PC operation; how the 8088 microprocessor works and what its instruction set is like; how to create, assemble, link, and run assembly language programs; and

(continued)

the uniqueness of the PC assembly language and applications written in assembly language.

Technical users will welcome the discussions of binary arithmetic and data representation. Bradley also covers the 8088 microcode, the special utilities available through the PC Macro Assembler, the additional data types and instructions provided by the 8087 coprocessor, and the registers of the 8088 and how addressing works. A generous use of assembly-language code examples makes the discussion easy to assimilate. Despite the technical nature of the topic, the writing style is clear and straightforward.

Dao, Lanny V. *Mastering the 8088 Microprocessor*. Blue Ridge Summit, PA: Tab Books, 1984. $15.95. 330 pages.

This technical handbook is more of a reference book on the 8088 than a text to learn about its operation. Dao divides his discussion into several areas: the 8088 assembly language, the 8088 instruction set, integer-arithmetic logical functions, data movement, control transfer, and how to shift and rotate memory contents and addresses.

Mastering the 8088 Microprocessor is well organized and thorough. Discussion of hardware interfacing with the 8088 focuses on the minute level of pin-out signals. Dao also discusses the 80286 and 80186 chips and their instruction sets. He includes more than 20 sample assembly-language programs to illustrate the discussions.

Equipped with an extensive index and appendixes, this is an excellent technical reference to the Intel 8088 and 8086 microprocessors and how they work. The writing style is quite readable.

Khambata, Adi J. *Microprocessors/Microcomputers: Architecture, Software, and Systems*. New York: John Wiley and Sons, 1982. $24.95. 577 pages. Hardcover.

Designed for classroom use, this book also functions as a technical reference book. It covers just about all the ground you can think of for the technically oriented PC user. The text moves from an introductory chapter on digital computers to technical discussions of hardware architectural features of different microprocessors (the Intel 8088/8086 chips are used as examples) and how they interface with random-access read/write memory (RAM).

The 8088/8086 is further examined from the point of view of its functional parts, including its control and timing unit (CTU) and arithmetic and logic unit (ALU). How the microprocessor deals with serial and parallel I/O and direct memory addressing (DMA) is also covered, and there is a chapter on BASIC and how it is implemented.

This book is an excellent, easy-to-read introduction to the digital electronics that make the PC work.

Lafore, Robert. *Assembly Language Primer for the IBM PC and XT*. New York: New American Library, 1984. $21.95. 501 pages.

This book from the Waite Group offers a practical introduction to implementing assembly-language code quickly and easily on the PC and PC XT. The discussion covers how PC Macro Assembler works and how to write macros and control peripheral devices with assembler routines.

Lafore teaches you to write one-line programs using the debugger and DOS function calls. From there, the discussion moves to the Macro Assembler and how to write and assemble longer programs. Numerous sample programs and troubleshooting help are provided throughout the book. The book moves you to consider more interesting assembler projects as you learn more about the language (one project, a program to create color graphics and sound effects on the PC, is particularly entertaining).

Lafore has written for the PC programmer who has some familiarity with BASIC programming and DOS but who is not necessarily a sophisticated technical user. Overall, the style and structure of this book work well.

Metcalf, Christopher D., and Marc Sugiyama. *Compute!'s Beginners Guide to Machine Language on the IBM PC and PCjr*. Greensboro, NC: Compute! Publications, 1985. $14.95. 330 pages.

Another guide to assembly language on the IBM PC, this book serves as both a reference guide and a learning text. Specific instruction is provided for EDLIN, DEBUG, LINK, and ASM. A separate section discusses using assembly-language routines from Pascal programs to improve their performance.

The authors provide a number of source-code listings to help in learning the language, most of which are heavily commented. The use of the stack and of subroutines is covered, as is a complete description of the 8088 instruction set and addressing modes.

Characterized by a clean style, with plenty of detailed explanations, this book provides a good introduction to PC assembly language for the technical user who has had some previous programming experience with high-level languages. Nontechnical users would be better off starting with a simpler introduction to assembly language, such as that found in the Markowsky book listed above.

Morgan, Christopher, and Mitchell Waite. *8086/8088 16-bit Microprocessor Primer*. New York: McGraw-Hill, 1983. $21.95. 355 pages.

This book is a very readable introduction to a complex technical topic. Morgan and Waite have put together a useful book (as a reference and as a self-teaching text) about the 1983 state of the art in microprocessor technology, using the Intel 8088 and 8086 as examples.

The authors explain how the 8088/8086 works, how 16-bit memory access and memory management differs from the 8-bit variety, and how coprocessors (e.g., the 8087) can be used to speed up system performance. There is also a discussion of multiprocessing and how it is implemented at the microprocessor level.

(continued)

Finally, the book surveys the most recent programming languages available on 8088/8086-based machines. The information contained here will be of more interest to technical audiences, although the writing is sufficiently clear.

Palmer, John F., and Stephen P. Morse. *The 8087 Primer*. New York: John Wiley and Sons, 1984. $16.95. 224 pages.

While this book, too, is aimed at the technical audience, the authors explain the subject in clear detail. The Intel 8087 coprocessor chip is analyzed from several different angles: its architecture, how it's programmed (its instruction set), and its usefulness to applications programs.

The authors are the inventors of the 8086 and 8087 chips, so their knowledge of the coprocessor is unsurpassed. The book explains how to program the 8087 for quick numeric computations. The interface between the 8087 and 8088/8086-based machines is also detailed. The authors also delve into the interface of the 8087 with high-level languages and discuss how to program and optimize Pascal and FORTRAN code for the 8087. The writing style is a bit clinical, but still usable.

Rector, Russell, and George Alexy. *The 8086 Book*. Berkeley, CA: Osborne/McGraw-Hill, 1983. $16.99. 624 pages.

"Copious coverage" is the best way to describe this book. The authors cover virtually all the technical aspects of the 8088/8086 microprocessors used in the IBM PC and compatibles. The major topic of the book is the Intel 8088/8086's instruction set. Well over half of this large book's pages are devoted to that topic.

Rector and Alexy wrote this book as a solid reference text for the 8088/8086. In this they have succeeded admirably. But the book goes beyond its reference format and is also useful for learning about 8088/8086 assembly language, the differences in the data buses of the 8088 and 8086 (and how that affects assembly-language programming and I/O), and a three-chapter discussion of the electrical-engineering concepts that drove the design and implementation of the 8088/8086 chips.

The largest single failing in this book is its organization and how background information is incorporated into chapters that require it. Even technically oriented readers will find that certain assumptions are made about prior knowledge that are unwarranted in a book with this broad a scope. A related problem is the order of presentation. Chapters that seem better placed at the

(continued)

Those who insist on C compiler performance are very big on Mark Williams.

And the compiler is just part of our <u>total</u> C Programming System.

NEW 3.0 ENHANCEMENTS

These and other powerful utilities now *included* in the C Programming System:

- make: compiles only what's necessary from multiple modules, a powerful programming discipline
- diff: identifies differences between two files
- m4: macroprocessor expression editing and substitution
- egrep: extended pattern search
- MicroEMACS: full screen editor with source

COMPILER FEATURES

- Runs under MS-DOS
- Full Kernighan & Ritchie C with recent extensions including void and enum
- Register variables for fast, compact code
- Full UNIX™ compatibility and complete libraries
- Large and small memory models
- MS-DOS linker compatibility
- 8087 Support
- One-step compiling
- English error messages
- ROMable code
- Linker, assembler, archiver
- Extensive third party library support

csd C SOURCE DEBUGGER

- Debugs at C source level without assembly language
- Separate evaluation, source, program and history windows
- Can execute any C expression
- Capabilities of a C interpreter, but runs in real time
- Set trace points on any statement or variable

Mark Williams' C compiler has earned a place in some very big companies for some very good reasons: it proves the benchmarks right with the speed, code density, consistent performance and expert support required in professional development environments.

But a total development tool shouldn't stop with compiling. Or go on and on with extras that add up and up.

Only Mark Williams' C Programming Systems *includes* the csd C Source Debugger with true source level debugging to speed your programming job.

And only Mark Williams' new 3.0 version *includes* utilities like "make" to make quick work of even the largest projects.

From source code to final product, only one takes you all the way: Mark Williams' C Programming System. All for only $495. Ask about our 30-day money back guarantee when you call **1-800-692-1700** to order today.* You'll be big on the total C Programming System from Mark Williams, too.

*In Illinois call 312-472-6659.

Mark Williams Company
1430 West Wrightwood
Chicago, Illinois 60614

beginning of the book are wedged in toward the end, after the meat of the technical discussion has been presented.

The overall content is good, however, and the writing style is readable, but the structural problems will probably limit this book's usefulness to mostly a reference role.

Startz, Richard. *8087: Application Programming for the IBM PC and the Other PCs*. Bowie, MD: Robert J. Brady Co., 1983. $22.95. 276 pages.

This is a first-rate book. Startz has provided a comprehensive overview of the 8087 coprocessor and how it's programmed. The discussion is useful for both 8087 assembler programmers and users of applications that work with the 8087. The text includes a nontechnical overview of the capabilities of the 8087, including speed benchmarks and buying guidelines for software that uses the 8087.

For programmers, Startz gives the full technical details on the 8087 and its instruction set and how to call 8087 routines from 8088 code and from BASIC programs. The discussion also covers matrix arithmetic, how linear systems work in comparison to nonlinear ones, and a guide to incorporating the 8087 into statistical-analysis programs.

Of all the technical books read for this article, this book was one of the easiest to read, and its style and stucture made reading it from cover to cover a pleasure.

Yeung, Bik Chung. *8086/8088 Assembly Language Programming*. New York: John Wiley and Sons, 1984. $19.95. 265 pages.

Less technical in its style and content than some of the other 8000-family assembly-language books reviewed here, this book still manages to cover the material from a solid technical viewpoint. The book is aimed at the programmer who has had some experience programming in high-level languages on the PC and compatibles (the ACT

(continued)

Hayes sets the standard for personal computer communications. Again. Smartmodem 2400.™

The new fast mover from Hayes. The telecomputing leader. When it comes to communications products for personal computers, we're the leader! Hayes Smartmodem 1200™ set the industry standards for quality, reliability and performance.

Now our new, faster Smartmodem 2400 goes even further to lower telephone line costs and improve user productivity. So, at twice the speed of a 1200 bps modem, it quickly pays for itself in any high-volume communications operation.

Smartmodem 2400 provides a quick link to minis and mainframes. Both synchronous and asynchronous transmissions are supported by an advanced version of the well-known Hayes "AT" command set. You can download from the IBM mainframe at the home office. Send data to the mini upstairs. And guarantee accurate transmission with information services.

With worldwide communications in mind, Smartmodem 2400 was designed to meet CCITT international standards. It provides a fast, cost-effective way to transmit data between approved countries.

New version of Hayes Smartcom II® communications software creates a complete telecomputing system with Smartmodem 2400. Our new Smartcom II, Version 2.1, is available for the IBM* PC and many popular compatibles. Smartcom II makes the most of Smartmodem's exceptional features, at the same time it makes communicating easy for you. And, if you're currently using an earlier version of Smartcom II, Hayes offers a $25 upgrade to Version 2.1.

So if you're looking for ways to streamline your communications, see your authorized Hayes dealer right away. For a hands-on demonstration of Smartcom II and our new Smartmodem 2400. Guaranteed to get you moving fast!

Hayes Microcomputer Products, Inc., 5923 Peachtree Industrial Blvd., Norcross, Georgia 30092. 404/441-1617.

Smartmodem 2400

• Direct connect • Asynchronous and synchronous communications • Accommodates Hayes-compatible modems of slower speeds • Meets CCITT worldwide standards • Keyboard control of all communications parameters • High speed indicator • Voice/data capabilities • Call progress monitoring • Two-year limited warranty with optional four-year extended warranty available.

Smartcom II

• Hayes Verification and XMODEM protocols • Emulates DEC* VT52 and VT100/102 • Totally unattended operation • Voice/data capabilities.

Hayes®

Apricot is also used in programming examples) and who now wants to learn PC assembly language.

In 18 well-packed chapters, Yeung covers all the necessary topics, from an introduction to the MS-DOS toolkit to how to use the assembler for Boolean instructions. Material includes data manipulation, binary arithmetic, program branching and looping, string manipulation, subprogramming, and direct I/O. The author also discusses macros, as well as how to interface assembly-language programs with BASIC. This is a good text on PC assembly language and can also function as an assembly-language reference manual.

High-Level Languages

The IBM PC *and its compatibles can work with a wide variety of high-level programming languages (through* PC- *and* MS-DOS *and* CP/M-86). BASIC *(interpreted and compiled),* Pascal, FORTRAN, COBOL, LISP, C, *and many others have all been written for the* PC. *The books listed here teach some aspects of high-level language programming on the* PC *while maintaining a technical focus.*

Brown, Jerald R., and LeRoy Finkel. *IBM PC Data File Programming*. New York: John Wiley and Sons, 1984. $14.95. 367 pages. Also available with a disk of BASIC programming examples for $39.95.

This text is a complete, accessible study guide for creating, using, and modifying BASIC data files on the PC. The book lists a number of ready-to-run BASIC subroutines that perform a variety of file access methods useful in building business applications in BASIC on the PC. These same routines are also available on disk for an additional charge.

Nashelsky, Louis, and Robert Boylestad. *IBM PC/XT BASIC Programming and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1984. $14.95. 304 pages.

A complete book of programming methods and how to build BASIC

(continued)

applications on the PC and XT (both interpreted and compiled BASIC). The discussion is structured in a manner usable to novice programmers and those already exposed to BASIC. All of the necessary BASIC data structures and control methods (selection, iteration, sequencing, etc.) are covered. Sample applications provide a good tie-in with the concepts being taught. Overall, it is a well-done effort that should prove its worth as an instructional text and a reference work.

Overgaard, Mark. *Personal Computing with the UCSD p-System.* Englewood Cliffs, NJ: Prentice-Hall, 1983. $21.95. 448 pages.
This book is valuable for learning the p-System and how UCSD Pascal works within this system. It can also serve as a reference text for the system. The book is easy to read and is aimed at both novices and experienced PC users. The structure moves from a general discussion of system features to a more technically oriented discussion of the development tools and utilities the p-System provides to programmers.

Topics covered include the system editor, filer, and operating-system menus and how to work in the modular environment of the p-System. *Personal Computing with the UCSD p-System* is must reading for anyone contemplating adding the p-System or UCSD Pascal to the PC.

Pollack, Lawrence, and Bryan Cummings. *Programming in C on the IBM PC.* Englewood Cliffs, NJ: Prentice-Hall, 1984. $16.95. 224 pages.
The UNIX operating system and the language it's written in (C) threaten to remake the world of microcomputing. Because UNIX and C are easily ported from mainframes and minis to micros, the ability to do large-scale application development on a mainframe or a minicomputer for eventual installation or marketing on a micro is particularly attractive to software developers.

A number of good books have been published on how to program in C, but few have been written for implementations of C language on the IBM PC (regardless of whether the PC is running under UNIX, the UNIX clone XENIX, or PC-DOS). This book addresses that void. The problem is that the C language covered here is a portable C. This means that no specific C compiler is discussed in any detail. So, while the book does a good job of teaching the structure and syntax of C language (just as do other C books), it does not give you any information about acquiring or using a commercially available C compiler for your PC. It is also unfortunate that the sections that discuss the PC specifically are crowded into the front of the book and are pretty thin.

Presley, Bruce. *A Guide to Programming the IBM Personal Computers.* Albany, NY: Lawrenceville Press (Delamar Publishers Inc.), 1985. 2nd edition. $18.50. 360 pages.
This book is an excellent textbook for teaching BASIC programming on the PC. The layout of the material and sequence of chapters also lends itself to self-instruction. Presley covers just about all the content you could reasonably expect in such an effort. The discussion moves from an introduction to programming concepts on the PC through increasingly complex topics, including subroutines, string functions, mathematical functions, sequential files, random-access files, and search/sorting algorithms.

Use of program examples throughout and student exercises at the end of sections enhance the learning experience. A teacher's guide is available as a companion volume. The book is written for the novice programmer and will also work as a BASIC reference guide. Altogether, it's a good effort at supplying the needs for a BASIC textbook on the IBM PC with a good substructure of technical information.

Stiegler, Mark, and Bob Hansen. *Programming Languages: Featuring the IBM PC and Compatibles.* New York: Baen Books, 1984. $9.95. 413 pages.
This book is a critical evaluation of the different programming languages available on the PC and its compatibles. The strengths and weaknesses of many languages are discussed, including BASIC, FORTRAN, APL, FORTH, RATBAS, C, Pascal, Modula-2, Ada, LISP, and assembly language.

The authors have worked out examples of each language and use charts and tables to analyze these samples. This book assumes considerable technical proficiency at programming, but it is well written and clear. This is an important reference book for anyone who does application development and programming on a PC.

GRAPHICS

From the first announcement of the PC, programmers and software users have been interested in its graphics capabilities. A number of recent books explore the medium- and high-resolution color graphics and the high-resolution monochrome text and graphics made possible by graphics boards from IBM and other vendors. The following are a few important books available on this subject, from the most technical discussions of graphics programming through more accessible, user-oriented graphics discussions.

Artwick, Bruce A. *Applied Concepts in Microcomputer Graphics.* Englewood Cliffs, NJ: Prentice-Hall, 1984. $29.95. 384 pages. Hardcover.
Although this book covers graphics applications on both the Apple II and IBM PC, the book is a useful one for the PC owner. A number of concepts are explained in a tutorial manner, including how to create custom color graphics on the PC. Artwick covers all aspects of microcomputer graphics, from creating business charts and graphs to producing real-time animation on the screen.

The discussion also covers display hardware, peripherals, design elements, interactive designs, and high-performance graphics. The book is written in a dry technical style and

(continued)

is useful primarily to technical audiences. More than 150 illustrations are included, 14 of which are in full color. If you are currently working on graphics programming or development work or are planning graphics work in the future on your PC, this book should prove to be an important reference tool. [*Editor's note: For additional comments, see the* Book Reviews *column, September* 1984 BYTE, *page* 66.]

Conklin, Dick. *PC Graphics: Charts, Graphs, Games, and Art on the IBM PC*. New York: John Wiley and Sons, 1984. $15.95. 182 pages. Also available with a disk of 40 BASIC programs for $40.90.

Dick Conklin was the product planner at IBM responsible for the system software on the PC. His familiarity with the internals of the PC and how best to utilize them for PC graphics shows in this small book. He covers material from programming samples of simple graphics images to more complex graphics applications (business charts, engineering data plots, computer art, maps, and games).

The book is written for the PC BASIC programmer (the features of BASICA are used most prominently) who wants to create a variety of graphics images and applications. It includes 40 ready-to-run graphics program listings that are also available on disk for an additional charge.

The author has produced a clear, detailed description of the graphics capabilities of the PC, from the ASCII (American Standard Code for Information Interchange) graphics characters in ROM (read-only memory) to making special characters from the keyboard. He covers animation, shading, slide shows, reading data from the screen, and both medium- and high-resolution graphics. Graphics input and output devices (joysticks, light pens, paddles, color printers, and videodiscs) are also discussed in considerable detail. In short, if you want to know anything about graphics capabilities and usage on the PC, this book is the place to start.

Hearn, Donald D., and M. Pauline Baker. *Computer Graphics for the IBM Personal Computer*. Englewood Cliffs, NJ: Prentice-Hall, 1983. $18.95. 330 pages. Hardcover.

These authors have written several books on computing and graphics. This book is among their best. Hearn and Baker discuss the topic from a comprehensive point of view, covering PC graphics from the basics of the PC's system unit through the
(continued)

differences between character-generated graphics (e.g., the standard ASCII graphics characters) and pixel-generated graphics.

Like other books of its kind, this book relies upon BASICA programming examples to illustrate the concepts the authors discuss. Altogether, this book offers a comprehensive introduction and survey of graphics programming techniques for the PC and its compatibles.

The book comprises some 14 chapters, organized around five thematic sections (hardware and system introduction, simple plots and color graphics, data transformations, three-dimensional displays, and application of graphics software tools). The authors have included more than 100 sample programs and 16 four-color illustrations.

The book lacks some important discussions of new graphics commands available in BASIC 2.0 (e.g., the commands on screen coordinates), but its content is still very useful. The book is intended for novice graphics programmers but also will serve more technically erudite audiences as a handy graphics reference work. Another potential use for this book will be as a college textbook for a semester-long course in microcomputer graphics. The writing style manages to avoid the typical academic dryness of such technical discussions.

Hearn, Donald D., and M. Pauline Baker. *Microcomputer Graphics: Techniques and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983, $18.95, 272 pages.

Written by the same team as the book above, this text takes a more general approach to the subject of microcomputer graphics. Besides the IBM PC, the authors discuss personal computers from other vendors (including Apple, TRS-80, Atari, and Commodore) and offer some valuable comparisons of the graphics capabilities of each machine.

This book is a valuable companion piece to the authors' PC-specific book. It is written in a more brisk, informational style than the previous book. The text is at its best when it combines programming examples (again, BASIC is the chosen language) with illustrations of the appropriate graphics screen output. The chapter on how to create interesting three-dimensional objects on the screen (e.g., chess pieces) is easily the best in the book.

The authors have aimed this book at a mixed audience of nontechnical novice programmers and of those more experienced in programming microcomputer graphics. The discussion of graphics applications (e.g., business graphics) is enlightening. In short, this is a well-written, easily accessible book that offers serious discussion of the graphics topics at hand.

Morgan, Christopher, and Mitchell Waite. *Graphics Primer for the IBM PC*. Berkeley, CA: Osborne/McGraw-Hill, 1983, $21.95, 430 pages.

These two old hands in the computer book field have turned out yet another important work about the IBM PC. This time the subject is high-resolution color graphics. They have produced an impressive, color-filled text that should be useful to a variety of readers.

The authors have written this book in a tutorial style, with copious examples to illustrate the graphics programming concepts being taught. Just about every kind of graphics effect that you can produce with the IBM PC color-graphics adapter is covered. Animation, charts, maps, business graphs, forms, games, and three-dimensional images are all discussed, and programming examples (BASIC) are provided. Accessing the graphics video chip of the color-graphics adapter is also discussed fully. I found this to be the most comprehensive PC-graphics programming book reviewed for this article. It is useful as a tutorial and as a reference work.

Traister, Robert J. *Graphics Programs for the IBM PC*. Blue Ridge Summit, PA: Tab Books, 1983, $15.50, 243 pages.

Once again, PC screen graphics are discussed from the point of view of the BASIC programmer. Unlike other books about PC graphics, Traister discusses text graphics for the IBM monochrome display as well as the standard chapters on color graphics.

The author's writing style is best termed "chatty." This makes the material easy to understand, but it limits the intended audience of the book as well as its usefulness as a graphics reference. The tone of this book may annoy more advanced, technically oriented PC users and programmers. In a 243-page book, the author needn't take almost two thirds of the pages to get to the point of the book, as Traister does here. Once he arrives, though, he is definitely on target. He gives you competent discussions of animation, filling regions with color, and drawing shapes on the screen.

I'd like to see more emphasis on the printed aspects of graphics output in future editions, as well as a greater use of color illustrations. Finally, the author's stated aim of allowing you to implement the "full graphics potential of your IBM PC" is simply not achieved.

DATA COMMUNICATIONS

Connecting PCs to a variety of other computing and database resources is becoming an increasingly important use of personal computers. The accessing of remote mainframes and minicomputers, public-access data networks (e.g., CompuServe, The Source, MCI Mail, etc.), *and other microcomputers is one of the most rapidly growing fields in an industry noted for rapid growth. A number of logistical and technical considerations are important in connecting a* PC *through a telephone to another machine.*

The few books listed here cover many important related issues, including the technical aspects of data communications (parity, data-transmission rates, handshaking protocols, and the like), the kinds of terminal-emulation and file-transfer protocols (like Kermit *and* XMODEM) *that can be used to upload and download computer files, and a discussion of the remote services available through data-communication hookups with your* PC.

(continued)

Derfler, Frank J. *Microcomputer Data Communications Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1982. $17.95. 129 pages.

This book is not a specific work for the PC owner, but it gives you the vocabulary and background information needed to explore the intricacies of PC data communications. The book is, overall, a well-written effort, although it does suffer from a clear lack of focus.

Microcomputer Data Communications Systems is really intended as an informational reference book rather than a textbook overview of microcomputer-based data-communications issues. If you use it this way, it will supplement nicely the other books available on PC data communications. Particularly useful are the discussions about communications terminology. The book suffers, however, from a cramped design and a small, hard-to-read typeface.

Glossbrenner, Alfred. *The Complete Handbook of Personal Computer Communications*. New York: St. Martin's Press, 1983, $14.95. 325 pages.

Although not a technical overview of data communications for the PC owner, this book covers the logistics issues of PC communications quite well. In 12 chapters crammed with information, Glossbrenner gives you all the initial information you'll need to select communications equipment for your PC (modems, communications software, etc.) and a remote data or access service to connect with.

The author covers the important services, including The Source, CompuServe, The Dow Jones News/Retrieval Service, Dialog, NewsNet, and a number of others, including public bulletin-board systems. Technical and cost information are both provided.

The writing is well done, combining a well-developed structural approach to the contents and an information-packed format. The information can be used by technical and nontechnical microcomputer users. Intended as a first-line introduction to the topic, Glossbrenner's book also has considerable utility as a reference book. Most PC owners who contemplate telecommunication connections should own this book. Because the book was published in 1983, a revised, updated edition would be a good idea.

Schwaderer, W. David. *Digital Communications Programming on the IBM PC*. New York: John Wiley and Sons, 1984. 224 pages. $16.95. Also available with a disk of BASIC program examples for $47.90.

With a good group of BASIC programming examples, Schwaderer shows you how to program your PC for a variety of data-communications functions. It's an all-inclusive book covering the programming techniques required, and it serves as a reference book about data communications on microcomputers.

The book covers modems, communications protocols, various communications applications, and miscellaneous communciations hardware. Although written as a self-paced instruction book, the material is organized in such a way that the book also becomes an important reference book once you've mastered its instructional objectives. The writing style is clear prose, without a lot of jargon. The book is aimed at the programmer with a fundamental knowledge of the PC and of BASIC who wants to expand that knowledge to the data-communications domain.

MISCELLANEOUS

Wesse, Rick. *Decision Making: A Management Science Guide for the IBM PC*. New York: John Wiley and Sons, 1984. $15.95. 224 pages. Also available with a disk of 34 BASIC programs illustrating the examples in the book for $46.90.

This book offers a practical introduction to using management-science techniques to solve common management problems with the PC. It starts with a concise overview of management-science methods and then moves onto BASIC program examples of forecasting, inventory, production and allocation, quality control, distribution, marketing, and project planning.

The writing assumes some experience with management-science topics and familiarity with PC operation, as well as a firm background in BASIC. The style manages to convey the information without a reliance on management-science jargon. The primary audience, however, is the technical manager who is already grounded in management-science techniques.

REFERENCES

One of the reasons we compiled this short annotated bibliography of technically oriented books for the user or owner of the IBM PC and its compatibles was the lack of any other current annotated list of such books. There are, however, several sources for further information about IBM PC books, as well as microcomputing books in general. We have listed a few of the most important sources below.

Bowker, R. R., and Company, eds. *in Print*. New York: R. R. Bowker Company, 1985 (published quarterly). Six volumes. Hardcover. $215. Number of pages varies with each issue but is usually greater than 8000.

The standard reference books for tracking newly released books, these works are usually available at most college, university, and public libraries, as well as at major bookstores.

Bowker, R. R., and Company, eds. *Bowker's Complete Sourcebook of Personal Computing 1985*. New York: R. R. Bowker Company, 1985. $19.95. 1050 pages.

This book contains more information than just book listings, and it is an invaluable sourcebook to have around. This issue lists about 5000 titles in print (most softcover) on microcomputing topics (about a quarter of these titles are related to the IBM PC).

Bowker, R. R., and Company, eds. *Computer Books and Serials in Print 1985*. New York: R. R. Bowker Company, 1985. $59.95. 660 pages.

This is a reference book of the highest order, listing vital statistics of all computer books and serials in three cross-referenceable categories: author, title, and subject. A handy listing of all publishers in computer-related fields is included in the back of the book.

Norton, Peter. *The Peter Norton Programmer's Guide to the IBM PC*. Bellevue, WA: Microsoft Press, 1985. $19.95. 426 pages.

This book is a solid reference manual for the IBM PC owner who expects to tinker with the hardware and experiment with assembly-language programming. Norton gives you an overview of the hardware components of the entire PC family: PC, XT, AT, Portable, and PCjr. He covers floppy disks, the ROM BIOS (including the support for video and graphics), program linking, DOS functions, peripheral device drivers, and the whole gamut of system support software.

The *Programmer's Guide* is a comprehensive reference work for the entire PC family. Although there is no glossary, the index is thorough enough to allow you to specifically search just one bit of information at a time. The scope of this book is so broad it covers the internals of the whole of the IBM family, including languages, service summaries, illustrative charts, DOS information, ROM BIOS services, sound, and three appendixes. Scattered throughout the text are mentions of the differences of the AT.

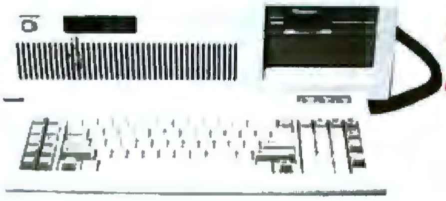
The chapters on the ROM BIOS for the PC family are particularly useful for the application programmer who is trying to write programs that will run on all IBM PCs. Using special symbols to indicate a passage about a particular machine, Norton's exposition can keep you from writing code on a PC that works differently on a PCjr.

The Programmer's Guide to the IBM PC also contains numerous code examples in both assembly language and high-level languages like BASIC. ■

PRICES AND MACHINES THAT OUTRUN THE COMPETITION.

PC'S LIMITED AT™

High Performance Competitive Price



$1995

Includes: 80286-based System Unit, 1024K on Mother Board, 1/ 1.2 Meg Drive, Combined Floppy and Hard Disk Controller Card, AT Keyboard, 192W Power Supply, 2 Serials and 1 Parallel Port, and Clock/Calendar with Battery Backup, all on Mother Board.

Runs all Major Software written for the IBM PC™, PC XT™, and PC AT™. (Processor) Intel 80286 running at 6MHZ. (Expansion Slots) :8. 7 are available in above configuration. Same Bus Configuration as IBM PC AT™.

PC'S LIMITED TURBO PC™

High Performance Competitive Price

$795

Includes: System Unit, 640K on Mother Board, 1 360K Drive, Keyboard, 135W Power Supply.

Runs all Major Software written for the IBM PC™ and PC/XT™, 40% faster, without modifications. (Processor) 16-bit 8088-2, 4.77 or 6.66MHZ Clock Speed. (Expansion Slots) :8; 7 are available in above configuration.

SOLVE YOUR POWER PROBLEM.

XT™ POWER 135W

$89

Fully XT™ compatible.
One Year Warranty.
Directly replaces power supply in PC.™

25 or more $75 each

150W $119

DISK DRIVES

TEAC $95
55-B, Half-Height, DS/DD

MITSUBISHI $85
Half-Height, DS/DD

Quantities of 10 or more $75 each

PC'S LIMITED Six Function Card

w/384K $149

Two Year Warranty

- Upgradable to 384K
- Clock/Calendar
- Software included
- Parallel Port
- Serial Port
- Game Port

Irwin Tape Backup System

$595

10 or more $495 each

- Uses Floppy Controller Card
- 10.35 Meg Formatted Capacity
- Used in Compaq Deskpro.
- Half Height
- Low Power

PC'S LIMITED

OUTSIDE TEXAS, ORDERS ONLY, CALL 1-800-426-5150
7801 N. Lamar, #E-200, Austin, Texas 78752
All sales-related calls inside Texas — (512) 452-0323.
All calls for technical support, phone (512) 452-0372, 452-0373 or 452-0374.

Telex No. 9103808386 PC LTD

Ad number 410 i

Compaq, TEAC, IBM, Irwin, and Intel are trademarks of their respective companies. All brand names are registered trademarks. We are an independent sales organization.

Public-Domain Utilities

Build an extensive software library for free

BY JON R. EDWARDS

THE EXTENSIVE public-domain collection for the IBM Personal Computer and compatibles is a very valuable resource. It is easily possible to build an extensive software library and incorporate the utilities into your home projects or to save considerable time and effort by installing a RAM (random-access read/write memory)-disk and print spooler. Most programs in the public domain provide source code; you can learn from the code and, more important, you can customize the routines for your own requirements. Undoubtedly, some of the software will fill your needs, and the more obscure programs may simply trigger your imagination.

The notion that "free means shoddy" does not necessarily apply to this software. I suspect that most of the free utilities were originally written to fill individual needs and as part of the "hacker ethic" have been shared with the public. The programs adequately fill many needs, and they have a tendency, as the user community modifies and expands them, to become more and more bug-free and sophisticated. Most public-domain programs provide limited functionality, and their user interfaces and documentation are generally less polished than commercial products, but it is amazing how many commercial products do very little more than integrate the capabilities of programs that already exist in the public domain. If nothing else, exposure to these programs will make you more aware of what to look for and expect from the products you buy. And who knows—in the short descriptions that follow, you may find software that's perfectly suited to your needs. At least the price is right.

Free Software

To the best of my ability, I have concentrated on free, no-strings-attached software and not on shareware or user-supported software. There is, to be sure, a growing amount of shareware for the IBM family, and much of it is excellent (see "Public-Domain Gems" by John Markoff and Ezra Shapiro, March BYTE, page 207), but the products often do not provide source code, and their authors usually request a contribution; most users legitimately feel that the products deserve financial support.

Naturally, I cannot guarantee that the software you download will function as you hope it will. I certainly hope you find dozens of interesting utilities here and that your investigations lead you to new and exciting things, but I take no responsibility if the the programs you download do nothing or turn your screen inside out.

Locating free software is getting easier and easier. There are more users groups, bulletin-board systems (BBSs), and public-domain copying services than ever before, and the

(continued)

Jon R. Edwards is a BYTE *technical editor. He can be reached at* BYTE, POB 372, Hancock, NH 03449.

ILLUSTRATED BY SHELLEY JORDON

number appears to be growing daily. In assembling my list, I have concentrated on some of the larger bulletin boards and copy services, but local bulletin boards have a surprisingly large selection; many have well over half of the programs on this list. Give them a call and see. You might not find precisely the program I've mentioned, but another program there may fill your needs or you may unexpectedly find new and exciting listings. There is in the public domain, for example, a large and growing number of .txt and .doc files that review new software and hardware, give programming tips, identify bugs in hardware and software, explain patches to existing software, and summarize on-line discussions. Download, read, and be informed.

The largest boards may require a long-distance call, but you are very likely to locate 80 percent of all the utilities. Among many, you might try Kingcomm at (713) 360-1316, The Eclectic BBS at (914) 221-2248, the Fargo Board at (701) 293-5973, the IBM PC Information Exchange at (312) 882-4227, or the Boston Computer Society at (617) 353-9312. There are, of course, hundreds of other boards with substantial collections; many of the bulletin boards maintain lists of active BBSs. Readers who do not yet have modems can still obtain many of the programs they want through their local users group or through the mail from services such as PC Software Interest Group (1030 East Duane, Suite J, Sunnyvale, CA 94088) or from the New York Amateur Computer Club (Box 106, Church Street Station, New York, NY 10008), which will ship you the programs you need for a copying fee.

Most of the utilities are available from a wide variety of sources, but some are more difficult to find because they seem to have appeared only on CompuServe. I have marked these with a [C]. I have divided the utilities into the following categories: disk utilities and DOS aids, memory and system status, keyboard assistance, text/file manipulation, graphics and screen control, application software, printer utilities, telecommunications, languages, and language aids. Happy public domaining.

Disk Utilities and DOS Aids

These utilities are among the most numerous and useful. There are RAM disks, applications to examine and modify sectors and tracks on your disks, directory and DOS enhancements, utilities for your hard disk, and a variety of utilities to test, fix, or modify your hardware.

First, a number of flexible RAM disks are available. RAMDSK16.COM and FREE.COM provide a 160K-byte RAM disk; later versions of FREE.COM automatically save data when you reset. Other RAM disks include QD.COM, which requires the Quadram board, HDS320.EXE and RAMDSK32.COM, which set the RAM disk for 320K bytes, and HDS512.EXE, which gives you a 512K-byte RAM disk. RAMDISK2.LBR lets you vary the size of the RAM disk.

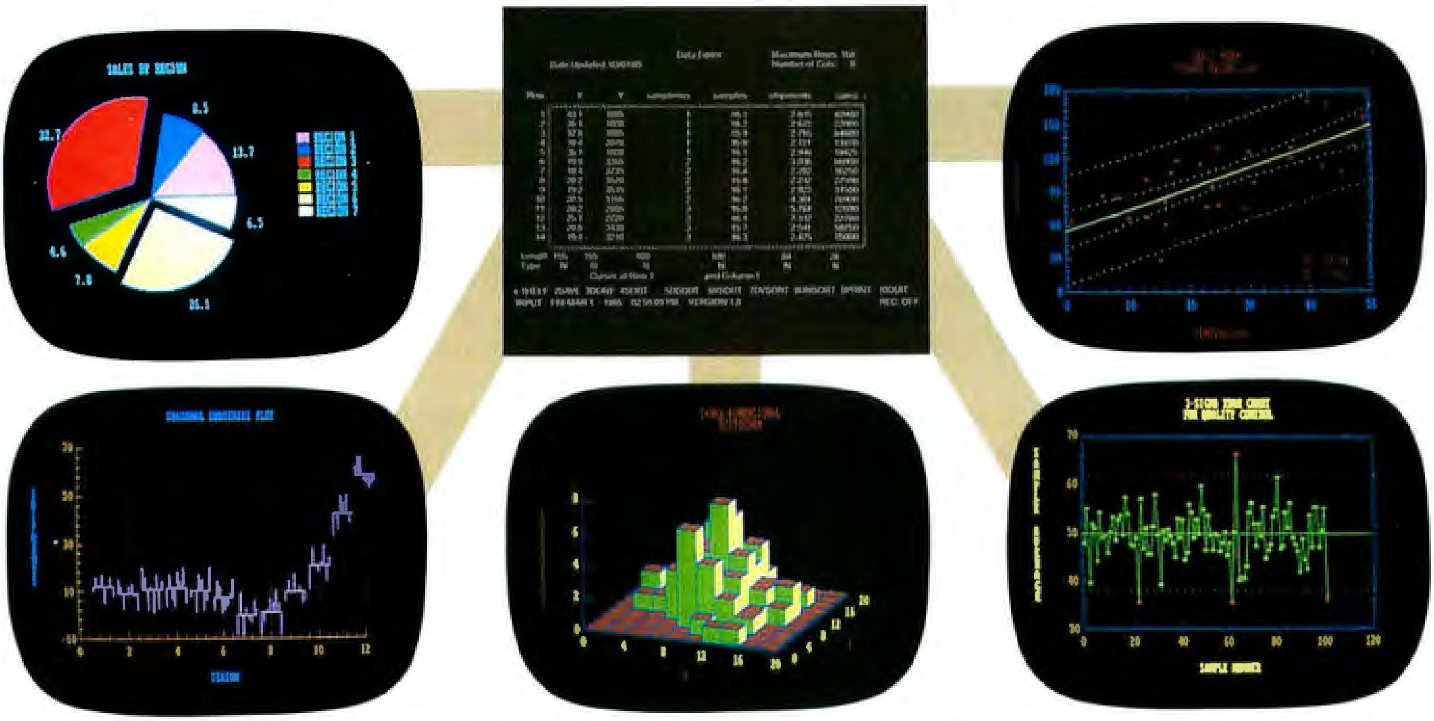
A number of utilities simplify DOS-related tasks. SHELL.COM keeps COMMAND.COM resident. FOR-

(continued)

STATGRAPHICS.®

NEW VERSION 1.1

ONE KEYSTROKE TURNS LIFELESS DATA...



INTO VITAL STATISTICS.

Data. You live with it every day. What you need are the tools to make it come alive. Comprehensive analytical tools. Mathematical precision. And graphics capabilities to broaden your insights—to help you visualize, analyze, and communicate your findings quickly and convincingly.

And now you've got it. With STATGRAPHICS—a new PLUS★WARE™ product from STSC. STATGRAPHICS is the only PC software to fully integrate a wide variety of statistical functions with high-resolution color graphics. Giving you the power and precision you used to find only with mainframe software.

STATGRAPHICS' unique interactive environment allows you to generate graphs from within the statistical procedures. You can change a variable and see the effect—immediately. So your job is easier and you're more productive.

MORE OPTIONS FOR MORE COMPREHENSIVE ANALYSIS.

STATGRAPHICS features over 250 functions for analyzing data—including analysis of variance, regression analysis, experimental design, quality control procedures, multivariate techniques, nonparametric methods, and extensive forecasting and time series analysis.

It also lets you enter data at the keyboard or access data stored in standard ASCII files, LOTUS® 1-2-3 worksheets, and DIF files used by other popular software packages.

TELL A CLEARER, MORE DRAMATIC STORY.

STATGRAPHICS includes a wide variety of graphics programs such as two and three-dimensional line and surface plots, bar and pie charts, histograms, time sequence plots, and quality control charts.

The graphics are supported on color and monochrome graphics boards, dot-matrix printers, and pen plotters for presentation quality graphics.

SOPHISTICATED STATGRAPHICS IS SURPRISINGLY SIMPLE TO USE.

STATGRAPHICS is completely menu-driven, and includes a comprehensive user's guide, online HELP screens, tutorial, and handy reference card. And STSC's HELP-Line is only a phone call away—giving you ready access to our staff of technical experts.

TAKE A LOOK AT STATGRAPHICS. AND LOOK NO FURTHER.

STATGRAPHICS is the most advanced statistical graphics software available for PC's. And the complete system is just $695.

To order STATGRAPHICS, or for more information, contact your local dealer or call **800-592-0050** (in Maryland, **301-984-5123**). Or write STSC, Inc., 2115 E. Jefferson St., Rockville, MD 20852. Major credit cards accepted.

Inquiry 188

Problem-solving at the speed of thought.

STSC
A Contel Company

STATGRAPHICS operates on IBM PC's and compatibles under DOS, Version 2.0 or later. A minimum of 384K RAM, a graphics adapter and compatible monitor, and two double-sided disk drives or a hard disk are required. An 8087 math coprocessor and 512K RAM are recommended.
STATGRAPHICS is a registered trademark of Statistical Graphics Corporation. PLUS ★ WARE is a trademark of STSC, Inc. LOTUS is a registered trademark of Lotus Development Corporation.

MAT2.BAT formats disks in the A and B drives at the same time. CAT.COM, SDIR26.ASM, SD.COM, and LF.COM sort directories by filename, extension, date, time, file size, and other criteria. XDIR displays an "extended directory," including file attributes such as whether the file is a system file, a read-only file, or a hidden file. GETDIR.PAS provides the Pascal source code for reading the directory. PARTCOPY.EXE copies files on double-sided disks to two single-sided disks. GDEL.EXE and PURGE.COM allow global file delete with yes/no verification. VCOPY.COM is a similar global file-copy utility. FDATE.LBR allows you to change the file dates. RO.COM marks files as read-only. RW.COM marks files as read/write. CV2.COM and VOLSER.COM can write or change the volume label on your disks. QDCOPY.DOC explains how to modify DISKCOPY in order to eliminate the prompts, and DCBA.HOW explains a patch to DISKCOPY for those who need many copies of a disk. DSKPAT.TXT |C| explains how to patch DISKCOPY and DISKCOMP for large-memory PCs. FILECMD2.BAS provides several DOS commands from BASIC. CLS.COM allows you to clear the screen from the DOS prompt.

DISKMOD.BAS, DISKMODF.BAS, and DISKREAD.EXE allow you to examine and modify disk sectors. Pascal source code is available for the latter. DISKRTN.EXE and HIDEFILE.BAS are dedicated to the task of examining and modifying the disk directory. Use them to recover lost or deleted files. DIRSORT1.BAS permanently sorts the disk directory. Finally, several programs provide a "switch" from DOS to different graphics and text modes. COLOR40.COM and COLOR80.COM switch to the 40- and 80-column color-graphics text modes. MONO.COM switches from DOS to monochrome 80-column text. COLORG.COM switches to monochrome medium-resolution graphics.

Several programs provide some measure of file security. DIRHIDE.LBR hides and "unhides" directories. CRYPT.EXE encrypts text files. PWORD.BAS requires that you enter a password before using the system.

There are several utilities for hard-disk users. FF.EXE and WHEREIS.LBR search through all subdirectories for a specified file. If there is more than one copy of the program, WHEREIS.COM will identify the directories in which you can locate the file. FORMAT.FIX prevents the accidental formatting of a hard disk. IBU.LBR is an incremental hard-disk backup utility. BACKSTAT.EXE checks the status of hard-disk files that have not yet been backed up. RED.COM and MOVEFILE.LBR move files from one subdirectory to another. DIRENAME.COM renames subdirectories. ARAQ.EXE erases files across subdirectories. DIRCTY.COM provides sorted directories of your subdirectories. LF.COM displays the files on the current subdirectory. MENU.LBR is a hard-disk menu system. DISKPARK.LBR places hard-disk heads over the shipping cylinder.

Most bulletin boards have "squeezed" files that occupy less disk space and download more quickly. To use these files, which are usually marked with a Q in their extension, you will need to use an "unsqueezer." Most services

(continued)

WOULD YOU BUY A PACEMAKER FROM MATTEL?

IBM/PC-based LOGIC LINE-1™ artificial intelligence software is so advanced, that competitive products are toys by comparison.

There aren't very many who'd stake their lives on a toy-store pacemaker. Luckily, we won't have to worry about their rancid cells polluting mankind's gene pool very long anyhow. Such brain-damaged geeks tend to die young.

If you've recently spent money on artificial intelligence software, you might be wishing a few programmers had croaked before writing that blithering swill they named AI and palmed off onto you. What they call an "inference engine" is nothing more than an IF-THEN decision tree that can't even do a very good job of arithmetic*

We're Clarity Software, and we're introducing a product that can take a massive amount of text from any machine-readable source, and mathematically distill thought processes for query and analysis. This process, as distinguished from *expert systems*, is referred to as *natural-language intelligence*.

In this ad, we're going to explain to you why you'd be crazy not to have our package in your portfolio of data-manipulation assets.

DERIVE THE DNA/RNA OF ANY WRITER'S THOUGHT PROCESSES.

LOGIC LINE-1 was the result of the interaction between a couple of cyberneticists and a physicist, with combined experience in high-performance learning and pattern-recognition programming. The physicist was responsible for one of the first DNA/RNA-tracking systems (the *RNA-of-thought* assertion is more than just an advertising creation). We are not your ordinary bunch of yahoos.

Imagine having the collected thoughts of Voltaire online. If you were interested in viewing Voltaire's thoughts on "job security," you would enter that term in the search menu.

Now you're thinking: "Nuts! These yahoos are trying to sell me something my supposedly *toy* text editor can do with a search command. Right?"

Wrong, pussycat. Your inference process was a little quick on the trigger. Never, in any of Voltaire's writing, was there ever the phrase "job security."

"Ok," you reply. "You have a dictionary of synonyms, eh?" Wrong again. LOGIC LINE-1 has no dictionary. Interesting?

Essentially, LOGIC LINE-1 uses a series of mathematical transformations on text, the output of which is cataloged in a database analogous to a biological DNA/RNA imprint of that text.

There are approximately one dozen parameters that make up a *thought's DNA/RNA*. Some transformations fingerprint syntax patterns; some look at subject/predicate relationships via a small dictionary of several dozen *noise* words.

After setting up the above Voltaire "job security" query, LOGIC LINE-1 will present you with high-possibility "hits." You will type "Y" when they are relevant, and "S" for skip, when they are not.

The first several "hits" might be rejected, since the term "job security" will not be found. Once you get an acceptable entry, however, and lock onto an acceptable *RNA-of-thought* pattern, the accuracy of LOGIC LINE-1 will be staggering. Or we'll refund your money. Simple enough?

"I'M NOT INTO VOLTAIRE," YOU SAY. "WHY DO I NEED LOGIC LINE-1?"

How would you like to be able to turn any textbase into an expert system? For example, most PC users rely upon word processing. The problem is, we store our correspondence in files with names like "LT062185" or "REJECT21" or "RANDOMTH." As a result, we reinvent the wheel with each letter we compose. Why do this?

Using LOGIC LINE-1, you can append all your written correspondence into one textbase file. Sure, any word processor could do a straight search for a term like "quality." But none can do this search as fast as LOGIC LINE-1. And none could turn up the "quality" references if queried for the term "workmanship!"

Thus, when writing letters and speeches, use LOGIC LINE-1 to collect previously articulated thoughts on any given subject. Whatever your profession, be it law, medicine, engineering or information management, LOGIC LINE-1 is an indispensable tool for true thought processing.

Now that thousands of textbases are publicly available, LOGIC LINE-1 is the key to their intelligent use. Once the computer establishes associative links in a large body of material, many creative applications are possible.

TRUE AI! CLONE THE GREAT MINDS ON YOUR PC.

Did Bacon write Shakespeare? Did Albert Einstein have anything relevant to say on the subject of "generation gap?" How many congressmen consistently contradict themselves in the *Congressional Record?*

The uses of LOGIC LINE-1 are limited only by the availability of textbases, and by your imagination.

Do you dare send a student to college without LOGIC LINE-1 and a PC? Would YOU care to compete in business with someone else who has this leading-edge, decision-support software?

Einstein should write your papers on relativity. Alfred North Whitehead should write your papers on philosophy. And Shakespeare should write your love letters.

1985'S SPECIAL $100 LIMITED-TIME OFFER WILL BE HISTORY IN 1986.

This is our very first, and only, announcement ad. For the rest of this year, you may obtain LOGIC LINE-1 for only $100 (note, we didn't try to fool you with a $99.95 price tag; we *do* give you, as well as your computer, some credit for intelligence). As of January 1, 1986, LOGIC LINE-1 will retail for $250.

To find out more about LOGIC LINE-1, call 216/729-1132. You may order by phone by calling 800/621-5839 (or in Illinois, call 800/972-5855). Return the product for any reason whatsoever within 30 days for a full refund. If you are with the US government or one of the intelligence *spook shops*, LOGIC LINE-1 is available to federal/DOD buyers through IBIS Corporation, 131 Elden Street, Herndon, VA 22070 (call 800/532-3344 or 703/478-0300).

You COULD spend your next $100 for other software. You could also ask Weird Al Yankovic to install that terrific new pacemaker from your kids' *Young Doctor* toy kit. Call today and save more than just money.

CLARITY SOFTWARE.

A Div. of EXPANSION PGMS INT'L, Inc.
PO BOX 839 • Chesterland, OH 44026 • USA
(216)729-1132

*The only other package out there worthy of the label "artificial intelligence" is from Teknowledge in Palo Alto, and we don't compete with them. They build "expert systems," while our emphasis is on "natural language" systems. We mention them in passing, because they're one of the few "good guys." Copyright © 1985 by Clarity Software. LOGIC LINE-1™ is a trademark of Clarity Software, and requires a 128K IBM PC with at least one diskette drive and runs under MS/DOS 2.1 and up. IBM™ IBM Corp. MATTEL™ Mattel. We hope the swell people at Mattel can take a joke. Advertising & PR by TRBA, 408/258-2708.

provide them, and it may be necessary to use the utility recommended by the particular board from which you acquire your software. Unsqueezers include USQ.EXE, NUSQ110.COM, and ALUSQ.COM. Much of the software also appears in library files, easily recognized by the .LBR extension. To use these files, you may have to extract the files and then unsqueeze them. LSWEEP13.COM and NULU.COM are flexible utilities that allow you to perform both operations at the same time.

Other utilities include CLEAN.COM, CLEANDRV.BAS, and WASH.COM, which exercise the disk during the use of disk-head cleaners. SPEEDUP.COM purports to speed up disk I/O. TWOSIDE.COM allows you to address two double-sided drives as a, b, c, and d. There are a number of disk-drive maintenance programs, including DRIVETST.BAS and DSKTST.BAS; they can determine the speed of your drives and if your disks have bad sectors. ALIGN.BAS, a disk-head alignment utility, provides control over disk-arm movement.

Memory and System Status

Many public-domain programs, including LOOK.COM, SCREEN.PAS, MEMDUMP.BAS, and CORELOOK.COM, examine the contents of specified addesses in memory. SYSTAT.COM and PCS.COM display general system status. EQUIP.COM and CHECKOUT.BAS assess your serial and parallel interfaces and equipment. 8087TEST.EXE checks the 8087. REGDISP.ASM displays 8088 registers in real time. MTEST.COM detects PC memory errors.

JOYSTEST.BAS is one of many joystick testers. LPENTEST.BAS checks your light pen. SEESWITCH.BAS displays the computer switch settings. DIPSET.BAS lets you configure them. MEMORY.COM sets memory size without your having to alter the switch settings. PARITY.COM disables parity checking, and both PARINT.COM and PARCHK.LBR provide a parity-check intercept handler. With the latter two, you can save your files before testing the system for the problem that caused the parity check. PARBNK.HEX, a modification of PARCHK, reports the bank number that had the error. Finally, MEM640.ZAP modifies the BIOS to allow 640K bytes of memory.

Keyboard Assistance

There is a general assortment of utilities that provide keyboard assistance. A number of routines, including BASICKB.EXE, KEYBOARD.EXE, and PCKEY.COM, purport to provide collections of keyboard utilities. Several programs, including KYBD.BAS, output keyboard input to the printer like a typewriter. Other programs, like BUFFER.COM, KBD_FIX.COM, BUFF159.COM, and KBBUFF.BAS, extend the keyboard buffer. CED, for command editor, provides a command stack in DOS and allows you to edit DOS commands with the cursor keys. QUICKREF.BAS creates templates for keyboard reference.

Many programs provide extended keyboard control. BIGANSI.SYS, an enlarged version of ANSI.SYS, lets you

(continued)

redefine up to 40 keys. ANSI2K.LBR is another form of ANSI.SYS with a 2K-byte buffer. KBRATE.COM and SPEEDKEY.COM change the rate at which the keys repeat. The latter also claims to speed keyboard I/O. NEWKEY.BAS reassigns the 10 function keys, while FKREST.COM resets the keys to their original functions. FKEYS.LBR redefines the Alt–function key combinations. KEYSBAS.BAS and OPKEYS.BAS program the function keys for BASIC programming. SETKEY.EXE and DEFKEY.COM are keyboard reassignment programs; ORIGINAL.KEY contains the original keyboard configuration. CNTRLBK.BAS demonstrates how to disable Ctrl-Break. SHIFTBS.COM converts the backslash key into a left shift key; SHIFTIBM.COM converts it back. KEYLOC.EXE and KEYHELP.COM make the shift, Alt, and Ctrl keys into toggles, a useful feature for handicapped users. FLIP.COM, UPNUM.COM, and KEYFLAGS.COM provide further control over the Caps Lock and Num Lock keys.

TEXT/FILE MANIPULATION

You may find some useful public-domain utilities if you frequently have to convert from one text format to another or if you commonly need to manipulate text files. TXTPRO.EXE, ASCFILTR.BAS, and FILTER.BAS are filters that remove control and non-ASCII characters from ASCII files. Use them, for example, to remove control characters from downloaded files. CR.BAS filters BASIC files. ADDCR.BAS and CRETURN.BAS, other useful programs for manipulating downloaded files, add carriage returns to the end of every line in a text file. WS-FIX.COM, WS-ASCII.BAS, UNWS.EXE, and many other programs strip the high-order bits from WordStar files to convert them to ASCII. WS-INDEX.COM indexes WordStar files. EZW2-ASCI.BAS converts EZWriter files to ASCII.

CHOP.LBR sections large text files. ADD-LF.BAS adds linefeeds to files that contain only carriage returns. JUSTIFY.EXE allows right and left justification of text files. DETAB.EXE expands tabs in a file to a specified number of spaces. LOWER.COM and UPPER.COM convert text files from and to lowercase and uppercase. WC.EXE counts the number of words in a text file. DOCANAL.LBR (C) analyzes the line length to help you decide how to print the file. DUMPFILE.PAS produces hexadecimal and ASCII dumps of disk files. BIHEX.BAS, HEX.BAS, and HEXCONV.BAS convert files among binary, hexadecimal, and ASCII.

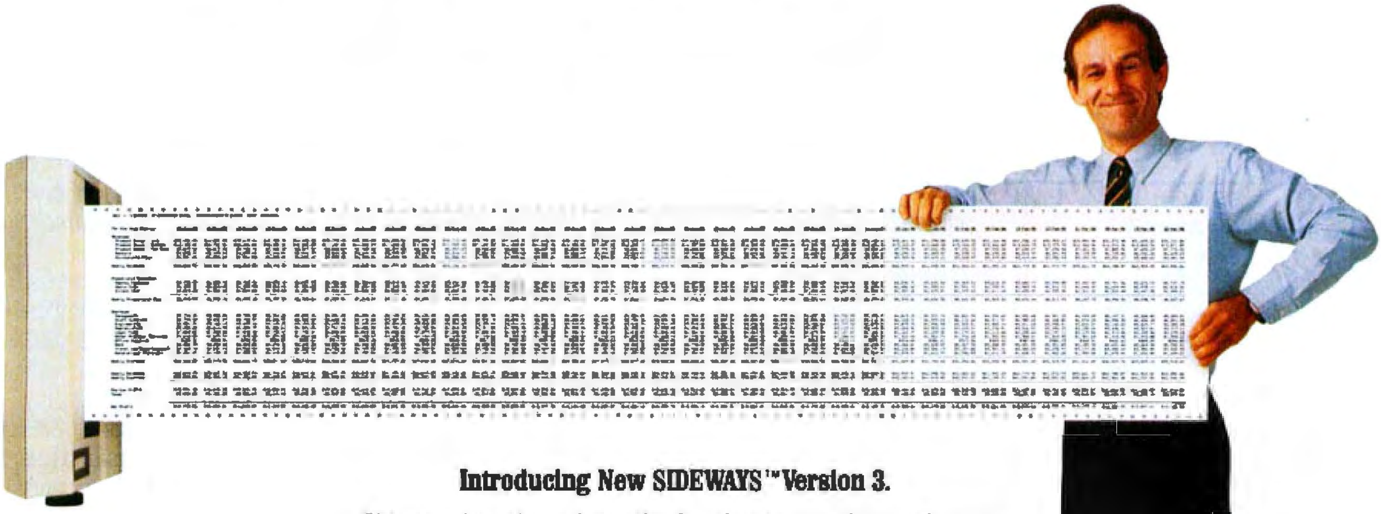
Several utilities ease string-related tasks. DEFINE.EXE is a string-replacement macro. KWIQ.LBR, FIND.EXE, and UTSCAN.EXE scan for occurrences of text strings in files; SORT.EXE sorts them; CHANGE.EXE replaces them. There are several sequential file utilities. For example, DATAFIX.BAS adds or removes line numbers, and KILLNULL.BAS removes nulls.

GRAPHICS AND SCREEN CONTROL

There are also a number of useful public-domain graphics and screen-control programs. If you're concerned about

(continued)

We've gone to great widths to make you look good.



Introducing New SIDEWAYS™ Version 3.

Sideways has always been the favorite way to print nearly everything that's too wide for a printer. Not surprising.

It's quick. It's easy. And it makes clean, one-piece printouts as wide as you like. So you get spreadsheets that really spread out. Pert charts that peer far into the future. All kinds of extra-wide reports without staples, glue or tape.

But now Sideways Version 3 goes even further.

Especially for 1-2-3™ and Symphony.®

Now you can make short work of those wide 1-2-3 spreadsheets. No need to make print files. Just highlight a range and print right from your worksheet file using familiar Lotus-like menus.

Mix bold, underlined and expanded type, all on the same page. Print through or skip over perforations. Make the finest extra-wide printouts this side of typesetting.

Using Symphony? Sideways chimes right in to print any worksheet you've loaded without ever leaving your Symphony session.

No matter what software you use.

Spreadsheets, databases, word processors, project planners. Whatever you use, you'll get more out of new Sideways.

Choose from nine different type sizes, from "minuscule" to "mammoth." Print bold, underlined or expanded – whatever your software can create. Even foreign language and line-drawing characters. Plus, there are so many ways to adjust margins, spacing and page size, it stretches the imagination.

Sideways Version 3 for the IBM® PC and all popular dot-matrix printers. Available now at computer dealers nationwide. Or directly from Funk Software with your check, Visa or Mastercard.

At just $69.95 a copy, you'll never want for width again.

Special upgrade offer!

Just send $20 along with your current Sideways program disk and manual directly to Funk Software. We'll send you Version 3 right away.

Funk Software, 222 Third Street, Cambridge, MA 02142. Phone (617) 497-6339.

burning out the phosphors on your monitor, get COLBLNK1.COM or MONBLNK1.COM, which blank the color or monochrome display when no keys have been pressed for about five minutes. If you are bored with the default screen, try CL.COM, which sets the background to blue, sets the foreground to yellow, and provides a block cursor, KOLOR.COM, which allows you to set the foreground, background, and border colors, or CLOCK.COM, which puts a digital clock in the upper right-hand corner of your screen. CRL.COM alters the size and shape of the cursor; NOBLINK.COM stops the cursor from blinking.

GFX.COM provides a graphics driver for color screens. MACHII.EXE is a graphics editor. MONITOR.COM toggles between monochrome and color display. The set of BASIC subroutines in MONITOR.BAS helps you to format different screen displays. AMBER.COM sets the color display to amber. EDITNO.BAS formats numeric output. ZIPCLEAR.BAS clears and SCROLL.ASM scrolls a designated screen area. RE-VIEW.LBR allows you to scroll all displays up or down. VIDEO.ASM provides a Pascal-callable routine for BIOS video interrupts. PC-COLOR.BAS is a color monitor test. OVERHEAD.BAS assists you in making overhead transparencies for display.

Application Software

A large and growing body of shareware often provides an excellent alternative to commercial software. BYTE has already reviewed much of this software, but free public-domain software is also a good source for text editors, database managers, mailing-list programs, and the like.

DBMS, U-MIND.BAS, EFS.BAS, and AUTOFILE.BAS are database systems. EDITXT.BASE (C), RV-EDIT.BAS, FSED.LBR, FULLEDIT.BAS, EDIT.EXE, FRED.EXE, and SCREEN.EXE are several of the text editors. PROOFER.BAS is a proofreader for text files that claims to learn. MAILIST2.BAS and MAIL1.BAS are mailing-list programs; the first has search and alphabetic-sort capabilities, while the second sorts indexes on four fields. PIBCALC1.LBR is a programmable calculator. BIGCALC is a calculator with 100-digit precision. PC-PAD and MINICALC.BAS are spreadsheets. SPELL.LBR is a spelling checker for ASCII text files. Several utilities, including PC-LIB.BAS, PC-DISK.PAS, DC4.LBR, and DIR201.EXE, catalog your disk library. BIBLIOGHY.LBR generates bibliographies. PRG-TIM.EXE (C) keeps a log of your user time.

CALCHK.EXE provides a memory-resident calendar. CALENDAR.LBR has a calendar and a phone book/dialer. There is also a set of book-indexing programs, INPUT.EXE, SORT.EXE, MERGE.EXE, and BUILD.EXE, all with Pascal source code. INDEX13.LBR creates a keyword index for text or WordStar files. TESTER.BAS is a mini course-authoring system; it creates question-and-answer files on any subject.

Printer Utilities

The public domain can provide you with a wide variety of print spoolers, screen-dump utilities, personalized print

(continued)

routines, label makers, print formatters, and programs to set the parameters of your printer.

There are many versions of print spoolers. MSPOOL-2.COM, for example, allows spooling of up to four printers simultaneously, supports both parallel and serial printers in any combination, has a user-definable buffer up to 63K bytes in 7K-byte chunks, and permits the immediate cancellation of ongoing print jobs. Other spoolers include SPOOLER1.COM for monochrome display cards and SPOOLER2.COM for color display. QSPOOL.COM provides a print spooler for those with a Quadram board.

Many programs dump screens and graphics to the printer. GRAFTRAX.COM, for example, permits screen dumps for Epson, NEC, and C. Itoh printers. GDUMP dumps medium-resolution graphics to the C. Itoh 8510 and the NEC 8023. OKIDUMP.COM converts graphics for Oki printers. PRINTGR1.BAS and PRINTGR2.BAS print medium- and high-resolution graphics upright and sideways. QUIK-GRAF.EXE allows those with color graphics cards to use their printers to make graphs. FASTPRT.COM speeds the print-screen function, and SERIAL.BAS allows the PrtSc key to function with a serial printer.

If you have recently purchased a new printer, there are several programs that aid installation. IBMPRINT.BAS and EPSON.BAS set options from a menu for Epson printers; FX_SETUP.BAS and SETPRTR.C can set the options for your FX-80 and MX-80. The first of these can also load user-defined characters. CPRINT.BAS and GPRINT.BAS do the same for the C. Itoh 8510 and some IBM printers, respectively. GEMINI.BAS works for Gemini printers. WordStar users may find useful DM.BAS and LQ.BAS, which set up dot-matrix and letter-quality printers under WordStar.

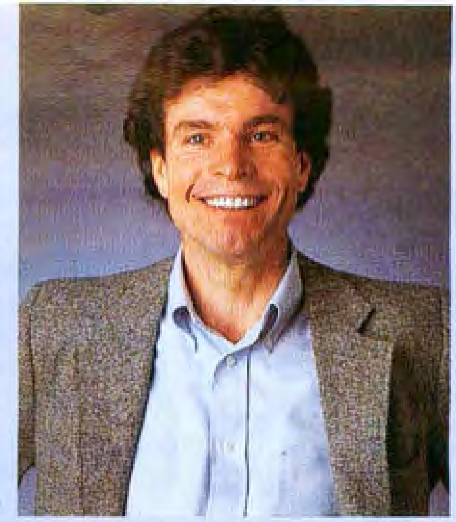
There are a variety of other utilities. TWRITE.COM turns your PC into a typewriter. PRINT.BAS and NICELIST.BAS print ASCII files with title banners and page numbers. PRINTNEC.BAS functions like PRINT.BAS for the NEC 8023 and the C. Itoh 8510. NECLABEL.BAS and LABELS.BAS print mailing or other types of labels; the first of these works for the NEC 8023A. SPOOLDSK.LBR and VPRINT.COM redirect printer output to a disk file, a useful program if you have an application that will print specific formats but will not save them to the disk. SERIAL.BAS routes printer output to the serial port. SWAPPRTS.LBR and LPT12SET.BAS toggle between LPT1 and LPT2. QSWAP.COM does the same thing for those with Quadram boards. ADDRFLCD.BAS and 3BY5.LQR allow you to print text on 3 by 5 cards. BANNER.BAS prints sideways banners. COVER.LBR prints directory summaries in condensed print for disk jackets.

TELECOMMUNICATIONS

Not surprisingly, there are numerous communication systems in the public domain, and virtually every bulletin board and users group can provide you with sufficient capabilities to send and receive files, to set up your PC in terminal mode, or to act as a host computer.

(continued)

Let David Rose tell you about our powerful new ProKey 4.0.



David Rose, inventor of ProKey.

Spend 2 minutes learning how to add a powerhouse of function to your keyboard with a terrific new version of ProKey!

"Suddenly others have discovered what RoseSoft has known since we invented the ProKey concept. If your keyboard could work a whole lot harder, your life would be a whole lot easier.

The original idea behind ProKey was a program that would let you store strings of keystrokes you could recall at a touch. Now there's ProKey 4.0 and it's better than ever."

ProKey 4.0 keyword macros unsnarl PC use.

"If you're a serious PC user, you probably use a variety of software programs. Spreadsheet. Word processing. Database. Communications. Plus the special-function programs you use to run your business or department.

And all these programs have different complicated commands that are tedious to type and easily forgotten. How much easier life would be if all you needed to operate your program was a small vocabulary of sensible words.

Want to log onto Dow Jones with your communications package to check on how your IBM stock is doing? Just type the keyword you previously selected for this function, "IBM". Need to print a profit and loss statement? Typing "p&l" is all it takes.

Behind the scenes, ProKey 4.0 checks its memory for the string of commands required to carry out the task and executes them automatically. Forgotten what functions you have available? Just call up the built-in menu that describes them in plain English.

Need to add a function? Enter the commands. Enter a description. Store them both away without leaving the program you're working with.

Built-in editing, English language menus, multi-character macro names and the ability to instantly change sets of functions. These are just four of the exciting new features that make ProKey 4.0 the most powerful macro processor you can buy."

ProKey 4.0 lets you customize your keyboard.

"Only ProKey has the unique Layout program that lets you redesign your keyboard to correct design flaws the manufacturers built in. You can change the location of any key, create a DVORAK layout, or make a duplicate set of arrow keys away from the keypad."

ProKey Is the Industry Standard.

"ProKey has all the features you need to make your life a whole lot easier. And RoseSoft has three years experience supporting customers who use macros. Including over 80% of the Fortune 100.

Don't be misled by the recent claims of ProKey imitators; they're comparing themselves against old versions of ProKey.

The choice is simple. Do you want an unknown, untested product in its first generation, or ProKey 4.0's power and features? The difference in price will hardly pay for dinner. The difference in performance is like night and day."

ProKey 4.0 runs on IBM and other compatible MS-DOS computers, and works with 99% of the software they use. $130 suggested retail price at software dealers everywhere. Or call RoseSoft. (206) 524-2350.

RoseSoft™

4710 University Way N.E.
Seattle, WA 98105
(206) 524-2350
Inquiry 174

MODEM.COM, MODEM7.COM, and XMODEM.COM, rewritten versions of MODEM7 for CP/M machines, incorporate Ward Christiansen's XMODEM protocol. Other systems include QMODEM.COM, which uses windows, MINITEL.EXE, BCOM.EXE, and IBMODEM.BAS. CIS.EXE and CSDEC.COM are terminal programs that specifically support CompuServe protocol file transfers. KERMIT.EXE is the Kermit Communications System that includes the sophisticated Kermit file-transfer protocol developed at Columbia for downloading files from mainframes to micros (see the two-part article "Kermit: A File-Transfer Protocol for Universities" by Frank da Cruz and Bill Catchings, June and July 1984 BYTE). SIM3278.BAS, developed at the University of Missouri, is another PC-to-mainframe communications package. CMS.BAS is a CMS/TCO telecommunications program. GLASSTTY.PAS is a simple dumb terminal. HOST.BAS permits remote use of your PC if you have a Hayes Smartmodem.

If you're interested in starting your own bulletin board, FIDO (approximately 550K bytes) and RBBS-PC (approximately 300K bytes) provide you with sufficient software. The RBBS software has source code in BASIC and adequate documentation. The latest versions permit ring-back bulletin boards and contain menus and sign-ins both with and without graphics.

Other utilities include SEARCHER.BAS, which auto-dials up to 999 numbers. CHRONSET.BAS and COMCHRON.BAS set and read the Hayes stack chronograph. APPLECOM.BAS facilitates communications with Apple computers. ASYNC.ASM [C] drives the asynchronous port (COM1) via interrupts.

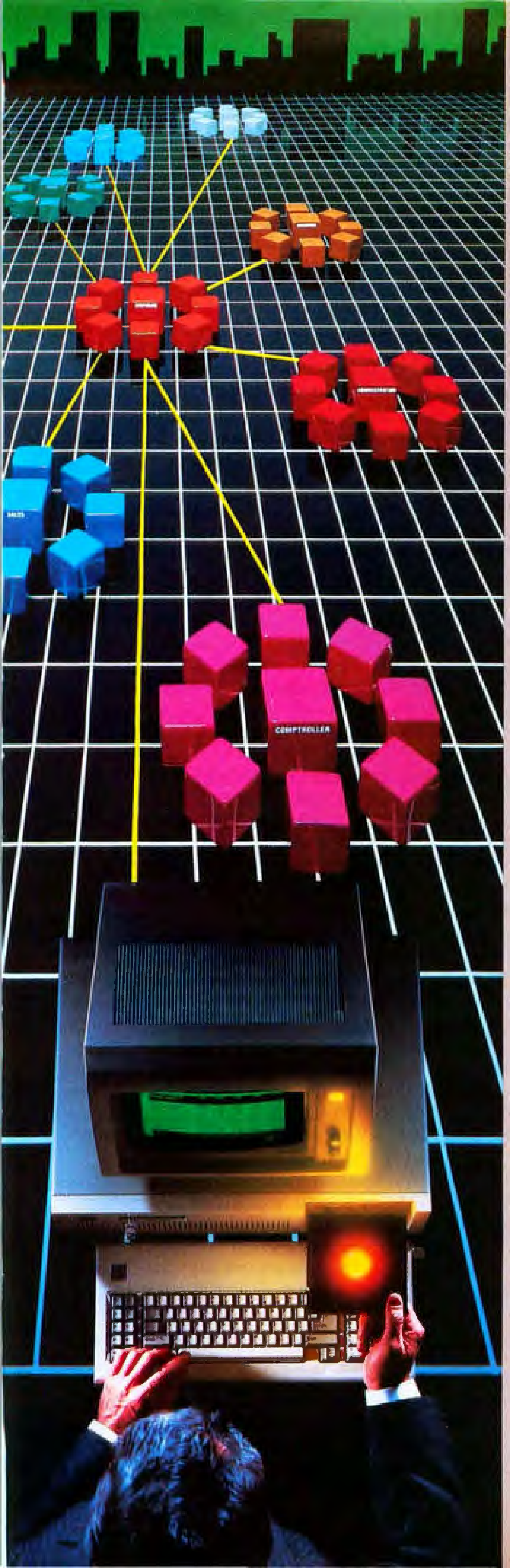
LANGUAGES AND LANGUAGE AIDS

The public domain also offers several programming languages and a large number of programming aids. For the FORTH community, Mountain View Press released MVP FORTH, including the source code, into the public domain; there are also several versions of FIG-FORTH: F83 is an 8086 FORTH, and FORTH.BAS is an implementation of FORTH in BASIC. XLISP is an experimental LISP. RATBAS is a structured BASIC preprocessor. TURTLE.LSP is an interactive Logo programmed in LISP. PIL.LSP is a micro-PROLOG interpreter written in LISP. C-COMP.LBR and CPC.EXE are Small-C compilers.

ASMGEN.LBR is an 8086/8087/8088 disassembler; it creates an assembler input file from an executable program. COM2ASM2.BAS is another disassembler. MPUBLI.C [C] reads in all your variables and helps to build a full macro file for use with symbolic debuggers. WINDOW.LBR [C] demonstrates how to use color windows in your assembly-language programs. CVTBIN.SAL [C] provides a tool for structured assembly code. Those interested in learning assembly language can also benefit from the numerous tutorials in the public domain.

There are a great many aids for BASIC programming. BASICAID.BAS provides BASIC program development

(continued)



tools. B-SIMPLE.BAS is an aid for structuring BASIC. PROFILE.MEM is a coresident BASIC optimizer. FORMAT.BAS [C] automatically reformats the screen so that words are never split over two lines. XREF, BASXREF, PCREF.BAS, and CROSSREF are cross-reference utilities. EXPLIST is an expanding lister utility. REMREM, PSQUISH, and SQUISH remove REMs from programs. SPLTSC.BAS [C] permits split screens in BASIC. SCREEN provides BASIC input routines. GS-UNNUM.EXE (and PRECOMP.EXE) and GS-RENUM.EXE remove and replace line numbers from unreferenced lines prior to compiling. EDITNO.BAS is a numeric editing subroutine. BASTODOS.BAS allows you to access DOS functions from BASIC. You can interface DIR4.ASM [C] with a BASIC program to read the disk directory into arrays of names, lengths, dates, and times. BASPRM.BAS [C] reads parameters entered at the DOS command level. INPUTSUB.BAS is a BASIC input subroutine. COLORPIC.BAS helps programmers to choose colors. LORES.LBR contains low-resolution graphics routines. MONITOR.BAS provides a full-screen menu facility. WINDOWS.LBR allows you to set up windows anywhere on the display. ADVANCED.BAS sees if a program requires BASICA. ADVBAS.LBR has advanced functions for compiled BASIC. TRS2PC.BAS partly converts TRS-80 BASIC to PC BASIC. BASTOFOR.BAS claims to convert BASIC to FORTRAN.

The advent of Turbo Pascal brought numerous programming aids into the public domain. THELP.COM provides help windows. PTOOLENT.LBR has a number of data-entry routines. PTOOLSCR.LBR provides data-entry routines. PTOOLWIN.LBR is a window routine. PTOOLTIM.LBR is a time-conversion routine. PIBMENUS.LBR is a pop-up window and menu facility. TURBOCOM.PAS provides several communications routines. CRFONTS.PAS is a color graphics font generator. SHRINK.PAS is a run-time size shrinker. XREFPAS.PAS is a cross-reference program. SCREEN.LBR and TBVIDEO.LBR have Turbo Pascal screen functions.

C users may enjoy MATRIX.LBR and STRING.LBR, which provide matrix and string operations. CC.LBR is a C debugger. XC.C is a cross-referencer. DISASSEMB.C is a C disassembler. WINDOW4C.LBR provides window routines. You can interface DIRGT.ASM [C] with IBM Pascal or Lattice C to read the disk directory into data structures. It functions much like DIR4.ASM for BASIC.

The following are some of the C utilities on disk 216 of PC-SIG. READS.C reads a string from STDIN. CHOSIT.C displays a menu and a prompt and waits for a response. SCRINIT.C intializes the screen and keyboard arrays. CURSOR.C moves the cursor to given coordinates. GETLINE.C reads a record from a file to a string. CURDOWN.C, CURUP.C, CURFOR.C, and CURBACK.C move the cursor in the indicated direction. MENCON.C opens and displays a menu file on a screen.

I hope, as you have read through this list, that you have found a number of useful utilities and aids. Look for them at your local users groups and in regional and national BBSs. Enjoy the downloads and, by all means, upload a few programs of your own. ■

From MEGAOMEGA SYSTEMS INC.

"MEMORY FULL!" That's what happens when your spreadsheet or database gets too big for your PC. So why not address the problem directly. Get **THE COMPANION CARD.™**

THE COMPANION CARD™ was designed to meet the specification jointly developed by Intel and **Lotus Development Corporation**, and more.

THE COMPANION CARD™ works with current releases of Lotus 1-2-3™, Symphony™, Framework™, and many other current programs.

MEGA-OMEGA SYSTEMS, INC., 5217 Ross Avenue, Suite 710 LB 122 Dallas, TX 75206, (214) 828-0960

FOR BOTH IBM PC™ and AT™

THE COMPANION CARD™, can dramatically increase the productive life of your IBM PC™, AT™, and compatibles. It's really two memory cards in one. It can fill in conventional memory below 640K and if you need to expand your system memory, you can do so. Up to four megabytes can be addressed beyond 640K (2 megabytes per companion card) for newly released programs such as Symphony™ release 1.1, Lotus 1-2-3™ release 2, and Framework™ (support during third quarter of 1985). It's all available as regular work space memory, so that spreadsheets can grow up to fifteen times their current size! Better yet, we know how valuable slots are in the system and **THE COMPANION CARD™** occupies only one slot.

THE COMPANION CARD PLUS™ does all this; but in addition, it can provide communications which are so vital to the system. The card comes with a serial port, parallel port, game port, real time clock, and only takes up one slot.

MORE ABOUT INTEL-LOTUS

Expanded Memory Specification

To go along with **THE COMPANION CARD™** hardware, **MEGA-OMEGA SYSTEMS, INC.** provides Expanded Memory Manager software. It's implemented as a true DOS driver making software development extremely straight forward.

THE COMPANION CARD™ also provides a unique memory diagnostics feature. Every time you boot your system, this feature checks for any defective memory chip, notifies you of their existence, and bypasses them. When the chip is eventually replaced, the Expanded Memory Manager senses it and the new memory is made available.

CONVENIENCE FEATURES

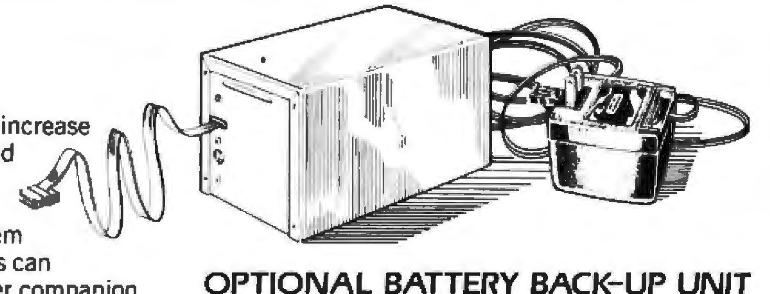
THE COMPANION CARD™ includes a print buffer, RAMdisk, and menu driven installation programs.

Print Buffer.

One key feature is the ability to pause or stop from within your application program thru the use of pop-up menus provided.

RAMdisk.

The RAMdisk can be configured as large as eight megabytes and partitioned as many as eight RAMdisks. The problem with most RAMdisk is that a sudden flicker of the lights or sudden loss of power to the system would result in data being totally lost. **THE COMPANION CARD™** comes with a unique feature. By attaching the optional battery back-up unit to the card you can virtually have a **nonvolatile RAMdisk**.



OPTIONAL BATTERY BACK-UP UNIT

Installation.

THE COMPANION CARD™ comes with a menu driven installation program that will allow you to install the print buffer to the AUTOEXEC.BAT file and the Expanded Memory Manager DOS driver or RAMdisk to the CONFIG.SYS file.

Warranty.

MEGA-OMEGA SYSTEMS, INC. offers a five year limited warranty on **THE COMPANION CARD™** and **THE COMPANION CARD PLUS™**.

Compare features and price at your favorite computer store and then buy **THE COMPANION CARD™** to resolve your Memory Full problem today.

MEGA-OMEGA™

THE COMPANION CARD™

	Suggested Retail
0K	$177
256K	$204
512K	$231
768K	$258
1 MB	$285
2 MB (Includes MXT Card)	$480

THE COMPANION CARD PLUS™

0K	$297
256K	$324
512K	$351
768K	$378
1 MB	$405
2 MB (Includes MXT Card)	$600

Includes serial, parallel, game port, and real time clock.

THE COMPANION MXT CARD™	$195

(Optional card with 1 MB installed for both THE COMPANION CARD and THE COMPANION CARD PLUS)

The BBU™ (Battery Back-Up Unit)	$177

MEGA-OMEGA SYSTEMS, INC., 5217 Ross Avenue, Suite 710 LB 122, Dallas, TX 75206, (214)828-0960

F.O.B. DALLAS • TERMS - C.O.D., NET 30 UPON APPROVED CREDIT. • Prices and terms subject to change without notice. Effective 7/26/85 • All Prices U.S. Funds
VISA, MASTERCARD & AMERICAN EXPRESS ACCEPTED

A NEW SOLUTION from MicroComputer Accessories, Inc.

THE FEED & FOLD SOLUTION

Solved at last, Watson!

THE MICROFOLD™ PRINTER STAND

What's that, Holmes?

The proliferating paper problem. Now we can feed and refold paper in a space hardly larger than our printer's footprint.

Footprint, you say?

Look under the printer, Watson. The paper tracks from that lower tray, feeds up through the bottom or rear of the printer, and refolds back underneath, into the upper tray.

Tracks? Footprints? Holmes, I'm baffled.

Very observant, Watson. These cunningly designed baffles are the secret of its precise, smooth operation.

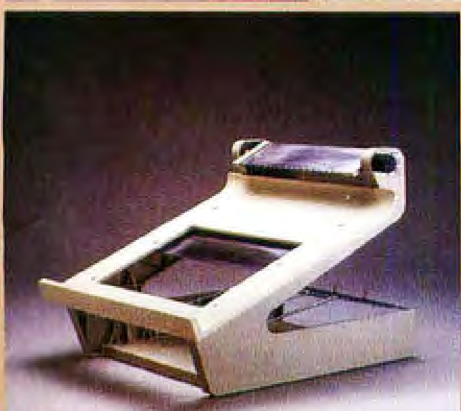
A smooth operator?

Even with its back to the wall, Watson!

Are we tracking the footprints of a smooth operator? Holmes, I haven't a clue...

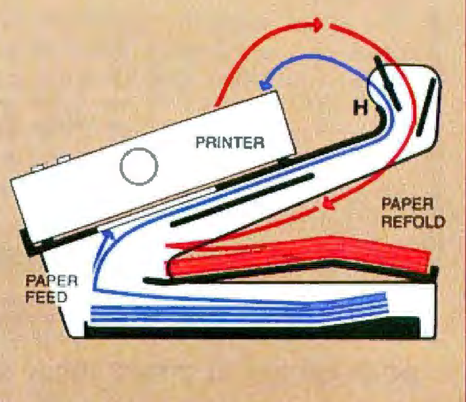
It's no secret, Watson—all other printer stands are now obsolete. Meet the MicroFold!

MicroComputer Accessories, Inc.™



Model #930
80 column printer stand

Model #940
132 column printer stand



USA: P.O. Box 3725, Culver City, CA 90231. Telephone 213/641-1800. EUROPE: Rue de Florence 37, 1050 Bruxelles, Belgique. Telephone 02/538.61.73.

Inquiry 131 for End-Users. Inquiry 132 for DEALERS ONLY.

ROM BIOS EXTENSIONS FOR THE PC AT

Excerpts from The Peter Norton Programmer's Guide to the IBM PC

BY PETER NORTON

Editor's note: The following excerpts are reprinted from The Peter Norton Programmer's Guide to the IBM PC, *published in June by* Microsoft Press *of Bellevue, Washington. The excerpts, which detail the* BIOS *extensions specific to the* IBM PC AT, *were taken from chapters* 10, 12, *and* 13 *of Mr. Norton's book. For a roundup of other technical books for the* IBM PC, *see* Don Crabb's *annotated bibliography on page* 11.

THE STANDARD ROM BIOS DISK SERVICES

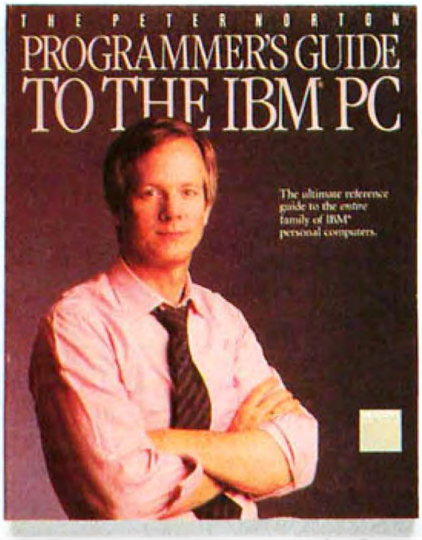
Since a disk drive can do only a few simple things, there are only six standard BIOS disk services common to all IBM PC models. The IBM PC AT, having introduced a more complicated disk drive, has added several new services to the ROM BIOS. These additions are discussed separately.

THE AT FIXED-DISK SERVICES

The AT uses disk drives different enough from the drives used in the other models that several new BIOS disk services were added. They are designed to support the high-capacity disks and the variety of fixed disks that the AT can use. We'll outline the new services here, but we won't go into any great detail. Our main concern is to explore general principles and programming practices. [*Editor's note: The hexadecimal equivalents of the interrupt and service numbers are given in parentheses.*]

SERVICE 8

Get Current Drive Parameters: Service 8 returns disk-drive parameters. DL reports the number of disk drives (from 0 to 2); DH reports the maximum head/side number; CH returns the maximum cylinder/track number; and CL returns the highest sector number.



SERVICE 9

Initialize Fixed-Disk Parameter Tables: Service 9 is used to set the disk base tables for two hard-disk drives. The interrupt vectors for interrupt 65 (41 hexadecimal) and 70 (46) are used to provide the table addresses. This service would be used only to install a "foreign" disk drive.

SERVICES 10 AND 11 (A AND B HEXADECIMAL)

Read and Write Long: Service 10 reads, and service 11 writes, "long" sectors on 20-megabyte fixed disks. A long sector includes a 4-byte error code called the ECC, which provides high-level error checking and error correction of the sector's data.

SERVICE 12 (C)

Seek to Cylinder: Service 12 performs a seek operation that positions the disk read/write heads over a particular cylinder on the hard disk. Register DL provides the drive ID, DH the head number, and CH the cylinder number.

SERVICE 13 (D)

Alternate Disk Reset: Service 13 performs an alternate drive-reset operation for the fixed-disk drives. The drive is specified in register DL. This service operates the same way as disk service 0.

SERVICE 16 (10)

Test for Drive Ready: Service 16 tests to see if the fixed-disk drive is ready. The drive is specified in register DL, and the status is returned in register AH.

SERVICE 17 (11)

Recalibrate Drive: Service 17 recalibrates individual fixed-disk drives. The drive is specified in register DL, and the status is returned in register AH.

SERVICE 20 (14)

Controller Diagnostics: Service 20 invokes an internal diagnostic routine in the AT's disk controller. The status of the controller is returned in register AH.

SERVICE 21 (15)

Get Disk Type: Service 21 is used to inquire about the type of disk drive installed. Given the drive ID in register DL, it returns in register AH one of four disk-type indicators: If AH is 0, it means no drive is present; if AH is 1, it indicates the presence of a disk drive that cannot sense when the disk has been changed (typical of most disk drives); if AH is 2, it indicates the presence of a disk drive that can sense a change of disks (such as the AT's high-capacity disk drives); finally, if AH is 3, it means that a fixed-disk

(continued)

Peter Norton is a well-known author of books, magazine articles, and software utilities for IBM PCs. He can be reached c/o Microsoft Press, 10700 Northup Way, Box 97200, Bellevue, WA 98004.

Table 1: A *partial summary of the* ROM BIOS *services (interrupts given in hexadecimal).*

Service	Interrupt	Register Input	Register Output	Description
Write string; don't move cursor	10	AH = 13 AH = 00 BL = attribute BH = display page number DX = starting cursor position CX = length of string ES:BP = pointer to start of string		
Write string; move cursor after string	10	AH = 13 AL = 01 BL = attribute BH = display page number DX = starting cursor position CX = length of string ES:BP = pointer to start of string	none	
Write string of alternating characters, attributes; don't move cursor	10	AH = 13 AL = 02 BH = display page number DX = starting cursor position CX = length of string ES:BP = pointer to start of string	none	
Write string of alternating characters, attributes; move cursor	10	AH = 13 AL = 03 BH = display page number DX = starting cursor position CX = length of string ES:BP = pointer to start of string	none	
Disk Services				
Get current drive parameters	13	AH = 08	DL = number of drives DH = maximum number of sides CL = maximum number of sectors CH = maximum number of tracks CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Initialize two fixed-disk base tables	13	AH = 09	CF = success/failure drives AH = status code	Interrupt 41 points to table for drive 0 Interrupt 46 points to table for drive 1 Status codes in AH: see disk service 01

Service	Interrupt	Register: Input	Register: Output	Description
Read long	13	AH = 0A DL = drive ID DH = head number CH = cylinder number CL = sector number ES:BX = pointer to buffer	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Write long	13	AH = 0B DL = drive ID DH = head number CH = cylinder number CL = sector number ES:BX = pointer to buffer	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Seek to cylinder	13	AH = 0C DL = drive ID DH = head number CH = sector number	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Alternate disk reset	13	AH = 0D DL = drive ID	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Test for drive ready	13	AH = 10 DL = drive ID	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Recalibrate drive	13	AH = 11 DL = drive ID	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Controller diagnostics	13	AH = 14	CF = success/failure flag AH = status code	Status codes in AH: see disk service 01
Get disk type	13	AH = 15 DL = drive ID	AH = disk type CX, DX = number of 512-byte sectors when AH = 3	Disk types: AH = 0: disk not there AH = 1: disk, no change detection present AH = 2: disk, change detection present AH = 3: fixed disk
Change of disk status	13	AH = 16	DL = drive that had disk change AH = disk change status 00 = no disk change 01 = disk changed	
Set disk type	13	AH = 17 AL = disk type		Disk type set in AL: AL = 00: no disk AL = 01: regular disk in regular drive AL = 03: high-capacity (1.2-megabyte) disk in high-capacity drive

(continued)

Service	Interrupt	Register		Description
		Input	Output	
Serial Port Services				
Initialize serial port parameters	14	AH = 00 DX = serial port number	AX = serial port status	Status bit settings: 00, 01 = word length 10 = 7 bits; 11 = 8 bits 02 = stop bits: 0 = 1; 1 = 2 03, 04 = parity 00, 01 = none; 01 = odd; 11 = even 05, 06, 07 = data-transmission rate (bits per second): 000 = 110 001 = 150 010 = 360 011 = 600 100 = 1200 101 = 2400 110 = 4800 111 = 9600 (4800 on PCjr)
Extended Services for the AT				
Device open	15	AH = 80 BX = device ID CX = processor type	none	
Device close	15	AH = 81 BX = device ID CX = process type	none	
Device program terminate	15	AH = 82 BX = device ID	none	
Event wait	15	AH = 83 AL = subservice 0 = set interval 1 = cancel ES:BX = pointer to caller's memory CX, DX = number of microseconds to wait	none	
Joystick support	15	AH = 84 DX = 0 get current switch settings DX = 1 read inputs	AL = switch settings AX = A(x) value BX = A(y) value CX = B(x) value DX = B(y) value	
System-request key press	15	AH = 85 AL = 00 press AL = 01 break	none	
Wait	15	AH = 86 CX, DX = number of microseconds to wait before return	none	

(continued)

Service	Interrupt	Register Input	Register Output	Description
Move block	15	AH = 87 CX = number of words to move ES:SI = pointer to table	none	
Get extended memory size	15	AH = 88	none	
Switch to virtual memory mode	15	AH = 89	none	Caution: See BIOS listing before use
Device busy loop	15	AH = 90 AL = type code	none	See BIOS listing
Set flag and complete interrupt	15	AH = 91 AL = type code	none	See BIOS listing

Table 2: *The 12 extended services for the AT available through interrupt 21 (15).*

Service (hexadecimal)	Description
80	Device open
81	Device close
82	Program termination
83	Event wait
84	Joystick support
85	System-request key press
86	Wait
87	Move block
88	Get extended memory size
89	Switch to virtual memory mode (Caution: See BIOS listing before use)
90	Device busy loop
91	Set flag and complete interrupt

drive is installed. When the drive type is 3, the register pair CX:DX acts as a 4-byte integer that gives the total number of disk sectors on the drive.

SERVICE 22 (16)

Change of Disk Status: Service 22 is used to inquire about a change of disks for drives that can sense when a disk has been changed, like the AT's high-capacity drives. Register AH is set to 0 to indicate no disk change and to 6 to indicate a change of disk. Register DL returns the number of the drive that had a disk change.

The change-of-disk sensing in services 21 and 22 is very useful to programs that need to know if a disk has been changed. For certain critical disk operations, such as reading a file allocation table (FAT), it helps to know if the disk has been changed or not. If it has been changed, then any disk data held in memory may have to be discarded and reread. When a disk drive can't report a disk change, the program usually has to assume that it might have been changed and react accordingly—at a cost to program efficiency. When designing programs that control a disk drive, it is clearly useful and more efficient for them to be able to check this information.

SERVICE 23 (17)

Set Disk Type: Service 23 is used to set the disk and drive combination for the AT. If AL is 0, there is no drive; if AL is 1, it indicates a regular disk in a regular drive; if AL is 3, it indicates a high-capacity disk in a high-capacity drive. This service is used with the format service to set the disk type to be formatted.

ROM BIOS SUMMARY

Table 1 presents ROM BIOS services and shows the register usage for input and output parameters for the AT's BIOS.

EXTENDED SERVICES FOR THE AT

Several new BIOS services were introduced with the AT to support the AT's extended memory and some of its more advanced features. They are called through interrupt 21 (15) just like the cassette I/O (input/output) services, with the service number (ranging from 80 through 91) placed in the AH register (see table 2).

THE AT TIME-OF-DAY SERVICES

As in other members of the IBM PC family, interrupt 26 provides the time-of-day services. Services 2 through 6, also invoked through interrupt 26, were introduced in the AT version of the BIOS. Services 2, 3, and 4 read and set the real-time clock, providing both time-of-day and date information, and services 5 and 6 set an alarm to interrupt up to 24 hours from the present time. ■

Why did 61,862 non-programmers send for a Cornerstone™ demo disk?

Cornerstone™ is the only sophisticated database management system created specially for non-programmers.

The 61,862 people who asked for our demo disk were all looking for the same thing: A truly flexible database system that understands how business people think. One that you never have to program. One that makes database building easy – without sacrificing the power you need to effectively manage information.

"...Far easier to use than any other database package released so far."

*–Erik Sandberg-Diment, The New York Times**

Created by Infocom, Cornerstone was designed to break down the barriers between business people and computers.

Cornerstone makes building and using a database easier than you've ever imagined. Because Cornerstone is different from anything you've ever seen.

It's the only system with personalized, interactive "Help" and "Options" keys to assist you. Menus, prompts, and questions guide you through the entire operation.

Cornerstone won't let you make a mistake. It tells you what's happening – why – and how to fix it.

"Cornerstone could easily save months of dBASE II or dBASE III programming... I'm sold on Cornerstone."

–Ira Krakow, Business Computer Systems

The key to Cornerstone's power is its flexibility.

You can make changes anywhere, any time. Even with data already in the database. You can add a file, field, or index, change a report or relationship.

Cornerstone features variable length fields. You never have to determine field length in advance – and you can enter pages of notes in your database. Any field can be multi-valued, and you can even have records within records. There are no artificial constraints. You can search, or sort, on any field.

Cornerstone's flexibility extends to reporting. With no limit to the number or kinds of reports you can create.

"Cornerstone is the best program I've ever used. If you need a relational database, buy Cornerstone."

–Kerry Leichtman, PC Week

Cornerstone is now available for the IBM® PC, PC XT,™ PC AT™ and compatibles, and soon for the Tandy® 1000, 1200HD and 2000. Suggested list price $495.

Free Cornerstone Information Kit and demo disk.

Call today, 1-617-574-0644. Or complete and mail the postage-paid card to Infocom, Cornerstone Demo Disk, Department "D," 125 CambridgePark Drive, Cambridge, Massachusetts 02140.

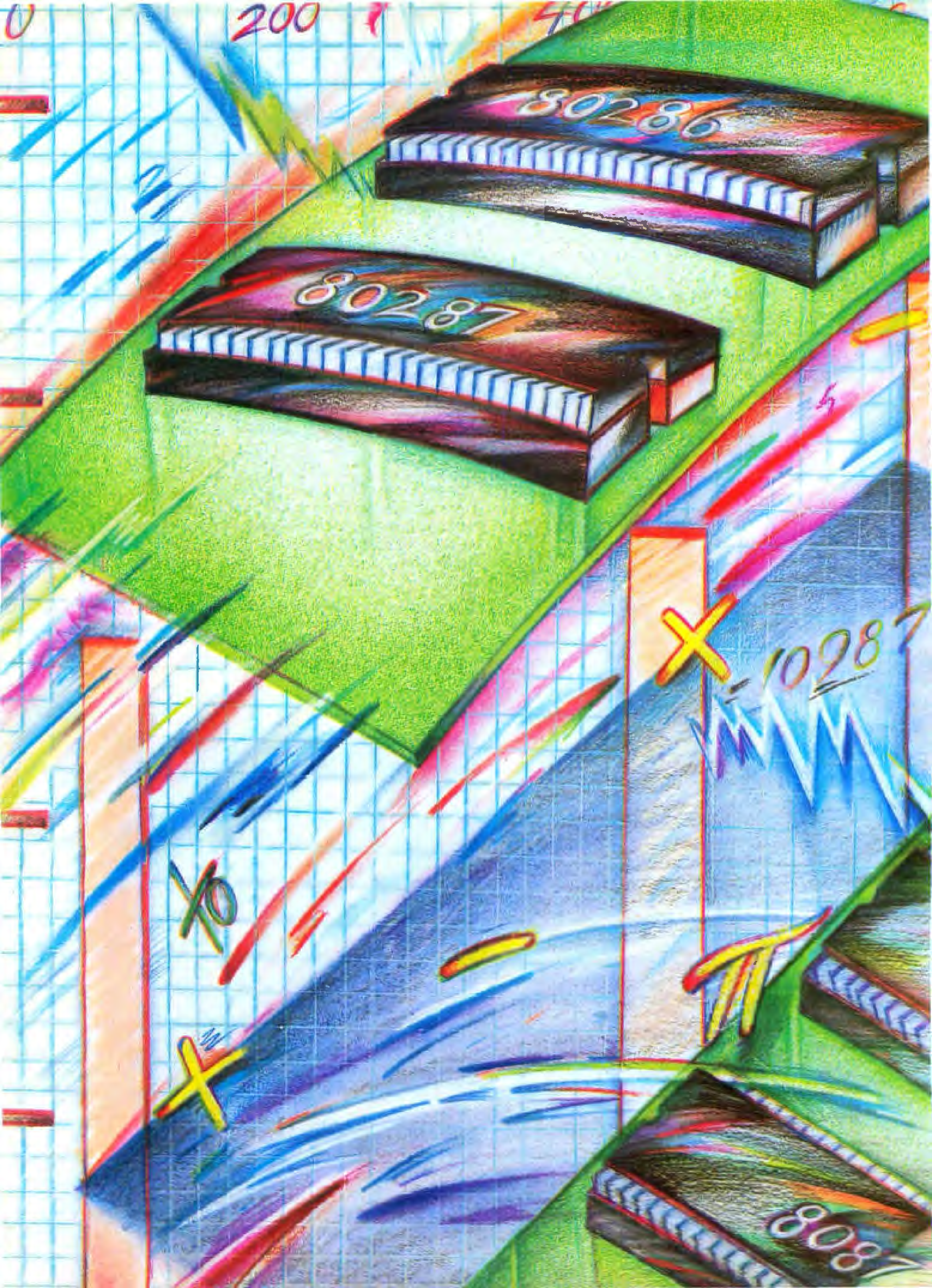
We'll send you a free Cornerstone Information Kit, including a free demo disk. You'll see why this database system has received outstanding reviews from business and PC publications – why business people are making it their way to manage information – and why Cornerstone has quickly made its way to the Softsel® Hot List® and other software best seller lists.

Find out for yourself why 61,862 other non-programmers are asking about Cornerstone. Call or mail the card today. (Better hurry! This offer lasts only while supplies last.)

INFOCOM™

dBase II is a registered trademark of Ashton Tate. dBase III is a trademark of Ashton Tate. All remaining products mentioned are trademarks or registered trademarks as indicated.

*The New York Times quote from the May 5, 1985 issue.


0
200
80286
80287

THE 8087/80287 PERFORMANCE CURVE

80287 performance does not provide the same degree of processor enhancement as did the 8087

BY STEPHEN S. FRIED

THE INTEL 8087 is a specialized 80-bit coprocessor that extends the 8088 and 8086 microprocessors to include floating-point operations. Because of the size of its registers and internal data paths (80 bits), it does floating-point arithmetic 50 to 100 times faster than the 16-bit processors it works with. Today, an IBM Personal Computer software product is expected to include 8087 support, and most do. However, a lot of 8087 software is still only 5 to 15 percent efficient and just barely takes advantage of the chip's speed and accuracy.

In 1985, however, a lot of folks who own IBM PC ATs are discovering that much of their software shows no gain or even slows up when they install an 80287, the AT's counterpart to the 8087. Needless to say, if some salesman just sold you an AT based on its blinding speed, you might be a little annoyed when you discovered that adding an 80287 had little effect on programs that were advertised to have 8087 support. In fact, the only reason lots of users never complained about 80287 performance is that they got the same numeric speed out of the AT's 80287 as they were getting from their PC's 8087. The dilemma of why a supposedly better processor fails to improve PC performance is part of a complicated story that is the main theme of this article: evaluating the performance of processor/coprocessor systems as the "quality" of the numeric processor and its support is changed.

This article builds on concepts introduced in my previous article ("Evaluating 8087 Performance on the IBM PC," BYTE *Guide to the IBM Personal Computers*, Fall 1984). The purpose of this article is threefold. First, I'll review some of the myths and history of the 8087 and 80287, touching on manufacturing, labeling, reliability, and chip validation. Second, I'll pick up where the first article left off by reviewing significant new 8087 software that has come on the scene in the last year, including three spreadsheets and a number of BASICs. Third, I'll explore the effect on overall process speed of changing the throughput of either processor in a processor/coprocessor pair, such as the 8088/8087, 8086/8087, or 80286/80287.

One of the things that we will uncover below is that 80286-based machines are more sensitive to the quality of 80287 support than 8088-based

(continued)

Stephen Fried is principal scientist and vice president of R&D at MicroWay Inc. (POB 79, Kingston, MA 02364), which produces 87BASIC and 87BASIC Inline.

ILLUSTRATED BY KATHERYN HOLT

machines were to the quality of 8087 support. The bottom line for the AT and its look-alikes is that the range of applications that really need 80287 support has been reduced to those that are generated with native-code in-line compilers or make extensive use of transcendental and trigonometric functions. On the positive side, the 80286 really shines in problems in which the calculation of the addresses of floating-point numbers takes longer than the arithmetic being performed. For example, multidimensional (typically three or more dimensions) array arithmetic really benefits from the 80286's increased integer throughput, especially when the indexes of the arrays are 32-bit integers.

8087 History, Myths, and Validation

A number of different grades of 8087s are available. Some are more reliable than others, and a number of them will not run on machines that they theoretically ought to. Using a chip that has not been thoroughly tested in the machine it is installed in is very risky in the case of an 8087. In addition, validating 8087 operation in microcomputers is an important subject that manufacturers have completely overlooked.

The 8087 is an 80-bit Intel sole-source (i.e., Intel has a monopoly) microprocessor that works in conjunction with the 16-bit Intel 8088 or 8086. It uses its 80-bit registers (there are a total of 8) to store and manipulate floating-point numbers, which it holds internally in an extremely precise format called "temporary real." The 8087 can perform simple operations such as addition, subtraction, multiplication, division, comparison, and finding square roots in a single operation about 100 times faster than an 8088 or 8086 using floating-point software routines. In addition, it can be programmed to calculate all transcendental and trigonometric functions a factor of 50 times faster than software routines. Accomplishing these feats requires a large number of gates (transistors), and the 8087 has roughly three times as many as the 8088 or 8086. Lots of gates imply lots of heat: A typical 8087 dissipates 1.5 watts and will have a surface temperature that ranges from 45°C to 55°C in an IBM PC. The surface temperature of the 8087 depends on the heat dissipated by other nearby devices, the cooling provided (the newer machines have larger fans), and the outside air temperature. When the surface temperature exceeds the Intel-rated temperature of the chip, anything can happen (and occasionally does).

Because of the large size of the die (the piece of silicon that makes up the chip), the first 8087s were very difficult to fabricate. Intel's yields on the component started off in the 15 percent range; i.e., only one in seven of the chips on a silicon wafer passed all tests and made it to market. The chips that did pass all tests were divided into two bins, those that worked at 5 MHz and 70°C and those that worked at 4 MHz and 70°C. About every six months after initial production started, a better version of the chip was developed and a new "step" of the component came out. With each succeeding step came higher yields, smaller devices, less heat dissipation, better speed, and lower prices. Each step also involved shrinking the die and would also be accompanied by an interruption in 8087 supplies, leading some people to speculate that 8087s were always in deliberate short supply. The most recent step, which occurred in November 1984, also involved a change in the semiconductor fabrication technique or process. It resulted in much faster 8087s but also required computer manufacturers to use better decoupling and sockets. The last step was such a significant improvement in the chip that Intel decided to change the way it labeled 8087s and 80287s. The labeling of Intel NDPs (numeric data processors) is summarized in table 1.

The components in the left-hand column of table 1 were manufactured using the first three steps. There is an incredible difference in operating and electrical characteristics between the first three steps and the fourth (right-hand column). For example, the components in the left-hand column often fail right at their upper temperature limits. On the other hand, the newer components often run to 130°C, which is 60 degrees above specification. We expressed this by placing a + after the official Intel upper temperature limit. Of course, running a component over specification is not recommended practice and often cannot be accomplished without the use of special sockets and other components.

Looking over table 1, one notices a "funny" component, tested to run at precisely the frequency of a PC. The -6 is an 8087 that didn't pass the -3 specification but was found to work in a PC if the 8087 surface temperature did not rise above 50°C. According to Intel, this obsolete component was never put on the open market, although about 20,000 were sold to a large OEM (original equipment manufacturer) who decided to use them "in house." We believe the -6 is actually a retested -4, and while it may work in your PC, it should be used with caution in any machine that is "loaded up" or in machines that run in hot en-

(continued)

Table 1: 8087 *and* 80287 *specifications.*

OLD STEPS HMOS1			NEW STEP HMOS3		
Part #	Max Freq	Max Temp	Part #	Max Freq	Max Temp
C8087-4	4.00	70			
C8087-6	4.77	50			
C8087-3	5.00	70	C8087	5.00	70+
			C8087-2	8.00	70+
			C8087-1	10.00	70+
C80287-3	5.33	70			
			C80287-8	8.00	70+

vironments. The fact of the matter is that even a -3 can be run out of limits if you push your machine too hard or neglect to keep the room temperature below 85°F. To guarantee that you don't get stuck with a potential lemon, make sure the 8087 you are buying is a C8087 with no dash number or a -2 if you have an 8-MHz machine.

Finally, I'll end with a discussion of validating 8087/80287 performance. When IBM decided to venture into the PC business, it assembled a special quality-assurance team to develop acceptance standards for the semiconductors to be used in the new machine. They also designed a suite of programs to test the components used in the PC. This suite is in the ROM BIOS, where it is run every time your PC is booted.

Unfortunately, there is no ROM BIOS routine that validates correct 8087 performance. The odds are your NDP works great all the time. But there is about a 0.1 percent probability (this number may be smaller and is a function of your machine and the 8087) that when you run your 8087 hard or near its upper limits it will lose accuracy. Since any failure of a processor to yield repeatable results is a complete failure, this means even a loss of 1 bit in 63 is a failure of your 8087. Because the 8087 starts failing gradually, it is impossible to detect a bad processor without running an 8087 processor diagnostic program.

When failures are induced by raising the surface temperature or operating frequency of an 8087, they initially appear about once in 100,000 operations. As the condition gets worse, the errors get more frequent and increase in magnitude until a very high percentage of all 8087 operations are erroneous. Often, the errors are small, only a few bits out of 63. As the test continues, the accuracy continues getting worse. Finally, some sort of catastrophe will occur, probably involving 8088–8087 synchronization, and the PC will crash. Cool the 8087 back down, and it will be as good as new. The point here is that a calculation made with a chip that is on the edge of failing could seem perfectly okay on the surface until compared with a calculation made with a properly running chip. The best way to guarantee accuracy is to stay well within the chip's tested operating guard band.

The 8087/80287 NDP is a valuable addition to an IBM PC or AT. Like all the other parts of your machine, it should be prudently used. It pays to buy the highest quality processors and use them within the manufacturer's operating envelope. It is also very important to validate processor operation in your PC at both the cold and hot operating points and before and after any critical run.

```
Savage evaluates:
        A = 1
        FOR I=1 TO 2499
        A = TAN(ATAN(EXP(LOG(SQR(A*A))))) + 1
        NEXT I
The Savage expression for cell A21 in 1-2-3 is:
        @TAN(@ATAN(@EXP(@LN(@SQR(A20*A20))))) + 1
Common Subexpression evaluates:
        A = 0
        FOR J = 1 TO 10
        FOR I = 1 TO 1000
        B = (A+1)*(A+2)/(A+3) + (A+3)*(A+1)/(A+2) + (A+2)*(A+3)/(A+1)
        NEXT I
        NEXT J
Megalopolis evaluates each of the following expressions 1800 times:
        1.2 + 3.4;    2.3 - 4.5;    2.3 * 4.5;    2.3/4.5
The FORTRAN benchmark evaluates array manipulations:
Subroutine 1, one-dimensional array (10,000 loops):
        DO 100 N = 1, 1000
        DO 100 I = 1, 100
        W(I)=Z(I) + X(I)*Y(I)*Y(I)/Z(I)
        R(I)=W(I) + Z(I)*Y(I)
100     CONTINUE
Subroutine 2, two-dimensional array (20,000 loops):
        DO 100 J = 1, 1
        DO 100 N = 1, ,1000
        DO 100 I = 1, 100
        W(I,J) = Z(I,J) + X(I,J)*Y(I,J)/Z(I,J)
        R(I,J) = W(I,J) + Z(I,J)*Y(I,J)
100     CONTINUE
Subroutine 3, three-dimensional array (30,000 loops):
        DO 100 K = 1, 1
        DO 100 J = 1, 1
        DO 100 N = 1, 1000
        DO 100 I = 1, 100
        W(I,J,K) = Z(I,J,K) + X(I,J,K)*Y(I,J,K)/Z(I,J,K)
        R(I,J,K) = W(I,J,K) + Z(I,J,K)*Y(I,J,K)
100     CONTINUE
```

Figure 1: *The benchmarks used to evaluate 8087/80287 performance. The Savage and Common Subexpression code is in BASIC, with a Savage expression also coded to work with Lotus 1-2-3.*

Comparing New Processor Implementations and Software

Unfortunately, the only way to learn about the speed of processors and programs, short of counting cycles or buying a logic analyzer, is to write and run benchmarks. This article uses five benchmarks run on four different processor combinations to gain insight into the relative speed of both the processors and the software used to

(continued)

code the benchmarks. It was our intention here to create benchmarks that test the different floating-point properties of the coprocessors and the ability of different compilers to produce high-quality floating-point code. The benchmarks are displayed in figure 1. They were written in BASIC, FORTRAN, and using Lotus 1-2-3 and Symphony 1.1. One aspect of the benchmarks that we did not pursue in this article is accuracy. The Savage benchmark in particular is used as a measure of both speed and round-off error. All of the products tested benefited from the accuracy of the 8087 and showed reduced round-off error when run with the 8087.

This article will compare three processor combinations, the 8086/8087, 80286/80287, and the 8088/8087 used in the original IBM PC. The first two processor pairs will be compared with the third, a standard PC.

Even though the 8086 preceded the 8088, it was not the first Intel 16-bit microprocessor to get used in large commercial numbers. The 8088 with its 16-bit internal registers and 8-bit data bus was the first processor that was widely accepted. That was a natural consequence of its choice by IBM for the IBM PC.

The first processor combination I'll compare is the 4.77-MHz 8088/8087 of the standard PC with a card featuring a 9.54-MHz 8086/8087. Why, for example, does this card execute some things two times faster than a PC and others four times faster? In the process of comparing these two CPUs (central processing units), you should develop a better feeling about what defines a good 8087 program.

The performance of the 9.54-MHz 8086/8087 board is comparable to that of a number of microcomputers that use the 8086 or 80186 CPUs at frequencies of up to 10 MHz.

The 8086–8088 comparison is best understood if we break up all computer operations into two groups: register operations that are limited by the ALU (arithmetic and logic unit) in the processor and I/O (input/output) operations that are limited by the time required by the processor to access memory or ports. Since all register operations are synchronized with the system clock, all register operations increase linearly with the clock speed; double the clock and you'll get twice as many operations. Both the 8086 and 8088 have identical registers and both use identical 8087s, so doubling the clock speed and changing from an 8088 to an 8086 would only double the speed if a program were register-bound. However, it's rare to find a microcomputer program that is so well written that it is register-bound, and it turns out that the time to fetch information from memory ends up playing an important role in most programs.

Memory fetches in an 8088 or 8086 are controlled by the depth of the prefetch queue and the data-bus bandwidth. The data-bus bandwidth is a measure of a bus's speed at transferring information. In the case of the 9.54-MHz 8086/8087 board, the bandwidth of the board is a factor of 4 greater than that of a standard PC. This increase in speed has two contributors, each of which adds a factor of two: the doubling of the clock speed and the doubling of the data-bus width. To determine the improvement that a 9.54-MHz 8086/8087 board provides over a PC, one must consider the mix of operations that a particular line of code performs. If one chooses a non-floating-point program that is written with a middle-of-the-road compiler, one discovers that a typical program executes a factor of 2.8 to 3.0 times faster on a 9.54-MHz 8086/8087 board than on a PC. This speedup can be explained by the following naive model: On average, our hypothetical program spends half its time accessing memory and half its time doing register-intensive operations. In the 9.54-MHz 8086/8087 board, memory accesses run a factor of 4 faster, while register operations run a factor of 2 faster. The average of 2 and 4 is 3, which explains why most nonnumeric programs run a factor of 2.8 to 3.0 times faster.

This idea can be quantified using the notion of code kinetics developed in my previous article. Readers content with our simple explanation are permitted to skip the equations. We assume that any computer program can be broken into pieces that can be thought of as either I/O-bound or register-bound. We now gather together all the I/O-bound parts of a program and execute them in a line. If each piece is assigned a weight that is proportional to the time it takes to execute it, we can describe the time to execute the entire I/O portion as

$$T_{io} = \frac{n_{io}}{R_{ibmio}} = \frac{N_{io}}{R_{ibmio}} \qquad (1)$$

where n_{io} represents the number of I/O operations for each step, N_{io} is the total number of I/O operations. R_{ibmio} represents the rate at which an IBM does I/O (units of I/O operations per second), and T_{io} is the time in seconds. We can write a similar rate equation for register operations:

$$T_{reg} = \frac{N_{reg}}{R_{ibmreg}} \qquad (2)$$

where all the symbols have nearly identical meanings, except that we have replaced the subscript *io* with *reg* (register) to indicate that these processes are register-bound. For a particular stretch of code, we can now express the time to execute both the I/O- and register-bound portions as

$$T_{ibm} = \frac{N_{io}}{R_{ibmio}} + \frac{N_{reg}}{R_{ibmreg}} \qquad (3)$$

This puts us in a position to compute the relative speed of a 9.54-MHz 8086/8087 board with a standard PC. Replacing the *ibm* in equation 3 with *ns* for the 9.54-MHz 8086/8087 board, we can write an equation for the 9.54-MHz 8086/8087 board to compute the same line of code:

$$T_{ns} = \frac{N_{io}}{R_{nsio}} + \frac{N_{reg}}{R_{nsreg}} \qquad (4)$$

and now dividing equation 3 by equation 4 yields the ratio of speeds of the two processors:

$$\text{Relative Speed} = \frac{T_{ibm}}{T_{ns}} = \left(\frac{N_{reg}}{T_{ibmreg}} + \frac{N_{io}}{T_{ibmio}}\right) \div \left(\frac{N_{reg}}{R_{nsreg}} + \frac{N_{io}}{R_{nsio}}\right) \qquad (5)$$

If we wanted to, we could divide any program into I/O and register portions and, using the standard op code timings in the Intel iAPX processor handbooks, figure out the values of *N*. This would be a long, arduous task most suited to a computer. Instead, we will examine the equation and make some simplifying assumptions that lead us to interesting numbers in a hurry.

First, notice that N_{io} and N_{reg} are characteristics of the program we are using to test the relative speeds, while R_{ibmio}, R_{ibmreg}, R_{nsio}, and R_{nsreg} are rates that are characteristic of the processes and the two machines we are comparing. We can immediately conclude that the relative speed is a function of the program used to benchmark the systems. Equation 5 can be simplified if we assume that the rate at which the IBM does I/O is the same as the rate at which it does register operations:

$$R = R_{ibmio} = R_{ibmreg} \qquad (6)$$

and also take into account what was said above about the relative speeds of the register operations and data-bus bandwidth:

$$R_{nsio} = 4R_{ibmio} = 4R \qquad (7)$$

$$R_{nsreg} = 2R_{ibmreg} = 2R \qquad (8)$$

Combining equations 5 through 8 and simplifying yields an expression for the relative speed of a 9.54-MHz 8086/8087 board to a PC:

$$\text{Relative Speed} = \frac{4(1 + N_{io}/N_{reg})}{(2 + N_{io}/N_{reg})} \qquad (9)$$

that is a function of the ratio of the number of I/O to register operations. If this function is reasonable, it should asymptote to 4 for I/O-bound sequences of code and to 2 for register-bound sequences. If the underlying assumptions are accurate (and theoretical models are only as good as the guesses they are based on), then we can infer from this equation that a program that executes three times as fast on a 9.54-MHz 8086/8087 board has a ratio of two I/O-bound operations to every register-bound operation.

Having developed a pretty model, we decided to examine the premises it was built on. First, were there any pure I/O-bound operations that executed a factor of 4 faster? The most obvious place to look for I/O-bound operations ought to be fast I/O instructions. The 8086 has a whole set of built-in string instructions, one of which can be used to move blocks of bytes or words from one location in memory to another. To use the instruction you set up pointers to the two blocks, tell the processor how

(continued)

many words or bytes you want to move, and then turn the block move on. Unfortunately, the speed increase was exactly a factor of 2, indicating that block moves are bound by the time it takes the CPU to compute and increment the addresses of the blocks being moved. The next candidate for an I/O-bound operation would be instructions that execute very fast once loaded into the 8088. Any instruction whose internal execution time is much less than the time required to fetch the instruction into the processor will have a total execution time that is limited by the time required to fetch instructions. We made up a simple program that consisted of a loop that executed 30 inter-register moves in a line:

```
MOV    BX,AX
```

This line of 30 moves executed 3.96 times faster on the 9.54-MHz 8086/8087 board. When we examined individual instructions with a logic analyzer, the ratio turned out to be 4.0 for individual moves. For those interested, the MOV instruction is 2 bytes long and executes internally in two cycles—420 nanoseconds (ns) in the case of the PC and 210 ns in the case of the 9.54-MHz 8086/8087 board. However, fetching in 2 bytes of code takes 1680 ns for the PC but only 420 ns for the 9.54-MHz 8086/8087 board. Since the moves take place in much less time than the fetches, this instruction is fetch (I/O) limited, Q.E.D.

This is a good example of the classic von Neumann bottleneck: A processor's execution speed can be limited to the time required to fetch instructions from memory. The other interesting situation that the logic analyzer pointed out is that the increase in speed is a strong function of the state of the prefetch queue. As we reduced the number of MOVs in our loop, the speedup was reduced because of the fact that the processor would have to refill the queue at the end of every loop. As the number of MOVs approached zero, the speedup approached 2 because the LOOP instruction is a long-running register-bound operation.

From benchmarking average non-numeric programs we know that the average 9.54-MHz 8086/8087 board speedup is a factor of 3 and is composed of operations that run anywhere from a factor of 2 to a factor of 4 faster. The question now arises: What kind of speedups should be obtained from 8087 code? At first glance, one would expect 8087 code to be limited by the increase in clock speed. Most 8087 instructions take at least 100 cycles to execute and are dominated by the time it takes the 8087 ALU to perform floating-point arithmetic. Even 8087 operations that

(continued)

NOTHING DOWN.

Even the best of personal computers can have a bad day. Blackouts happen. Usually at the worst possible time.

But power interruptions don't need to be business interruptions. And lost power shouldn't result in lost data. Not with the POWERMAKER® Micro UPS from Topaz.

Designed specifically for hard disc and critical-use business systems, Powermaker Micro UPS ensures a continuous supply of smooth sine wave power even during a total blackout.

And there's more. Powermaker Micro UPS also removes spikes and other error-producing transients from incoming power, protecting your PC's sensitive circuitry.

So, if your PC needs a little help once in a while, give it Powermaker Micro UPS protection. For nothing down. Call us today at (619) 279-0831, or contact your local Square D distributor.

TOPAZ®
SQUARE D COMPANY

Inquiry 199 for End-Users.
Inquiry 200 for DEALERS ONLY.

load or store numbers spend at most 30 percent of the time doing I/O, indicating that 8087 operations on average ought to be register-bound.

One of the most number-intensive benchmarks available, the Savage, is a good test of the ability of the 8087 to perform trigonometrics and transcendental functions. Our intuitive feeling about 8087 code being register-bound is borne out by the Savage benchmarks in table 2. This table indicates that the best (fastest) Savage benchmarks run 2.14 times faster on the 9.54-MHz 8086/8087 board than on an 8087-equipped PC.

Row 2 contains the times for BASCOM, the IBM BASIC COMPILER. The fact that the compiler speeds up the calculation by only a factor of 5 over the interpreter indicates that Savage really bogs down BASCOM with arithmetic (comparisons for other programs will generally show that compiled BASIC usually runs a factor of 10 to 30 faster than BASICA). Neither row 1 or 2 used an 8087. Rows 3 and 4 show results for 87BASIC and 87BASIC Inline. Comparing rows 2 and 3 demonstrates that the first-generation code makes a 30 to 1 improvement for the benchmark on the PC. We deduce this by comparing the 169 seconds for BASCOM running on a PC without any 8087 to the 5.74 seconds of 87BASIC and the 3.5 seconds of Inline.

Rows 5, 6, and 7 give the times for three other well-known BASICS: True BASIC, BetterBASIC, and Professional BASIC. These other products include 8087 support, but their forte lies in other areas such as easy debugging, modularity, large arrays, or incremental compilation. Even though they are compiled BASICs, they compile to p-code instead of in-line native code. In general, to really get the true speed of an 8087 requires efficient code, and p-code is just not that efficient.

Lines 8 and 9 are the times for two of the more popular FORTRANs, Microsoft 3.3 and Ryan-McFarland 1.0 (both are also separately marketed by IBM). The Savage benchmark is more dependent on the quality of the trigonometric and transcendental libraries of the compiler being checked than the efficiency of the code that links library calls together.

Table 2: *The Savage benchmark results for the* IBM PC, *in both a standard configuration and with an* 8086/8087 *board for testing the* 8086.

SAVAGE BENCHMARK
IBM PC
(time in seconds)

		Standard PC		9.54-MHz 8086/8087 board	
	Frequency of 8088/8086	4.77 MHz	4.77 MHz	9.54 MHz	9.54 MHz
	Frequency of 8087	0 MHz	4.77 MHz	0 MHz	9.54 MHz
(1)	BASICA (no 8087)	891	—	757/318*	—
(2)	BASCOM (no 8087)	169.5	—	64.7	—
(3)	87BASIC (with 8087)	—	5.74	—	2.53
(4)	87BASIC Inline compiler (with 8087)	—	3.35	—	1.56
(5)	True BASIC	160	10	—	35
(6)	BetterBASIC	873	12	—	5.2
(7)	Professional BASIC	420	15	335	6
(8)	MS FORTRAN	—	6.92	—	3.26
(9)	Ryan-McFarland FORTRAN	—	3.85	—	1.81
(10)	1-2-3 Retrieval Recalculation	23 374	 —	—	12 177
(11)	1-2-3 with 8087 support Retrieval Recalculation	23 —	— 10.2	—	12 4.8
(12)	Symphony 1.1 Retrieval Recalculation	 478	25 17.5	— —	14 10.2

* FAST ROM enabled

SPREADSHEET BENCHMARKS

There are two different kinds of problems spreadsheet users encounter. The first involves recalculating worksheets that contain many transcendental and financial functions, such as NPV or IRR. The second involves recalculating worksheets that are filled up with cells that contain financial information but that are at best connected by formulas that involve the four simple numeric operations. To analyze the relative performance of these sheets with different processors, we again fell back onto

(continued)

PERSONAL COMPUTER PHOTOCOPIER

Have you ever wished that there was a machine for your PC that could scan your photographs, artwork or documents just like a photocopier? And was as easy to use and understand? As well as copy and store in color or black-and-white onto your hard drive for editing?

Well, there is such a machine now. It's called the SpectraFAX digital photocopier. We got tired of trying to use digitizing tablets, cameras, mice and everything else that has been invented. The SpectraFAX machines will scan any and all of your art, photos, forms or text for storage in your PC. Images that you can cut-and-paste into your word processing or database programs. Our SpectraFAX 200 will scan and store your color or black-and-white images for editing and printing at resolutions up to 200 dots per inch. Then the SpectraFAX Graphics Editor software that we bundle with the scanner will allow you to cut-and-paste your images, rotate them, enlarge and reduce whatever pictures you choose.

That's not all. Our digital photocopiers are based on open-architecture design, which means that as new uses for this technology develop, add-on cards can enhance our SpectraFAX digital photocopier. But we didn't want to wait for third-party vendors, so we invented two optional boards of our own: the SpectraFAXimile™ Card and the SpectraFAX TEXreader™.

The SpectraFAXimile card takes advantage of the 200 dots per inch resolution of the scanning device to turn it into a facsimile machine, compatible with all Group II and Group III facsimile machines worldwide.

The TEXreader board is a revolution in itself — full-scale OCR (optical character recognition) to read typewritten pages with the SpectraFAX machine into your word processor without having to re-type what somebody else has already typed.

What's most amazing about the SpectraFAX product line, though, are the prices. Our digital photocopiers cost about the same as a regular office copier. Because we know high tech has to be affordable.

• IBM is a registered trademark of International Business Machines, Corp.
• SpectraFAX, SpectraFAX 200, DS-200, SpectraFAX Graphics Editor, SpectraFAXimile & TEX reader are trademarks of SpectraFAX, Corp.

SpectraFAX Corp.

2000 Palm Street S. • Naples, Florida 33962 • (813) 775-2737 two benchmarks, the Savage and a simple expression of constants evaluated 2500 times, Megalopolis.

Because the time required to go from cell to cell is much less than the time required to evaluate the Savage function, Savage results for spreadsheets with 8087 support can approach that of good compilers. Lines 10, 11, and 12 of table 2 are the times for Savage run on 1-2-3, 1-2-3 with 8087 support, and Symphony 1.1 with 8087 support. Two quick observations: 1-2-3 seems to be tighter than Symphony and as a result is a better candidate for 8087 support. 8087 support for 1-2-3 improves Savage by a factor of 36.66, while Symphony 1.1 runs the Savage 27.3 times faster when run with an 8087.

The Savage is the ultimate 8087 benchmark in that it is so number-intensive that it can make even inefficient compilers look good, provided they have good run-time libraries.

Table 3 compares the ability of the same BASIC compilers, FORTRANs, and spreadsheets to evaluate expressions. The spreadsheets were compared with a single worksheet that added, subtracted, multiplied, and divided four constants and was repeated 2500 times. We call this Megalopolis because it is typical of the kinds of long-winded sheets that end up accumulating when inexperienced users keep adding cells to their first and only worksheet (a real problem, I am told). Looking at the results for 1-2-3 (15.9 seconds) and 1-2-3 with 8087 support (7.8 seconds), we see that there is only a factor of 2 increase associated with adding an 8087. Looking at the results for Symphony 1.1, we see that adding an 8087 makes an improvement of only 30 percent. Megalopolis is actually a little severe. Normal use seems to deliver speedups of between 3 to 1 and 10 to 1 and higher. What this says is that most worksheets incorporate at least a few of the built-in 1-2-3 functions or a lot of built-in functions. In general, the larger the number of built-ins called, the greater the speedup.

The BASICs and FORTRANs were compared running 10,000 loops of an expression that has three common subexpressions repeated three times each. This benchmark was clearly designed to highlight the use of the 8087 stack. The more use a product makes of the stack, the faster it will run this benchmark. True BASIC does the best of the p-code compilers in both this and the Savage. It should be pointed out that the stack could also be used to hold "hot" global variables, an optimization not being done by any of these compilers but that is a normal technique employed by assembly-language programmers.

In this benchmark we intended to force the compiler to generate a lot of long-lived temporaries, which in a good compiler will end up on the 8087 stack. We did this using common subexpressions. However, a good compiler evaluates a common subexpression only once, and it is

(continued)

Table 3: *The Common Subexpression and Megalopolis benchmark results for the* IBM PC *and the* IBM PC *with the* 8086/8087 *board.*

COMMON SUBEXPRESSION BENCHMARK
IBM PC
(time in seconds)

		Standard PC		9.54-MHz 8086/8087 board	
	Frequency of 8088/8086	4.77 MHz	4.77 MHz	9.54 MHz	9.54 MHz
	Frequency of 8087	0 MHz	4.77 MHz	0 MHz	9.54 MHz
(1)	BASICA (no 80287)	862	—	769/274*	—
(2)	BASCOM (no 80287)	53.17	—	18.92	—
(3)	87BASIC (with 8087)	—	27.29	—	10.11
(4)	87BASIC Inline compiler (with 8087)	—	4.73		— 2.16
(5)	True BASIC	74	48	—	17
(6)	BetterBASIC	310	93	—	42
(7)	Professional BASIC	1023	129	320	53
(8)	MS FORTRAN	—	5.92	—	2.9
(9)	Ryan-McFarland FORTRAN	—	5.1	—	2.62
	MEGALOPOLIS BENCHMARK				
(1)	1-2-3 (no 8087)	15.9	—	6.9	—
(2)	1-2-3 with 8087 support	—	7.8	—	3.3
(3)	Symphony 1.1	21.2	16.2	8.3	6

* FAST ROM enabled

possible that some of the BASICs (we know this is true of BASICA) may not have performed this optimization, thus contributing to the discrepancies between the p-code compilers.

On the other hand, RM/FORTRAN clearly added an optimization that we didn't expect, forcing us to change our benchmark so that RM/FORTRAN would generate 8087 code that was evaluated in the loop. RM/FORTRAN discovered that the variable *A* was loop invariant and could be moved ahead of the loop, thus effectively eliminating the entire execution time associated with repetitively evaluating the expression.

INTERPRETING 80287 BENCHMARKS

The 80287 is the coprocessor designed to work with the 80286. It is identical to the 8087 from the standpoint of how you write programs to run it, except for the issue of identifying that there is a coprocessor in the system. At least one product with automatic 8087 sensing does not work with the 80287 as a result of this difference. To understand the performance-related issues that determine whether or not you should add an 80287 to your PC AT, it is necessary to first look at the 80286 and the interface between the 80286 and the 80287. The 80287 also adds a new feature to the processor/coprocessor system: It can be used with an optional asynchronous clock that can be used to speed up the 80287 ALU.

The 80286 is an upgraded version of the 8086 that is designed for use with multiuser and multitasking operating systems. It has two modes of operation, real and protected, and is substantially faster than the 8086. The added speed is obtained by increasing the amount of parallelism in the 80286; it has four internal units that each independently do a piece of the total processing job, in comparison to the two independent units of the 8086. In protected address mode the 80286 can address 16 megabytes of memory and manage a virtual address space of a gigabyte. In real address mode, it emulates an 8086, addressing a megabyte. Most of the programs and operating systems currently in vogue work in the real address mode only and do not use either of the extended addressing capabilities of the 80286.

The 80286 is a substantial upgrade of the 8086 in all respects except the interface to the 80287. In the 8086/8087 pair, the 8086 and 8087 each have their own BIU (bus interface unit) that allows either processor to independently acquire data from the system. When an 8087 gets passed an

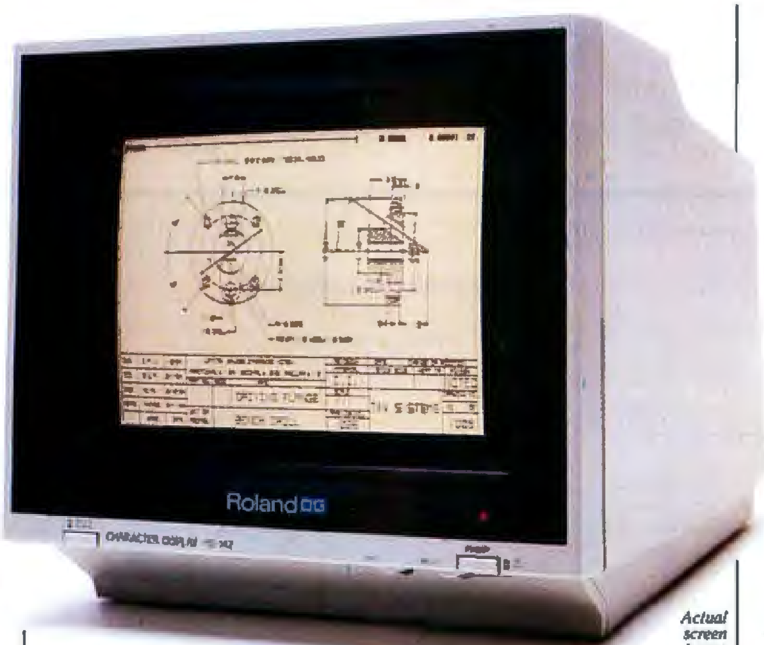
(continued)

Now Showing In Black And White

If you own an IBM-PC or PC work-alike, Roland's new MB-142 monitor lets you show off your text and graphics in today's hottest colors—black and white. That's right! The MB-142 gives you black characters on a paper-white background—just like people have been reading for centuries. You can also have white characters on a black background with just the touch of a button.

Both of these black and white display formats are easier on the eyes and less fatiguing than the green or amber phosphor used in standard monochrome monitors. The MB-142's large 14-inch screen, combined with its ultra-high 720 x 350 resolution, can display characters that are larger and more legible than what you can get with ordinary monochrome monitors. Another great plus is that the MB-142 plugs directly into the monochrome board of your IBM or compatible—just like your present monochrome monitor, with nothing more to buy.

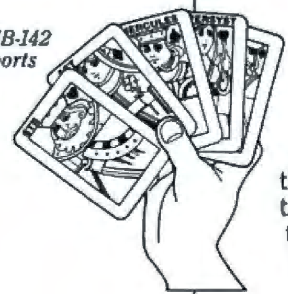
Because of the MB-142's advanced electronic circuitry, you even have the ability to mix graphics and text on the same display when using graphics and text boards from leading manufacturers such as Persyst, STB, Paradise, Hercules, AST and many others. What makes it all possible? The same sophisticated technology used in color monitors.



Actual screen image

MODEL
MB-142

Push a button for instant reverse screen



The MB-142 supports all the winning cards

For business, black and white makes more sense than green and black

The big difference is that the MB-142 monitor does the job for significantly less money. The MB-142 is designed to interface economically, too. Imagine seeing your favorite business graphics or CAD/CAM packages, such as Lotus 1-2-3, Energraphics, Chart-Master, AutoCAD, CADDraft and VersaCAD, in ultra-high resolution black and white. Also, take full advantage of your program's windowing capability using the large 14-inch screen.

Take a good look at the differences that set the MB-142 apart from the rest. No other monochrome monitor gives you the fatigue-free black and white viewing, text and graphics capabilities and easy interface. Naturally enough, the MB-142 is from Roland DG—the new computer peripherals company that's pointing the way to the future. Look for this and other Roland products at fine computer dealers everywhere.

For more information, contact: Roland DG, 7200 Dominion Circle, Los Angeles, CA 90040. (213) 685-5141.

The software programs listed are trademarks of the following companies: AutoCAD, AUTODESK, Inc.; CADDraft, Personal CAD Systems, Inc.; Chart-Master, Decision Resources, Inc.; Energraphics, Enertronic Research, Inc.; Lotus 1-2-3, Lotus Development Corp.; VersaCAD, T&W Systems, Inc.

Roland DG

instruction, it uses its BIU to compute the addresses of operands and then generates the required signals to load or store the specified operands. While the 8087 is doing this, the 8086 is free to execute other instructions, including ones that access memory.

The 80286 substantially complicates this scenario. It has a memory-management unit (MMU) that is made up of two of the four 80286 processor units. Duplicating the 8086/8087 interface with the 80286 processor would have required a second MMU on the 80287. However, duplicating an MMU on the 80287 would have been a much bigger task than placing the BIU on the 8087. The designers of the 80286/80287 therefore opted for an I/O arrangement for passing op codes and data to the 80287. This interface is implemented with several dedicated ports on the 80287 that are fed by a dedicated DMA (direct memory access) channel on the 80286. The first word of every 80287 op code tells the 80287 how many (if any) words of data will follow. Once the 80287 has decoded its op code, it waits for its operands and then goes and performs the desired operation. One of the conclusions we draw from our benchmarks is that both pro-

Table 4: *The Savage benchmark results for the* IBM PC AT *running at* 6 *and* 9 MHz. *The* 80287 *runs at* 4, 6, *and* 8 MHz.

SAVAGE BENCHMARK
IBM AT
(time in seconds)

	Frequency of 80286	6 MHz	6 MHz	6 MHz	6 MHz	9 MHz	9 MHz	9 MHz
	Frequency of 80287	0 MHz	4 MHz	6 MHz	8 MHz	0 MHz	6 MHz	8 MHz
(1)	BASICA (no 80287)	303	—	—	—	196	—	—
(2)	BASCOM (no 80287)	44.76	—	—	—	29.2	—	—
(3)	87BASIC (with 80287)	—	5.55	4.16	3.85	—	3.68	3.14
(4)	87BASIC Inline compiler (with 80287)	—	3.46	2.42	2.2	—	2.31	1.87
(5)	True BASIC	50	50	50	50	32	32	32
(6)	BetterBASIC	29	7	6	5	150	5	4
(7)	Professional BASIC	—	9	7	7	—	6	5
(8)	MS FORTRAN	—	6.39	5.18	4.58	—	4.24	3.62
(9)	Ryan-McFarland FORTRAN	—	4.15	3.15	2.64	—	2.77	2.28
(10)	1-2-3 Retrieval Recalculation	123 104	—	—	—	84 68	—	—
(11)	1-2-3 with 80287 support Retrieval Recalculation	—	27 8.4	24 7	24 6.8	—	22 5.5	21 5.1
(12)	Symphony 1.1 Retrieval Recalculation	119.9	38 12	37 11.2	36 10.8	78.3	23 8	24 7.5

cessors get tied up during the interface procedure and that for fast executing 80287 instructions the interface time is a substantial percentage of the total execution time.

To help compensate for this loss in parallelism, the designers of the 80287 decided to incorporate an optional asynchronous clock that could be "turned up" when faster 80287s became available. To take advantage of this feature requires the computer manufacturer to build in an asynchronous clock. Since IBM and other manufacturers did not, I developed a card that allows the user not only to run an 80287 asynchronously but also to increase the clock speed as better 80287s are made available by Intel. Without such a card, the 80287 in an AT does arithmetic at 4 MHz. With the card, the speed of the 80287 ALU can be increased up to 8 MHz or higher. In all the benchmarks in this article I give AT figures for the 80287 running at 4, 6, and 8 MHz.

Tables 4 and 5 are the Savage, Common Subexpression, and Megalopolis benchmarks run above for the PC AT board. We now start to examine the bizarre side of the 80287. Looking at the second and third rows in the Com-

(continued)

Table 5: *The Common Subexpression and Megalopolis benchmark results for the* IBM PC AT *running at a variety of clock speeds.*

COMMON SUBEXPRESSION BENCHMARK
IBM AT
(time in seconds)

	Frequency of 80286	6 MHz	6 MHz	6 MHz	6 MHz	9 MHz	9 MHz	9 MHz
	Frequency of 80287	0 MHz	4 MHz	6 MHz	8 MHz	0 MHz	6 MHz	8 MHz
(1)	BASICA (no 80287)	304	—	—	—	195	—	—
(2)	BASCOM (no 80287)	18.34	—	—	—	11.12	—	—
(3)	87BASIC (with 80287)	—	17.9	15.38	14.66	—	11.81	10.71
(4)	87BASIC Inline compiler (with 80287)	—	5.82	4.23	3.79	—	3.9	3.25
(5)	True BASIC	24	24	24	24	16	16	16
(6)	BetterBASIC	228	43	39	39	58	29	27
(7)	Professional BASIC	—	63	54	53	—	39	36
(8)	MS FORTRAN	—	7.2	5.73	4.86	—	4.79	4
(9)	Ryan-McFarland FORTRAN	—	6.51	5.05	4.32	—	4.33	3.6

MEGALOPOLIS BENCHMARK

(1)	1-2-3 Recalculation	5.3	—	—	3.5	—	—	—
(2)	1-2-3 with 80287 support Recalculation	—	3.8	3.6	3.5	—	2.4	—
(3)	Symphony 1.1 Recalculation	7	5.9	5.6	5.5	4.4	3.7	—

mon Subexpression benchmark results (table 5), we notice that the IBM BASIC compiler code running on the AT takes 18.34 seconds and that 87BASIC took 17.9 seconds. Put another way, adding a 4-MHz 80287 increased the speed by only 2.4 percent. Examining the 6- and 8-MHz times, we see that even doubling the 80287 clock results in only a 25 percent speed improvement over the unaided compiler in line 2. The 9-MHz times show that the unaided 80286 running at 9 MHz is actually faster than a 9-MHz 80286 running with a 6-MHz 80287.

We did not include a sample benchmark that slows up when an 80287 is added to an AT. However, such programs exist and typically perform little floating-point arithmetic. What they actually do is use the 80287 for making floating-point comparisons. It turns out that the old Microsoft binary representation for floating-point numbers is more convenient for comparing numbers than the IEEE format. Consequently, a BASCOM program whose floating-point time is dominated by comparisons runs faster if the comparisons are done in the 80286 than if they are done in the 80287. For compiler writers who like to keep all their floating-point routines running on the same processor, it means there is now a reason to consider writing special code for the AT that uses the 80286 for floating-point comparisons. However, I don't think you'll ever see anyone wasting time on this project. Other bizarre results include the Megalopolis result for Symphony 1.1: adding an 80287 improved this benchmark by only 18 percent on an AT. Again, increasing the 80287 speed to 8 MHz only increases the 18 percent speedup to 27 percent. To understand these results we have to examine the relative speeds of the two processor sets that we are comparing.

To explain the small or nonexistent speedups observed when an 80287 is added to an AT and run with inefficient code, we have to look at the relative power of the processor combinations at doing floating-point arithmetic. If we add an 8087 to a PC and get a factor of 3 speedup, this tells us that the 8087 does floating-point arithmetic in hardware three times faster than an 8088 using software. If we now take the same program and run it on a machine with a slower numeric processor and a faster main CPU, the speedup upon adding the numeric processor will be less significant. This is exactly what happens in the AT. The 80286 is three times as fast as an 8088 at doing arithmetic, and this makes the slightly slower 80287 look much less effective at speeding up programs. A program that gives us a factor of 36

(continued)

Table 6: *The* FORTRAN *array-manipulations benchmark results for both the* PC *and the* AT.

MS FORTRAN ARRAY-MANIPULATIONS BENCHMARK
(kiloflops)

Small Memory Model, Integer*2

	IBM PC		IBM PC AT				
Frequency of processor	4.77 MHz	9.54 MHz	6 MHz	6 MHz	6 MHz	9 MHz	9 MHz
Frequency of coprocessor	4.77 MHz	9.54 MHz	4 MHz	6 MHz	8 MHz	6 MHz	8 MHz
Subroutine 1 One dimension	13.7	32.2	13.9	17.8	20	21	24.6
Subroutine 2 Two dimensions	5.9	14.4	11.85	14.16	15.2	17.8	20.4
Subroutine 3 Three dimensions	2.8	6.6	8.4	9.06	9.4	12.7	13.5
Large Memory Model, Integer*2							
Subroutine 1 One dimension	8.3	21.3	12.5	14.9	16.5	18.9	21.3
Subroutine 2 Two dimensions	4.5	11.2	10.7	11.8	12.5	16.1	17.5
Subroutine 3 Three dimensions	1.7	4.1	5.5	5.6	5.8	8.2	8.5

PROVEN!

PC/FOCUS outloads, outruns, outperforms and outreports dBase III™ and R:Base™ 5000.

Don't believe us. Believe National Software Testing Laboratories of Philadelphia. They proved PC/FOCUS to be clearly superior to dBase III™ and R:Base™ 5000. Here's how:

Several real-world, business situations were created to examine each system's speed and capability.

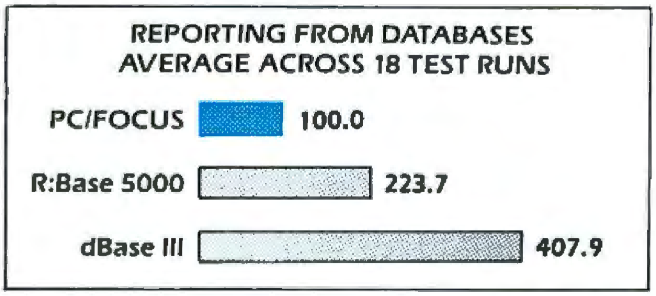
Their findings are published in an extensive report now available to you free of charge. Their results, as stated by NSTL, showed that..."PC/FOCUS was faster overall than the other programs tested." For example...

Database Loading:

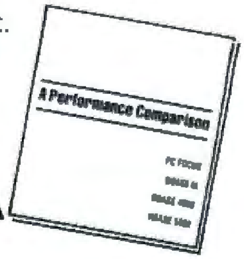
In this test series, database loading time for each DBMS was measured by loading three transaction files. Eight separate test runs were conducted, with PC/FOCUS averaging 20.2% faster than dBase III,™ and 37.3% faster than R:Base™ 5000.

Database Reporting:

Report requests of various levels of complexity were run against single and multiple databases. Eighteen separate test runs were conducted, with PC/FOCUS averaging 45.2% faster than dBase III,™ and 40.0% faster than R:Base™ 5000.



There's more to the study than can be described in this ad. Lots more tests...and much more proof. So, we've reproduced the NSTL Performance Comparison in full. To obtain your free copy, just fill out the coupon and mail it to Donald Wszolek, Dept. U9, Information Builders, Inc., 1250 Broadway, New York, NY 10001.



dBase III is a trademark of Ashton-Tate.
R:Base 5000 is a trademark of Microrim.

Dear Don:
I want proof! Please send my free copy of "A Performance Comparison" to:

Name____________________
Title____________________
Company____________________
Address____________________
City__________ State______ Zip______

PC/FOCUS
INFORMATION BUILDERS, INC.

New York (212) 736-4433 • Washington, D.C.: (703) 276-9006 • Chicago: (312) 789-0515 • St. Louis: (314) 434-7500 • Dallas: (214) 659-9890
San Jose: (408) 293-6600 • Los Angeles: (213) 615-0735 • Houston: (713) 952-0260 • Boston: (617) 272-8600 • Atlanta: (404) 980-0474

Inquiry 94

speedup upon adding an 8087 to a PC gives us a factor of 12 speedup upon adding an 80287 to an AT. However, a program that gives a factor of 3 speedup on a PC gives us a factor of 1, i.e., nothing. For example, the Savage mark for 1-2-3 showed a speedup of 36.6 when run on a PC but only 12.4 when run on an AT. This rule of a 3-to-1 reduction does not apply strictly as speedups get small. For example, Megalopolis evaluated with 1-2-3 speeds up 100 percent on a PC but only 39 percent when run on an AT. Looking at the Symphony results, we see that the 30 percent speedup for the PC goes down to 18 percent for the AT.

For the Common Subexpression benchmark the excellent performance of the 8087 in comparison to the 80287 is a result of the nature of the code. Because it makes good use of the stack, in-line code consists mostly of internal stack operations. These operations involve very little 8087 I/O, and because they execute the fastest they are also the most affected by the 80286–80287 interface losses. For the sake of argument, assume that the 80286–80287 interface takes up 30 percent of the numeric time in a standard AT running with a 4-MHz 80287. Doubling the 80287 clock to 8 MHz would have an effect on the numeric portion of the cycle only. Since our assumption is that the interface takes 30 percent, that means the arithmetic at 4 MHz takes 70 percent. Halving the 70 percent results in a process that takes 65 percent as much time (instead of 50 percent). This is virtually the result we get comparing the 4- and 8-MHz columns in table 5 for fast-running compilers.

Looking at table 4, we finally see an area where the AT really shines: running inefficient spreadsheets fast. For example, while the fastest time for Savage is Lotus 1-2-3 on a 9.54-MHz 8086/8087 board (4.8 seconds versus 5.1 for the souped-up AT), the fastest Megalopolis time is on the AT (2.4 seconds versus 3.3 on a 9.54-MHz 8086/8087 board). Comparing Megalopolis run on Symphony, we now see that the souped-up AT comes in at 3.7 seconds versus 8.3 for the 9.54-MHz 8086/8087 board. In going from an efficient spreadsheet to a not-so-efficient spreadsheet, our souped-up AT went from a 50 percent advantage to a 100 percent advantage. Unfortunately for the AT, we had to run the 80286 well out of limits to get the good times.

The last result was to be expected. Early in our AT experiences we discovered that the one area in which the AT shined was in computing indexes in multidimensional arrays. This is clearly shown in table 6, which compares three FORTRAN subroutines that do simple array arithmetic.

The core of the first subroutine has an inner loop that gets executed a total of 100,000 times. The operations performed in the loop include multiplication, division, and addition between members of five single-dimension arrays. The total arithmetic performed is 500,000 floating-point operations. Dividing the number of the operations by the time to perform the total benchmarks yields the processor throughput in kiloflops (thousands of floating-point operations) per second. The second subroutine is nearly identical except that it performs the same sequence on members of two-dimensional arrays. The third subroutine repeats using three-dimensional arrays. All three routines were compiled using MS FORTRAN, integer*2 indexes, and the small and large memory models. The results are similar for integer*4 indexes but not shown. Comparing the AT with a PC, we see that the times for both machines for vector operations are nearly the same, while the AT takes the lead for the matrix operations and takes a commanding lead for the tensor operations. For the small model, these results can be interpreted as meaning that the PC and AT are 8087- or 80287-bound for the vector case, but the PC is index-bound for the matrix and tensor cases, while the AT is almost number-bound for the matrix case and definitely index-bound by the tensor case. Looking at the large model, we see that the indexes dominate the PC execution and are just starting to dominate the AT vectors, while they definitely dominate the matrix and tensor operations. Although we do not show it, the times for RM/FORTRAN are better. In general, the souped-up AT fares better against the 9.54-MHz 8086/8087 board, tying it for vectors, running 50 percent faster for matrices and 100 percent faster for tensors. These results are similar to those for Symphony, where the souped-up AT was 100 percent better than a 9.54-MHz 8086/8087 board for running Megalopolis. In fact, since addressing and floating-point arithmetic are going on concurrently in both of these examples, we can conclude that array addressing takes twice as long on a 9.54-MHz 8086 as it does on a 9-MHz 80286.

While the work just done suggests that the AT is superior for matrix work, that's really not the case. These benchmarks were written to test the quality of micro-generated indexing. In a mainframe FORTRAN it would have been possible to remain number-bound through all three subroutines. In addition, hand-coded routines that use pointers to address matrices are number-bound for the 9.54-MHz 8086/8087 board and run faster on it than the AT. While it is not always practical to rewrite floating-point programs so that they are really number-bound, it usually is possible. If this is done we can make a general statement: Any problem that can be cast into a number-bound algorithm will run fastest on the machine with the fastest numeric processor. We make this last statement by deduction. Numeric processors by definition are devices that do floating-point arithmetic faster than the processor they work with. If a floating-point processor were beat by a non-floating-point processor, then it could hardly claim to be a floating-point processor. Therefore, any problem that is bound by numerics will be solved fastest by the fastest numeric processor.

The last benchmark typifies the kind of problem I really wanted to concentrate on in this article. For integer*4 indexes, the vector marks are dominated by floating-point operations, while the matrix and tensor marks are dominated by the overhead associated with computing the addresses in the matrices and tensors. If we change to integer*2, we discover that the vector and matrix times are the same, indicating that they are both floating-point-bound and the tensors bind in addressing. This is the kind of problem that can be handled well with analytic techniques, and the ability to change relative speeds of processors provides a nice basis for verifying concepts about throughputs for processor/coprocessor systems as a function of the clock speeds and code being run.

CONCLUSIONS

For number-bound programs, the fastest computer will always be the one with the fastest numeric processor. This was the case for all the benchmarks we ran when they were coded with the most efficient compilers. Unfortunately, if a program is not number-bound but bound by other system bottlenecks such as addressing variables, then the fastest machine is not always the one with the fastest numeric processor. Often problems that appear to be number-bound turn out to be CPU-bound. This is especially the case with programs that do not use native in-line 8087 support.

It was also discovered that the 80286 in the AT was much closer in numeric speed to an 80287 than the 8088 was to an 8087. In general, this resulted in a reduction in the speedup obtained when an 80287 is added to an AT. As a rule of thumb, the 80287 speedups observed for an AT are one-third those obtained when an 8087 is added to the PC. In situations where programs speed up less than a factor of 2 on a PC, they can slow up when run with an 80287, especially when floating-point comparisons become an important part of a problem.

One of the disappointments of the 80287 support is spreadsheets. While 8087 support for spreadsheets is usually good enough to be worthwhile, adding an 80287 often results in speedups that are less than 50 percent. This is especially true of the types of worksheets that many users like to generate. That's because most spreadsheets run as interpreters, which are much less efficient than native-code compilers. On the positive side, the 80286 does execute interpreted code much faster than an 8088, and the best times for the huge sheet, Megalopolis, were turned in by a souped-up AT running a 9-MHz 80286 with an 8-MHz 80287.

After examining 80287 performance we came to the conclusion that the 8087 is slightly more efficient

(continued)

than the 80287 for executing well-written programs. Even our experimental AT turned out to be 40 percent slower than the 9.54-MHz 8086/8087 board for running number-bound problems. Extrapolating this result to future machines, it takes an 11-MHz 80286/80287 to keep up with the 9.54-MHz 8086/8087 running in-line code. However, since we are currently running experimental 9.54-MHz 8086/8087 boards at 12 MHz, it will really take a 14-MHz 80287 to keep up. We conclude that for number-bound applications the 8086/8087 upgrade path is a viable alternative to 80286-based machines and will remain so for the next one to two years and possibly longer depending on the availability of 8087 upgrades.

With improved 8087s or their equivalent, it is theoretically possible to achieve floating-point improvements of up to a factor of 10 over the current 8087 without changing the existing processor/coprocessor interface or software. Whether or not this will happen is a function of the large semiconductor manufacturers. In their eyes the 8087 is a niche market, which may or may not warrant additional R&D expenditures. From a marketing standpoint, souping up your competitor's architecture is not necessarily good business. However, because the Intel architecture has now assumed the stature of a standard, it is fair game as an aftermarket. In fact, at least one semiconductor house, NEC, has an 8087 upgrade in the pipeline that is supposed to be pin-for-pin- and software-compatible and a factor of 2 faster than the 8087. The other contenders in this marketplace include National Semiconductor and Motorola. The late-to-the-market Motorola 68881 runs up to a factor of 7 faster than a 5-MHz 8087 doing simple operations and up to a factor of 25 faster doing trigonometrics. The Motorola part is not pin- or code-compatible with the 8087. While it might be possible to glue it into a PC bus, the overhead would end up eliminating a good part of the speed gained.

If a user is running his or her own programs and doesn't mind recompiling, there are a number of new devices coming on the market that will make it possible to achieve PC floating-point performance in the 1- to 4-megaflop (millions of floating-point operations)-per-second supermini range for about $2000/megaflop. All of these new devices require the use of boards, special interface software, and/or special compilers. The compilers will be trickier than even the current 8087 compilers because these new devices perform floating-point arithmetic about as fast as an 8086 performs an inter-register move. Since code and data will have to get moved onto such a board for it to run up to its potential, it will have to have large chunks of data and code preloaded and repetitively used.

An example of the latest generation of floating-point devices that can be used to construct a coprocessor is the Weitek 1164/1165 chip set. Like the 68881, it would have to be glued into a PC with discrete logic. This processor can be memory-mapped and designed with a high-speed register cache that holds up to 1000 double-precision constants or variables. Leaving hot variables in a processor reduces I/O, and, since the instruction set of this processor includes primitive branches, it is possible to download whole code fragments that will execute in the coprocessor at very high speeds. In fact, the Weitek chip set discussed here is capable of achieving floating-point throughputs in excess of 2 megaflops, which is a factor of 50 greater than an 8087.

In the last three years microcomputers have gone from floating-point speeds of 1000 operations per second up to 80,000 operations per second. The next three years promise an explosion in this field with new players, new devices, and new methods of organizing floating-point calculations. The expectation of having user workstations whose speeds are measured in megaflops by the year 1988 seems quite probable. Some OEMs are talking about dedicated boards running in microcomputers that perform specific operations such as matrix inversions and fast Fourier transforms in the 100-megaflop speed range. While much of this new technology will be expensive, it will still be very cost-effective. ■

Let's
Stop Playing
"Cat & Mouse" ...

good reasons Manager Mouse™

Manager Mouse is an innovation in Mouse technology.

It is indispensable to many windowing, computer aided design, graphics, project management and word processing programs. Its unique independent suspension* system delivers smooth, skip-free performance on any surface, at any angle with precise control. It plugs directly into your computer without any external power supply and requires no special tablets or pads. It's totally portable. Take it with you in its travel pouch anywhere in the world.

Manager Mouse's enclosed drive mechanism makes for carefree maintenance as well. Manager Mouse is designed for use with IBM PC,™ XT,™ AT™ and a growing number of compatible personal computers, as well as an ever increasing list of software. It comes with its own TMouse™ software, a comprehensive user's manual and technical reference data, a six month limited warranty, and a protective travel pouch.

*Patent Pending

A standard RS-232-C interface* makes Manager Mouse compatible with these computers.

IBM PC,™ PC-XT,™ PC-AT™ and PC Portable™

COMPAQ Portable,™ Plus,™ 286™ and DeskPro™

ATT 6300™

Eagle PC,™ PC Plus,™ Spirit™

Ericsson™

Columbia™

Corona™

TI Professional™

DEC Rainbow 100™

ITT Extra™

Datavue 25™

Sperry PC™

Leading Edge Model D,™ Model M™

Morrow Pivot™

Televideo 1605™

Tandy 1000,™ Tandy 1200 HD,™ Tandy 2000™

Panasonic Senior Partner,™ Executive Partner™

Burroughs ET-2000™

Kaypro™

*Please check the type of connector on your computer's serial card to be sure to buy the right Manager Mouse for your computer.

MANAGER™ mouse

Call 1-800-982-0030 for your nearest dealer.

TORRINGTON

Part of worldwide Ingersoll-Rand

59 Field Street
Torrington, CT 06790

Inquiry 202

MOVING FROM THE 8088 TO THE 80286

Important differences you need to know to make your programs transportable

WILLIAM J. CLAFF

THIS GUIDE to writing assembly-language code for the IBM PC family pays particular attention to the impact of the 80286 on current programming methods. With the growing number of 80286-based machines, including the IBM PC AT (and probably the anticipated IBM "PC2"), the differences between the 80286 and the 8088 become quite important to the software developer. This is especially true when multitasking and multiuser operating systems acquire a larger installed base. These differences also highlight the ever-present need to structure programs for change.

This article is not a primer. It assumes that you are familiar with assembly-language concepts and does not cover the expanded instruction set of the 80286 or how to write systems software. Its primary aim is to acquaint you with the specific differences between the 80286 and the 8088. It also attempts to convey that change is inevitable and programs must be structured accordingly.

Since the 80286 is a superset of the 8088, let's review that chip first and then cover those features of the 80286 that are important to applications, as opposed to systems, programmers.

THE 8088 MICROPROCESSOR

Figure 1 shows an elementary block diagram of the 8088. This processor has two separate processing units: the execution unit (EU), which executes instructions, and the bus interface unit (BIU), which is responsible for the 8088's communication with the outside world. The EU provides a logical address to the BIU, which translates it into a physical address. This translation, called the physical-address computation, uses two 16-bit quantities: a segment register and an offset. The notation used for logical addresses is *segment:offset*. The segment registers (parts of the BIU) are code segment (CS), data segment (DS), stack segment (SS), and extra segment (ES). The offset is usually supplied by the EU.

To compute the physical address, the 8088 shifts the segment register left 4 bits and adds the offset in the BIU's dedicated adder, Σ. Segments are

(continued)

William J. Claff (7 Roberts Rd., Wellesley, MA 02181) is manager of special projects for Spinnaker Software Corporation. He has also run his own microcomputer retail and consulting business since 1978. Mr. Claff holds an M.S. in applied mathematics from Harvard University.

For increased efficiency, the BIU pipelines bytes.

64K-byte relocatable pieces of the 1-megabyte physical-address space. They are located on 16-byte boundaries called paragraphs. Since assembly-language programs are written in logical segments, the placement of these segments in memory is a function of the linker and DOS. They can be overlapped, contiguous, or disjointed.

The address of the next instruction to be executed is CS:IP (code segment:instruction pointer). For increased efficiency, the BIU *pipelines* bytes (prefetches them and puts them into a queue). To facilitate this calculation, the instruction-pointer register is kept in the BIU.

The EU contains eight 16-bit registers, any of which can be used in computations. Four of these registers comprise the data group. They are the accumulator (AX), base (BX), count (CX), and data (DX) registers. The 8088 can also access the high and low 8 bits of each data register. The two halves of the accumulator register, for example, are AH (accumulator high) and AL (accumulator low). The respective halves of the BX, CX, and DX registers are similarly named.

The next two general registers, the stack pointer (SP) and the base pointer (BP), constitute the pointer group. These registers manipulate the stack. When a subroutine is invoked, SS:SP (stack segment:stack pointer) stores the return address on the stack. SP points to the top of the stack and BP to the base. SP is automatically decremented by calls and incremented by returns. The stack is also used to pass subroutine parameters. BP accesses these parameters.

Two other general registers, the destination index (DI) and the source index (SI), make up the index group and are used primarily in string operations. Two segment registers are re-

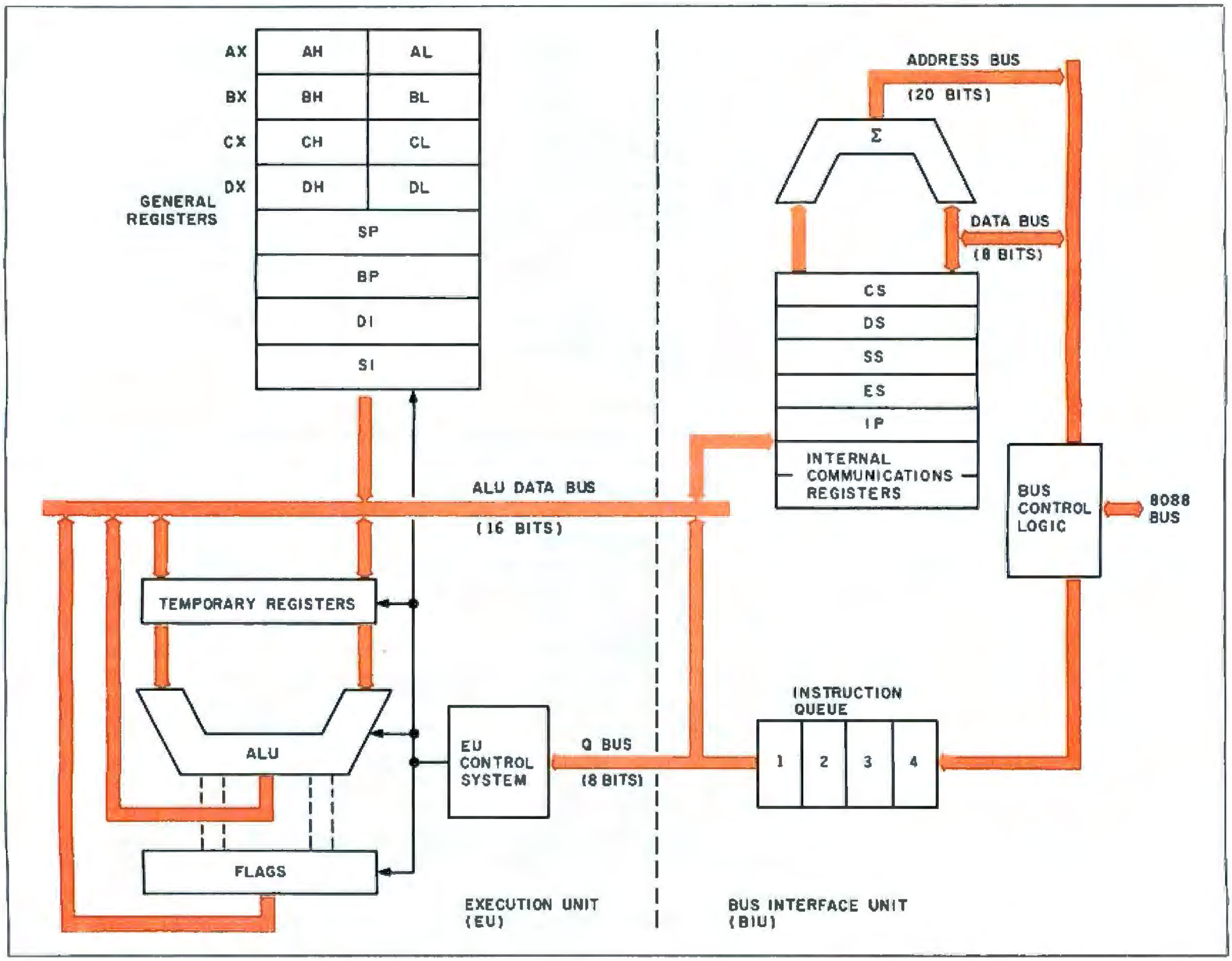


Figure 1: *An 8088 elementary block diagram (see reference 1).*

quired to perform moves or comparisons on memory more than 64K bytes apart. This is why there is an ES in the BIU in addition to the DS. The destination in a string operation is always ES:DI (extra segment:destination index).

Many of the registers in the EU have special uses. Table 1 shows these registers and their uses.

8088 Addressing

The EU generates an effective address (offset) using one of several methods called addressing modes. An effective address has one or more of the following: base, index, and displacement. A base can be BX or BP; an index can be either SI or DI; and a displacement is a 16-bit signed number.

If you do not specify a segment register, the 8088 uses the DS register. If you specify the BP register as the base, it uses SP as the segment register. Supplying a segment register other than the default is called using a *segment-override prefix*. However, you cannot override the IP, SP, or DI register in string operations. Figure 2 shows how the various addressing modes in the EU and the BIU combine to form the physical address.

8088 Interrupts

The 8088 does not distinguish between interrupts invoked by the assembly-language instruction INT and those generated by the hardware. There are 256 interrupts, vectored through a table of double words found at location 00000:00000 hexadecimal. Each double-word entry in the table corresponds to the CS:IP of the subroutine that the interrupt invokes. The 8088 uses interrupts 000 through 004 hexadecimal for the following errors: divide by zero, single step, nonmaskable interrupt, breakpoint, and overflow.

Table 1: *The 8088's implicit use of general registers (see reference 1).*

Register	Operations
AX	Word multiply, word divide, word I/O
AL	Byte multiply, byte divide, byte I/O, translate, decimal arithmetic
AH	Byte multiply, byte divide
BX	Translate
CX	String operations, loops
CL	Variable shift, variable rotate
DX	Word multiply, word divide, indirect I/O
SP	Stack operations
I	String operations
DI	String operations

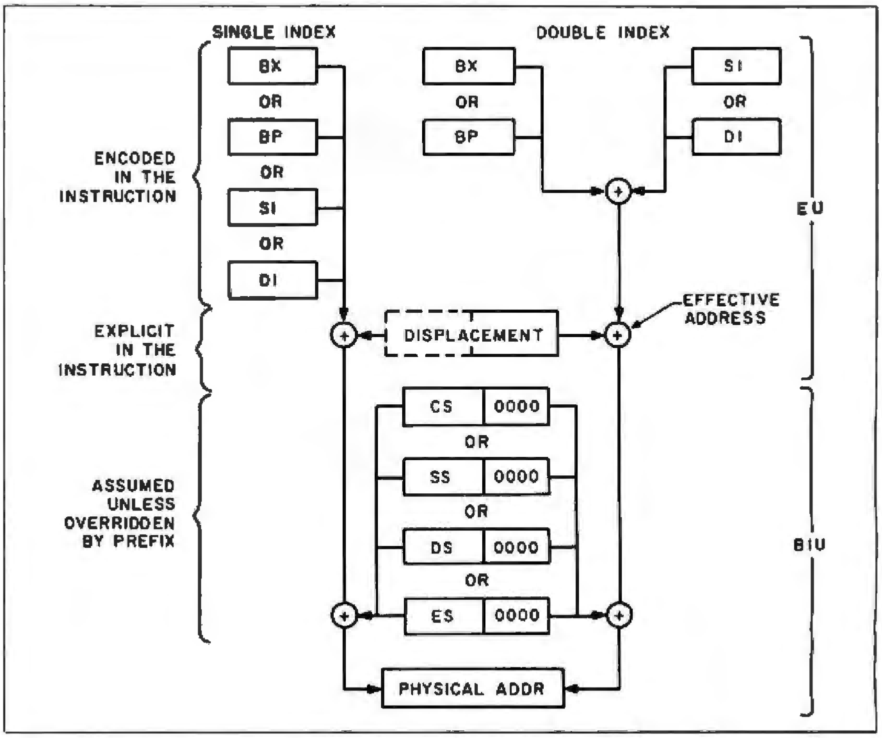


Figure 2: *The 8088 memory-address computation (see reference 1).*

The 80286 Central Processing Unit

Figure 3 shows an elementary block diagram of the 80286. This processor has four separate processing units: the EU, the bus unit (BU), the instruction unit (IU), and the address unit (AU). The 80286 operates in either real-address mode or protected virtual-address mode (protected mode). Bits in a new register, the machine status word (MSW), control the processor mode. The machine status word also activates a feature of the 80286 that allows for the emulation of coprocessors such as the 80287. Let's examine the components as they operate in real-address mode first.

The IU is a further refinement of pipelining. The BU prefetches up to 6 bytes of instructions; however, instead of being decoded *by* the EU as in the 8088, the IU decodes them in parallel execution *with* the EU. This improves the speed of the 80286 but has no impact on programming.

The 80286's BU and AU operate in essentially the same way as the 8088's BIU does. Specifically, the AU calculates the physical addresses in the same manner.

80286 Execution Unit

While the EU executes a superset of the 8088 instruction set, some instruc-

(continued)

80286 Memory-Management System

BY STEVE SATCHELL

In designing the 80286 in 1982, Intel implemented what most of us think of as virtual memory. But how does it work? Quite simply, a virtual-memory system takes the address of a memory request from a processor and translates it to reference a location in RAM (random-access read/write memory). The translation from the logical address to the physical address is performed by dedicated hardware. The logical-memory space can be smaller than, the same size as, or larger than the physical-memory space.

Let's illustrate with the 8086 family of processors. First, the 8086 has a physical-address space of 1 megabyte, but a logical-address space of four 64K-byte segments, or 256K bytes. Therefore, we are mapping a smaller logical-address space onto a larger physical one. Whenever the 8086 executes an instruction that accesses memory, the base address in the desired segment selector is added to the offset from the instruction. The segment selection is either explicitly specified or implied by the instruction executed. The resulting 20-bit address is placed on the system bus as the address of the operation. The map function is simple in the case of the 8086: Take the segment selector, multiply by 16, and add that to the offset to give you the physical address.

When the total logical-address space is larger than the physical-address space, part of the data is stored outside of physical RAM, usually on a disk or drum store. The memory system then needs to know what locations are present in physical RAM. When the memory system cannot find a requested location, it generates an interrupt to the operating-system software. The operating system then takes corrective action, usually by loading the desired data from disk into physical RAM. This is called "swapping."

High-performance mainframes use two or more levels of memory. The first level is very fast—10-nanosecond ECL (emitter-coupled logic) cache RAM—and very expensive. The second level is slower—400-nanosecond NMOS (negative-channel metal-oxide semiconductor) RAM—and much less expensive. The third level is disk, which is very cheap in comparison to the cost of the ECL RAM.

Most modern virtual-memory systems also implement access checking. When the program requests access to a memory location, the memory hardware makes sure that the access is valid. Customary tests are for read access, write access, and instruction fetch (execute access). The 8086's virtual-memory system does *not* have any form of access checking at all.

Types of Virtual-Memory Systems

Nonmapped Memory: The simplest memory system has identical logical and physical addresses. The translation is fixed by design. You wouldn't call this a virtual-memory system, but it is a very basic form of one. The 8080 and 8085 are good examples of this scheme. A given logical address always references the same RAM location, regardless of the state of the processor.

Paged Memory: Early designers first increased the amount of physical memory on a system by setting aside a block of logical memory and mapping two or more physical blocks—called banks—into that one logical space. This enabled systems to grow above the limited address space and was usually used for multiuser systems. Then the designers defined multiple blocks that allowed two or more banks to be mapped in at the same time. You could then move data from one bank to another instead of switching banks all the time.

If you carry this process to its inevitable conclusion, the designer defines a set of pages, all the same size, in both logical and physical space. The system maps logical pages into physical ones using a table. The logical address is then broken into a page selector and an offset. The address-translation hardware replaces the page selector with the upper *n* bits of the physical-page address and passes the offset through as is. This then becomes the physical address.

Fitting new processes into a paged memory system is relatively easy since the operating system doesn't have to find a block of memory big enough to hold the process; it must only collect enough pages to satisfy the space requirements for the program. Need to add memory for the program? Get another page.

Such a system also allows limited data sharing, especially if access checking is implemented. If you have multiple users of the same program, you can save time and RAM by having everyone use the same copy. Then you need one set of pages to hold the program and a small working set of pages for each user's data.

The main problem with paged memory is wasted RAM. If you have many programs and each program is not optimized to fit exactly in one or more pages, you lose on the average half a page per program. If you choose too big a page size, the losses mount rapidly. If the page size is too small, you increase the size of the mapping table, thus increasing system cost and decreasing system reliability.

Segmented Memory: You can also vary the size of each memory area to fill your needs. As a minimum, you need to know the segment's physical starting address. Some segmented memory systems use a portion of the logical address to select the segment (as in the paged system) while the remainder defines the offset within the segment. Others, like the 8086 and the 8088, keep the segment specification separate from the rest of the logical address. In reality, the 8086 doesn't have a 16-bit logical address; it has an 18-bit one: 16 bits of offset and 2 bits of segment specification.

Systems with access checking also carry the size of the segment to be sure that the access request is within bounds; if it is not, the operating system is informed.

Since the segment is only as large as it needs to be, you can tightly pack RAM with code and data. However, in multiuser systems, programs come and go with great frequency, so you can end up with a lot of chunks of free memory too small to use.

If you don't have enough contiguous

space for a new program, you need to garbage-collect your RAM to gather those chunks of unused memory together and readjust the segment definitions accordingly. This is a time-consuming process and, if done frequently, can really kill system throughput. Sometimes, however, it can mean the difference between a system that runs and one that doesn't.

THE 80286 MEMORY MODEL

The 80286 implements segmented virtual memory by using in-memory tables that contain copies of information about segments currently in use on the RAM chips. Each memory segment has associated with it a descriptor containing the segment's physical-memory base address, its length in bytes, its access restrictions, and two bits for memory systems that swap segments in and out, present and accessed. Whenever the system loads a selector into a segment register, it saves the old descriptor and loads the new descriptor from memory into the 80286.

These descriptors are grouped into tables, which are described by higher-order descriptors. There is one descriptor table for each task in the system and a global descriptor table as well.

The actual implementation involves a chain of memory segments and tables. For complete information on how these work, see reference 4.

EFFECTS ON OPERATING SYSTEMS AND APPLICATIONS

Operating systems and application programs are sensitive to the environment in which they run. Since the memory system for the 80286 protected mode differs significantly from the memory system for the 8086, there are some effects.

MS-DOS: MS-DOS was developed specifically for the 8086 family of processors. Microsoft tended to ignore the Intel guidelines for program structure. Intel developed these guidelines to minimize the pain of upgrading from one processor to the next, such as from the 8086 to the 80286.

Further, for the sake of speed many applications were written in assembler, instead of in high-level languages like C or Pascal. This means that you must examine your code and, where you made use of specific details of the 8088/8086, modify that code to run in the 80286 environment. High-level language programs require considerably less modification.

The operating system is most affected by the 80286's new design. MS-DOS must be extensively modified to run in the protected environment, even for single-user, single-task work. The tables required to let programs run in protected mode must be set up, making it difficult for existing programs and MS-DOS to move into the 80286's protected environment. Since most software changes required by the 80286 center on the operating system, it could mean a complete rewrite.

UNIX: The 80286 was developed with UNIX in mind, and many of its features are appropriate to a UNIX kernel. The 80286 is best suited to UNIX tasks that are small and numerous, as opposed to gigantic programs, since the 64K-byte-per-segment restriction is still present with the 80286.

The IBM PC AT, however, is a single-processor machine with no easy method of inserting additional 80286 processors. Coprocessors cannot use the PC AT system RAM directly. This means the AT must remain as a high-end, single-user and low-end, multiuser machine. Most serious UNIX machines use one processor for applications programs and one or more for I/O (input/output). For example, the Sun Microsystems 2/120 system uses a 68000 microprocessor for computation and at least one 80186 processor for I/O functions.

GRAPHICS, SCIENTIFIC APPLICATIONS: Graphics applications and the 80286 are not suited for each other. I'm not talking about Lotus 1-2-3 pie charts, but CAD (computer-aided design), picture processing, and windows on large displays. These graphics applications require large arrays to store and manipulate the information they need. Just to display 1024 by 1024 dots in 256 colors requires 1 megabyte of display memory alone!

Scientific processing must also take a back seat, since some statistical algorithms require very large arrays. Currently, the systems using the Motorola 68000 family—such as the Macintosh and the AT&T UNIX PC—and the National Semiconductor 32000 family are better suited to those tasks.

INTEL 80386 VS. 80286

The 80386, Intel's second 32-bit microprocessor, is forthcoming. Unlike the iAPX 432, the 80386 will build on the concepts started with the 80286 and add some more interesting capabilities.

If you combine the two different methods of implementing virtual memory, segmented and paged, you can get the best of both worlds. Segment descriptors carry all the access-checking information, and it is easier to swap fixed-size pages to and from disk. If a page-modified or "dirty" bit is assigned on a page basis, you don't need to swap out an entire segment just because you changed one word; you can just swap the page.

Limited garbage collection is much easier with this combined memory scheme, since you only need to manipulate *parts* of pages instead of entire segments; this is especially true if a segment starts in the middle of a page. Full garbage collection is just as tedious, however, since you need to move data around in all the pages of the segment.

Intel has shown plans to implement a combined segmented and paged virtual-memory system on the 80386 that is upward-compatible from the 80286 system. Segments carry access restrictions just as they do on the 80286, but a paged system makes memory allocation and deallocation easier. Instead of swapping segments, you have your choice of swapping pages, segments, or some combination thereof.

Stephen Satchell is the owner of Satchell Evaluations (16 Searing Ave., Morristown, NJ 07960). He has spent 12 years in computer product development and evaluation.

tions operate slightly differently, posing a compatibility problem. These differences are summarized as follows but are of no consequence in most applications programming:

• Shift counts are masked to reduce the maximum instruction time. The CL register is masked when it is used as a shift or rotate count.
• PUSH SP works differently. Due to protected mode, the value of PUSH SP is different on the 80286. If this is important, you should use the following instructions:

```
PUSH    BP
MOV     BP,SP
XCHG    BP,[SP]
```

• Flag word has a different value. The upper 4 bits of the flag word are 1111 on an 8088 and 0000 on an 80286 operating in real-address mode. (Note: This provides a way of telling these processors apart programmatically.)
• Quotients of 80 or 8000 hexadecimal are possible.
• Divide error is restartable.
• Segment wraparound causes exception D hexadecimal.
• External interrupt handlers cannot be single stepped. The priority of the single-step interrupt has been changed. This keeps an external interrupt from being single-stepped if it occurs while single-stepping through a program.
• Interrupts can occur after MOV/POP DS/ES. The 80286 only ignores interrupts after a MOV/POP SS instruction.
• Do not rely on NMI (nonmaskable interrupt) interrupting the NMI handler. The 80286 disables NMI and processor-extension interrupts after recognizing an NMI; they remain disabled until the first IRET is executed.
• Place a far jump at FFFF0 hexadecimal. The 80286 starts execution at F000:FFF0 hexadecimal as opposed to FFFF:0000 for the 8088.
• Do not duplicate prefixes. The prefetch and instruction unit impose a 10-byte instruction-length limit that you can reach only if you code redundant prefixes.

Table 2: *The 80286's predefined interrupt vectors (see reference 3).*

00	Divide-error exception
01	Single-step interrupt
02	Nonmaskable interrupt
03	Breakpoint interrupt
04	INTO detected overflow exception
05	Bound RANGE exceeded exception
06	Invalid op-code exception
07	Processor-extension not-present trap
08	Double protection exception
09	Processor-extension segment overrun exception
0A	Task segment format exception
0B	Segment not-present exception
0C	Stack under-/overflow exception
0D	General protection exception
0E–0F	Reserved
10	Processor-extension-error interrupt
11–1F	Reserved

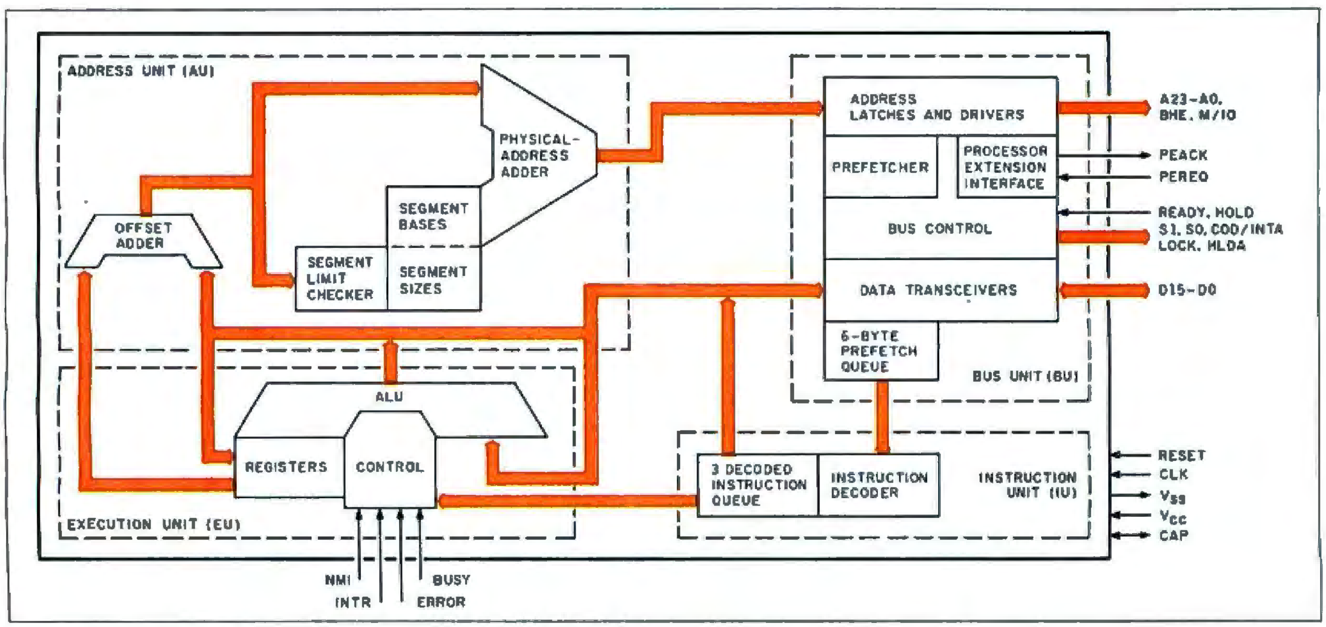


Figure 3: *An 80286 internal block diagram (see reference 2).*

• Do not use undefined op codes, in particular POP CS or MOV CS,op or POP/PUSH mem with undefined encodings.
• Self-modifying code may not work. Since the 80286 can prefetch further ahead, you should jump to self-modifying code. (Note: Self-modifying code will not be compatible with the 80386 microprocessor.)
• The numeric-exception handler must use interrupt 10 hexadecimal.
• The numeric-exception handler must allow for prefixes.
• The numeric-exception handler must not use the 8259A-chip INT signal.
• FNDISI and FDISI do not disable numeric interrupts.
• Do not perform I/O (input/output) to ports F8–FD hexadecimal.
• Avoid operations that the iAPX 286 may restrict to ensure system integrity, low interrupt latency, or low bus-request latencies (for example, shift/rotate with shift count greater than 31, locked CMPS/STOS/SCAS/LODS, STI, CLI, HALT, and I/O instructions).
• Do not rely on the value pushed onto the stack by PUSH SP.
• Do not rely on processor instruction-execution times.

80286 Interrupts

Another difference between the 8088 and the 80286 operating in real-address mode is in their handling of interrupt vectors. Table 2 lists the 80286's predefined interrupt vectors. These new interrupt vectors would not be a problem if IBM had avoided using Intel's reserved interrupts in designing the PC. For example, a processor-extension error (interrupt 10 hexadecimal for the 80286 in table 2) causes a random video interrupt (the IBM PC's use of interrupt 10 hexadecimal) to occur. These new interrupts are unlikely to occur in real-address mode, and you can trap them in protected mode, so this may not turn out to be as great a problem as it appears.

Protected Mode

Examination of the processor components as they operate in protected mode brings the most significant difference between the 8088 and the 80286 to light. Since most 80286-based systems are currently operating in real-address mode, this difference has not yet become a major problem.

Figure 4 shows the complete 80286 register set when operating in protected mode. There are several new registers, some of which are not programmer accessible. In protected mode the AU provides full memory management, protection, and virtual-memory support. To do this, the AU sets up operating-system control tables in memory that describe all of the machine's memory, and then the hardware enforces the information in these tables.

The 80286 extends the 8088's 16-bit segment registers into 64-bit segment selectors by appending a 48-bit segment descriptor taken from a descriptor table that uses the segment register as an index. Using the segment descriptor to hold this informa-

(continued)

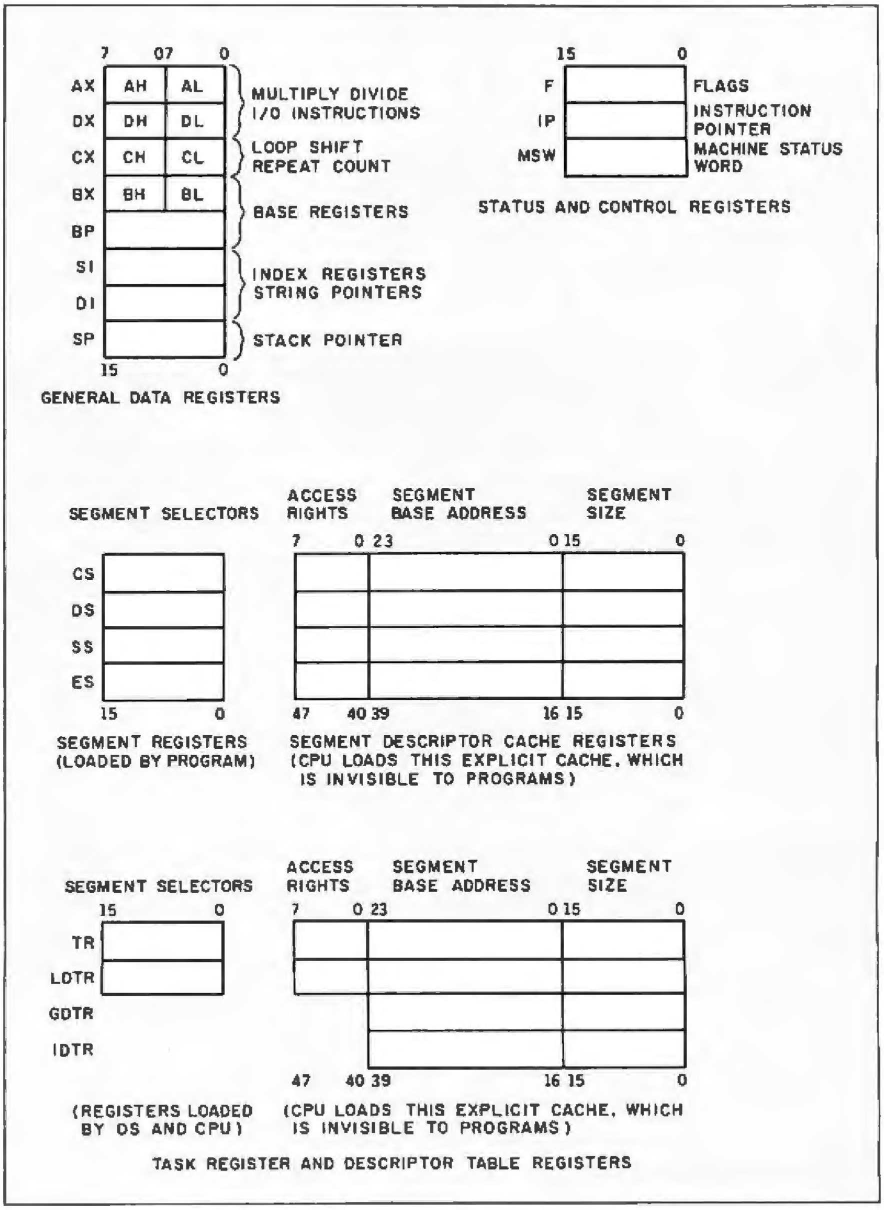


Figure 4: *The full register set for the* 80286 *(see reference* 3).

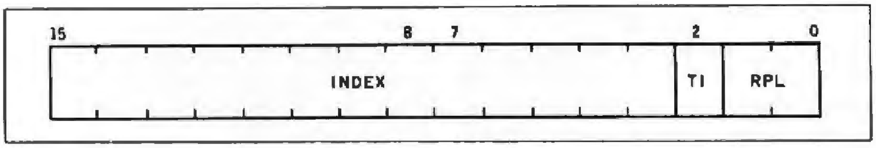


Figure 5: *The format of a selector, where* TI *means table indicator and* RPL *means requested privilege level (see reference 4).*

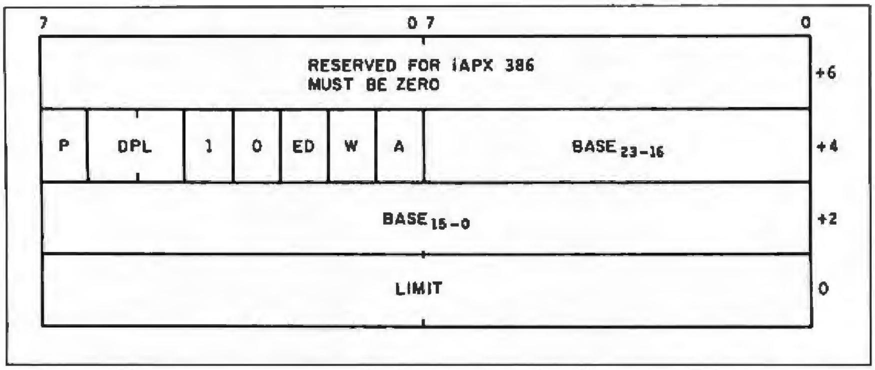


Figure 6a: *Data-segment descriptor, where* P *means present bit;* DPL, *descriptor privilege level;* ED, *expansion direction;* W, *writable; and* A, *accessed (see reference 4).*

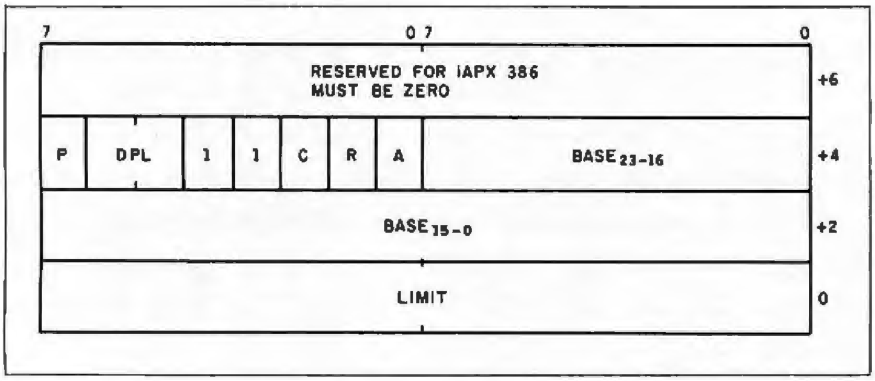


Figure 6b: *Executable-segment descriptor, where* P *means present bit;* DPL, *descriptor privilege level;* C, *conforming;* R, *readable; and* A, *accessed (see reference 4).*

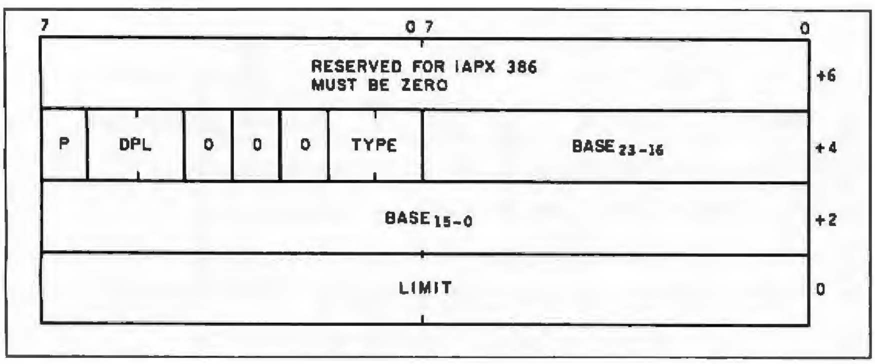


Figure 6c: *System-segment descriptor, where* P *means present bit and* DPL, *descriptor privilege level (see reference 4).*

tion is called an *explicit cache*. This is more efficient since the indexing takes place only when you *set* the segment-register value, rather than each time you use it.

Figure 5 shows the format of the selector. The requested privilege level refers to reducing the privilege level required to access a particular segment. The table indicator lets you choose between using the global descriptor table (GDT) and the currently active local descriptor table (LDT). There is only one global descriptor table; it is established when you enter protected mode. The currently active local descriptor table, if any, is a segment within the global descriptor table. The 14-bit index portion of the selector and the 16-bit offset combine to allow a 1-gigabyte logical-address space.

Descriptor tables are segments and can contain up to 8192 8-byte descriptors. There are four types of descriptors: data-segment, executable-segment, system-segment, and gate. Figures 6a through 6d contain the formats of these descriptors.

Data-segment descriptors contain system or application data including stacks. Executable-segment descriptors refer only to segments that contain instructions. System-segment descriptors contain data structures that are recognized directly by the hardware such as the descriptor tables themselves. A gate descriptor provides a pointer to an exported entry point. The call gate offers an additional level very much like a software interrupt. A particular call gate can represent the entry point of an operating-system function by number so that no explicit binding of addresses is required.

The present bit (P) and the accessed bit (A) are used in implementing virtual memory. The other bits hold protection and privilege information of interest if you are writing operating systems. The 24-bit base address for a segment means that the 1-gigabyte logical-address space is mapped into a 16-megabyte physical-address space. The 16-bit limit means that segments in the 80286 address space can be less than 64K bytes.

There is also an interrupt-descriptor table (IDT). This makes it possible for different tasks in a multitasking environment to have their own interrupt handlers. This table is conceptually like the real-address mode vector tables except that the entries are descriptors and not double words. Inspecting or changing interrupt vectors in protected mode is necessarily an operating-system function.

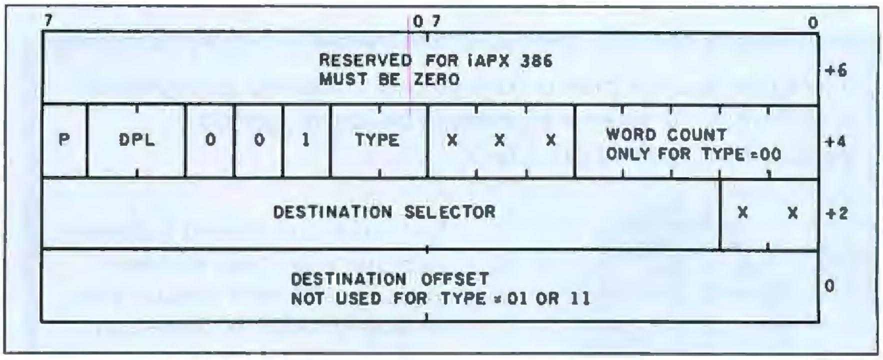


Figure 6d: *Gate descriptor, where* P *means present bit;* DPL, *descriptor privilege level; and* X, *not used (see reference 4).*

DIFFERENCES

The different view of memory that each of these machines takes implies several rules for programming on the 80286.

- Since segments can be less than 64K bytes, keep all references within the logical-segment boundary. For example, do not use a label at the end of a data segment as if it were the offset to free memory.
- Keep all data and code references within logical-segment boundaries and consistent with the segment's attributes.
- Do not rely on the iAPX 86 relationship between the value in a segment register and the selected physical memory. Programs should be as independent of the physical-memory address in which they reside as possible.
- Do not write self-modifying code.
- Do not use overlapping segments.
- Do not store temporary values in segment registers.
- Use intersegment calls to invoke operating-system functions.

CONCLUSION

By applying good programming techniques and by paying attention to the differences between the 8088 and the 80286, you can greatly simplify the writing of easily transportable programs. As microcomputers become more complicated, proper programming structure and practice become more important. You should write modular programs that use operating-system-provided facilities as heavily as possible. For the 80286 this is especially true of memory management and I/O facilities. Many software developers have already adopted this approach because it results in "well-behaved" programs in the IBM TopView and Microsoft Windows environments. Your modifications are more likely to be localized and well defined if you take this approach. ■

REFERENCES

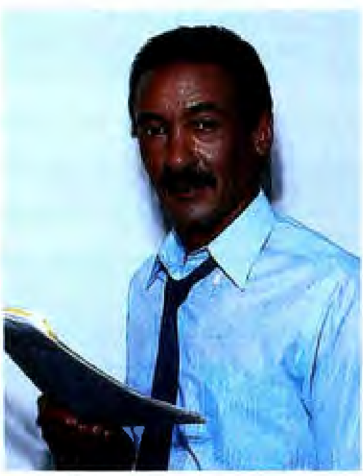
1. *iAPX 86,88 User's Manual.* Santa Clara, CA: Intel Corp., 1981.
2. *iAPX 286 Hardware Reference Manual.* Santa Clara, CA: Intel Corp., 1983.
3. *Introduction to the iAPX 286.* Santa Clara, CA: Intel Corp., 1982.
4. *iAPX 286 Operating Systems Writer's Guide.* Santa Clara, CA: Intel Corp., 1983.

Discover ~~what~~ who 60,000

It takes more than a low price to keep customers satisfied. It takes the service and support from the Staff of Qubie'.

Kellie is just one of a dozen people who can answer your questions about any of our products. She can explain our *exclusive* money-back ***No Risk Guarantee*** and extended warranty ***Preferred Customer Plan.*** All of our products include a one year limited warranty. Besides explaining and answering, Kellie can also take your order.

Don is part of a team of technical support people who can answer questions, after you buy. All of our products come with an easy-to-understand user's manual. And don't worry about getting lost during installation, Don is as close as your phone. If you should have a problem, our ***48 Hour Turnaround*** on warranty repairs insures you'll be back up in no time.



Chuck is the guy responsible for the ***Order Today, Shipped Tomorrow*** promise. He controls a small army of product testers and order packers. And Chuck's crew of billing clerks won't charge you extra for UPS surface freight and insurance or credit cards. If you're in a hurry, 2-day UPS service is available.

HARD DISK SUBSYSTEMS

PC20 (Internal) $599
PC20E (External) $749
With Spooling Tape Backup, Add $400

20 Megabyte Capacity • Boot From The Hard Disk -- No Software Patches Or Drivers To Install • Runs All The Popular Software -- DOS 2.0 Or Later • Low Power Consumption • 5 1/4" Controller Card • High Reliability And Durability -- Specially Plated Drives • Faster Access Time Than XT -- Changeable Interleave Value • Full Height And Half-Height Bezels • *Includes* Cables, *1DIR* "Visual Shell" and *ZyINDEX* "Info Searcher" Software

External Model (PC20E) Includes Case And Power Supply • Auxiallary Power Supply For Internal Model (PCPWR) Available

PC owners now know.

"The user manual is the best written and organized of those reviewed...its bargain basement price, excellent documentation, drive replacement policy, and changeable interleave, may be the best deal going."
PC WORLD
August 1985

"Perfect for anyone wanting to get XT performance without paying the price."
PC WEEK
September 25, 1984

ENHANCED KEYBOARD

ff5161 $159 NEW

Solid-State Capacitive Key Switches • Familiar Key Legends • Keys Give Tactile Feedback • Extra Wide Return Key • Finger Homing Bumps On Home Row Keys • LED Indicators On All Lock Keys • Separate Cursor Control Cluster • 3-Position Height Adjustment • Left Hand "Control" Key Is Adjacent to The "A" Key • Clustered Function Key Row • Sturdy Coiled Cable • Plugs Into PC, PC/XT, PC AT or Compaq Deskpro

SPOOLING TAPE BACKUP

BT/SP10E $499

Handy Portable Case • Utilizes Self-Threading Tape Spools Which Hold Up To 10 Megabytes Each • Includes Cable And Software That Allows You To Backup Larger Drives Requiring Multiple Tapes • Plugs Into Rear Of Floppy Drive Controller • Can Be Accessed Like A Floppy Drive • Tape Reels ($12.95 Each Or 6 For $59.95)

CARTRIDGE TAPE BACKUP

BT60 Streamer $895

Utilizes Standard DC300 Cartridges • Backup 5 Megabytes Per Minute • Both Backup And Restore Is Performed With Read/Write Verification • Includes Software, Cables, 5 1/4" Controller Card • Tape Cartridges Available -- Different Densities and Lengths

NEW

No Risk Guarantee

If you are not completely satisfied with your purchase, you may return it within 30 days for a full refund, including the cost to send it back. If you can get any dealer or competitor to give you the same No Risk Guarantee, buy both and return the one you don't like.

For fastest delivery, send cashier's check, money order, or order by Mastercard/Visa. Personal checks, allow 18 days to clear. Company purchase orders accepted, call for prior authorization. California residents, add 6% sales tax.

Hours: M -F 8 am - 5 pm PTZ
Sat 8 am - 12 pm PTZ

London (01) 223-4569
Paris (01) 321-5316
Sydney (02) 579-3322
Canada (403) 434-9444

Outside California
800-821-4479

Inside California
805-987-9741

QUBIE'

Department B
4809 Calle Alto
Camarillo, California 93010

Inquiry 166

MICROSOFT.
PRESS

The Classic Returns.

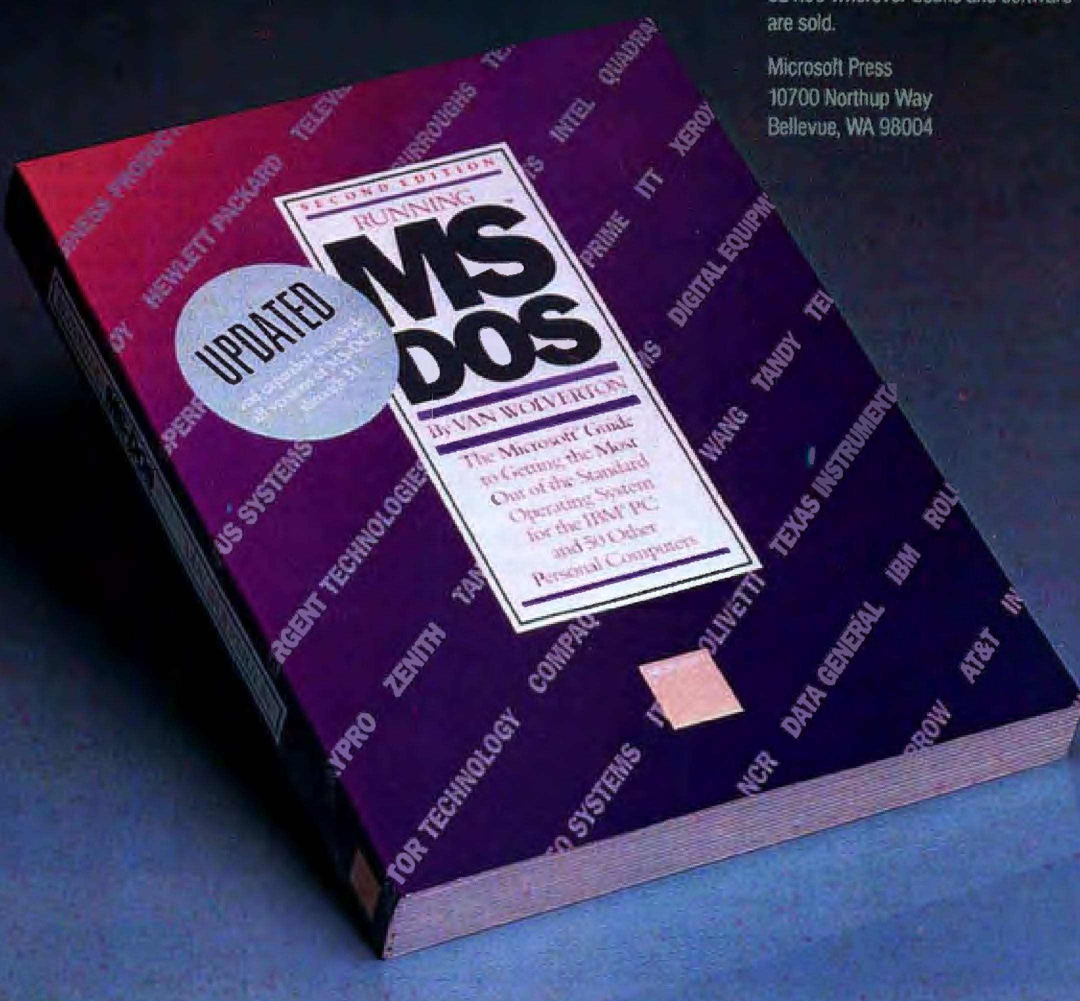
Now... newly udated to include all versions of MS-DOS through 3.1... the 2nd edition of RUNNING MS-DOS is your ultimate sourcebook for mastering the full power of the software that tells your computer how to think. It continues the tradition of the first edition, proclaimed by Jim Seymour of PC Week magazine to be the "Best PC-Related Book of 1984."

Here is the inside information—directly from the company that developed MS-DOS—that will help you streamline your computer, set up a multi-level filing system, use the MS-DOS text editor, build and sort a miniature database, create customized commands, and manage your monitor, printer, disk drives and modem. Plus new information on using the IBM PC AT's high-capacity diskette drive (MS-DOS 3.0) and using MS-Networks (MS-DOS 3.1).

If you use one of the more than five million personal computers with MS-DOS... then you need RUNNING MS-DOS.

$21.95 wherever books and software are sold.

Microsoft Press
10700 Northup Way
Bellevue, WA 98004



WRITING DESK ACCESSORIES

Design your own memory-resident programs for the IBM PC, *available at the touch of a key*

BY TOM WADLOW

YOU'RE EDITING A Pascal program and need to multiply two hexadecimal numbers. You could search around the house for your HP 16C Programmer's Calculator, or you could save your current file and write a quick and dirty program to print out the answer you need. Instead, you call up SideKick, and a little window appears in the center of the screen. You select the Calculator, set the mode to Hexadecimal, and perform the calculation. You leave SideKick, and you're back in your edit with the answer. Elapsed time: just a few seconds.

You're working on the budget for the next quarter. The phone rings. You could search for paper and pencil to take notes, or you could drop out of the spreadsheet program and call up the text editor. Instead, you call up the Spotlight notepad and type the notes into a file directly. After hanging up the phone, you pop back into the spreadsheet, exactly where you left off.

In many jobs, work is often a series of interrupted tasks rather than a single operation pursued to completion before another is begun. Most programs, whether business products or programming tools, are not designed to be interrupted. Most programs are expensive, in terms of your time, to start and stop. So you probably would not use your personal computer to take notes while on the telephone because it takes a long time to stop what you're doing and start a text editor. And it probably takes a comparably long time to return to what you were doing before the phone rang.

On more expensive computer systems, this problem might be solved with concurrency. With the ability to have several programs running at the same time, you can leave a text editor running in a separate window and then switch to that window when it is time to take notes.

True concurrency is quite difficult to achieve on the IBM PC, however. This is due to several technical reasons, primarily the way in which the IBM operating system, PC-DOS, was designed. Because of these design limitations, only one task can be active at a time. If you are willing to abide by this limitation, however, you can simulate concurrency on the IBM PC.

Several products, most notably Borland International's SideKick and Software Arts' Spotlight, have done exactly that. They provide you with a program that "hides" in your processor's memory. A special keystroke activates it, suspending whatever you were doing before. When you have finished, the suspended task is reactivated as though nothing had happened at all. Typically, these products provide several functions, such as a notepad.

(continued)

Tom Wadlow works as an engineer at the Lawrence Livermore National Laboratory. He can be contacted at POB 2755, Livermore, CA 94550.

ILLUSTRATED BY LAURA CORNELL

a calculator, an appointment calendar. In other words, the sorts of things that you normally have to interrupt other work to deal with.

These "desk accessory" programs work in two phases. Typically, you receive a program such as a .COM file that you execute once when you boot your system. By placing the command in your AUTOEXEC.BAT file, this will happen automatically. The actual desk-accessory program is surrounded by an envelope of code that loads the program, initializes it to work on the correct activation signal, and then terminates, leaving the program permanently stored in memory. Running the installation program slightly decreases the amount of memory available to later programs. The desk-accessory code, in effect, becomes a part of your operating system (until you reboot or the system crashes).

The second phase is execution. Depending on the design of the desk accessory, the newly installed code is run when a specific event takes place. In many cases, this event is a special sequence of keystrokes. For example, SideKick is activated by pressing the Ctrl and Alt keys at the same time. Other signals besides keystrokes are possible. For instance, you may want to run a clock display on your screen. In that case, you would choose the timer interrupt rather than the keyboard interrupt (see the text box "Interrupts" on page 120). Even so, the theory behind the construction of interrupt-driven programming is the same.

If you want to write your own desk accessories for an IBM PC, you should acquire several tools. First, you will need the IBM technical reference manual. Even if you are an experienced assembly-language programmer, you will probably find this manual to be a bit difficult to read; but there are valuable technical details buried in there that are to be found nowhere else. (All page references in this article are to the first edition, August 1981. They may differ from subsequent printings.) Second,

(continued)

Portability without compromise. It's yours with the Kaypro 2000 lap top computer. In fact, the sleek Kaypro 2000 is the equivalent of a full-sized IBM PC - for thousands of dollars less.

The Kaypro 2000 has an 8088 CPU for power and speed. The exceptionally readable LCD screen features a 25 line by 80 column display with 200 x 640 graphics resolution, a low glare lens, and tilt adjustment. The IBM PC compatible keyboard is detachable. The generous RAM (256K standard) is expandable to 768K internally. The computer uses 3.5" disks, each with 720K of storage, formatted.

Internal battery pack and a charger are included with the Kaypro 2000. Hardware expansion is a snap, since the IBM PC bus is user accessible. And a complete set of business software comes with every machine.

Visit your Kaypro dealer and discover the deal of the century - the Kaypro 2000!

Ask about **Kaypro Quick Credit. You can finance a Kaypro 2000 for as little as $75 per month** (excluding sales tax on a revolving charge, 10% down).

KAYPRO®
CORPORATION

For the KAYPRO Dealer nearest you, call: TOLL FREE Nationwide 1-800-4KAYPRO

Trademarks: IBM, International Business Machines, Inc.

photo features simulated screen

Listing 1: *This is a trivial and fairly useless example of a basic keyboard interceptor. The whole purpose of this routine is either to detect the typing of an uppercase Z and return a lowercase z or to detect a lowercase z and return an uppercase Z. It's not very useful, but it is a good (and fairly harmless) way of finding out if you've successfully installed the code.*

```
; TRIVIAL.ASM — A trivial example of a keyboard interceptor
; Written April 1985 by Tom Wadlow
;
; Definition of constants used by this code
;
DOS_keyboard_io          equ   16H   ; The keyboard I/O vector number
DOS_replace_vector       equ   15H   ; Make this 15H for debugging
DOS_function             equ   21H   ; The DOS function interrupt
DOS_terminate_resident   equ   27H   ; Terminate-but-stay-resident interrupt
get_vector               equ   35H   ; DOS function number for get-vector
set_vector               equ   25H   ; DOS function number for set-vector
;
CSEG SEGMENT
        assume   cs:cseg,ds:cseg       ; These govern choices made by MASM
        org      100H                  ; All .COM files start at 0100H
;
; The next instruction will be the first one executed when TRIVIAL.COM is
; run from the DOS prompt. Since the only function of TRIVIAL.COM is to
; install the interceptor, we don't want to do anything but that. This
; is still in the part of the code that will stay resident, so the only
; thing we should do here is jump past the resident code to the transient
; installation program.
;
start:
        jmp      initialize
;
; Application code should start here
;
; Any variables that our application needs can be placed here. This section
; will remain resident. Don't forget that these variables must be in the
; CSEG, not the DSEG, since we are sneaking all this code in the back door
; via the DOS terminate-but-stay-resident interrupt.
;
old_keyboard_io  dd      ; Set aside a double word for the old keyboard vector
;
;   This is the actual keyboard interceptor routine. It checks to see
; what keyboard function is requested. If the function is READ (AH = 0),
; then it simulates a DOS interrupt to the old function, which returns
; the character read in AL. If that character is a lowercase Z, then it
; is replaced in AL by an uppercase Z, and the interceptor returns
; from the interrupt.
;   If the request is not a READ, we simply jump into the old function
; that processes the request as it normally would and does our IRET for us.
;
keyboard_interceptor          proc far
        assume   cs:cseg, ds:cseg   ; Tell the assembler about default segments
        sti                         ; Turn interrupts back on
        or       ah,ah              ; Is this a READ request (AH = 0)?
        jnz      ki2                ; If so, then do the special routine
        pushf                       ; Push flags, to simulate an INT call
        assume   ds:nothing         ; so that we need not rewrite the old
        call     old_keyboard_io    ; keyboard handler, we just use it.
; Check for lowercase z.  If found convert to uppercase Z
        cmp      al,'z'             ; Old handler returns, char in AL
        jne      ki0                ; If not lowercase z, do next test
```

(continued)

you will need an assembler. I used version 1.0 of MASM, the Microsoft/IBM Macro Assembler.

It is possible to write desk accessories in some high-level languages, but in many ways it is neither feasible nor desirable to do so on the IBM PC. Remember, this program is going to permanently use up part of your available memory, so you want it to be as small as possible. A Turbo Pascal version of TRIVIAL.ASM (listing 1) would compile to approximately 12K bytes. Turbo loads many Pascal library functions for even the smallest program. The assembly-language version of TRIVIAL.ASM is only 81 bytes. Also, the code generated by many high-level languages has undesirable side effects when running from within an interrupt handler. It is possible, however, to build a toolkit of high-level language programs to help you write desk accessories. Later in this article I discuss two such tools, TEST.PAS and VECTORS.PAS, both written in Turbo Pascal. (All the Turbo Pascal programs were compiled to COM files using the Compiler Options setting.) Turbo is powerful enough to do the job. It also has quite a few built-in functions for dealing with the idiosyncrasies of the IBM PC and PC-DOS. The whole package is small enough to copy to your working disk. In fact, all the assembly-language examples in this article were written and debugged using the Turbo text editor. The fast built-in editor and compiler make Turbo Pascal a pleasant system for developing tools.

A Basic Desk Accessory

TRIVIAL.ASM shows the code for an extremely simple-minded desk accessory. The function of this code is to detect when the letter Z has been typed and invert its case. Thus, an uppercase Z becomes a lowercase z and a lowercase z becomes an uppercase Z. Not very useful, but it produces an easily detectable and harmless effect when installed. In addition, except for the few lines of code that do the actual case shifting, the bulk of the code can be used to install any

(continued)

```
        mov     al,'Z'                  ; If our special char, then change it
        jmp     ki1                     ; Go to interrupt return
; Check for uppercase Z.  If found convert to lowercase z
ki0:    cmp     al,'Z'                  ; Is this an uppercase Z?
        jne     ki1                     ; If not, return unchanged character
        mov     al,'z'                  ; If so, then lowercase it
; Return from the interrupt
ki1:    iret                            ; Return from the interrupt call

; This processes an AH <> 0 call to the interrupt handler by simply
; transferring control to the old handler, which already knows about
; doing these requests
;
ki2:
        assume  ds:nothing              ; Turn off all assumptions (IMPORTANT)
        jmp     old_keyboard_io
;
keyboard_interceptor    endp
;
; Application code should end here
;
initialize:
        mov     bx,cs                   ; Make DSEG = CSEG through BX because
        mov     ds,bx                   ;   of inadequate 8088 instruction set
;
; Get the old keyboard I/O vector and put it in its variable
;
        mov     al,DOS_keyboard_io  ; The vector we want is keyboard I/O
        mov     ah,get_vector       ; Use the DOS function call to get it
        int     DOS_function        ; See page D-33 of DOS 2.0 manual
        mov     old_keyboard_io,bx  ; Old vector is returned in ES:BX
        mov     old_keyboard_io[2],es ; Save it to variable
;
; Set the keyboard I/O vector to our routine
;
        mov     bx,cs                   ; DOS function wants vector in DS:DX
        mov     ds,bx                   ; so put seg:ofs of interceptor there.
        mov     dx,offset keyboard_interceptor
        mov     al,DOS_replace_vector  ;Use the DOS function call to set it
        mov     ah,set_vector           ; See page D-28 of DOS 2.0 manual
        int     DOS_function
;
        mov     bx,cs                   ; First expendable address is that of
        mov     ds,bx                   ; initialize, so put seg:ofs in DS:DX
        mov     dx,offset initialize  ; End program but leave keyboard
        int     DOS_terminate_resident  ; interceptor resident
;
CSEG ENDS
        END START
```

Listing 2: *This program is intended to test a replacement for the keyboard I/O interrupt handler. Since that handler is a critical portion of the operating system for the IBM PC, direct testing of untrustworthy code is difficult. By assembling the handler to replace the cassette I/O vector (interrupt 15 hexadecimal) rather than the keyboard I/O vector, the new handler can test in relative safety, keeping the system mostly operational.*

```
(* TEST.PAS    — Interrupt handler test program
                 Written by Tom Wadlow, May 1985 *)
program test;
```

(continued)

keyboard-driven desk accessory.

While it might be possible to simply type in TRIVIAL.ASM and run it, you may have a hard time extending that basic program to do anything interesting. The reason is this: You will probably make a mistake (or several). When you are debugging a regular program, a mistake means that your program dies and control reverts to the operating system. But here, you are changing the operating system. If you replace the keyboard interrupt handler with a bad one, your keyboard will no longer work. You must reboot; but since Ctrl-Alt-Del involves keystrokes, you may not be able to reboot that way. Since the IBM PC has no reset button, you must turn the machine off and on again and wait for it to reboot. It doesn't take many of these delays to cause you to give up desk-accessory programming.

Since it is inconvenient to replace a vital system interrupt handler with an undebugged program, why not use an unimportant interrupt until the program works? I chose the cassette-tape interrupt handler INT 15H (read as interrupt 15 hexadecimal) because on older PCs (like mine) an electromechanical relay gives an audible click when activated by the function AH=0 of the ROM (read-only memory) cassette handler. Chances are good that you do not make use of your cassette I/O (input/output) port (if you own an older PC that has one). If you run the test program and the relay clicks, you've installed it incorrectly. In this case you should terminate the program and turn the PC off for a few seconds to let the relay open. Leaving the relay activated for a short time should not damage it, as it was designed for such use. To avoid the necessity of turning the PC off, you might write a version of this program that does an INT 15H function AH=1, which turns the relay off. Rapid cycling of the relay should be avoided. Newer PCs may not have the built-in cassette relay, in which case any noncritical interrupt may be used. Both TEST.PAS and your application must agree on the interrupt, however.

(continued)

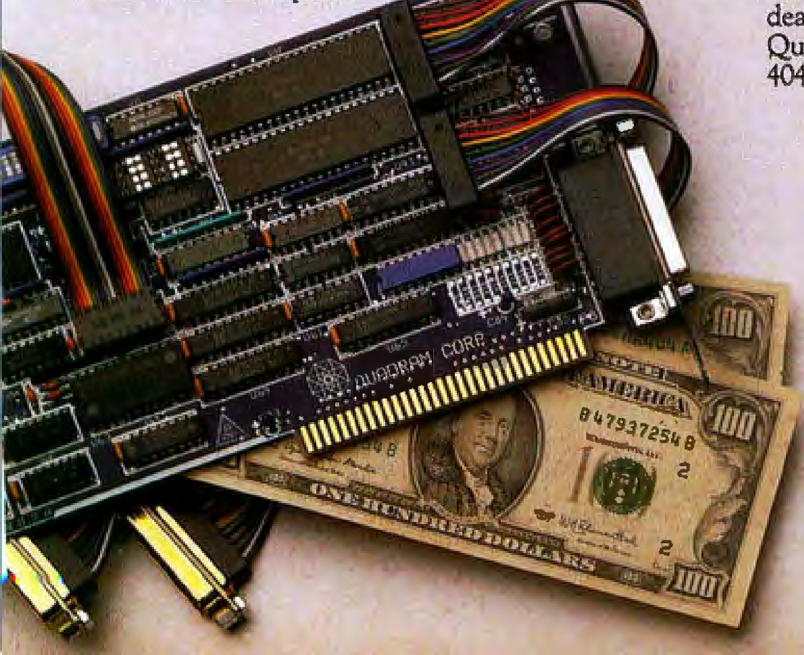
Quadram® presents the multifunction test that AST failed.

Silver Quadboard®	SixPakPlus™
☑ **Reliability:** Advanced 256K technology combines with 64K technology for greater reliability.	☐ **Reliability:** Limited to 64K technology.
☑ **Features:** Two serial ports standard. Game port standard.	☐ **Features:** One serial port maximum. Game port costs extra.
☑ **Warranty:** 5 year extended warranty with 384K (factory installed).	☐ **Warranty:** 2 year warranty.
☑ **Compatibility:** 100% PC compatible.	☑ **Compatibility:** 100% PC compatible.
☑ **Software:** RAM Drives, spooler and productivity software.	☑ **Software:** RAM Drives, spooler and productivity software.
☑ **Price:** Hundreds of dollars less than you would expect to pay.	☐ **Price:** Hundreds of dollars more than you should pay.

The Silver Quadboard by Quadram

Check the facts. You'll see why Silver Quadboard outperforms the competition everytime. Only Silver Quadboard combines so many standard features (including powerful PolyWindows™ DeskPlus desk top software) to deliver maximum power and expandability to your IBM PC, PC XT, and PC-compatible system. And, with its advanced-technology design, Silver Quadboard does it better than any other board on the market, for hundreds of dollars less than what you'd expect to pay.

So before you buy a new multifunction board, check the facts yourself. Then check with your local Quadram dealer for a free demonstration. Or contact us at One Quad Way, Norcross, Georgia 30093-2919; 404-923-6666.



SPECIFICATIONS:
Silver Quadboard: Memory expansion in 64K or 256K increments, two serial ports, parallel port, game port, clock/calendar, QuadMaster III with RAM drive and spooler software, PolyWindows DeskPlus software.

QUADRAM®
An Intelligent Systems Company

Inquiry 162

```
const
  INTERRUPT = $15;                    (* The interrupt we are testing *)
type
(* The following variant record is for 8088 registers and allows you
to read and write the A-D registers as either words or byte registers.
rL means low-order byte, rH means high-order, where the register (A through D)
is r. See page 208 of the Turbo Pascal 3.0 manual. *)
  PCregisters = record
  case Integer of
    1 : (AX, BX, CX, DX, BP, DI, SI, DS, ES, Flags : integer);
    2 : (AL, AH, BL, BH, CL, CH, DL, DH : byte);
  end;
  hexWord = string[4];                (* Four-character string for the hex converter *)
var
  registers : PCregisters;            (* Record for use in calling DOS *)
(* The function hex takes a 16-bit integer as its argument and returns a four-
character string that is the hexadecimal equivalent of that number *)
function hex(v : integer) : hexWord;
const
  map : array [0..15] of char = '0123456789ABCDEF';
var
  tmp : hexWord;
begin
tmp[4]:=map[(v and $7fff) mod 16];     (* The AND masks off the sign bit *)
tmp[3]:=map[(v shr 4) mod 16];         (* SHR means shift right *)
tmp[2]:=map[(v shr 8) mod 16];
tmp[1]:=map[(v shr 12) mod 16];
tmp[0]:=#4;        (* Byte 0 is the length of this string *)
hex:=tmp;
end;
procedure showRegisters;
begin
writeln;
write(' AX:',hex(registers.AX));
write(' BX:',hex(registers.BX));
write(' CX:',hex(registers.CX));
write(' DX:',hex(registers.DX));
write(' BP:',hex(registers.BP));
write(' DI:',hex(registers.DI));
write(' DS:',hex(registers.DS));
write(' ES:',hex(registers.ES));
writeln;
              (* Show the flag bits. *)
if ((registers.Flags and $0001) > 0) then write(' CF');
if ((registers.Flags and $0004) > 0) then write(' PF');
if ((registers.Flags and $0010) > 0) then write(' AF');
if ((registers.Flags and $0040) > 0) then write(' ZF');
if ((registers.Flags and $0080) > 0) then write(' SF');
if ((registers.Flags and $0100) > 0) then write(' TF');
if ((registers.Flags and $0200) > 0) then write(' IF');
if ((registers.Flags and $0400) > 0) then write(' DF');
if ((registers.Flags and $0800) > 0) then write(' OF');
writeln;
end;
(* Testing code begins here *)
(* This function prompts for a character, then reads it by
performing an interrupt with the appropriate function code.
State of the registers is shown before the call and afterward. *)
```

(continued)

By using INT 15H you leave the vital keyboard driver untouched and working. For purposes of illustration, let's say you were debugging TRIVIAL.ASM. You could make the constant DOS__replace__vector equal 15H and use TEST.PAS (see listing 2) to simulate a call to the keyboard I/O handler through INT 15H instead of the correct INT 16H. TEST.PAS also prints the entry and exit states of the PC registers around each call. Thus, when your experimental code fails (and it will), the chances are pretty good that your system will not crash with it. When you feel confident that your code is working properly, reassemble it to replace the constant DOS__replace__vector with INT 16H, the keyboard I/O interrupt.

Another tool you may find helpful, VECTORS.PAS (listing 3), is a Turbo Pascal program that dumps the current interrupt vectors to the screen or to a file. By comparing these hexadecimal numbers to the default settings listed on page 3-3 of the IBM technical reference manual, you can see if your installation went as planned.

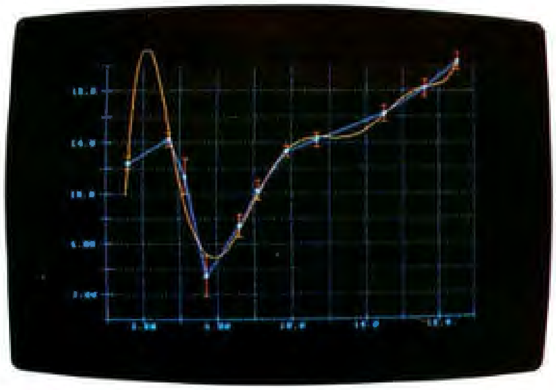
Loading the New Interrupt Handler

Surrounding the code that actually performs the application is an envelope of code that does the installation and setup of the new interrupt handler. As I said earlier, this is the code that is actually run during phase one of the execution of this desk accessory.

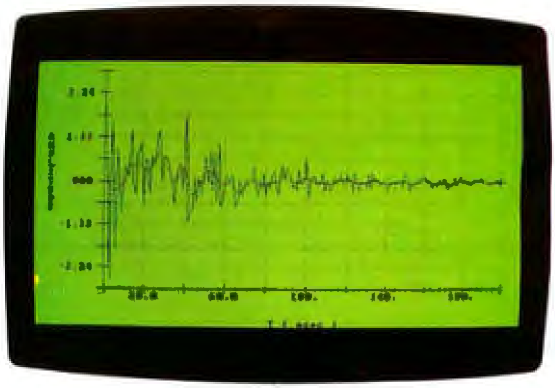
When you ask PC-DOS to run a .COM file by typing its name at the DOS prompt, it does two things. First, it copies the entire file into memory. Second, it does a subroutine call to location 100H in the new code segment, running whatever code is there. So your initialization code need not actually copy the new program from a separate disk file. DOS does that for you.

The function that makes these desk-accessory programs possible is INT 27H (terminate process and remain resident). You provide this function call with a pointer to the next free ad-

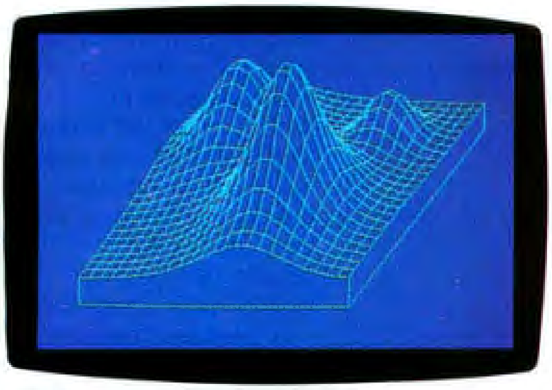
(continued)



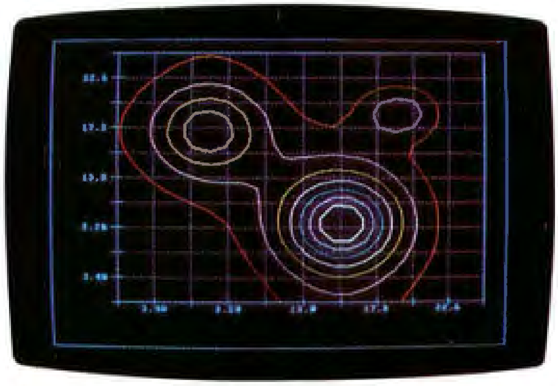
ASYST high-resolution graphics now include error bars, labels, axes, grids, and multiple colors.



New color graphics enhance comparison of experimental data with filtered data.



ASYST axonometric plots simplify analysis of complicated 3-dimensional surfaces.



Contour plotting adds an alternative approach to meaningful 3-D representation.

ASYST™ adds new muscle.

More hardware support, more analysis capabilities for the IBM PC.

ASYST™ Scientific Software turns your IBM PC, XT,™ AT, or compatible into a complete scientific workstation. And now it's even more versatile, with:

- Expanded analysis capabilities
- High-resolution color graphics
- GPIB/IEEE-488 hardware support
- Axonometric and contour plotting
- Additional A/D hardware support

Minicomputer speed and precision—at a fraction of the cost.

ASYST on an IBM PC does a 1024-pt. FFT in less than 3 seconds (as fast as 1.2 on some compatibles). For the same task, an optimum performance routine on a DEC 11/23+ minicomputer using FPF 11™ took 2 seconds—at 5 times the price!

Built-in routines. Full programmability.

Straightforward, pre-programmed commands, such as XY.DATA.PLOT, FFT, and A/D.IN, put you in total control right away. Commands can be used interactively, or combined and modified as needed—from simple macros to fully customized programs. And all commands co-reside in memory—no disk shuffling.

ASYST is four separate, fully-integrated modules:

Module 1: System/Graphics/Statistics establishes the environment. It provides basic mathematics operators, descriptive statistics, array manipulation and control, automatic plotting and color graphics support (including IBM standard/enhanced and Hercules boards), a text editor, file I/O, and a built-in programming language.

Module 2: Analysis reduces and analyzes data. Includes eigenvalues, eigenvectors, polynomials, ANOVA, axonometric and contour plotting, least squares approximations, curve fitting, convolutions, integration, differentiation, smoothing, and fast Fourier transform.

Module 3: Data Acquisition allows communication with lab equipment and analog signal sources. Includes A/D and D/A conversions, digital I/O, timing, and triggering. Supports standard interface boards including IBM DACA.

Module 4: GPIB/IEEE-488 allows additional interfacing to some 10,000 instruments through a variety of plug-in cards.

- Purchase Module 1 alone—or with any combination of the other modules—to tailor the system to your specific applications.

Try ASYST for 30 days. For details, call **(800) 348-0033;** in NY, (212) 702-3241.

ASYST

MACMILLAN SOFTWARE CO.
An Affiliate of Macmillan Publishing Company
866 Third Avenue, New York, NY 10022

Inquiry 116

```
procedure testCode0;
var
  c : byte;
begin
c: = 0;
while (c < > 3)                    (* Exit with CTRL-C *)
DO  begin
      registers.AH: = 0;   (* Set the function code *)
      registers.AL: = 0;   (* Zero everything else, so that change is obvious *)
      registers.BX: = 0;
      registers.CX: = 0;
      registers.DX: = 0;
      showRegisters;
      write('Enter a character: ');
      intr(INTERRUPT, registers);        (* Turbo function to perform INT *)
      showRegisters;
      writeln('Character = [',ord(registers.AL),'] (',chr(registers.AL),')');
      writeln('----------------');
      c: = registers.AL;
      end;
end;
begin
testCode0;
end.
```

Listing 3: *This program displays the current values of the interrupt vector settings on the* IBM PC. *The initial values of these settings can be found in the* IBM *technical reference manual on page* 3-3. *Their addresses and additional descriptive information can be found on page* 3–21.

```
{$P256}
(* VECTORS.PAS written in Turbo Pascal 3.0 by Tom Wadlow
The P256 compiler option is used to permit the table printed by VECTORS to be
re-directed into a file. If you use an early version of Turbo Pascal, this option will
not work. If your Turbo compiler supports redirection, then the vector table can be
written to a file by compiling to a .COM file, and then typing:
  vectors > VECTOR.LST
at the DOS prompt, where VECTOR.LST is the name of the file that you want the
table to go in. *)
program vectors;
type
  vectorName = string[30];
  hexWord    = string[5];
  (* HEX takes a 16-bit integer as its argument and returns a five-
character string that is the hexadecimal equivalent of that number *)
function hex(v : integer) : hexWord;
const
  map : array [0..15] of char = '0123456789ABCDEF';
var
  tmp : hexWord;
begin
  tmp[5]: = map[(v and $7fff) mod 16];   (* The AND masks off the sign bit *)
tmp[4]: = map[(v shr 4) mod 16];
tmp[3]: = map[(v shr 8) mod 16];
tmp[2]: = map[(v shr 12) mod 16];
tmp[1]: = '$';          (* Turbo hex constants are preceded by $ *)
tmp[0]: = #5;           (* Byte 0 is the length of this string *)
hex: = tmp;
```

(continued)

dress in memory; it will terminate the current program, resetting the base address (where the next program will be loaded) to the address you specify.

In TRIVIAL.PAS, the first code executed is a jump to the initialization code at the end of the program. In order to minimize the amount of memory permanently taken up by the interrupt handler, put all the initialization code at the end of the .COM file. When you call INT 27H with a pointer to the start of this code, all the initialization code is effectively flushed, but the new interrupt handler is not. Only the few bytes of the first jump remain as "overhead," and this is usually quite acceptable.

In this case, the initialization code must perform three separate functions. It must save the old value of the keyboard interrupt. It must replace the old keyboard interrupt with the address of the new one. And it must terminate gracefully, leaving the interrupt code but allowing the space taken up by the installation code to be reclaimed for later use.

One rule of thumb for writing these interrupt handlers is: Never reinvent the wheel. There is no point in wasting space duplicating something that DOS can do for you. That is why it is a good idea to save the old value of the keyboard interrupt.

The reason that DOS uses software interrupts to call functions is modularity. If DOS used a predefined address for a given routine, it would be impossible to rewrite different portions of the code, as we are attempting to do in this article. Every program would have the special address for each function "wired in," and it would be impossible to change without changing every piece of code written for the IBM PC. By using software interrupts and a jump table, every piece of code is independent of the current settings of the interrupt vectors. But software interrupts are not magical. In fact, they differ only from subroutine calls in that the processor automatically pushes all of the system state flags onto the stack before the return address. So if you know the address

(continued)

```
end;
  (* PRINTVECTOR takes an interrupt number and a descriptive string
  and prints the value of the corresponding interrupt vector on the
  standard output device *)
procedure printVector(v : integer; s : vectorName);
    (* Since the addresses are stored in with their bytes in least
    significant order, we can get them a byte at a time and rearrange
    them to word byte order. This could probably be done in a single
    complex statement, but I prefer to separate it for clarity. *)
var
  a,b,c,d : integer;
begin
a: = Mem[0000:v*4];        (* MEM is a Turbo array that can be used to *)
b: = Mem[0000:v*4+1];  (* read or write any byte in memory *)
c: = Mem[0000:v*4+2];
d: = Mem[0000:v*4+3];
writeln(hex(v),s:30,'   ',hex((d shl 8) + c),':',hex((b shl 8) + a));
end;
  (* The main program simply calls printVector repeatedly with information
  gathered from the technical reference manual. Interrupts 6,7,A,B,C,D are listed
  as RESERVED by IBM and are not shown here, though it would be trivial to
  add them. In addition, table entries for the Video Initialization ($1D)
  and Disk Parameters ($1E) were omitted so that the remaining
  listing would fit entirely on one screen. *)
begin
printVector($00,'Divide by zero');
printVector($01,'Single step');
printVector($02,'Nonmaskable interrupt');
printVector($03,'Breakpoint');
printVector($04,'Overflow');
printVector($05,'Print screen');
printVector($08,'Timer');
printVector($09,'Keyboard');
printVector($0E,'Disk');
printVector($10,'Video I/O');
printVector($11,'Equipment check');
printVector($12,'Memory check');
printVector($13,'Disk I/O');
printVector($14,'RS-232 I/O');
printVector($15,'Cassette I/O');
printVector($16,'Keyboard I/O');
printVector($17,'Printer I/O');
printVector($18,'ROM BASIC entry');
printVector($19,'Bootstrap loader');
printVector($1A,'Time of day');
printVector($1B,'Get control on KBD break');
printVector($1C,'Get control on timer');
printVector($1F,'Graphics character table');
end.
```

of an interrupt routine, you can call it directly by pushing the flags yourself. A normal interrupt call to get a character from the keyboard and return it in the AL register that looks like

```
; Function 0 is read-character
  MOV      AH, 0
; Keyboard I/O software interrupt
  INT       16H
```

can be simulated by

```
; Function 0 is read-character
  MOV      AH, 0
; Push processor flags on stack
  PUSHF
; Call ROM routine directly
  CALL      <address>
```

where <address> is where the original routine is located. When the IBM PC is first booted, that address is $F000:$E82E (using Turbo Pascal's notation for a hexadecimal segment and offset as the program VECTORS.PAS would print it out). You could code that address directly into a keyboard interrupt handler, but that would make it impossible to run your handler with any of the other commercially available interrupt-driven programs, like SideKick or Spotlight, that replace the keyboard vector.

DOS provides a function, 35H, that reads the current value of a given interrupt vector from the jump table and returns it to your program. By using this function, you can place the old value (which may not be the same as the ROM value) into a variable in memory and call through that variable to run the old handler. There may be other reasons why SideKick or Spotlight or the others will not run with your specific code, however.

Since DOS has a special function to read an interrupt vector, you might guess that there is a similar function to set an interrupt vector. Indeed there is: function 25H. (A complete listing of all the DOS functions can be found in section D of the DOS manual.)

Writing an Interrupt Handler

A little skepticism is a healthy thing when writing a desk accessory that hides inside an interrupt handler. Many functions that are perfectly legitimate inside a regular program are difficult and dangerous inside an interrupt handler. By calling a system function incorrectly from within an interrupt handler, you may change the state of the system that the interrupted application returns to. This can have disastrous results. For example, if your desk accessory moves the cursor (by writing to the screen, for instance), the application may not detect this change. Subsequent input based on cursor location would be incorrect. Thus, you should be careful to place the cursor back where you found it before leaving the interrupt code. Probably the most dangerous

(continued)

Listing 4: *This program installs a keyboard interceptor into* PC-DOS. *The interceptor keeps a record of the last full command typed to* DOS *(or any other program that uses the* DOS *keyboard* I/O *calls). Pressing Ctrl-\ makes the previous command repeat, one character at a time. A more extensive version of* KEYSTROK.ASM *is available from the author for* $25 ($15 *if you send your own disk and a self-addressed stamped disk mailer).*

```
; KEYSTROK.ASM
; written April 1985 by Tom Wadlow
;
; Definition of constants used by this code.
;
DOS_keyboard_io          equ   16H   ; The keyboard interrupt number
DOS_function             equ   21H   ; The DOS function interrupt
DOS_terminate_resident   equ   27H   ; Terminate-but-stay-resident interrupt
get_vector               equ   35H   ; DOS function number for get-vector
set_vector               equ   25H   ; DOS function number for set-vector
;
CSEG SEGMENT
        assume   cs:cseg,ds:cseg      ; These govern choices made by MASM
        org      100H                 ; All .COM files start at 0100H
;
start:
        jmp      initialize
;
; Application code should start here
;
old_keyboard_io  dd        ; Set aside a doubleword for the old keyboard vector
outline          db   80 dup(0)  ; Where the last line is stored
                 dw   0          ; Just to make sure it is terminated
inline           db   80 dup(0)  ; Where the current line is stored
                 dw   0          ; Just to make sure it is terminated
outptr           dw   0          ; Pointer into outline
inptr            dw   0          ; Pointer into inline
;
keyboard_interceptor        proc far
        assume   cs:cseg, ds:cseg  ; Tell the assembler about default segments
        sti                        ; Turn interrupts back on
        push     ds                ; Save the registers used in this code
        push     bx
        push     di
        mov      bx, cs            ; Set the Data Segment = Code Segment
        mov      ds, bx
        cmp      ah, 0             ; Is this a READ request (AH = 0)?
        jne      ki1               ; If not, then use the ROM handler
;
; Simulate an interrupt call to the old keyboard I/O handler
;
ki:     pushf                      ; Push flags, to simulate an INT call
        assume   ds:nothing        ;  so that we need not rewrite the old
        call     old_keyboard_io;  keyboard handler, we just use it
        cmp      al, 28            ; Look for CTRL-\
        je       ki3
        cmp      al, 8             ; Look for backspace
        je       ki4
        cmp      al, 13            ; Look for carriage return
        jne      ki0
;
; After CR we reset output to point to start of keystroke buffer
;
ki00:   assume   ds:cseg
        push     si                ; Save the registers used
```

(continued)

Skepticism is a healthy thing when writing an accessory that hides in an interrupt handler.

area is file I/O. It is possible to safely open files from a desk accessory, but great care must be taken not to interfere with any file activities of the interrupted program. It would be easy to confuse DOS by opening the same file with your pop-up notepad and your normal text editor at the same time.

You can perform many useful desk-accessory functions without using any disk I/O. Calculators or clocks are a good example. If you can possibly avoid going to the disk in a desk accessory, it is probably a good idea to do so. If not, then test your code exhaustively for bugs and side effects.

Some DOS functions may not work at all from within an interrupt handler because of the way PC-DOS is written. For any function you intend to use, it is important to thoroughly test the calls from within an interrupt handler.

TRIVIAL.ASM shows a minimal desk accessory with almost no I/O at all. Surrounded by the envelope of installation code, the application itself is rather small. Note that the interrupt handler checks to see if the function is a read (register AH = 0). If it is not a read, then the old interrupt-handler code handles any other functions, including IRET (return from interrupt). If the function is a read, then a simulated interrupt call to the old keyboard interrupt handler returns the next character in the input buffer. By doing the simulated interrupt, you can let the DOS and ROM routines do the work they were designed to do and keep your desk accessory confined to its special function. The old routine is supposed to return the character read in register AL, so compare that with the target characters (in this case, uppercase or lowercase Z.) By modi-

(continued)

Wide spreadsheets?
*Side*Step *is in your future!*

Good News! $60

People have gone to some pretty great extremes to solve their wide printing problems.

Luckily, you don't have to. *Side*Step™ lets your dot matrix printer print as wide a document as you need by turning each letter on its side — electronically! *Side*Step frees you from gluing, taping, and stapling wide printouts together. You get a document of virtually infinite width automatically.

*Side*Step does everything other sideways printing programs do, and more:

- Fast operation
- East to learn and use
- Context-sensitive online help
- No limit on file size or print width
- 18 combinations of size and quality
- Allows double-strike and emphasized printing*
- Letter quality on Toshiba and LQ1500
- Supports high-speed draft mode*
- Speed optimized for each printer
- Features electronic spreadsheet "pasting"
- Prints IBM graphic characters
- Prints Greek and math symbols
- Fully compatible with TopView
- Command options can be saved for reuse
- Can run from the command line
- Complete hard disk path support
- Allows wild cards
- Directories always available
- Can run from batch files
- Supports output to disk files
- Supports color monitors

*Not available on all printers

*Side*Step your wide printout problems. Order *Side*Step today!

*Side*Step runs on the IBM PC, PC/XT, PC/AT, PC*jr*, and compatible computers under DOS 2.0 and later and requires 128K memory. *Side*Step works with popular printers from IBM, Epson, HP, Toshiba, IDS, OkiData, TI, C. Itoh, NEC, Dataproducts, OTC, Citizen, and more. Ask for *Side*Step from your local dealer or call today and order *Side*Step directly.

Side Step

To Order Call Toll Free
1-800-255-7526

30-DAY MONEY-BACK GUARANTEE
VISA AND M/C ACCEPTED
(Dealer inquiries invited)

SoftCorp®
We make it easier for you to manage.

2340 State Road 580, Suite 244, Clearwater, Florida 33575
In Florida call (813) 799-3984 Inquiry 179

The new standard in sideways printing

YES! Send me SideStep today! I enclose $60 plus $3 shipping and handling. (Florida residents, add 5%)
☐ Visa ☐ M/C ☐ Check Printer ______
Name ______
Address ______
City ______ State ______ Zip ______
Phone ______

INTERRUPTS

PC-DOS is an interrupt-driven operating system. It might be possible to write a similar system without the use of interrupts, but that would involve the software equivalent of trying to look in all directions at once. Interrupts greatly simplify the way an operating system is written.

An interrupt, as the name implies, is an event that diverts the flow of a program from what it normally would be doing. Interrupts can happen at any time; they are, in technical parlance, asynchronous events. They happen to people dozens of times in the course of a day; to computers, they usually happen many thousands or millions of times. For people, the scenario is a familiar one: You are writing your memoirs when your daughter comes in and asks you to tie her shoelaces. You do so, she leaves with her request satisfied, and you continue with your writing. For computers, it is similar: Your program is executing merrily away, computing the next Mersenne prime number, when you strike a key. Your prime-number task is suspended; the interrupt handler reads the keystroke and places it in the input buffer to be given to your application program the next time it requests a keyboard read operation (which may be several million years hence if you are computing Mersenne primes on an IBM PC). With the character safely in the input buffer, the keyboard interrupt handler relinquishes control to your application.

One major use of interrupts is the coordination of external hardware (such as keyboards and disk drives) that performs functions and must report the results to the processor. Another use of interrupts is internal. Machines like the IBM PC use software interrupts to allow application programs to communicate with the operating system. If each application were required to know the address of each specific operating-system function, it would be very difficult to change the operating system (to fix bugs, for example). By using software interrupts, an application program need not know where a function is located in order to have it performed.

The IBM PC operating system provides two types of interrupt-driven utilities: ROM interrupts, which are built into the hardware of the machine, and DOS interrupts, which come with the disk operating system. DOS interrupts are higher-level functions that use the low-level ROM interrupts as building blocks to do more complicated and useful things. Functions refer to an interrupt that does one of many things, depending on its parameters. Most DOS functions are performed by INT 21H. The different functions are chosen by selecting the appropriate function number (from 0 to 57H for DOS 2.0) and placing it in the AH register.

An interrupt signals the processor to perform some special operation. On the IBM PC, the interrupt vector table governs the choice of operation. This table is located in the first few hundred bytes of memory. Each entry in the table is an address that's 4 bytes long.

This address indicates the absolute location in memory of the interrupt-handling routine for that particular interrupt. When an interrupt occurs, the 8088 processor first pushes the current value for all system flags onto the stack and then executes a long subroutine call to the appropriate interrupt handler. The processor determines which handler to use by multiplying the interrupt number by 4 (the number of bytes in a table entry) and fetching the long word at that address. When the interrupt routine is finished, it executes an IRET instruction, which pops the flag values off the stack, restoring them to the state they were in before the interrupt, and then does the equivalent of a return-from-subroutine instruction to return control to the interrupted task.

Murphy's Law says that interrupts, being asynchronous events, will occur at the most unfortunate time during the execution of a program. So computer designers allow the programmer a method by which interrupts can be held at bay for a short time. The CLI (clear interrupt-enable flag) instruction prevents any further interrupts from being recognized, to allow some particularly critical code to execute. One of the most disastrous times for an interrupt to occur is during the actual processing of another interrupt. To prevent this, the 8088 is designed to immediately suspend interrupt processing when an interrupt occurs. Interrupts are turned on again explicitly by the STI (set interrupt-enable flag) instruction, which you usually find right at the beginning of an interrupt handler.

You might think that you would rather turn on interrupts at the end of an interrupt handler, just before returning control to the main task. This turns out not to be the case, however. Most computers have only a limited ability to remember what interrupts have occurred while interrupt processing was suspended. Any external activity, such as keyboard input or disk I/O, may be lost since the interrupts were not processed in time. It is very important for the design of an operating system to minimize the time in which interrupts cannot be processed. It is usually possible to interrupt an interrupt handler with another interrupt request. These routines are designed so that this is possible because the inability to do so will drastically reduce the performance of the operating system.

On the IBM PC, one special case of this interruptibility is not available. It is not possible for many interrupt handlers to be interrupted by themselves. In industry jargon, they are called nonreentrant routines. For example, two keyboard interrupts spaced too closely together would fail in strange and mysterious ways. The reasons for this usually lie in the special-purpose buffers, such as the keyboard input buffer, that each routine maintains. Imagine many people writing a message on the same small blackboard, each unable to see what the others are writing. You might get all the messages clearly, but chances are you would simply end up with an unreadable mess. For single-user systems, this is not much of a problem. Reentrant routines are primarily used for multiuser systems.

Interrupt programming is primarily done for operating-system functions, and thus it usually involves some of the trickier facets of both the operating system and the hardware involved.

For any function you intend to use, it is important to test the calls from within an interrupt handler.

fying register AL, you can change the character that will be returned to DOS or the application program. If you choose, you could find a special, seldom used character (such as Ctrl-\ , for example) and use that to start up a much more complicated desk accessory.

KEYSTROK.ASM (listing 4) is a somewhat more complicated desk accessory that employs Ctrl-\ as its activation character. This program uses exactly the same installation code as TRIVIAL.ASM. KEYSTROK.ASM is a simple line editor with a one-command history function. As you type a command line into DOS, the new keyboard interrupt handler copies it into an internal buffer. KEYSTROK.ASM remembers all (or at least the first 80 characters) of the previously typed line. You can feed that old line back as a new DOS command by repeatedly pressing Ctrl-\ . For example, if you find you've mistyped one character at the beginning of a long complicated DOS command, it is possible to redo the command with a minimum of fuss. It also allows you to repeat various commands with a minimal variation, such as

```
ERASE FOO.OBJ
ERASE FOO.COM
ERASE SAMPLE.FOO
```

Note that KEYSTROK.ASM is intended more as an instructional example for this article than an actual production tool. For purposes of clarity, I've ignored certain special cases that must be considered in an actual tool.

It is always useful to elaborate the exact steps necessary to turn some-

(continued)

```
        push    di
        push    cx
        push    ax
        push    es
        mov     si, ds                  ; Source segment in DS
        mov     es, si                  ; Destination segment in ES
        mov     si, offset inline       ; Source offset in SI
        mov     di, offset outline      ; Destination offset in DI
        mov     bx, inptr               ; Null terminate the input string
        mov     byte ptr [si+bx], 0
        mov     cx, 80                  ; Repetition count in CX
  rep   movsb                           ; Move byte string
        pop     es                      ; Restore registers
        pop     ax
        pop     cx
        pop     di
        pop     si
ki02:   mov     outptr, 0               ; Restart in and out buffers
        mov     inptr, 0
        jmp     ki01
;
; Put a character in the incoming buffer
;
ki0:    assume  ds:cseg
        mov     di, offset inline       ; Use input buffer
        mov     bx, inptr               ; Go to next empty space
        cmp     bx, 80                  ; Take no more than 80 chars
        jg      ki01
        mov     [di+bx], al             ; If less than 80 chars, add char
        inc     inptr                   ; and advance pointer
;
; Return from interrupt after restoring the register state
;
ki01:   pop     di
        pop     bx
        pop     ds
        iret                            ; Return from the interrupt call
;
; Let the ROM routine handle everything, including the return from interrupt
;
ki1:    pop     di
        pop     bx
        pop     ds
        assume  ds:nothing              ; Turn off all assumptions (IMPORTANT)
        jmp     old_keyboard_io
;
; Send the next byte of a string back as though it had been typed at the kbd
;
ki3:    assume  ds:cseg
        mov     di, offset outline      ; Use output buffer
        mov     bx, outptr              ; Advance to next char to be sent
        mov     al, [di+bx]             ; Put char in AL
        cmp     al, 0                   ; Have we used up buffer?
        je      ki01                    ; If so, return NULL
        inc     outptr                  ; If not, advance output pointer
        jmp     ki0
;
; Don't save the backspace. Instead, back up input pointer by 1 (unless it's 0)
;
ki4:    assume  ds:cseg
        cmp     inptr, 0                ; Is input buffer empty?
        je      ki01                    ; If so, just return
        dec     inptr                   ; If not, back up input pointer
        jmp     ki01                    ; And return
```

(continued)

```
keyboard_interceptor      endp
;
; Application code should end here
;
initialize:
        mov     bx,cs                   ; Make DSEG=CSEG through BX because
        mov     ds,bx                   ; of inadequate 8088 instruction set
;
; Get the old keyboard I/O vector and put it in its variable
;
        mov     al,16H                  ; The vector we want is keyboard I/O
        mov     ah,get_vector           ; Use the DOS function call to get it
        int     DOS_function            ; See page D-33 of DOS 2.0 manual
        mov     old_keyboard_io,bx      ; Old vector is returned in ES:BX
        mov     old_keyboard_io[2],es   ; Save it to variable
;
; Set the keyboard I/O vector to our routine
;
        mov     bx,cs                   ; DOS function wants vector in DS:DX
        mov     ds,bx                   ; so put seg:ofs of interceptor there.
        mov     dx,offset keyboard_interceptor
        mov     al,DOS_keyboard_io      ; Use the DOS function call to set it
        mov     ah,set_vector           ; See page D-28 of DOS 2.0 manual
        int     DOS_function
;
        mov     bx,cs                   ; First expendable address is that of
        mov     ds,bx                   ; initialize, so put seg:ofs in DS:DX
        mov     dx,offset initialize    ; End program but leave keyboard
        int     DOS_terminate_resident  ; interceptor resident
;
CSEG ENDS
        END start               ; Starting address is start
```

thing as finicky as an assembly-language source into an executable application program. Quite often the reader and the writer make different basic assumptions about what is going on; explicit instructions can help put everybody on the right track.

All the assembly-language programs in this article were assembled using this basic set of commands:

```
MASM <name>;
LINK <name>;
EXE2BIN <name>
RENAME <name>.BIN
   <name>.COM
ERASE <name>.EXE
```

where <name>.ASM is the name of the source code.

You'll notice that the example assembly-language files are assembled to .COM files rather than .EXE files. This is to simplify the code and speed the execution of the installation programs. The .COM format for PC-DOS programs is intended for small, fast utilities. Thus, they have no separate stack segment and are limited in their choices of code and data segments. They begin execution at location 100H rather than at 0H. When writing a program intended to be a .COM file, you should not define a stack segment, and the first line of executable code should come after an ORG 100H statement. When this code is being assembled, the IBM assembler generates a warning that your code has no stack segment. This is no cause for concern, and you should expect it to occur.

DEBUGGING AN INTERRUPT HANDLER

One of the most difficult parts of writing a replacement interrupt handler is getting it to work. It is, in effect, a new part of the operating system. Small bugs in a new keyboard handler or video interrupt handler will be magnified since much of the rest of the operating system depends on these routines. A technique I discussed earlier involves replacing a less consequential interrupt vector for part of the debugging phase. Another important technique is prototyping. If you are writing a really large interrupt handler, you should try to build and test each portion of your code as a normal program before placing it in an interrupt routine. For example, if you design a scheme for pop-up windows, be sure that it works from a normal program first. If you can, push the system dependencies as far down in your code as possible. If you need to get a character from the keyboard rather than scatter DOS calls *all* throughout your code, encapsulate the DOS call in one special routine and then have your application call that. Don't be afraid to write little test programs that try out one thing at a time from within an interrupt handler.

Assembly-language programming brings out a tendency to optimize, to try and make programs as small and as tight as possible. This can be good, but it can be dangerous to be too tricky. A program that is too big or too slow isn't of much use, but one that is that is impossible to understand a month after you write it, or impossible to modify, is equally bad. A sense of purpose and balance is important here. Be tricky only when you have to.

These techniques should be familiar to anyone who has ever written modular code before. They are basic principles of software design. A good book on this subject is *The Mythical Man-Month: Essays on Software Engineering* by Frederick P. Brooks Jr. (Addison-Wesley, 1974).

Assembly language is not for everyone. And interrupt handlers are by far the most finicky type of assembly-language programming you can attempt. But if you are careful and persistent, you can have a lot of fun adding special-purpose functions to your IBM PC. ■

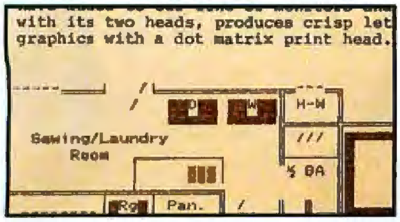
Editor's note: The source-code files for the listings in this article are available for downloading from BYTEnet Listings. *Prior to* November 1, *phone* (617) 861-9774; *thereafter, phone* (617) 861-9764.

THE DUAL HEAD PRINTER

THE FORTIS DH45, THE START OF A NEW BUSINESS GENERATION

DAISY OR DOTS... *Have it your way. Don't settle for one or the other... get both.*

The revolutionary **FORTIS DH45** *dual head printer combines the speed and bit image graphics of a dot matrix with a high quality daisy wheel for crisp letters. All this in one compact unit that saves desk space and at a price that is less than you would pay for one printer. It is also compatible with IBM* PC and most other personal computers.*



Imagine the advantage of having two heads in one printer. Need graphics or condensed print spreadsheets or a rough draft of a long letter, just touch the control panel. How about important correspondence to impress a potential client, again, just touch the control panel to switch to the letter quality daisy wheel. Indeed,
the old saying **TWO HEADS ARE BETTER THAN ONE,** *really holds true in the* **DH45.**

*IBM is a registered trademark of IBM Corp.

FOR THE DEALERS NEAREST YOU CALL (800) 447-4700

FORTIS **Dynax™, Inc.**

DYNAX, INC. OFFICES

- **HEADQUARTERS** 6070 Rickenbacker Rd., Commerce, CA 90040 • (213) 727-1227
- **NEW JERSEY** One Madison St., East Rutherford, NJ 07073 • (201) 471-0100
- **TEXAS** 6012 Campus Circle, Suite 250, Irving, TX 75062 • (214) 257-1700
- **ILLINOIS** 533 West Golf Rd., Arlington Heights, IL 60005 • (312) 228-0707
- **MASSACHUSETTS** 400 W. Cummings Park, Suite 5300, Woburn, MA 01801 • (617) 933-8162
- **N. CALIFORNIA** 1255 Oakmead Parkway, Sunnyvale, CA 94086 • (408) 730-1712

YES! Please send me more information on the FORTIS DH45 Dual Head Printer

Name ______ Title ______

Company ______ Phone ______

Address ______ City ______ State ______ Zip ______

Mail to: **DYNAX, INC. Customer Service / FORTIS DH45**
6070 Rickenbacker Road, Commerce, CA 90040

PRINTOUTS TH

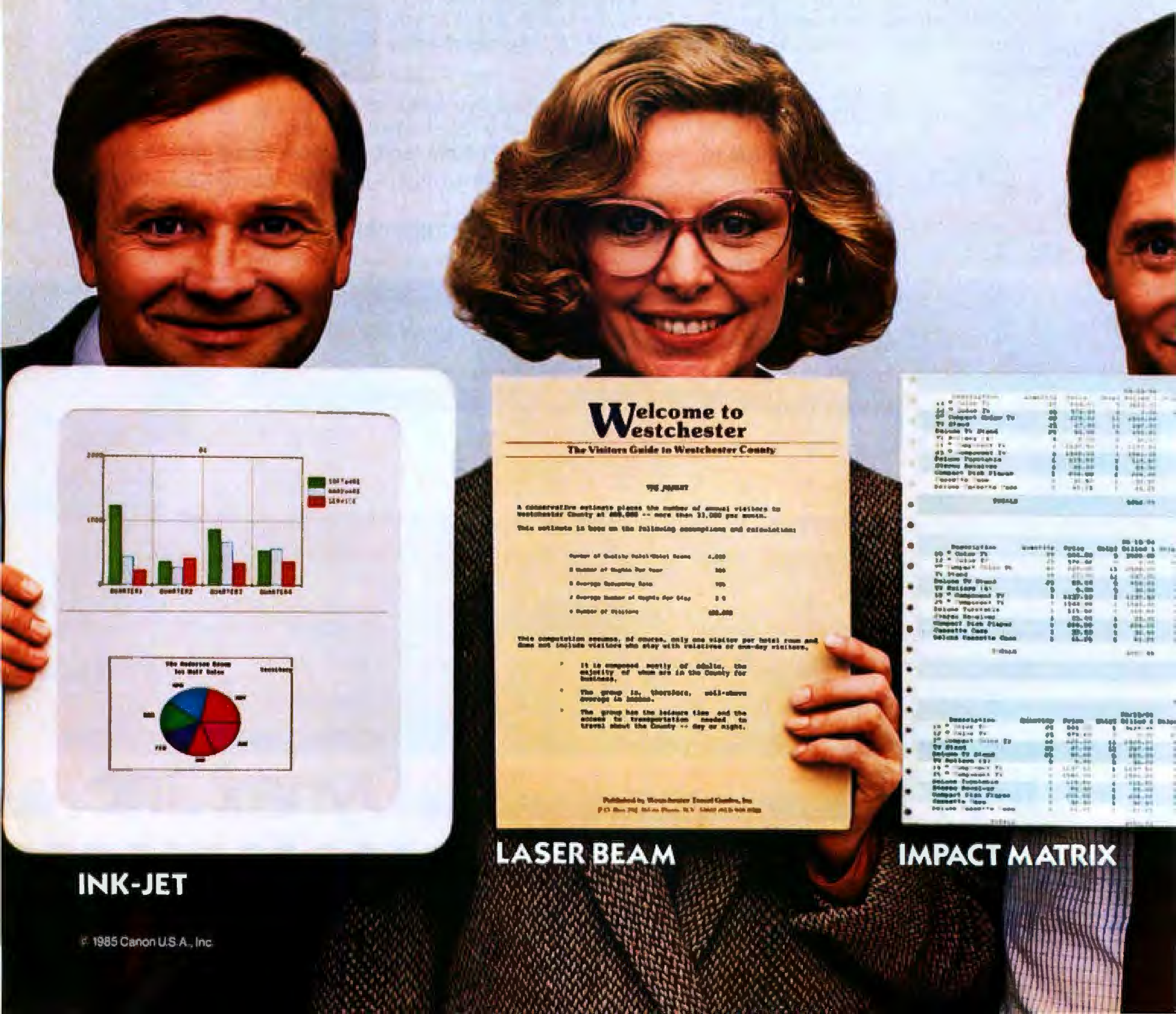
Take a close look at Canon's line of printers. The first thing you'll notice is how good they make you look.

That's because Canon's experience in high-tech optics really shows. For instance, Canon was among the first to make a desktop Laser Beam Printer. At a price that's within any business's reach, the Canon Laser Printer can produce eight pages a minute with a quality that's more in common with a professional print shop than a personal computer.

Canon also developed the ingenious Bubble-Jet Printer technology, finally making it possible for a printer to not only work extremely quickly but also incredibly quietly.

Our Ink-Jet Printer can produce exceptionally sharp, high-resolution graphics in seven colors, even on transparencies. While the Thermal Transfer Printer has three kinds of print modes, including letter quality, at a very competitive price.

And the Canon Impact Matrix Series, for regular and extra-wide column paper, makes printing very fast, very economical.



A MIDI Recorder

Store and play back keyboard music with your IBM PC

BY DONALD SWEARINGEN

COMPUTERS HAVE been used in music and related areas for some 30 years, but, until recently, only by those with access to the costly and scarce computing resources available at large institutions such as universities. Now, the technological advances that gave rise to the personal computer are being applied to a new generation of musical instruments, promising to bring computer music within reach of a wide range of eager participants. At the center of this activity is the musical instrument digital interface, or MIDI, a specification for the exchange of information in digital form among intelligent musical instruments. The specification, standardized by several prominent manufacturers of synthesizers and electronic musical instruments in 1983, defines both a hardware standard for connection of MIDI devices and a software protocol for data interchange.

In the past two years a flood of new MIDI products, from keyboard synthesizers to music computers, has appeared. As has been the case with personal computers, MIDI hardware has often preceded the software. However, we are seeing more and more MIDI software, ranging from simple recording and playback programs to elaborate composing systems capable of converting music played on a synthesizer keyboard to standard music notation.

The subject of this article is a simple MIDI recorder program, capable of recording a performance played on the keyboard of a MIDI-equipped music synthesizer and faithfully reproducing the performance on playback. My program, MPU401.PCF, written in Laboratory Microsystems' PC/FORTH for the IBM PC, provides for the recording and playback of MIDI data on any of eight separate tracks. It communicates with a MIDI-equipped music synthesizer (see figure 1) via the Roland Corporation MPU-401 MIDI Processing Unit (MPU), an intelligent controller that relieves the IBM PC of much of the processing load associated with the real-time MIDI information. (*Editor's note: The source code for* MPU401.PCF *is available for downloading via* BYTE*net* Listings. *Call* (617) 861-9774 *before* November 1; *thereafter, call* (617) 861-9764. *You will need* PC/FORTH *in order to use the program.*)

The basic record/playback function is a good starting point for exploring the possibilities of MIDI software. In addition to MIDI considerations, my MIDI recorder program addresses several other issues of interest to IBM PC programmers. Among these are the processing of real-time interrupts, the use of execution vectors, and memory allocation via calls to PC-DOS.

Before I discuss the software itself, you should be aware of a few of the

(continued)

Donald Swearingen (100 Valencia #261, San Francisco, CA 94103) is a freelance software developer, consultant, musician, and composer.

ILLUSTRATED BY KATHERINE MAHONEY

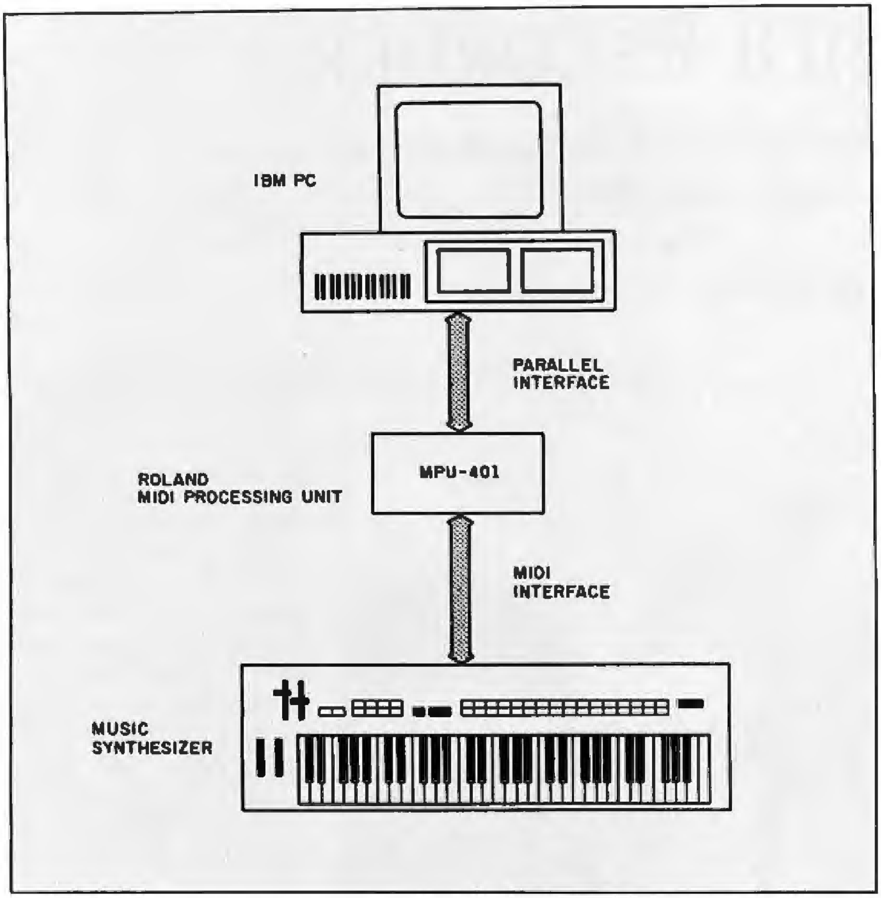


Figure 1: *Interfacing a music synthesizer with an* IBM PC *via the* MIDI *Processing Unit* (MPU).

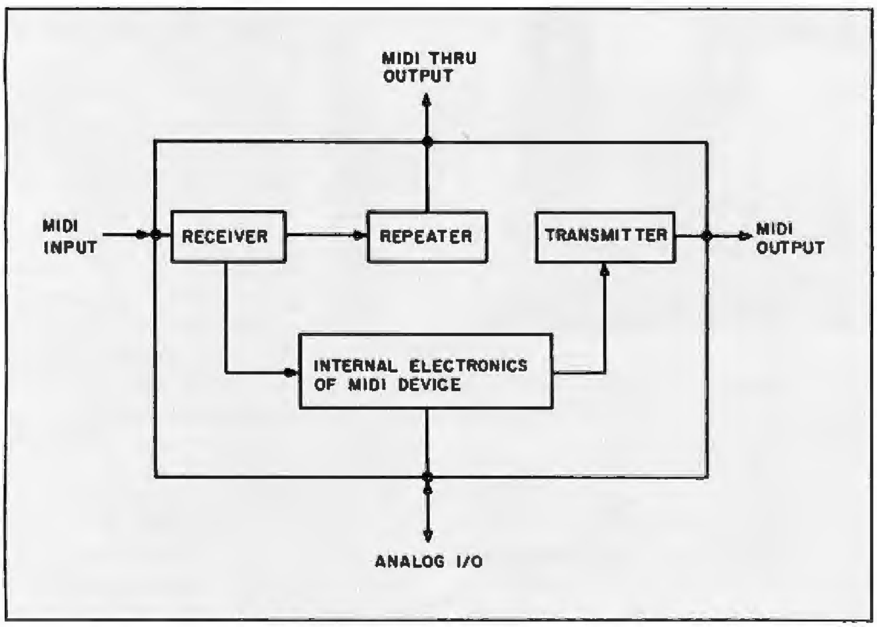


Figure 2: *The internal components of a typical* MIDI *device.*

technical details of the MIDI standard and of the MPU-401. For the sake of brevity, I will limit my discussion to those items that are crucial to an understanding of the application at hand.

THE MIDI STANDARD

The MIDI standard allows you to connect MIDI-equipped devices in a simple bus network via an asynchronous link. The standard MIDI interface operates at 31.5K bps (bits per second) with 1 start bit, 8 data bits, and 1 stop bit, for an effective transfer rate of about 3K bytes per second.

A MIDI device (see figure 2) normally contains both a receiver (*MIDI* input) and a transmitter (MIDI output). Some MIDI devices also contain a repeater module, which routes the receiver section's unaltered input to an additional output (MIDI Thru output) for connection to other MIDI devices. By daisy-chaining the Thru output of one MIDI device to the input of the next, you can connect up to 16 receivers in a MIDI bus network. However, you can only have one transmitter in a given network because there is no bus arbitration. You can configure MIDI devices in many other interesting ways; basically, any combination of output to input connections is allowable, even in the same MIDI device.

All communication among MIDI devices is carried out by the transmission of MIDI *messages* containing both data and commands for MIDI devices. A MIDI transmitter originates MIDI formatted messages, while a MIDI receiver recognizes and acts upon messages in standard MIDI format. MIDI messages consist of a *status byte* and one or two *data bytes*. Status bytes, 8-bit values with the high-order bit always set (i.e., >= 80 [hexadecimal]), indicate the type of command or function to be performed (or being performed) by a MIDI device. Data bytes, 8-bit values with the high-order bit always reset (i.e., < 80 [hexadecimal]), follow the status byte and serve as parameters or modifiers to the commands.

(continued)

Back up 10 megabytes in 8 minutes! (or less)

with Fastback™

If you've ever lost valuable data, you know backing up your hard disk is a *MUST.* For your business, this necessity can become an expensive nightmare. Until now. Fastback™ (Version 5.0) is the hard disk backup (and restore) software utility that eliminates the need for expensive and time consuming tape backup systems. It's so fast you can back up a full 10-megabyte hard disk on standard 5¼" floppies in less than 8 minutes or 10 megabytes of data on the IBM™ PC-AT in less than 4 minutes! This can mean a 95% savings of time and money. And it's easy to use.

Fastback is fully self-contained and functions with any make or size hard disk. It works with PC-DOS™, or MS-DOS™ version 2.0 or higher and requires no additional hardware to take up valuable expansion slots in your IBM PC/XT/AT or compatible. FASTBACK USES ADVANCED *ERROR CORRECTION* TECHNIQUES TO RECOVER DATA FROM DAMAGED DISKETTES. So if your disk is damaged after the backup procedure, Fastback (Version 5.0) will still be able to recover the data.

Fastback allows rapid lookup of archived data and prevents accidental overwriting. Fastback is completely file oriented, not an image mode backup.

Suggested Retail Price: *$179.*

INTRODUCTORY OFFER

Yes, send me________Fastbacks at the Special Introductory price of $149 each. That's $30 off the regular retail price of $179.

I am enclosing: ☐ BankDraft ☐ Check **$30 OFF**

Name: __________
Address: __________
City: __________
State: __________ Zip: __________
Telephone (____) __________

These prices include shipping to all U.S. cities. All foreign orders add $10 per product ordered. COD's and purchase orders WILL NOT be accepted by Fifth Generation Systems. Outside USA: payable in U.S. dollars drawn on a U.S. bank. FASTBACK is also available at better dealers nationwide. Call (213) 439-2191 for the dealer nearest you.

Offer Expires December 31, 1985

Fifth Generation SYSTEMS
7942 Picardy Avenue
Baton Rouge • LA • 70809
Telephone: (213) 439-2191

Inquiry 63

Dealer Inquires Invited

Marketing and Sales by CSSL, Inc.

IBM-PC, IBM-XT, IBM-AT and PC-DOS are registered trademarks of International Business Machines, Inc. MS-DOS is a registered trademark of Microsoft, Inc. Fastback is a registered trademark of Fifth Generation Systems.

MIDI messages are generally divided into two types: *channel messages* and *system messages*. To understand my MIDI recorder program, you need only be concerned with a subset of the channel messages—the *channel voice messages*. Channel voice messages (see table 1) include Note On/Note Off events (i.e., a key being pressed or released), control changes (volume, modulation, etc.), and program changes (the selection of unique "patches" on music synthesizers).

All channel messages contain, in the low-order bits of the status byte, a 4-bit number that directs the MIDI message to one of up to 16 MIDI devices attached to the MIDI network (see figure 3).

Each MIDI device attached to the bus is responsible for recognizing its channel number and responding appropriately to the MIDI message. A receiver of a channel voice message always adopts the command status indicated by the last status byte it received. This is referred to as the channel's *running status* and the receiver remains in this state until a new status byte is received. Thus, if a given status is to be repeated, it may be optionally omitted, and only the proper number of data bytes for that status need be sent. Consider, for example, the *MIDI data stream* (MDS), consisting of several MIDI messages, shown in figure 4.

Note the manner in which the running status is used to reduce the number of bytes required to specify a MIDI message. When a device must send long strings of data for the same command (e.g., Note On/Note Off), it can realize a considerable savings in the number of bytes transmitted by using the running status.

Table 1: *The* MIDI *channel voice messages. All data bytes are in the range from* 0 *to* 127 *unless otherwise specified. A velocity of* 0 *for a* Note On *message indicates* Note Off, *allowing the use of running states for strings of* Note On/Note Off *messages.*

Command Field of Status Byte	Message Description	Data Byte(s)
000	Note Off	key note off velocity
001	Note On	key velocity (note off if velocity=0)
010	Polyphonic key pressure	key pressure
011	Control change	control # (0–121) control value
100	Program change	program #
101	Channel pressure	pressure
110	Pitch wheel change	pitch wheel least significant byte pitch wheel most significant byte

THE MIDI PROCESSING UNIT

Roland's MPU-401 (see figure 5) is an intelligent interface containing its own CPU (central processing unit) and an LSI (large-scale integration) handshake controller for the MIDI network. The MPU connects to the IBM PC via a parallel interface. It uses the *IRQ2* interrupt to initiate data transfer between itself and the IBM PC. The use of a smart interface allows the IBM PC to perform other tasks while the MPU is recording or playing. The MPU operates independently, managing the MIDI bus; it interrupts the IBM PC only when a data transfer is required. Because of this the IBM PC can perform such tasks as disk I/O (input/output) and screen update while the MIDI record/playback process is in operation. The use of IRQ2, which has a higher interrupt priority than disk and other I/O interrupts, ensures that no MIDI data will be lost.

When instructed to start recording MIDI data, the MPU transmits MIDI

(continued)

BIT	7	6	5	4	3	2	1	0
VALUE	1	COMMAND			CHANNEL			

Figure 3: *The* MIDI *status byte for channel voice messages.*

	1	2	3	4	5	6	7
MDS MIDI MESSAGES	90 20 50	20 00	21 60	21 00	C0 04	90 30 65	30 00
CH. 1 RUNNING STATUS	90				C0	90	

Figure 4: A MIDI *data stream* (MDS) *consisting of several* MIDI *messages with the running status shown.*

messages received at its MIDI input to the IBM PC for storage. A single MIDI data stream transmitted to the IBM PC during the record process is referred to as a track. Each of the eight tracks is allotted a separate area of the IBM PC's memory for storage and for later playback. In playback mode, the MPU requests the transfer of track data for performance at the appropriate times.

Through the use of an internal clock, the MPU manages all timing and synchronization for the MIDI data stream. The MPU's internal clock rate controls the basic tempo for the record/playback process and can be varied between 8 and 250 beats per minute. In turn, each beat is subdivided into smaller units according to the value of an internal *time base*. The internal time base can be set to seven values, ranging from 48 to 192 divisions per beat. At initialization, the internal time base defaults to 120 divisions per beat. Therefore, for a tempo of 60 beats per minute, each division of the internal time base would be equal to 1/120 of a second.

When the recording process begins, the MPU sets a *record counter*, used for timing of incoming MIDI messages, to 0. Thereafter, it increments the record counter by one for each count of the internal time base. The record counter is allowed to range in value from 0 to 239 (00 to EF [hexadecimal]) and can thus represent timing intervals in the range from 0 to 2 beats. When a MIDI message is received at the MPU's MIDI input, the current value of the record counter is copied and used as a prefix to the MIDI message to form a *track event*. The track event is then transmitted to the IBM PC and stored in the computer's memory. The record counter is then reset to 0 and the process is repeated for the next MIDI event. If the record counter reaches 239 before a MIDI message is received by the MPU, the counter is reset to 0 and a timing overflow indicator (F8 [hexadecimal]) is transmitted to the IBM PC indicating that no MIDI messages have been received during the preceding 240 counts of the internal time base (null MIDI event). Such a sequence of track events is referred to as a *track data stream*. With the addition of the relative timing values, the MIDI data stream above would be transmitted to the IBM PC as shown in figure 6.

Here the event times represent the sum of all preceding timing bytes in the MIDI event stream; for example, the sum of the timing bytes of the first three events (00+60+10) is 70 (hexadecimal), and the event time (0.93 second) represents the actual offset of the third event from the start of the MIDI event stream (70 [hexadecimal] = 112 [decimal], 112 ÷ 120 divisions per second = 0.93 second).

In playback mode, the MPU assigns a unique *play counter* to each active track. When the playback process is begun, the MPU requests a MIDI event for each active track. The timing value of the MIDI event is copied into the play counter for the track and the MIDI message is temporarily stored. The play counter is then decremented by one for each count of the internal time base. When the play counter reaches 0, the MIDI message is transmitted to the MIDI bus. The MPU then requests the next MIDI event for that track from the IBM PC and the process is repeated. A MIDI event stream received from the MPU in record mode can thus be sent unaltered to the MPU in playback mode for a faithful reproduction of the original sequence of MIDI messages.

The exchange of track events during the record and playback processes constitutes the bulk of the interactions between the MPU and the IBM PC. In addition to the track events, the MPU sends other *MPU messages* (see table 2) to the IBM PC indicating various conditions. The IBM PC, in turn, controls the actions of the MPU by sending *MPU commands* (see table 3). It should be emphasized that the tables represent only a small portion of the repertoire of commands and functions the MPU is capable of performing. A more complete picture of the MPU's capabilities is hinted at

(continued)

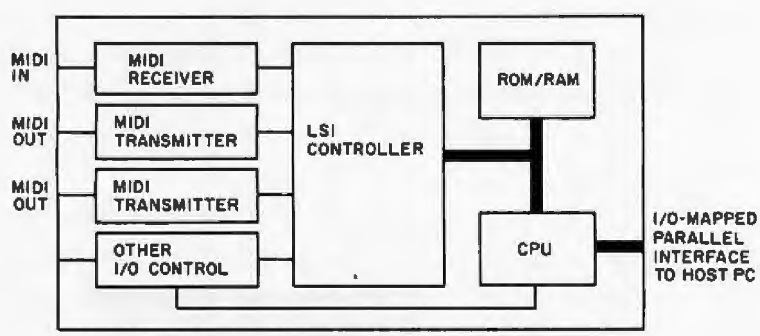


Figure 5: *The Roland* MPU-401 MIDI *Processing Unit.*

	1	2	3	4	5	6	7	8
TRACK EVENTS	00 90 20 50	60 20 00	10 21 60	80 21 00	F8	20 C0 04	40 90 30 65	A8 30 00
EVENT TIMES	0.00	0.80	0.93	2.00	4.00	4.26	4.80	6.20

Figure 6: A *track data stream based on the* MIDI *data stream from figure* 4 *with the addition of relative timing values.*

in the section of my program labeled "MPU-401 Control Words."

One further note concerns the MPU's use of a simple handshaking scheme for synchronizing its communications with the IBM PC. When it sends an MPU message to the IBM PC, the MPU remains in a software loop until the entire message has been read by the IBM PC. The MPU can accept no new commands or data from the IBM PC until it completes the message.

Without this condition the IBM PC could send a command to the MPU while the MPU was simultaneously sending a byte to the IBM PC (the first byte of an MPU message). The MPU would then expect the IBM PC to read this byte (as well as the rest of the MPU message) while the IBM PC would want to send the rest of its command.

To prevent this situation from occurring, the MPU employs a command acknowledgement (ACK) character (FE [hexadecimal]). When the MPU accepts a command from the IBM PC, it transmits the ACK character, indicating that the IBM PC can now transmit succeeding data bytes (if any) associated with the MPU command.

Table 2: *The* MPU *messages sent by the* MPU *to the computer to indicate various conditions. Note that the* MIDI *recorder program ignores the* MPU *commands* F9, FC, FD, *and* FF. *All message numbers are in hexadecimal.*

MPU Message	Message Description	Comments
00–EF	Track event	In record mode, the first byte (timing byte) of a track event to be read by the host IBM PC
F0–F7	Track data request	In playback mode, the MPU requests the next track event for tracks 1 (F0) to 8 (F7)
F8	Timing overflow	The record counter has reached 240 during record mode
F9	Conductor data request	The MPU requests the next track event for the "Conductor" track
FC	All end	Indicates all tracks have finished playing in playback mode
FD	Clock to host	Clock tick to host (see MPU commands 94, 95, and E7)
FE	Command ACK	Acknowledgment that the MPU has received a command from the IBM PC
FF	System-exclusive message	MIDI system-exclusive message follows

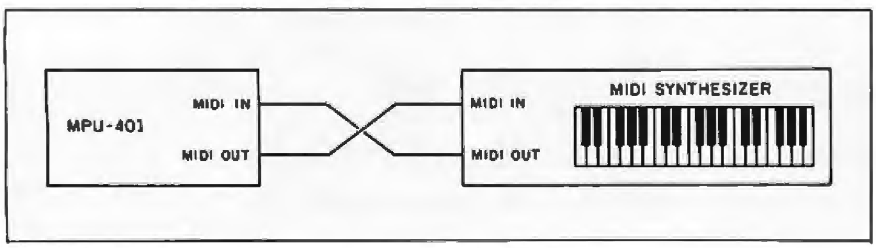


Figure 7: *Interfacing the* MPU-401 *to a* MIDI *synthesizer.*

MIDI Music Synthesizer

The music synthesizer used in our example has both a MIDI input and a MIDI output. It is connected to the MPU-401 by two cables, as shown in figure 7.

All notes played on the synthesizer's keyboard are transmitted to the MPU-401 through the synthesizer's MIDI output, and all notes received at the synthesizer's MIDI input are performed by the synthesizer as though they had actually been played on its keyboard.

The FORTH Environment

My program, MPU401.PCF, is made up of a number of objects that work together in the FORTH environment to perform the functions of a MIDI recorder. The program's organization reflects FORTH's prohibition of forward references. Since FORTH compiles new programs into its workspace in a one-pass operation (in the same manner that it interprets keyboard input), MPU401.PCF is organized "bottom-up," with the more primitive definitions preceding those at higher levels.

At first glance, the listing appears similar to a standard 8088 assembly-language program, and, indeed, many of the traditional elements of assembly language are present. On closer examination, however, you may notice a number of differences that will seem strange if you are unfamiliar with FORTH programming conventions.

To begin with, FORTH's use of a stack for the passing of arguments and data among procedures (*words* in FORTH) requires you to use reverse Polish notation to form executable FORTH statements. As a result, many FORTH programs appear to be "backward" to those accustomed to more traditional computer languages.

In FORTH, the key word CODE informs the FORTH compiler that the following statements, up to the word END-CODE, are to be compiled as code words using FORTH's built-in 8088 assembler. Ordinarily, code words end with the statement NEXT, allowing them to be used interactively with the FORTH interpreter in the same manner as other FORTH words. Code words intended for this kind of usage always end with the statement NEXT, rather than with a traditional return (RET) statement.

The MIDI recorder program uses two such code words, FREE_MEM and GET_MEM, which are intended for use as traditional FORTH words, receiving their inputs and returning their outputs on the FORTH stack. All other code words in the MIDI recorder program, however, are intended as objects of internal CALL statements and therefore contain RET statements, as would normal assembly-language subroutines. Regarding the calling of code words that contain RET statements, you must use a special sequence of instructions to ensure that you actually jump to the executable code (parameter field address) of the subroutine, skipping a pointer (code field address) FORTH places at the beginning of code words. This is accomplished by a statement of the form:

```
' MPU_RECD >BODY CALL
```

The same convention applies to jump (JMP) statements. Those familiar with traditional 8088 assembly language should have little difficulty with the format of the FORTH code words.

Until recently, FORTH programs were customarily created using "screens," 1K-byte blocks of memory that could readily be displayed, edited, and stored in a small-computer environment. Today, many versions of FORTH offer the alternative

(continued)

Table 3: *The* MPU *commands the computer uses to control the* MPU.

Command Byte	Command Description	Comments
22	Start record process	The record counter is cleared and the recording process is begun.
11	Stop record process	
0A	Start playback process	Playback is begun for all active tracks as set by the SET_AC_TRK command.
05	Stop playback process	
2A	Start overdub process	The record and play processors are activated simultaneously.
15	Stop overdub process	
FF	Reset MPU-401	Reset the MPU-401 to the power-up defaults.
33	Disable MIDI Thru function	MIDI Thru function from MIDI In to MIDI Out is disabled.
86	Screen bender data	Bender (continuous) data is not transmitted to the host IBM PC.
87	Allow bender data	
8C	Disable measure end	Measure end marks are not sent to the host IBM PC.
8D	Enable measure end	
8E	Disable conductor track	The conductor function is disabled.
8F	Enable conductor track	
94	Disable clock to IBM PC	
95	Enable clock to IBM PC	
96	Screen system-exclusive data	MIDI system-exclusive data are not transmitted to the host IBM PC.
97	Allow system-exclusive data	
B8	Clear play counters	The play counters are cleared for all active tracks.
BA	Clear record counter	The record counter is cleared.
E0	Set tempo	The tempo is set to the value indicated by the following data byte to the MPU.
EC	Set active tracks	The active tracks for playback are set according to the following data byte.

of compiling FORTH source code from traditional text files. This alternative is provided by PC/FORTH through its INCLUDE statement. I created MPU401.PCF using a standard text editor and compiled it using the INCLUDE feature. The appearance and maintainability of the source code are greatly enhanced by the use of a text editor that allows you to create code segments and commentary of arbitrary length.

FORTH allows the free use of comments between a freestanding left parenthesis and a right parenthesis. I made liberal use of comments in the program listing to improve its readability and maintainablilty.

THE MIDI RECORDER PROGRAM

MPU401.PCF begins with the definition of the data structures (constants, variables, buffers, pointers, etc.) to be used by the various routines. Items such as the I/O port addresses for the MPU, the track size, and the number of tracks are defined as constants. Following are the definitions of five arrays to be used in the management of track data. For each track, T__SEG will contain a pointer to the segment of the 8088's memory-address space where the data for that track are stored. T__RPTR and T__PPTR will contain, for each track, a pointer to the offset (within the track's memory segment) of the next location to be used for recording or playback of track events. T__RST is used to hold the current running status for each track, and T__NDAT stores the number of data bytes associated with the current running status of each track. The variable ACK__RCVD is used to coordinate the sending and receiving of MPU communications. PREV__IRQ2 is used to retain the previous contents of the IRQ2 vector for later restoration. REC__TRK contains the number (0–7) of the track currently being recorded. MPU__VEC is used by the IRQ2 interrupt routine to vector execution to the proper character handler, and MPU__VEC__RST contains the address of the routine that handles the first character of an MPU message, used to reset MPU__VEC after an MPU message has been processed.

The code definitions begin with two functions used to allocate memory outside the FORTH workspace. This memory will be used for the storage of track data in order to reserve memory within the FORTH workspace, which has a 64K-byte limit, for additional word definitions. Another benefit of this approach is that you can make your track-data buffers as large as you wish, subject only to the available memory and the 64K-byte limit imposed by 8088 segment addressing. FREE__MEM releases all of the IBM PC's unused memory for use by other processes, returning the number of available memory paragraphs (16-byte sections) on the stack. GET__MEM is used to allocate memory from this available pool. It accepts as input the number of requested paragraphs and returns a pointer to the allocated memory (seg), the number of paragraphs actually allocated (n__alloc), and a PC DOS error code (0, if no error).

The next section contains definitions of routines used to send data and command output to the MPU. Each object in this section is defined both as a FORTH word and a callable assembler subroutine so that the function it performs will be usable in either application. Of particular note here is the manner in which the word !MPU__CMND, which sends a command to the MPU, utilizes the ACK__RCVD flag. Before a command is sent to the MPU, the flag is set to 0. After sending the command byte, !MPU__CMND waits in a loop until ACK__RCVD is set to a nonzero value by the routine MPU__MSG, which handles messages from the MPU.

Following the MPU output words are the routines that handle entry and exit from IRQ2 interrupts. When an IRQ2 interrupt occurs, execution is vectored to IRQ2__INT. IRQ2__INT saves all appropriate CPU registers, reads the character from the MPU data port, and then jumps to the MPU character handler pointed to by the vector MPU__VEC. After the character has been processed, IRQ2__INT__END is executed. It sends an end-of-interrupt (EOI) signal to the PC's 8259 interrupt processor, restores the CPU registers from the stack, and returns from the interrupt.

The next section includes the basic routines used in the recording and playback processes, MPU__RECD and MPU__PLAY. Following that is the MPU message interpreter, MPU__MSG, which processes the first character of all MPU messages. These routines constitute the core of the recording and playback process.

MPU messages (see table 2) consist of either a track event (to be recorded) or a single-byte message.

```
Interpret MPU Message

(note: the MPU character handler vector is initially set to the address of MPU__MSG
to process the first character of an MPU message)

MPU__MSG:
begin
if command byte is MPU ACK then
        set command ack flag (ACK__RCVD) to signal command received
        jump to end-of-interrupt
else if command byte is a timing value then
        jump to track event recorder routine (MPU__RECD)
else if command byte is a play request
        jump to track event play routine (MPU__PLAY)
else
        jump to end-of-interrupt
end
```

Figure 8: *The algorithm for the* MPU *message interpreter,* MPU__MSG.

MPU__MSG is responsible for taking an appropriate action in response to the MPU message. The algorithm for MPU__MSG is shown in figure 8.

MPU__RECD is responsible for the recording of track events received from the MPU (see the algorithm in figure 9). Track events (see figure 10) may be from 1 to 4 bytes in length. In the interrupt environment, you only receive one character at a time, and it is not possible to scan ahead in the input stream in order to make decisions about how to handle multibyte messages. MPU__RECD solves this problem through the use of the MPU character handler vector, MPU__VEC. After a track-event character has been processed, MPU__VEC is set to the address of the routine that will handle the next character of the track event when the next interrupt occurs. After the entire track event has been processed, MPU__VEC is reset to the address of MPU__MSG in order to process the next MIDI message.

MPU__PLAY (see the algorithm in figure 11) is responsible for transmitting a track event to the MPU in response to a track-data request, the inverse of the operation performed by MPU__RECD. Here, by contrast, execution vectors are unnecessary since you are transmitting to the MPU.

The remainder of the routines in the program listing are written as standard FORTH colon words, using the words and data defined in the preceding sections.

The interrupt control words are responsible for initializing the IRQ2 and MPU character handler vectors and for enabling or disabling the IRQ2 interrupt. They demonstrate the sequence of actions necessary for the proper activation and deactivation of prioritized interrupts (IRQ0–IRQ7) on the IBM PC.

The MPU control words consist of the standard MPU commands and the record/playback control words. MPU__INIT initializes the state of the MPU at start-up to the state that is required for proper interaction with the MIDI recorder program. MPU__ON and MPU__OFF are short words that are used to activate and deactivate MPU communications.

```
Record Track Event

(process timing value of track event)
MPU__RECD:
begin
record timing value for track event
if not a null track event (timing overflow) then
        set MPU character handler vector to routine that will process
          the second character of the track event (MPU__RECD__2)
jump to end-of-interrupt
end

(process second character of track event)
MPU__RECD__2:
begin
if character is an MPU mark then
        record the mark
        reset MPU character handler vector to MPU__MSG
        jump to end of interrupt
else
        set MPU character handler vector to routine that will process
          the data bytes of the track event (MPU__RECD__3)
        compute and store the number of data bytes for the current running
          status of the track being recorded
        if character is not a status byte then
          jump to routine to process data bytes of the track
            event (MPU__RECD__3)
        else
          store the new running status
          compute and store the number of data bytes for the new
            running status
          jump to end-of-interrupt
end

(process data bytes of track event)
MPU__RECD__3:
begin
store data byte
decrement count of data bytes for current track event
if count is 0 then
        reset character handler vector to MPU__MSG
jump to end-of-interrupt
end
```

Figure 9: *The algorithm for the track event recorder,* MPU__RECD.

The record/playback control words are at the highest level in the hierarchy of modules in the MIDI recorder program. They are the words that constitute the program's simple user interface. Once compiled into the FORTH dictionary, they can be executed interpretively from a keyboard or included in more sophisticated programs.

RESET__RPTR resets the record pointer for a given track to 0, in effect erasing the contents of the track (a nonzero record pointer always points to the byte following the end of a track's data). RESET__PPTR resets a track's play pointer to 0 so that its track data will be performed from the start.

Before any track data can be recorded, you must allocate the memory to be used for track-data storage. RECORD__INIT invokes FREE__MEM and GET__MEM to obtain the necessary memory and stores pointers to the track buffers in the T__SEG array. It then loops to reset the record and play pointers for all

(continued)

tracks. Once MPU__ON and RECORD__INIT have been executed, you are ready to begin the recording and playback process.

RECORD is responsible for recording a performance on a single track. When RECORD is used, the MPU does not request track data for performance on other tracks. Therefore, only the track being performed is heard. RECORD is invoked with the number (0–7) of the track to be recorded on the stack. The statement 0 RECORD starts the recording process and records data on track 0. As a consequence of the interrupt scheme, the actual recording and playing of track data is carried out in the background, allowing you to continue executing other FORTH commands while the process is in operation. The recording process is ended by the command RECORD__OFF.

Recording and playing of track data goes on in the background, so you can execute other FORTH *commands as the process continues.*

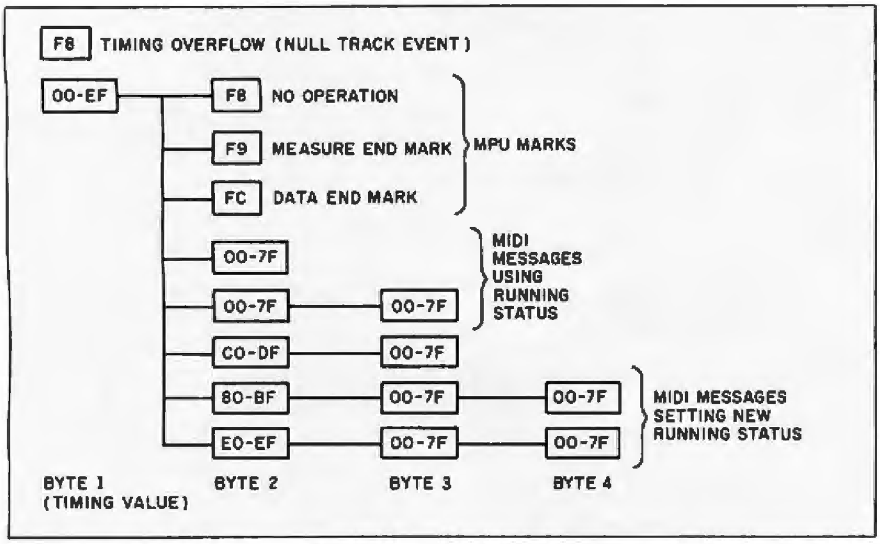


Figure 10: *The format of track events received from the* MPU. *All track events are in hexadecimal.*

```
Play Track Event

MPU__PLAY:
begin
if track buffer is empty then
        send data end to MPU (00,FC)
        jump to end-of-interrupt
read timing value from track buffer
send timing value to MPU
if timing value was overflow (F8) then
        jump to end-of-interrupt
if next character in track buffer is an MPU mark then
        read mark from track buffer
        send mark to MPU
        jump to end-of-interrupt
if next character in track buffer is a MIDI status byte then
        set running status for teack to new status byte
        read status byte from track buffer
        send status byte to MPU
compute and store the number of data bytes for the current running status
repeat
        read next data byte from track buffer
        send data byte to MPU
        decrement count of data bytes for current running status
until count is 0
reset play pointer for track to reflect data played
jump to end-of-interrupt
```

Figure 11: *The algorithm for the track event performer,* MPU__PLAY.

OVERDUB functions similarly to RECORD, except that all other active tracks are played back while the overdub track is being recorded. Using OVERDUB, performances on the MIDI synthesizer can be "layered" on up to eight tracks.

The PLAY command simply plays back the track data for all active tracks. No track data is sent to the IBM PC by the MPU.

CONCLUSION

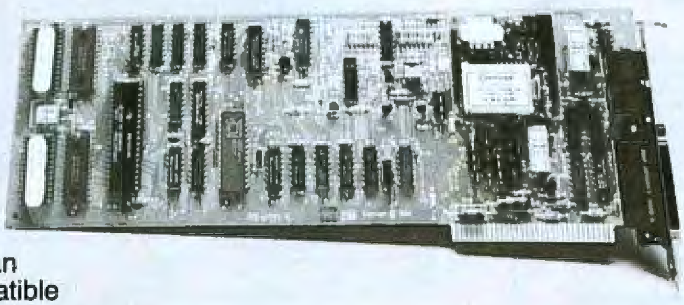
The MIDI recorder program provides a foundation for the development of an expanded programming vocabulary for the musical instrument digital interface. Although I have only touched the surface of issues pertinent to MIDI programming, the functions provided by MPU401.PCF constitute a useful and instructive introduction to this subject. In the near future, we can look forward to new developments in MIDI software that will greatly stimulate musical creativity. ■

The Business Modem
that speaks for itself...

"My **VOICE MESSAGE** capabilities help you stay in touch . . . with complete privacy. Personal ID codes make sure the right message gets to the right person."

"My **VOICE MAIL** system means business . . . 24 hours a day, 7 days a week. I let you take orders or exchange information from any touch-tone telephone, anywhere in the world."

"I'm a true **BUSINESS MODEM;** handling both VOICE and DATA. I'm also 100% Hayes compatible, 300/1200 Baud and use most popular communications software."



"My **TIME MANAGEMENT** system plans your business day. On-line features include a Voice Calendar, Alarm, Auto-Dial Phone Book, and much, much more!"

"Every business with an **IBM PC, XT, AT** or compatible needs me. I can even run in BACKGROUND MODE while you use your PC for other tasks."

Talk to Watson Yourself.
Call 1-800-6WATSON
(617-651-2186 in Massachusetts)
for a demonstration.

Watson
Voice + Data

Natural MicroSystems Corporation
6 Mercer Road, Natick, MA 01760 617-655-0700.

Inquiry 139

$100 COUPON OFFER

Purchase Watson prior to November 30, 1985 and receive coupons worth up to $100.00. See your dealer or call 1-800-6WATSON for details.

COMPLETE SYSTEMS
(Optional local warranty service available in most major metro areas)

IBM PC BASE SYSTEM
IBM PC w/256K
Floppy Drive Controller
2 Double Sided Double Density Disk Drives

$1,495.02*

IBM PC HARD DISK SYSTEM
IBM PC w/256K
Floppy Drive Controller
1 Double Sided Double Density Disk Drive
Half Height 10MB Disk Subsystem

$1,749.00*

CALL FOR LATEST IBM ANNOUNCEMENTS

CUSTOM CONFIGURATIONS WELCOME

COMPAQ™

PORTABLE HARD DISK SYSTEM
w/256K/1 Floppy/Hard Disk

$2,235.00* w/10MB
$2,369.00* w/20MB

DESKPRO SYSTEM WITH TAPE BACKUP
640K CPU/1 Floppy/Hard Disk
10MB Tape Drive/Monitor

$2,951.94* w/10 MB
$3,151.94* w/20MB

Wholesale Prices Change Rapidly...If you ever see a lower price advertised anywhere please call! You will find the Network's Price will always be the best!

64K IBM PC MEMORY EXPANSION KITS
Quantity Discounts Available.
Guaranteed for Life!

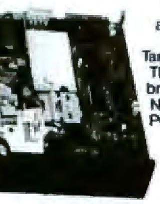
Quantity of 100 Sets **$3.96*** per set

Set of 9 Chips **$5.40***

BRAND NAME DISKETTES
DS/DD Box of 10 *Guaranteed for Life! Not Generic* **$8.95***

DS/DD Bulk Pack Special
Packaged in 50 *Guaranteed for Life!* .... Each **.79***

1/2 HEIGHT DS/DD DISK DRIVES



The Network buys direct and makes fantastic deals with manufacturers like Tandon/CDC/Shugart/Qume/TEAC/Hi-Tech and others to bring you fantastic prices on Name Brand drives for your PC/AT/XT/jr/or Compatible.

$69.00*

Quantity Discounts Available

INTERNAL PC HARD DISK
Low Power/Automatic Boot. Works on standard PC's and Compatibles. Includes drive/controller/cables/mounting hardware and instructions.

Full one year warranty!

10MB **$330.00***
20MB **$420.00***

Quantity Discounts Available

Lotus™ 1-2-3
$265.00*

MultiMate™
Professional Word Processor
$200.00*

***Members pay 8% above this wholesale price plus shipping.**

CALL TOLL FREE **1-800-621-S-A-V-E** (orders and memberships)

In Illinois call **(312) 280-0002 Validation code: B3XX**

TM—Registered trademark of IBM and COMPAQ

PC NETWORK

... WITH THESE 15 UNIQUE BENEFITS

1 COST + 8% PRICING—The NETWORK purchases millions of dollars in merchandise each month. You benefit in receiving the lowest price available **and all at just 8% above published dealer wholesale price.**

2 OUR 600 PAGE WHOLESALE CATALOG—Members receive our 600 page wholesale catalog containing over 20,000 hardware and software products for the *IBM PC, APPLE* and over 50 other popular computer systems. **THE NETWORK'S CATALOG IS THE LARGEST SINGLE COMPILATION OF PERSONAL COMPUTER PRODUCTS AVAILABLE TODAY. NOW UPDATED QUARTERLY!**

3 IN-STOCK INSURED FAST HOME DELIVERY—The NETWORK maintains a **giant multi-million dollar inventory of most popular products,** allowing us to ship many orders from stock. Non-stock items are typically maintained in local warehouses just days away from The NETWORK and YOU. We pay all insurance expenses on your shipment. **EMERGENCY OVERNIGHT SERVICE IS AVAILABLE ON REQUEST.**

4 10 DAY RETURN POLICY—If you are not satisfied, for any reason with any hardware component purchased from The NETWORK within 10 days of receipt, we will refund your entire purchase (less shipping) with no questions asked.

5 MEMBERSHIP SATISFACTION GUARANTEE—If for any reason you are not satisfied with your membership within 30 days, we will refund your dues IN FULL.

6 EXPERIENCED CONSULTANTS—The NETWORK hires consultants, not order takers, to aid you in product selection. Our consulting staff possesses in excess of 150 man years of personal computer product experience. **We back our consultants with our money back guarantee: IF ANY PRODUCT RECOMMENDED BY OUR CONSULTING STAFF FAILS TO PERFORM AS PROMISED—WE WILL TAKE IT BACK AT OUR EXPENSE FOR A 100% REFUND.**

7 FREE TECHNICAL SUPPORT—The NETWORK supports every product it sells. Our qualified TECH-SUPPORT staff will help you assemble your system, interpret vendor documentation and get your software and hardware to work. **WE WILL GIVE YOU ALL THE HELP YOU NEED, WHEN YOU NEED IT—FREE!**

†8 OPTIONAL BUSINESS RENTAL LIBRARY—All members can join our BUSINESS RENTAL LIBRARY ***featuring over 1000 available titles*** for just $25 PER YEAR above the base membership fee. **This entitles you to rent business software AT JUST 20% of the DISCOUNT PRICE FOR A 14 DAY PERIOD. If you decide to keep the software, the entire rental fee is deducted from the purchase price. VIP MEMBERS GET A FULL 30 DAYS for just $30 above the V.I.P. base fee. This also includes the game library privileges for a $5 combination savings.**

†9 OPTIONAL GAME SOFTWARE RENTAL LIBRARY—The Game Rental library is available to members for just $10 PER YEAR and permits evaluation (or just enjoyment) of any game or educational software product as above.

10 SPECIAL SAVINGS BULLETINS—THE PRINTOUT—Issued Quarterly at no charge to Network members only! The Printout contains all the New Product listings and price changes you need to keep your Catalog up to date. Also, we buy excess dealer inventories, and store bankruptcy closeouts, which we turn around and make available to our members at fantastic savings via *THE PRINTOUT.*

11 DISCOUNT BOOK LIBRARY—Working with numerous publishers and distributors, The NETWORK has assembled a library of over 1000 computer related books and manuals at savings of up to 75% from the normal store price.

12 MEMBERSHIP REFERRAL BONUS—Our most valuable source of new members is you! To date almost 40% of our members have been referred by word of mouth from other satisfied members. For those of you who refer new members, The NETWORK will credit a cash bonus to your account applicable to any future purchase.

13 CORPORATE ACCOUNT PROGRAM—Almost 50% of The NETWORK's members are corporate buyers and users *(see opposite page left).* The NETWORK can establish open account status and assign designated account managers to expedite orders, and coordinate multiple location shipments.

14 QUANTITY DISCOUNTS—For large corporations, clubs, and repeat or quantity buyers The NETWORK can extend additional single order discounts, when available to us from our manufacturers and distributors.

15 PRICE PROTECTION—The PC Industry is crazy!! Prices change not yearly or monthly or even weekly but often day by day! These changes are sometimes up *but are mostly down!!!* **THE NETWORK GUARANTEES THAT IN THE EVENT OF A PRODUCT PRICE REDUCTION, BETWEEN THE TIME YOU PLACE YOUR ORDER AND THE TIME THE PRODUCT SHIPS YOU WILL ONLY PAY THE LOWER AMOUNT!!**

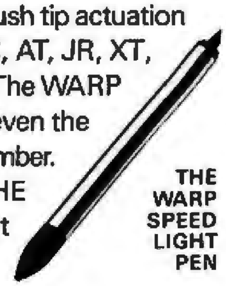
Inquiry 146

The Warp Speed™ Light Pen. Works As Fast As You Think.

First there was the keyboard. Then there was the mouse. Now the most powerful and versatile input device is the WARP SPEED LIGHT PEN with its powerful CONTROLLER software. What once required many keystrokes or taking your mouse for a walk, now occurs with the touch of a pen.

Instant Compatibility

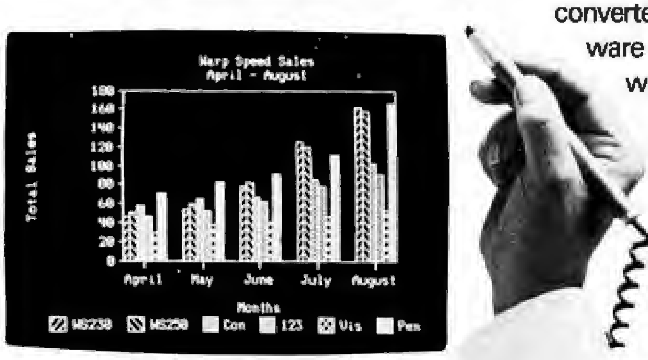
Instantly you're working faster. Simply plug the WARP SPEED LIGHT PEN into the light pen port on your monitor adapter card. The WARP SPEED LIGHT PEN with the push tip actuation switch is completely compatible with the IBM PC, AT, JR, XT, and almost all IBM compatibles and clones. The WARP SPEED LIGHT PEN works on all screens, even the difficult IBM P-39 green and Amdek 310 Amber. For additional speed and less fatigue, THE WARP SPEED LIGHT PEN'S lightweight Aluminum design is the lightest ever.



THE WARP SPEED LIGHT PEN

The Fastest Draw In The West

Run circles around the mouse with the new converter program PENMOUSE. It converts all Mouse operated software to light pen control. So now the light pen has more software applications than the mouse, including PC Paintbrush, PC Draw and Autocad. And it's only **$69.99**. Not copy protected.



Free Software and One Year Warranty

The WARP SPEED LIGHT PEN comes complete with its powerful CONTROLLER software and a full one year warranty. You can add even more versatility with driver programs for LOTUS 123, SYMPHONY, WORDSTAR, GEM or design a custom program with help from our expert staff of programmers.

The chance to work almost as fast as you think waits for you at your local dealer. Grab it.

To order direct call

1-800-874-4315

(In California 1-800-826-1563)

WARP SPEED LIGHT PEN Includes free CONTROLLER software. A complete file management system.	**$199.99**
CONDUCTOR 2.0 NEW VERSION Enhancements and Light Pen Controller for all versions of Lotus 123 and Symphony.	**$79.99**
PENMOUSE Converts all mouse software to Light Pen control.	**$69.99**
GEMLIGHT A Light Pen controller and enhancement driver for GEM.	**$69.99**
WARPSTAR A Light Pen controller and enhancement driver for WORDSTAR.	**$69.99**

All software not copy protected.

WARP SPEED™

SPEED FOR THOUGHT

WARP SPEED COMPUTER PRODUCTS
5555 Inglewood Blvd.
Los Angeles, CA 90230

Inquiry 212 for End-Users. Inquiry 213 for DEALERS ONLY.

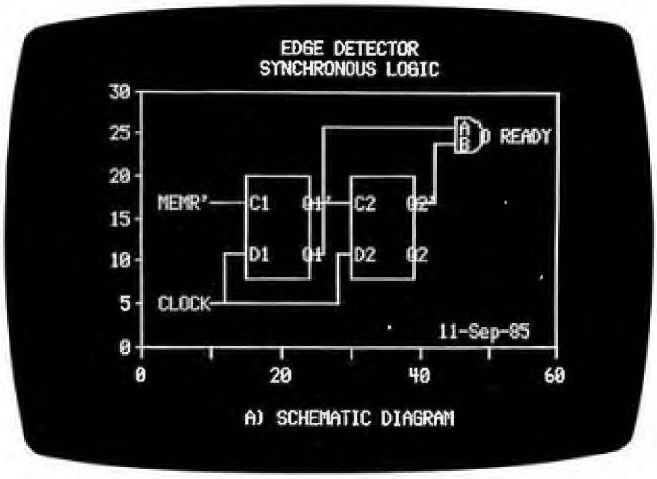
CIRCUIT DESIGN WITH LOTUS 1-2-3

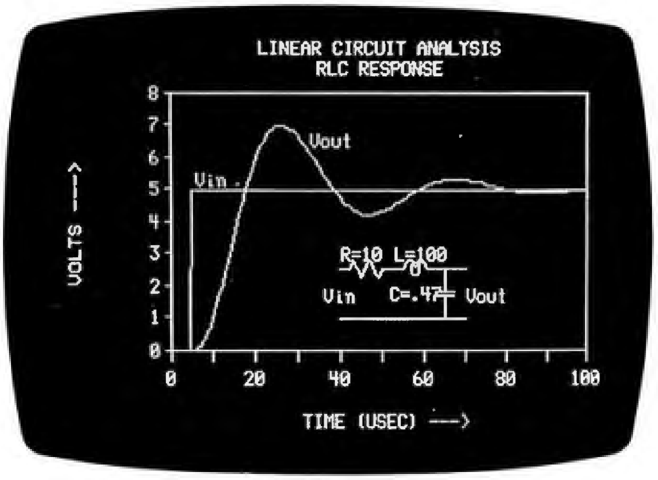
Use the famous spreadsheet to design circuits and print out schematics

BY JOHN L. HAYNES

SPREADSHEETS, especially those with graphics, are not just for business applications; they can be of great help to circuit designers or anyone else designing systems that can be described by equations.

As an example, let's take a look at the application of one spreadsheet, Lotus 1-2-3, to one technical problem, electronic circuit design and analysis. We'll look at both digital and linear circuits.





DIGITAL CIRCUITS

Digital circuits are built from logic building blocks—inverters, NAND gates, flip-flops, etc. We can simulate each of these components with the equations in a cell of a spreadsheet, using the spreadsheet's built-in logical operators shown in figure 1. For instance, in the spreadsheet portion of Lotus 1-2-3, the equivalent of an inverter is the operator #NOT#, structured as #NOT#(*A* = 1). This structure means the state of the operator #NOT# is not true, or equal to a logical 0, if the state in the parentheses is true. This is equivalent to the output of an inverter circuit whose input is *A*. Similarly, the model of a NAND gate, #NOT# (*A* = 1#AND#*B* = 1), is not true if input *A* and input *B* are both true.

The flip-flop is a bit more complex, since its output depends not only on its input conditions but on the transition of a clock pulse. For simplicity, let's assume that there is a narrow clock pulse that triggers the flip-flop whenever the clock pulse is true—in other words, whenever its logic state is a logical 1. The *Q* output remains in its present state until the clock is true; it then assumes the state of the input *D*. The *Q'* output is the logical opposite of *Q*.

These actions are easily simulated using the logical @IF function. It is structured as @IF(*A*,*B*,*C*) and means IF *A* THEN *B* ELSE *C*. That is, if the logical condition of *A* is true, then the function equals *B*. Otherwise, the function equals *C*. Setting the variables as @IF(*C* = 1, *D*,*Q*), we can interpret the state of the function as: If the clock *C* is true, the state is equal to *D*; otherwise, it remains *Q*. The *Q'* output is handled with the #NOT# operator.

Given the ability to simulate logic components with spreadsheet functions and operators, let's now look at how we can use this technique to build a simple digital circuit. The synchronizing circuit of figure 2 is a commonly encountered arrangement. Known variously as an edge detector, a synchronizing circuit, and a digital differentiator, it develops a pulse one clock period long when an external.

(continued)

John L. Haynes is associate director of the Becton Dickinson Research Center, Research Triangle Park, NC 27514. He has been "active in the R&D of computers since the vacuum-tube days (64 bytes of RAM!)." He has spent the last 18 years in health-care product development. His interests are photography, running, computers, and his new grandchild.

Table 1: *These are the named ranges in the worksheet of figure 3. Naming the ranges for the variables makes the equations in range* K6..K11 *easier to follow and understand.*

Named Range	Cell
CLOCK	K6
MEMR'	J7
QQ1	J8
QQ1'	J9
QQ2	J10
READY	J11

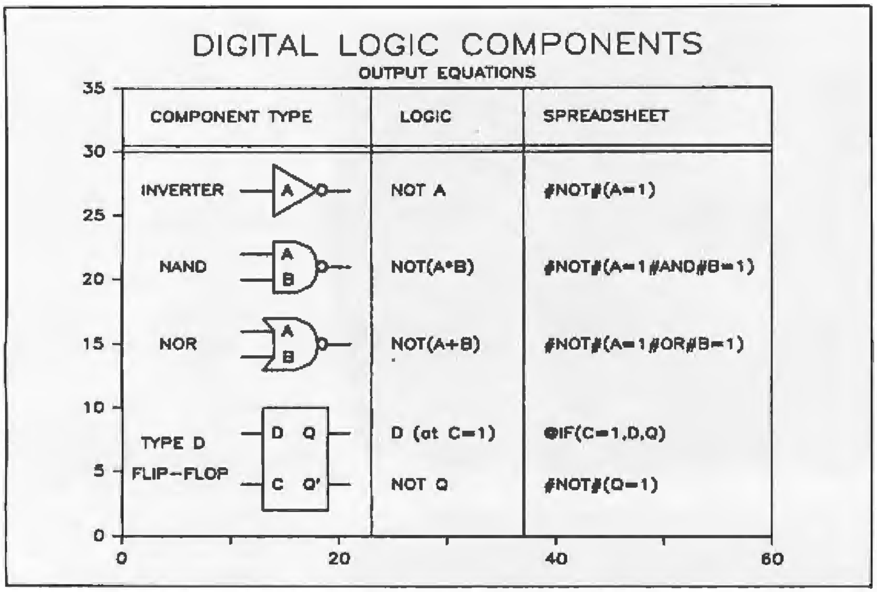


Figure 1: *Logic components simulated by* Lotus 1-2-3.

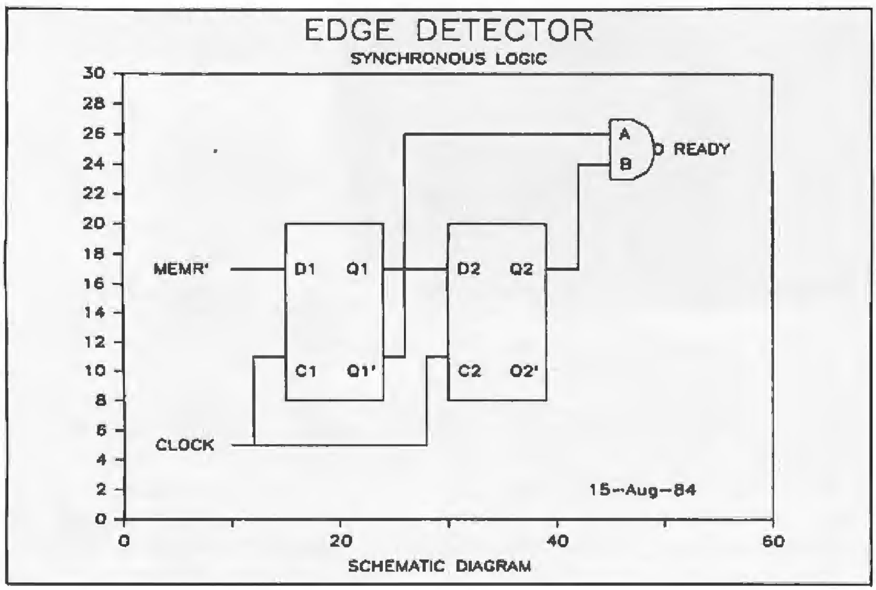


Figure 2: *Edge-detector schematic drawn by* Lotus 1-2-3.

asynchronous event occurs. In this case, it is configured to give a negative pulse, synchronized with the clock, whenever the MEMR' input goes negative. (In a microprocessor circuit, MEMR' might be a Memory Read signal.)

The edge-detector circuit (see the text box "Schematics on Lotus 1-2-3" on page 146 for an explanation of how the circuit was drawn) uses two D flip-flops and a NAND gate. Starting at the left side, we need to describe the input signals for MEMR' and CLOCK, use them as inputs for the first flip-flop, and then take the flip-flop's output to the second flip-flop until we reach the output of the NAND gate in the upper right. The spreadsheet to accomplish this is shown in figure 3. It is laid out in two major sections, Variables and Plot Values.

The equations for each of the outputs of the logic components of figure 2 are displayed in text format in column K of the Variables section. The cell at K8 gives the output equation for Q1. The formula used is that of the D-type flip-flop component described in figure 1, modified to show that the D1 input value is MEMR'. This has the effect of specifying the circuit connection from MEMR' to D1.

For clarity, the variables in the formulas in column K refer to signal names, that is, cell names, rather than cell coordinates. The names used in the formulas are shown in table 1 (you input them using the commands /RANGE, NAME, CREATE). Note that all the named ranges are in column J, with the exception of the CLOCK signal, which is in column K. This makes the state of the outputs in column K dependent on the values in the preceding column, J. Column J holds the initial values for each independent state. The flip-flop names used in the formulas are QQx rather than Qx to avoid confusion with column Q cells.

The formulas in cell range K6..K11 have been copied into L6..BM6. This creates a series of logic states that can be interpreted as the time sequence of states for each variable.

The two input signals are not

created by the logic components of our circuit but must be specified by us. The initial value of MEMR', in cell J7, is set to 1. Each succeeding cell in row 7 is a 1 until cell AA7, which has been set to 0. That value continues for the rest of the row.

The CLOCK signal is a little trickier. Its formula, in K6, is set to repeat the value of the cell A6, which is 10 columns back. Repeated along the whole row, this makes each clock state dependent on the state 10 cells back. This gives a series of zeros in cells L6 to S6, as each cell looks back to a zero-containing cell 10 columns back. At cell T6, however, things change; it looks back at the initial value in cell J6, which is set to @TRUE, so T6 is a 1. This sequence repeats each 10 columns. The next positive clock pulse is in AD6.

Looking now at the logic states of the logic components, as represented by rows 8 to 11, we see that the outputs remain the same at each column until the clock signal is positive, as in column AD. Then the outputs of flip-flop Q1 change state as MEMR' is strobed into it. (No change happened at the clock signal in column T because Q1 was already in the same state as MEMR'.) While very useful, the sequence of 1s and 0s is not nearly as satisfying as a graph.

To plot the results of the logic states, we need to stack them one above the other for clarity, as in figure 4. This is the function of the Plot Values section. The logic state of the CLOCK in cell L6 is added to the offset in cell A17 and placed in cell L17. The formula in L17, +L6+$A17, is copied to the entire region from L17..BM17. As a result, each logic state is added to its appropriate offset and placed in the Plot Values region. When the formula is copied into L18, the copy logic of 1-2-3 will make the cell formulas relative, so the formula in L18 becomes +L7+$A18, and the MEMR' logic state is added to its associated offset value of 9. The $A notation means that the column reference is absolute rather than relative; this ensures that when the formula is copied into succeeding columns it continues to use the offset values of column A. The six output states are now ready to be plotted as graphs A to F. The graph type chosen is Line, formatted as Line Only.

To plot the results, we must assign regions for each graph. Let's start by concentrating on the first line of the plot, the CLOCK signal, plotted as graph A. The region assigned to graph A is the entire row 17, A17..BM17, a total of 66 columns. The Line mode of 1-2-3 plots that as follows: The horizontal axis is divided into 66 equal segments, and the value of each cell is plotted at the appropriate vertical location. In Line Only format, only points in adjacent columns are plotted. Isolated points are not plotted.

These features determine the look of the graph of the CLOCK signal. There are no adjacent nonblank cells until column L, so the graph doesn't start to plot until the twelfth horizontal tick mark. That leaves a blank beginning space for the label.

The label is inserted using the Data-Label facility of 1-2-3. Selecting graph A, we assign the range as cell A6, aligned right. This will print the label "CLOCK" in cell A6, to the right of the value in cell A17.

Each of the remaining five graphs is plotted and labeled the same way. The names of A6..A11 also serve as the labels for each graph; the offsets of A17..A22 serve to locate the labels. The columns B to I serve two functions: They provide the space at the start of the graph for the label and allow the CLOCK formula in K6 to look 10 columns back to set up the clock.

(continued)

```
     A BCDEFGHI   J              K                      L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z AA AB AC AD AE AF AG AH AI
 1
 2    -------------- VARIABLES ---------------
 3                INITIAL
 4  NAME          VALUES   FORMULAS                     LOGIC STATE VS TIME ---->
 5  ------        -------- ----------------------       --------------------------------------------------------------------------
 6  CLOCK         @TRUE    +A6                           0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
 7  MEMR'         @TRUE    +MEMR'                        1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0
 8  Q1            @TRUE    @IF(CLOCK=1,MEMR',QQ1)        1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0
 9  Q1'                    #NOT#(QQ1=1)                  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  1  1  1  1
10  Q2            @TRUE    @IF(CLOCK=1,QQ1,QQ2)          1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
11  READY                  #NOT#(QQ1'=1#AND#QQ2=1)       1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0
12
13    -------------- PLOT VALUES ---------------
14
15  OFFSET        GRAPH:                                PLOT VALUE VS TIME ---->
16  -------       ------                                --------------------------------------------------------------------------
17  11              A:                                  11 11 11 11 11 11 11 11 12 11 11 11 11 11 11 11 11 11 12 11 11 11 11 11
18   9              B:                                  10 10 10 10 10 10 10 10 10 10 10 10 10 10 10  9  9  9  9  9  9  9  9  9
19   7              C:                                   8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  7  7  7  7  7  7
20   5              D:                                   5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  5  6  6  6  6  6
21   3              E:                                   4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4  4
22   1              F:                                   2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  2  1  1  1  1

     A BCDEFGHI   J              K                      L  M  N  O  P  Q  R  S  T  U  V  W  X  Y  Z AA AB AC AD AE AF AG AH AI
```

Figure 3: *Spreadsheet to analyze edge-detector circuit.*

SCHEMATICS ON LOTUS 1-2-3

A very crisp, professional-looking schematic can be drawn by using the graphics mode of a spreadsheet. The results in figure 2 speak for themselves. Chances are you've never considered using Lotus 1-2-3 to draw a schematic. So how is it done? Just a simple *XY* graph, drawing lines from point to point.

What's the secret? Lotus draws no line if a point is missing in a range; this allows us to put blanks between groups of data to draw numerous unconnected components. We can then interconnect them at will, using additional groups of data to draw the connections.

The spreadsheet in figure 10 contains the data to draw figure 2. It's organized in two groups; the first, A1..I38, handles the components and their labeling. It is shown in the left column of the figure. The group on the right, A40..I75, handles the interconnections and completes the labeling.

STEP 1: DRAW THE COMPONENTS

The seven pairs of numbers in the range B6..D12 define the first flip-flop, FF1. The pair (15,8) is the lower left-hand corner of FF1. The next five pairs of numbers define the four corners and mark the the two input ports. The last pair (15,8) closes the box back at the starting point. If components are used more than once, you may want to generate a component primitive table as in figure 11. The shapes can then be copied to the desired location by adding the appropriate offset to the *X* and *Y* values.

The next group, in range B14..D20, defines FF2 in the same way. The last group, in range B22..D37, outlines the NAND gate. The curved outline requires far more points than the rectangle of the flip-flops. The isolated point at (48,25) will provide the label to put an O on the NAND output to provide the little circle that signals that it is an inverting output. Using the O saves drawing another 8 to 10 points to draw a circle.

Defining the graph type as *XY*, set the *X* graph range to B6..B37. Set the *A* range to D6..D37. Set the scale manually to get a 30 by 60 plot.

STEP 2: ADD THE INTERCONNECTING LINES

The data in the right-hand group B45..D68 defines the connections. Each connection is labeled for function in the Comments column. To plot the interconnects, extend the *X* and *A* ranges to include their data; the *X* range is now B6..B68, and the *A* range is D6..D68.

STEP 3: ADD THE LABELS AND THE DATESTAMP

Extend the *X* and *A* ranges down to include rows 69 to 74 and pick up the date stamp and input signal label points. Then use the Data-Label option to define a label range for graph range *A*. This is the set of labels in column E. So the label range is set to E6..E74. Select location Right from the menu when defining the data labels to put the labels to the right of the selected points.

The label will plot about one character to the right of the plotted point in graph range *A*. Notice how nicely this puts the little circle on the NAND gate. The date in cell E70 is a @TODAY function, so it keeps the data sheet and schematic up to date automatically as changes are made and saved.

Use the Titles option to set up your titles at the top and bottom, and the graph of figure 2 results.

All the figures in this article were done using Lotus 1-2-3.

The resulting graph of figure 4 is very easy to follow. It's now clear how the circuit works to put out a synchronized pulse when MEMR' goes negative. The next CLOCK signal (the second pulse of our figure) strobes the logical zero state into Q1, driving Q1' positive. Since Q2 is unaffected at this clock time, its output is still a logical one. With both inputs of the NAND gate now positive, its output goes negative, starting the READY signal pulse. The next CLOCK pulse strobes the logical zero output of Q1 into flip-flop Q2, sending its output to zero. This pulls the *B* input of the NAND to ground, sending its output positive and terminating the READY signal. It's clear that the READY signal will be exactly one clock period long, synchronized with the CLOCK signal.

The circuit of figure 2 is a good example of the power of a spreadsheet program to diagram a digital circuit, analyze it, and plot the results. If the results aren't what you want, it's easy to change the design. For example, what if the input for the NAND gate is taken from the opposite outputs of the flip-flops?

LINEAR CIRCUITS

The analysis of transient effects and frequency response in linear circuits can also be modeled very effectively using spreadsheets. The added graphics of Lotus 1-2-3 can show the resulting waveforms at the press of a key, allowing an interactive modification of the circuit to get the desired response.

No differential equations need be solved, no integrals evaluated. All that is necessary is to model the incremental changes that occur in a brief time period.

The three important linear circuit elements—resistors, capacitors, and inductors—are modeled in figure 5. As in the digital case, we can set up equations that describe the behavior of the elements; those equations can then be modeled in the equations of a spreadsheet cell.

The simplest linear element is the resistor, whose behavior is described by Ohm's law: The ratio of voltage to

(continued)

Technical Bulletin
No. 2 in a series.

SUBJECT: Engineering a LAN for Maximum Flexibility.

Quantum Software Systems Ltd. proudly announces QNX 2.0 — the Ultimate Distributed Network Operating System. QNX 2.0 is now available for the IBM-PC, IBM-AT, PC compatibles, DEC Rainbow and TANDY 2000. If you have been waiting for a Real-time Multi-tasking Multi-user Operating system with fourth generation LAN support, then QNX 2.0 can offer you today what the competition can't even begin to promise for the future.

QNX 2.0 integrates the Local Area Network architecture right into the heart of the operating system, at the fundamental level of intertask communication allowing tasks to communicate transparently with other tasks across the whole network. This means that any task (program/application) may access ANY serial port, ANY printer or ANY disk on the network. There are no artificial restricitons. Every PC with a disk is a potential file server, PCs without disks will automatically BOOT over the network.

QNX on the IBM-PC AT:

QNX is the first Multi-tasking Multi-user Operating system available for the AT. It is available in both networked and single machine configurations. At about 2.5 times faster than the QNX 8088 PC based systems, and 10 times faster than other multi-tasking operating systems on the same processor. QNX is the ideal program development environment.

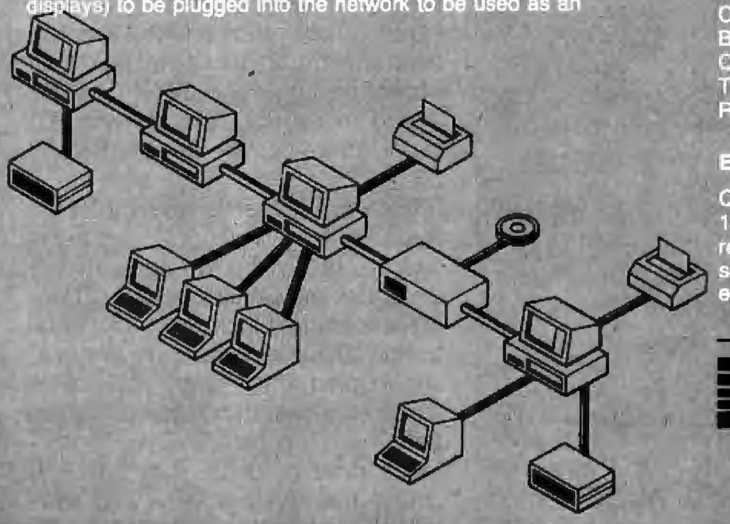
O/S	Computer	Processor	Measured time
QNX ™	IBM-PC AT	80286	480 usec
XENIX ™	Intel-286	80286	4,930 usec

File Security:

Designed with extensive file security features, QNX 2.0 provides login protection with network wide file permission checking based on 255 groups of 255 users. In addition, each PC user may control network access to devices attached locally to their machine.

Distributed Processing:

The QNX LAN supports distributed processing as well as distributed devices. Tasks may be executed on remote stations as easily as they may be executed on the local work station. This allows pure processing elements (PCs without keyboards or displays) to be plugged into the network to be used as an un-committed processing resource. This is ideal for real-time, process control, data acquisition and data communication applications.



Global Communications:

QNX supports a full implementation of X.25 allowing connection to public networks such as Telenet and Datapac. This allows you to link geographically separate LANs together providing true global area networking.

Cost Effective Growth and Flexible Solutions:

QNX is affordable, and will work with the PCs you use today and those you will use tomorrow. You may mix and match different brand PCs on the same QNX network with absolute ease. Multi-user expansion may be accomplished by adding terminals to PCs or PCs to the network. You can start your multi-user application on a single PC with 1 to 10 attached terminals. Once your single processor starts to show signs of degradation, add another PC and connect terminals to the new processor. If the disk becomes the major bottleneck, you may add hard disks to other attached PCs to distribute the processing. Applications which are very CPU intensive may wish to limit a single user to each processor and expand the system with low cost diskless PCs used as work stations. QNX does offer a truly cost effective and flexible solution to your applications needs.

Portability:

QNX 2.0 is portable. The operating system is independant of the physical local area network. It is available in a form suitable for porting to other 8088/8086/80186/80286 computers in the consumer, educational and industrial market place. QNX is ROMable and can operate in as little as 128Kb RAM.

DOS Compatibility:

PC-DOS version 2.1 can run as a task under the QNX 1.2 or 2.0 operating systems. QNX will also allow transparent access to the DOS file system partition and floppies.

QNX Products:

QNX Operating System	PC-DOS Emulator
Full Screen Multi-terminal Editor	Electronic Mail
Extended Utilities	Electronic Teleconferencing
C Compiler & 8086 Assembler	Full Screen Menu Developer
Basic Compiler	Isam File Utility
Qbol (dibol) Compiler	Networking Board
Text Processor	OEM Customization Kit (to port QNX)
Real Time Spelling Checker	

Established:

Quantum sold over 10,000 copies of its operating system during 1984, into all business systems environments, to developers of real time applications, government and educational systems, to software developers/integrators, universities and research establishments.

QUANTUM SOFTWARE SYSTEMS LTD.

Moodie Drive, HiTech Park, 215 Stafford Rd.
Ottawa, Canada K2H 9C1 (613) 726-1893

current is the resistance.

The behavior of a capacitor is a little more complicated. The nature of a capacitor is to resist changes in voltage. The equations of figure 5 show that the current *I* flows into a capacitor *C*, for a period of time *DT*, to build up the charge necessary to change the capacitor voltage *VC* by an amount *DV*.

The larger the capacitance, the smaller the voltage changes for a given current change. The new voltage is the sum of the previous voltage, *VC*, plus the incremental change caused by the previous current flow, *I*:

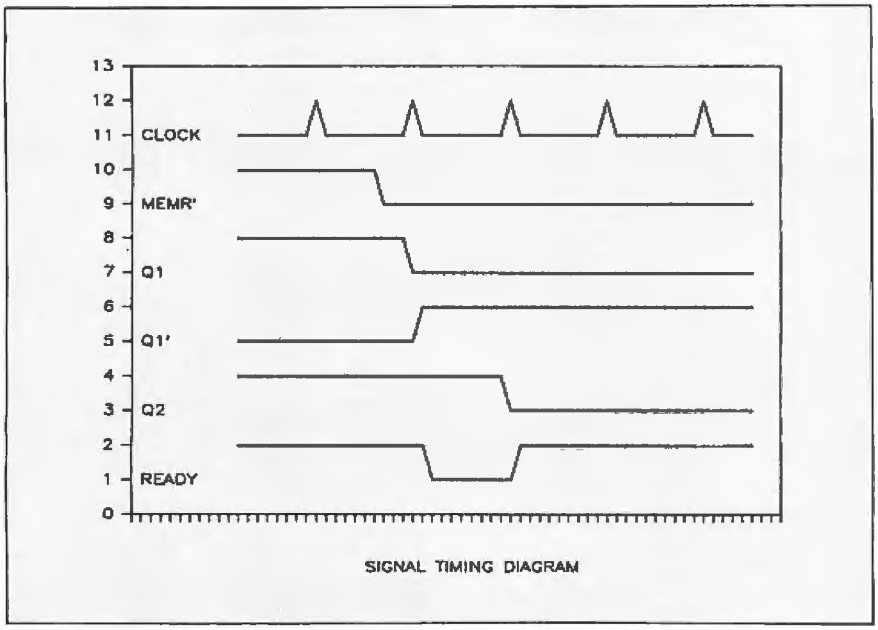


Figure 4: *Timing diagram for edge-detector circuit.*

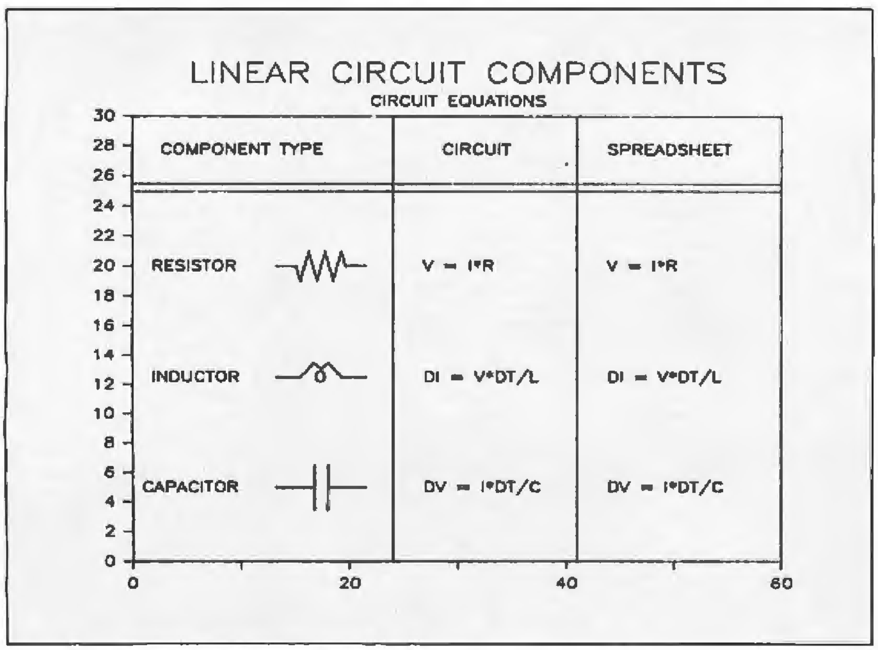


Figure 5: *Linear circuit components simulated by Lotus 1-2-3.*

$$VC = VC + DV$$
$$VC = VC + I*DT/C$$

The behavior of the inductor is the inverse of the capacitor behavior; that is, the inductor resists changes in current when a voltage is applied across its terminals. The voltage *VL* must be applied for a time *DT* to build up flux in the inductor and allow a change *DI* in current flow. The larger the inductance, the smaller the change in current for a given applied voltage. The new current is the sum of the previous current *I* plus the incremental change caused by the previous voltage *VL*:

$$I = I + DI$$
$$I = I + VL*DT/L$$

These circuit elements can now be combined to form a number of circuits that we can then analyze with our spreadsheet modeling technique.

For example, let's look at the simple circuit of figure 6. The capacitor *C* is charged through the resistor *R*. What is the resulting response to a step input V_{in}? Those with circuit experience will recognize that the capacitor does not change voltage instantaneously; it charges up at a rate related to the amount of current flowing into it through the resistor. This is the behavior shown by V_{out}, the output voltage taken across the capacitor.

The spreadsheet that generates figure 6 is shown in figure 7; it is similar to the sheet that generated figure 4. The section added at the top describes the circuit operating conditions. The input is a 5-volt (V) step, stored in cell E2. The resistor and capacitor values are in cells E4 and E5.

The Variables section assembles the formulas for the circuit components into a circuit; the resulting matrix describes the circuit operation. Cells in the formulas block, K13..K18, define the circuit. Once defined, the formulas are copied into range L13..DF18, which lays out those values versus time.

The time clock is in cell K13. It copies the previous value and adds to it the value of one clock tick, *DT*.

(continued)

Tired of waiting for your PC?

Special Offer Only

$49.95*

This SPECIAL PRICE OFFER absolutely ends on October 31, 1985.

End the wait!... DoubleDOS

Multi-tasking, concurrent processing using today's software and your PC, XT, or AT system!

No matter how good your IBM PC is, you still have to wait while it processes and calculates for a long report, spread sheet, payroll, or whatever. This waiting is costing you time and money.

Now, DoubleDOS lets your PC actually run programs concurrently!

Think of the ways DoubleDOS will save you time and money:

- BUSINESS. Run a spreadsheet, and at the same time, print the payroll checks or accounting reports.
- WORD PROCESSING. Work on another document while printing or spell checking previous work.
- PROGRAMMING. Compile programs while starting new ones immediately.
- COMMUNICATIONS. Receive or send information while you use the PC to do other tasks. DoubleDOS has endless applications!

Order Toll-Free:
1-800-272-9900

Listen to what users are saying about DoubleDOS:

- "It let's me get more done. Lotus and dBase run together like they were made that way and it only takes 30K in RAM!"
- "Great product! It is solid, painless, and bulletproof!"
- "Simply the best program I've ever had."
- "I'm thoroughly impressed. Fantastic . . . great value-to-cost, too!"
- "Excellent product at an excellent price (hard to find today)."
- "Great stuff! I use it to do compiles and links while editing. Saves a *lot* of time!"
- "Your programming staff deserves a standing ovation for their efforts and results!"
- "I can't believe how fast it works, rarely notice slowdown when using 2 programs."

*Plus $5.00 Shipping & Handling.

SOFTLOGIC SOLUTIONS Satisfaction Guaranteed!

Creators of the Disk Optimizer

530 Chestnut St., Manchester, NH 03101
1-800-272-9900. In NH, call 627-9900.

DoubleDOS works with PC-DOS and your existing software. Also supports many compatibles using MS-DOS.

DT is set at 1 microsecond (μs); we'll come back and review this choice later. The input voltage, V_{in}, is set to quasi-*DC*; it repeats whatever value it had previously. The exception is cell O14, which is set to the value in cell E2 in the Circuit Parameters section. This gives us the desired 5-V step input at $t=5$ μs.

Cell K15 contains the most complex of the formulas. It sets the value of the capacitor voltage *VC*. As described in the circuit element equations of figure 5, the value is the previous value J15, plus the incremental voltage change from the previous *I*, flowing in *C* for the interval *DT*.

Next we calculate the voltage available to the resistor. The sum of the voltage across the resistor and capacitor must equal the input voltage. Since the input and capacitor voltages are known, the resistor voltage can be calculated as their difference, $V_{in} - VC$. This is the formula in K16.

The current *I* is our easiest calculation—just divide the resistor voltage *VR* by the resistance *R*. Notice in column O that the initial effect of the step-input voltage is seen entirely across the resistor; a 0.5-ampere current flows as a result.

For clarity, the output voltage V_{out} is given its own row, 18. This is identical to the capacitor voltage, *VC*. Note how the value of V_{out} builds up as time progresses from column P through column X. Figure 6 plots the input voltage V_{in} from row 14 and the output voltage V_{out} from row 18.

PLOTTING THE GRAPH

To get the desired smooth curve, the format for all curves is set to Line Only. To get the maximum flexibility, the graph type has been set to XY. This will be necessary when we get to the schematic diagram and is a help in labeling the curves.

The column labeled Graph (E12.. E20) shows what we've plotted to generate the curves. This takes the place of the Plot Values section of the digital sheets. It lets us know that

(continued)

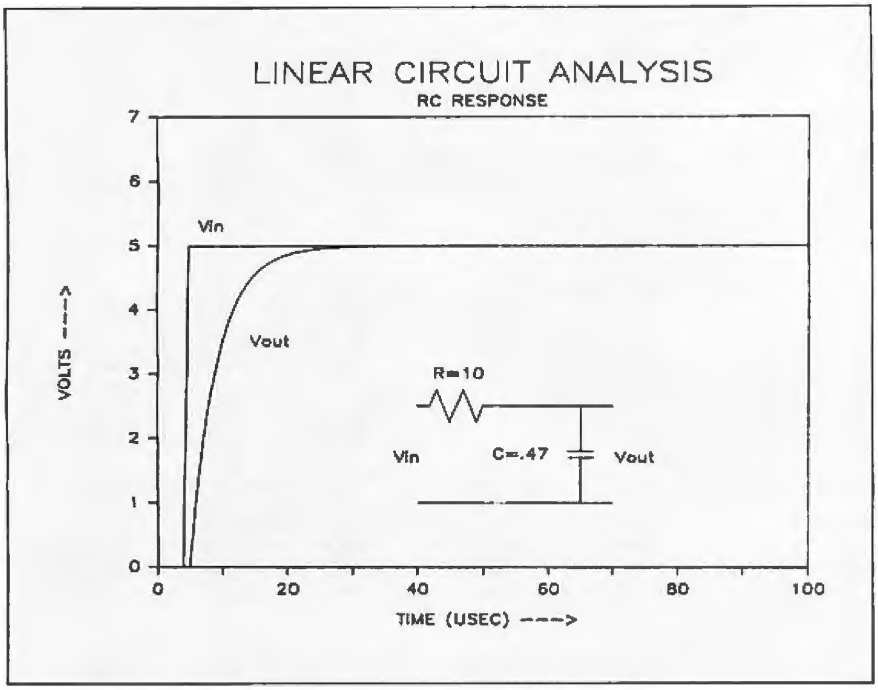
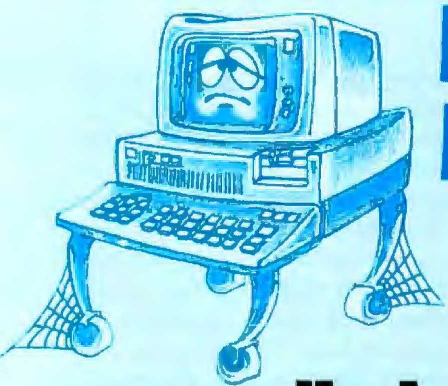


Figure 6: *Schematic and voltage-versus-time graph of* RC *circuit.*

```
   A BCD  E   FGHI    J              K           L     M    N    O    P    Q    R    S    T    U    V    W    X    Y    Z
1   -------------- CIRCUIT PARAMETERS ----------------                               ------- NAMED RANGES ---------
2 VIN =   5.0  volt step at t=5 microsec
3                                                                                  R  E4..E4                  X: J13..EF13
4 R   =  10.0  ohms                                                                C  E5..E5
5 C   =  0.47  microfarads                                                         DT  E6..E6                 A: J14..DF14
6 DT  =   1.0  microsec    DT MUST BE LESS THAN R*C =    4.7 microsec            VIN K14..K14                 B: J18..DF18
7                                                                                 VC K15..K15                 C: J20..EF20
8                                                                                 VR K16..K16
9   -------------- VARIABLES ----------------                                     I J17..J17        C:DATA-LABEL   J21..EF21
10                  INITIAL
11 NAME  GRAPH      VALUES   FORMULAS           VALUE VS TIME ---->
12 ----- -----      -------- ----------------   ----------------------------------------------------------------------------
13 TIME    X:       ZERO     +J13+$DT              2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
14 VIN     A:       ZERO     +J14                0.0  0.0  0.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0
15 VC               ZERO     +J15+I*$DT/$C       0.0  0.0  0.0  0.0  1.1  1.9  2.6  3.1  3.5  3.8  4.1  4.3  4.4  4.5  4.6
16 VR               ZERO     +VIN-VC             0.0  0.0  0.0  5.0  3.9  3.1  2.4  1.9  1.5  1.2  0.9  0.7  0.6  0.5  0.4
17 I                ZERO     +VR/$R              0.0  0.0  0.0  0.5  0.4  0.3  0.2  0.2  0.2  0.1  0.1  0.1  0.1   .0   .0
18 VOUT    B:       ZERO     +VC                 0.0  0.0  0.0  0.0  1.1  1.9  2.6  3.1  3.5  3.8  4.1  4.3  4.4  4.5  4.6
19
20 LABEL   C:                                                  5.3                                3.5
21 DATA-LABEL                                                  Vin                                Vout
22
23
   A BCD  E   FGHI    J              K           L     M    N    O    P    Q    R    S    T    U    V    W    X    Y    Z
```

Figure 7: *Spreadsheet to analyze* RC *circuit.*

SHIFT YOUR HARD DISK BACK INTO HIGH GEAR



with

disk optimizer

ONLY **$49.95***

Special Introductory Price

Tunes Your Disk Files For Highest Performance

The Disk Optimizer organizes your disk to make your PC run faster. You'll discover faster program loading, faster file loading and sorting, faster file creation, faster backups to diskette. And the disk Optimizer reduces the chances of error during your disk I/O.

A	CC	AA	DDD	D	E	BB	F	FF	GG	AAA	H	BB	JJ	JJJ	AAA	BBBB
A	CC	AA	DDD	D	E	BB	F	FF	GG	AAA	H	BB	JJ	JJJ	AAA	BBBB

AAAAAAAAA	BBBBBBBB	CC	DDDD	E	FFF	GG	H	JJJJJ
AAAAAAAAA	BBBBBBBB	CC	DDDD	E	FFF	GG	H	JJJJJ

DOS IS SLOWING YOU DOWN!

Whether you realize it or not, the more you use your hard disk, the slower it is likely to react. This is because the operating system (PC-DOS/MS-DOS) stores things (programs, overlays, batches, etc.) randomly, wherever unused disk space is available. Your operating system doesn't keep pieces of the same file together. This means slower access time. Lost speed. Lost productivity.

Order Toll-Free:
1-800-272-9900

TURN THE THROTTLE BACK UP!

Now, the Disk Optimizer puts the zip back into your hard disk! The Disk Optimizer collects all the scattered pieces of your files and re-stores them together in neat, well-organized areas on your disk. The result: immediate performance improvement . . . faster loading programs (and their overlays) . . . faster sorting . . . faster loading and storing spreadsheets or document files.

BONUS EXTRAS — at no extra cost

- **DISK ANALYZER**
  Displays a visual performance analysis . . . shown in percentages . . . for any disk or diskette.
- **PERSONAL FILE SECURITY**
  Password protect any file or program.
- **FILE PEEKER**
  Now you can visually examine any file on your system.

$49.95*

*Plus $5.00 Shipping & Handling

SOFTLOGIC SOLUTIONS
Creators of DoubleDOS
530 Chestnut Street, Manchester, NH 03101
1-800-272-9900. In NH, call 627-9900
Disk Optimizer works with IBM PC's and true compatibles.

curve X is taken from row 12, curve A from row 13, etc. The actual ranges plotted are shown in the range name table included on the sheet in the upper right-hand corner.

Lotus 1-2-3 does not have a command to display named ranges. The Named Ranges table is a good idea to include in all your spreadsheets as part of the documentation, to show which ranges have names. The formulas in column K have the names of a number of circuit parameters mixed in with the cell locations; this makes the formulas more self-explanatory.

Since there is quite a bit to label in figure 6, two rows have been added for labeling, rows 20 and 21. The C curve allows us to spot unplotted points and to plot line segments on the figure. The value in cell O20, for example, spots an unplotted point at $x=5$ and $y=5.3$. The Data-Label below it prints the label "Vin" on the figure to the right of this point. Remember that the point doesn't plot, because the Line format does not plot isolated points—it only plots cells that adjoin nonblank cells. The other labels are treated similarly. The data for the schematic diagram is in the region DH13..EA21. It's plotted in the way described in the text box.

How do we pick a reasonable value for the time interval *DT*? For our difference analysis to be accurate, the incremental change in the capacitor voltage must be small compared to the voltage *VR*, which determines the current *I*. Let's write that as an inequality:

$$I \cdot DT/C << VR$$

Rearranging the inequality to solve for *DT*:

$$DT << (VR/I) \cdot C$$

But from Ohm's law, we know that *VR/I* equals the resistance *R*, so:

$$DT << R \cdot C$$

This tells us that the time interval *DT* must be small compared to $R \cdot C$. The *RC* product is a term familiar to cir-

(continued)

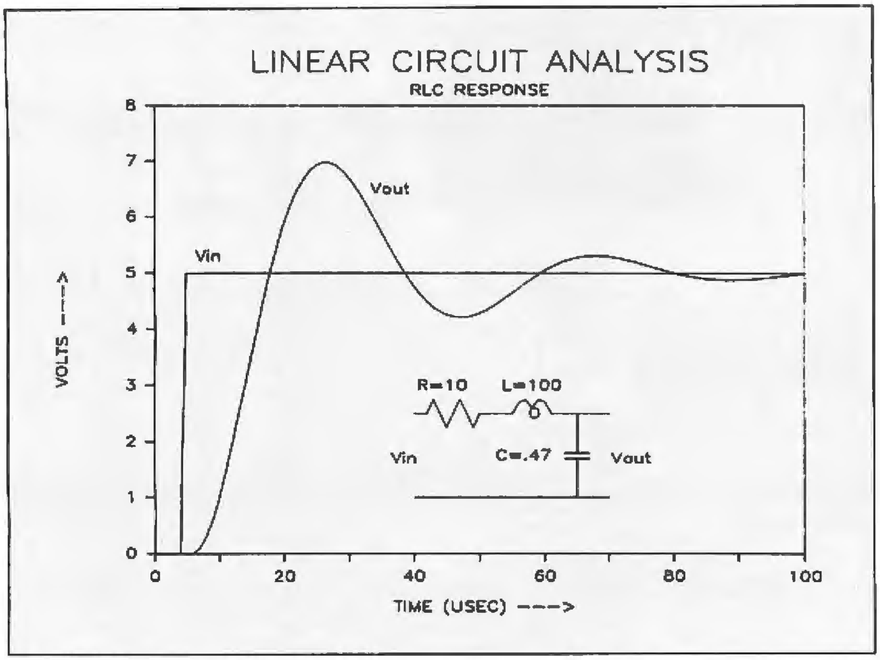


Figure 8: *Schematic and voltage-versus-time graph of* RLC *circuit.*

```
    A BCD  E    FGHI   J              K            L     M    N    O    P    Q    R    S    T    U    V    W    X    Y    Z
1    -------------- CIRCUIT PARAMETERS ---------------                                    ------- NAMED RANGES ---------
2 VIN =      5  volt step at t=5 microsec                                                  L  E3..E3                  X: J13..EF13
3 L   =    100  microhenries                                                               R  E4..E4
4 R   =     10  ohms                                                                       C  E5..E5                  A: J14..DF14
5 C   =   0.47  microfarads                                                               DT  E6..E6                  B: J19..DF19
6 DT  =      1  microsec    DT MUST BE << SQRT(L*C) =    6.85 microsec                   VIN K14..K14                  C: J21..EF21
7                                                                                         VC K15..K15
8                                                                                         VR K16..K16      C:DATA-LABEL   J22..EE22
9    ------------ - VARIABLES ---------------                                             VL J17..J17
10                    INITIAL                                                              I J18..J18
11 NAME   GRAPH       VALUES   FORMULAS           VALUE VS TIME ---->
12 ----- -----        -------- ---------------    -------------------------------------------------------------------------
13 TIME     X:        ZERO     +J13+$DT              2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
14 VIN      A:        ZERO     +J14                0.0  0.0  0.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0  5.0
15 VC                 ZERO     +J15+I*$DT/$C       0.0  0.0  0.0  0.0  0.0  0.1  0.3  0.6  1.0  1.5  2.0  2.5  3.0  3.6  4.1
16 VR                 ZERO     +I*$R               0.0  0.0  0.0  0.0  0.0  0.5  1.0  1.4  1.8  2.1  2.3  2.5  2.5  2.5  2.5
17 VL                 ZERO     +VIN-VC-VR          0.0  0.0  0.0  5.0  5.0  4.4  3.7  2.9  2.2  1.4  0.7  0.1 -0.6 -1.1 -1.6
18 I                  ZERO     +I+VL*$DT/$L        0.0  0.0  0.0  0.0  0.1  0.1  0.1  0.2  0.2  0.2  0.2  0.3  0.3  0.2  0.2
19 VOUT     B:        ZERO     +VC                 0.0  0.0  0.0  0.0  0.0  0.1  0.3  0.6  1.0  1.5  2.0  2.5  3.0  3.6  4.1
20
21 LABEL    C:                                                    5.3
22 DATA-LABEL                                                     Vin
23
24
    A BCD  E    FGHI   J              K            L     M    N    O    P    Q    R    S    T    U    V    W    X    Y    Z
```

Figure 9: *Spreadsheet to analyze* RLC *circuit.*

```
     A    B     C  D          E     F  G            H    I
 1  :----------:--------------------:--------------------:  1
 2  : GRAPH X  :      GRAPH A       :     COMMENTS       :  2
 3  :----------:--------------------:--------------------:  3
 4  :     DATA :  DATA      LABELS  :   COMPONENTS       :  4
 5  :==========:====================:====================:  5
 6  :       15 :     8              :NAME      FF1       :  6
 7  :       15 :    11  D1    Q1    :                    :  7
 8  :       15 :    17  C1    Q1'   :LOCATION:           :  8
 9  :       15 :    20              :      XO=        15 :  9
10  :       24 :    20              :      YO=         8 : 10
11  :       24 :     8              :                    : 11
12  :       15 :     8              :                    : 12
13  :----------:--------------------:--------------------: 13
14  :       30 :     8              :NAME      FF2       : 14
15  :       30 :    11  D2    Q2    :                    : 15
16  :       30 :    17  C2    Q2'   :LOCATION:           : 16
17  :       30 :    20              :      XO=        30 : 17
18  :       39 :    20              :      YO=         8 : 18
19  :       39 :     8              :                    : 19
20  :       30 :     8              :                    : 20
21  :----------:--------------------:--------------------: 21
22  :       45 :    23              :NAME      NAND      : 22
23  :       45 :    24        B     :INPUT LABEL         : 23
24  :       45 :    26        A     :INPUT LABEL         : 24
25  :       45 :    27              :                    : 25
26  :       47 :    27              :                    : 26
27  :    48.32 :  26.5              :LOCATION:           : 27
28  :    48.73 :    26              :      XO=        45 : 28
29  :    48.94 :  25.5              :      YO=        23 : 29
30  :       49 :    25              :                    : 30
31  :    48.94 :  24.5              :                    : 31
32  :    48.73 :    24              :                    : 32
33  :    48.32 :  23.5              :                    : 33
34  :       47 :    23              :                    : 34
35  :       45 :    23              :                    : 35
36  :       NA :    NA              :                    : 36
37  :       48 :    25 O READY      :OUTPUT LABEL        : 37
38  :----------:--------------------:--------------------: 38
     A    B     C  D          E     F  G            H    I
```

```
     A    B     C  D          E     F  G            H    I
40  :----------:--------------------:--------------------: 40
41  : GRAPH X  :      GRAPH A       :     COMMENTS       : 41
42  :----------:--------------------:--------------------: 42
43  :  DATA    :     DATA   LABELS  :  CONNECTIONS       : 43
44  :==========:====================:====================: 44
45  :       10 :    17              :   MEMR --> D1      : 45
46  :       15 :    17              :                    : 46
47  :----------:--------------------:--------------------: 47
48  :       12 :     5              :   CLOCK --> C1     : 48
49  :       12 :    11              :                    : 49
50  :       15 :    11              :                    : 50
51  :----------:--------------------:--------------------: 51
52  :       10 :     5              :   CLOCK --> C2     : 52
53  :       28 :     5              :                    : 53
54  :       28 :    11              :                    : 54
55  :       30 :    11              :                    : 55
56  :----------:--------------------:--------------------: 56
57  :       24 :    11              :  Q1' --> NAND A    : 57
58  :       26 :    11              :                    : 58
59  :       26 :    26              :                    : 59
60  :       45 :    26              :                    : 60
61  :----------:--------------------:--------------------: 61
62  :       39 :    17              :   Q2 --> NAND B    : 62
63  :       42 :    17              :                    : 63
64  :       42 :    24              :                    : 64
65  :       45 :    24              :                    : 65
66  :----------:--------------------:--------------------: 66
67  :       24 :    17              :     Q1 --> D2      : 67
68  :       30 :    17              :                    : 68
69  :----------:--------------------:--------------------: 69
70  :       42 :     2 26-Aug-84    :DATE STAMP          : 70
71  :----------:--------------------:--------------------: 71
72  :        2 :    17   MEMR'      :INPUT SIGNAL        : 72
73  :----------:--------------------:--------------------: 73
74  :        2 :     5   CLOCK      :INPUT SIGNAL        : 74
75  :----------:--------------------:--------------------: 75
     A    B     C  D          E     F  G            H    I
```

Figure 10: *Spreadsheet to draw edge-detector circuit.*

```
        C    D          E    F    G          H       I
 97  :------------------------------------------------: 97
 98  :             COMPONENT PRIMITIVES               : 98
 99  :=======================:========================: 99
100  :      RESISTOR         :       CAPACITOR        :100
101  :-----------------------:------------------------:101
102  :  X-AXIS      Y-AXIS   :  X-AXIS      Y-AXIS    :102
103  :  ------      ------   :  ------      ------    :103
104  :       0           0   :       0           2    :104
105  :    0.75           0   :    0.75           2    :105
106  :       1        -0.5   :   -0.75           2    :106
107  :     1.5         0.5   :                        :107
108  :       2        -0.5   :   -0.75         2.5    :108
109  :     2.5         0.5   :    0.75         2.5    :109
110  :       3        -0.5   :       0         2.5    :110
111  :     3.5         0.5   :       0           5    :111
112  :    3.75           0   :                        :112
113  :       6           0   :                        :113
114  :-----------------------:------------------------:114
        C    D          E    F    G          H       I
```

Figure 11: *Digital and linear component primitives.*

cuit designers; it's a measure of the response time of a circuit, historically called the circuit "time constant." So the comment in row 6 tells us that we should pick the time intervals on our spreadsheet analysis so that they are small compared to the circuit response time *RC*—a very reasonable restriction.

The results of the our work are summarized in the graph of figure 6. Sure enough, the output voltage rises to the input value as a familiar exponential curve. So far, so good. But, you say, I could have estimated that result far faster without all that work. True enough. A simple *RC* circuit isn't all that impressive, unless you remember that we did the analysis with a spreadsheet program.

Now all the "what if" power of the spreadsheet is at our disposal. Want to see the effect of a smaller capacitor? A different value resistor? The ef-

(continued)

fect of a 5 percent change in each of the elements? Just key them into the Circuit Parameters cells and push F10 to see the modified graph. Now that's power.

Let's add an inductor *L* in series with our resistor, as shown in figure 8. First, we add the inductor value in the Circuit Parameters section in row 3, then we add a new row 17 for the inductor voltage *VL* in the Variables section, pushing rows 17–23 down to make room. Next, we add an equation that describes the behavior of the inductor to the Formulas section, based on the equations of figure 5. We now modify the formulas for *VR* and *I* to reflect the dominating role of the inductor in setting the circuit current. The result is the spreadsheet shown in figure 9. Finally, we make some small changes to the schematic drawing section to add the inductor to the schematic of figure 8.

As before, the cell equations of the formulas in range K13..K19 are copied to the adjacent 99 columns. Now we're ready to analyze an RLC circuit.

Figure 8 shows the resulting graph. Still easy for our spreadsheet method, but it would have been a far-from-trivial task with any other method. The circuit values are easily modified in the Circuit Parameters section if the results are not satisfactory. Higher values of *R*, for example, will reduce the overshoot if that is undesirably large. The results of any value change are always available at the touch of F10, the LOTUS GRAPH command.

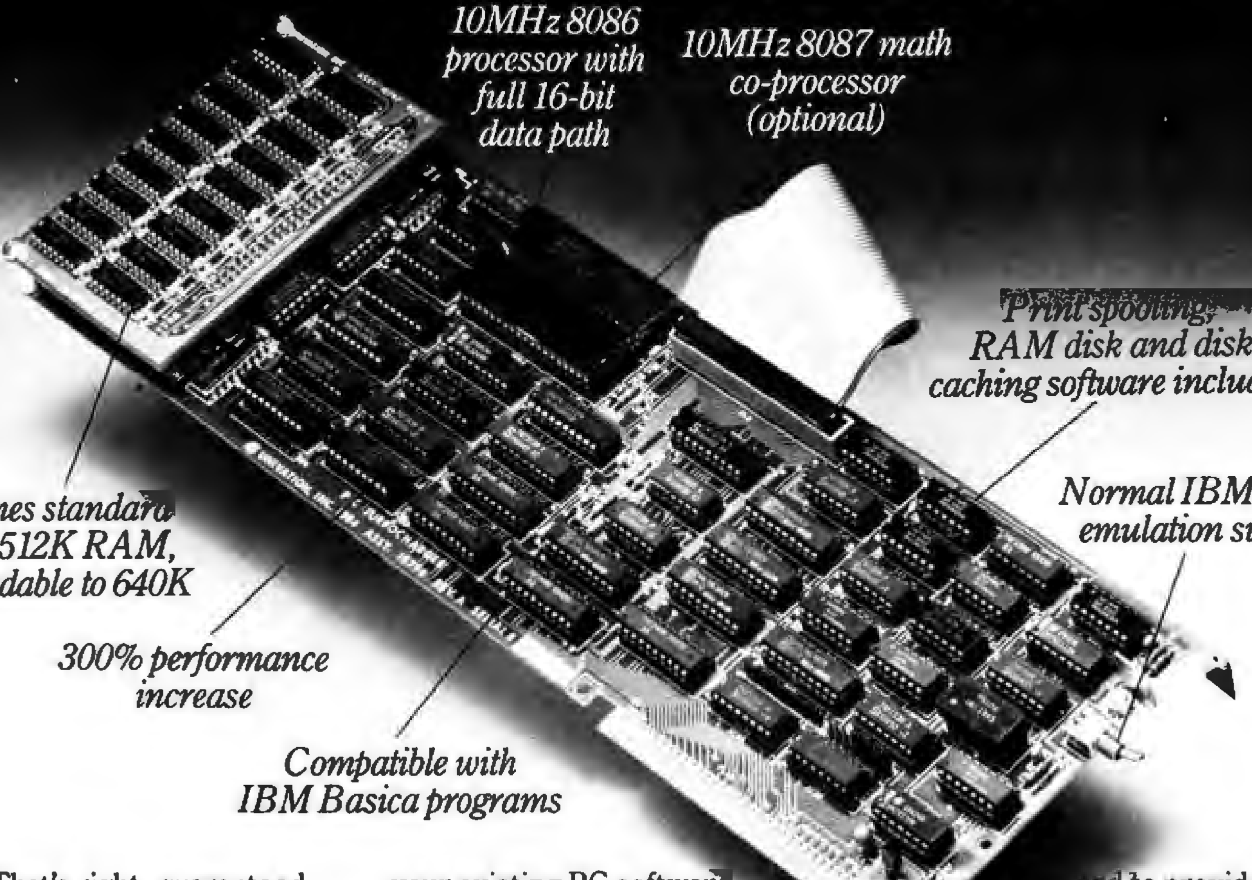
These circuits are only examples of what can be done with linear circuits. With a little more work, some very complex networks can be studied. Nonlinear elements such as diodes could be added; they are a natural for a table lookup. Circuit gain elements such as amplifiers are easily added.

The technique I've shown here can be expanded to more complex applications. However, where circuits of 20 or more nodes are involved, a specialized application program such as Micro-Cap or TUTSIM should be considered.

While the examples are all electronic circuits, nothing would prevent extension of the general technique to design and analysis of mechanical systems, chemical reactions, biological modeling, etc. Any process that lends itself to description by a simple model should be fair game for analysis by a spreadsheet.

The technique is ideally suited for the PC environment; all operations are out in the open and controlled by the user. The technique should be useful for students as well as expert designers. Try it—it's great for those small problems you want to look at in detail. ■

PC AT PERFORMANCE FOR A FRACTION OF THE COST. GUARANTEED.



That's right, guaranteed performance. The Univation Turbocharger will triple the processing speed of your IBM PC, PC/XT, or 100% compatible.

Just think, the data processing speed and performance of a PC/AT without the expense. Plug it in, and your PC will do everything it did before, only 300% faster.

Best of all, the Univation Turbocharger works with all your existing PC software automatically. Nothing to lea[redacted] nothing to change. Speed spread[redacted]sheets, databases, graphics, even IBM Basica programs. The Turbocharger will spark each and every program you use.

To ensure complete reliability, our Turbocharger is extensively tested. It's available for immediate delivery. And, best of all, it's guaranteed. Guaranteed to work exactly as promised [redacted] to provide all [redacted] processing power you'll ever need.

So why not make your life a little easier and do your work a lot quicker with the Univation Turbocharger.

See it at your local computer dealer today. For the dealer in your area call: (408) 745-0180.

UNIVATION

YOUR EXPANSION COMPANY

1037 North Fair Oaks Ave.
Sunnyvale, CA 94089
(408) 745-0180

Dealer and distributor inquires invited.

Inquiry 206

ADDING A HARD DISK TO YOUR PC AT

You can save money and get increased performance

BY JON SHIELL AND JOHN MARKOFF

THE IBM PC AT has rapidly become a new performance standard in the world of IBM and IBM-compatible personal computers. Because of the open-system philosophy of the IBM PC family, a thriving third-party industry has sprung up, offering both price and performance enhancements.

The PC AT comes in two models: the basic system without a hard disk and the enhanced system, which includes a hard disk, extra memory, and I/O (input/output) ports. It is possible to convert a basic model into an enhanced system merely by adding a third-party hard disk and multifunction board. Purchasing a hard disk independently can also offer cost savings and increased performance.

The multifunction card is particularly attractive for this application because of significant cost savings and increased system versatility. You can configure a PC AT with more than 8 megabytes of main memory. However, operating systems now available for the AT cannot productively use this memory. It is doubtful that even a 9-megahertz (MHz) PC AT can effectively use this much main memory. (See the text box "Crystal Change Enhances PC AT's Performance" on page 161.)

It is possible to put 230 megabytes of hard-disk storage on an 8-megabyte 9-MHz PC AT and reduce access time to half that of a factory-standard AT.

For most people, the best way to purchase a hard disk is in kit form. This precludes the necessity of hard-formatting the disk and generally simplifies the installation procedure. However, if you are scavenging a hard disk from another system or have bought a third-party product, you will need to obtain special mounting side rails and hard-format the disk (see the text box "Hard-Formatting a Disk Using the AT Advanced Diagnostics" on page 161).

Everything you need to know about physically installing and soft-formatting the disk is explained in the *AT Installation-and-Setup Manual*.

The first step is to determine the drive type. Refer to table 1 and table 2 and compare the parameters listed to the information supplied with your hard disk. Don't be surprised if you have to trim your disk to fit. Most kitted disks come with information about suggested drive types. IBM has predefined 14 types of disk drives (see table 1).

In addition, type 15 has been left open for user-defined drives. Table 2 shows disk-drive type numbers for some common hard disks. In some cases, the drive has been trimmed to fit by not using all available cylinders. It is also possible to trim a drive by not

(continued)

John Markoff, a technology writer for the San Francisco Examiner, *is a former* BYTE *editor. He can be reached at* 110 Fifth St., *San Francisco,* CA 94123.

Jon Shiell is a system architect and microprogrammer. He can be reached at POB 61195, *Sunnyvale,* CA 94088.

ILLUSTRATED BY BLAIRE THORNLEY

Table 1: *Definitions of predefined drive types.*

Disk type	Cylinders	Heads	Write precompensation	Landing zone (cylinder)	Capacity per drive
1	306	4	128	305	10 Mb
2	615	4	300	615	21 Mb
3	615	6	300	615	31 Mb
4	940	8	512	940	64 Mb
5	940	6	512	940	48 Mb
6	615	4	no	615	21 Mb
7	462	8	256	511	31 Mb $
8	733	5	no	733	31 Mb
9	900	15	no	901	115 Mb
10	820	3	no	820	21 Mb
11	855	5	no	855	36 Mb
12	855	7	no	855	51 Mb
13	306	8	128	319	21 Mb
14	733	7	no	733	44 Mb

$ 4 megabytes unused because only 462 out of 511 cylinders are used
Capacity per drive = cylinders • heads • 17 sectors/track • 0.5 Kb/sector
(heads is the same as tracks/cylinder)

Table 2: *Sample drive-type definitions.*

Disk type	Capacity per drive	Examples of this drive type
1	10 Mb	Cogito CGF912; MMI M212 and M312; Seagate ST412, ST212, and ST112; Rodime RO202; Tandon TM252 and TM502; Fujitsu M2233; Shugart 712 %; MiniScribe 2012 and 3412
2	21 Mb	Tulin TL226 %, Qume R200 %, Shugart 724 %
3	31 Mb	Tulin TL240 %, Qume R300 %, Rodime RO206
4	64 Mb	Atasi 3080 %
5	48 Mb	
6	21 Mb	Seagate ST4026
7	31 Mb	Quantum Q540 %
8	31 Mb	Seagate ST4038
9	115 Mb	Maxtor XT-1140 %
10	21 Mb	Micropolis 1302 %, Vertex V130 #%
11	36 Mb	Vertex V150 #%
12	51 Mb	Vertex V170 #%
13	21 Mb	Seagate ST425, MMI M225 and M312, Rodime RO204, Fujitsu M2235
14	44 Mb	

% Not all cylinders used; to use all you must define it as a type 15
# The Vertex drives are an extreme case of trimming to fit, as they actually have 987 cylinders, so only 83 percent of the V130 and 87 percent of the V150 and V170 are used

using all read/write heads, but this tends to be especially wasteful. Using only five out of six heads wastes about 17 percent of the drive's capacity.

After the disk is physically installed, use the AT diagnostic disk-setup option to set the drive-type nybble in the configuration RAM (random-access read/write memory). Unlike the PC and the XT, which use switches to tell the BIOS (basic input/output system)/DOS (disk operating system) what equipment is attached to the system, the AT uses a CMOS (complementary metal-oxide semiconductor) RAM with a battery backup. The RAM (50 bytes) is contained in a Motorola 146818 chip, which also contains a real-time clock. The RAM-configuration-data format and typical entries for 11 drives are given in table 3. Figure 1 is the CMOS RAM map, showing the address offset and contents of each byte in the configuration RAM.

Byte 12 holds the fixed-disk-type information for the C and D drives. Bits 0 through 3 (single hexadecimal digit) specify the drive type for drive C. Bits 4 through 7 (single hexadecimal digit) specify the drive type for drive D. A value of 0 hexadecimal indicates that no drive is present.

After the configuration RAM has been modified (using the diagnostic disk-setup option), complete the installation using the normal procedure for standard IBM fixed disks. Simply run the FDISK and Format programs, following IBM's instructions.

When partitioning your disk with FDISK, keep in mind that DOS currently supports a maximum of approximately 64K sectors per disk or partition. This requires that disks larger than 32 megabytes be split into a number of 32-megabyte partitions. Most kits for these large disks contain the software required to allow use of more than one active—but only one bootable—partition. Table 3a, for use with FDISK, shows the relationship between number of data heads, number of sectors per cylinder, and maximum number of cylinders allowed for 64K-sector DOS partitioning. Table 4

(continued)

Crystal Change Enhances PC AT's Performance

Changing the timing crystal voids the warranty! You should not change the crystal before the machine has had time to shake down for at least 90 days.

The speed of your PC AT is controlled by the crystal that determines the clock rate of the 80286 processor and its support chips. This clock rate is half of the crystal's frequency (the standard 12-MHz crystal gives a clock rate of 6 MHz).

Table A is a list of the common crystal frequencies and the clock rates they yield.

You should have a number of crystals with different frequencies because some ATs run faster than others. The ones we tested varied between 8 MHz and 9.8 MHz.

Your clock crystal should be tested with all add-on and multifunction boards present. Some high-performance multifunction boards will run at speeds of 8 MHz or higher, so test your system fully configured to be sure.

Table A: *Crystal frequencies and clock rates for the* PC AT.

Crystal frequencies (MHz)	Clock rate (approx., MHz)
12.0000	6.00
14.3181	7.16
15.0000	7.50
16.0000	8.00
18.4320	9.22
19.6608	9.83
20.0000	10.00

Crystals are available at most major electronics supply houses for less than $5 each. The ones we used were from Nymph and BME and had HC-18 cases.

The Nymph crystals had long, thin leads that needed to be trimmed.

To change the crystal:

1. Turn the machine off and unplug it.
2. Remove the cover and locate the 12-MHz crystal on the motherboard. It's above and to the left of the 80286 chip as you look from the front of the machine. Before removing the old crystal, touch the chassis to ground yourself. Use a thin flat-bladed screwdriver to remove the old crystal (be sure that you don't damage any traces on the motherboard).
3. Insert the new crystal (save the old 12-MHz crystal). Which side of the crystal is face up doesn't matter.
4. Close the cover and give it a try.

Try the fastest crystal first; if the crystal is too fast, the machine will not show the memory check or boot. In some borderline cases the machine will run fine after it has warmed up, but it may need to be rebooted first. Try each crystal starting with a cold machine.

Hard-Formatting a Disk Using the AT Advanced Diagnostics

Before beginning, note that the disk type must have been set in the configuration RAM prior to hard-formatting.

If you get a 17XX error when the system powers up or resets, press the F1 button to continue.

Enter the fixed-disk test menu and do an unconditional format by selecting the following:

1. System-checkout routine (option 0): Enter "y" if the options list is correct; otherwise, go back to setup and correct the list.
2. Run tests one at a time (option 0).
3. Test fixed-disk drive (option 17): If at this point another menu is not displayed, but instead the test begins, you know that you are not using the advanced diagnostics.
4. Select format menu and drive letter (options 7,c or 7,d).
5. Select unconditional format (option 2).

The current screen should ask you to enter the known flaws. A list of known flaws is printed on a label on the top of all hard disks. The list contains the cylinder, head number, and byte offset from the index, but you need enter only the cylinder and head numbers. After you have entered all the flaws listed on the top of the drive, press "y" to format the disk.

You now have a hard-formatted disk. Enter 9s to get back to the main menu. The next step is to run FDISK, then the normal format program on the DOS partition.

Offset	Contents
00	Seconds
01	Second alarm
02	Minutes
03	Minute alarm
04	Hours
05	Hour alarm
06	Day of the week
07	Day of the month
08	Month
09	Year
0A	Status register A
0B	Status register B
0C	Status register C
0D	Status register D
0E	Diagnostic status byte
0F	Shutdown status byte
10	Disk-drive type for drives A and B
11	Reserved
12	Fixed-disk-drive type for drives C and D
13	Reserved
14	Equipment byte (corresponds to switch 1 on PC and XT)
15–16	Base memory size (low,high)
17–18	Expansion memory size (low,high)
19–20	Reserved
21–2D	Reserved (not checksumed)
2E–2F	Checksum over bytes 10 through 20 (low,high)
30–31	Expansion memory size as determined by power-on routine (low,high)
32	Date century byte
33	Information flags (set during power-on)
34–3F	Reserved

The Alarm function is used by the operating system/BIOS to drive the Wait function, INT15h ah=90h

The drive-type bytes use bits 0:3 for the first drive and 4:7 for the other

Disk-drive types:

- 0 No drive present
- 1 Double-sided disk (360 Kb)
- 2 High-capacity disk (1.2 Mb)
- 3–F Reserved

The equipment byte is used to define the configuration for the power-on diagnostics

Base memory is all memory below the 1-megabyte line, the range is 256 Kb to 640 Kb

Expansion memory is all memory above (at) the 1-megabyte line, range between 0 (none) and 15 Mb, although you can currently get only to 3 Mb with 64K-byte RAMs without using an expansion chassis (which doesn't currently exist)

Bytes 00–0Dh are defined by the chip for timing functions, and 0E–3F are defined by IBM

To access the configuration RAM:

1) Write the byte address (00–3F) you want to access to I/O port 70h
2) Access (read/write) the data via I/O port 71h

Figure 1: CMOS RAM *map.*

Table 3: *Sample drive-type entries, for use as type-15 disks.*

Byte	MMI M206	Tandon TM503	Tandon TM703	Quantum Q540	Vertex V170	MiniScribe 6085	Micropolis 1324	Micropolis 1325	Maxtor XT-1105	Maxtor XT-1140	Largest defined disk	
	5 Mb	15 Mb	30 Mb	35 Mb	59 Mb	71 Mb	52 Mb	70 Mb	85 Mb	117 Mb	139 Mb	Disk capacity
0	306	306	695	512	987	1024	1024	1024	918	918	1024	Number of cyl.
2	2	6	5	8	7	8	6	8	11	15	16	Number of heads
3	0	0	0	0	0	0	0	0	0	0	0	Not used
5	0	128	0	0	0	0	0*	0*	0	0	0*	Write precomp.
7	0	0	0	0	0	0	0	0	0	0	0	Not used
8	0	0	0	256	0	0	0	0	8	8	8	Control byte
9	0	0	0	0	0	0	0	0	0	0	0	Not used
12	306	306	695	512	987	1024	1024	1024	918	918	1024	Landing zone
14	17	17	17	17	17	17	17	17	17	17	17	Sectors/track
15	0	0	0	0	0	0	0	0	0	0	0	Not defined

The TM503 is a "generic 15-megabyte hard disk"; many other disks, like the Seagate ST419, would also use this setup
The Maxtor XT-1140 appears here to show how it would be defined for maximum capacity
Largest defined disk is the largest (highest-capacity) disk that can be defined under the PC AT BIOS; a dedicated servo is assumed
Write precompensation depends on the actual drive used; an asterisk by a 0 write precompensation means that a dedicated servo is used and has a whole platter instead of just one side dedicated to it (thus the even number of data heads)

```
Offset
  0   Number of cylinders on the drive     [0–1023 allowed, for 1024 cyl.]
  2   Number of heads per drive            [0–15 allowed, for 16 heads]
  3   -n/u  (starting reduced write current cyl. on PC XT)
  5   Starting cylinder for write precompensation
  7   -n/u  (maximum ECC data-burst length on PC XT) {recheck not used}
  8   Control byte
        Bit
         7     Disable disk-access retries
         6     Disable ECC retries
        5–4    -n/d  (zero)
         3     More than eight heads
        2–0    -n/u  (drive option on PC XT)
  9   -n/u  (time-out values on PC XT)
 12   Landing zone, cylinder to use as a
 14   Number of sectors/track [0–17 allowed, 17 is the IBM standard]
 15   -n/d

-n/u   Field not used by PC AT
-n/d   Field reserved for future use
```

Figure 2: *Format of a drive-type entry.*

Table 3a: *Disk-partitioning data, for use with* FDISK.

Number of data heads	Number of sectors per cylinder	Maximum number of cylinders allowed in a 64K-sector DOS partition
3	51	1024
4	68	936
5	85	771
6	102	642
7	119	550
8	136	481
9	153	428
10	170	385
11	187	350
12	204	321
13	221	296
14	238	275
15	255	257
16	272	240

(continued)

Table 4: *Comparison of drive performance.*

	PC XT	Seagate ST425	PC AT	Qume R300	Quantum Q540	Tandon TM703	Vertex V170	MiniScribe 6085	Maxtor XT-1140
Type	1	13	2	3	7	15	15	15	9
Disk capacity	10 Mb	20 Mb	20 Mb	31 Mb	31 Mb	30 Mb	59 Mb	71 Mb	115 Mb
Number of cyl.	306	306	615	615	462	695	987	1024	900
Number of heads	4	8	4	6	8	5	7	8	15
Cyl. capacity	34 Kb	68 Kb	34 Kb	50 Kb	68 Kb	42.5 Kb	58.5 Kb	68 Kb	127.5 Kb
Access times $									
Track to track	16 ms	23 ms	14 ms	19 ms	10 ms	5 ms	5 ms	3 ms	5 ms
Mean	85 ms	65 ms	52 ms	93 ms	45 ms	45 ms	30 ms	30 ms	30 ms
Maximum	205 ms	170 ms	97 ms	213 ms	80 ms	65 ms	65 ms	50 ms	48 ms

$ Access times include head-settling time; average latency for all of the above disks is 8.33 ms
The access times for the XT are based on the Seagate ST412
The access times for the Maxtor XT-1105 are the same as those of the XT-1140

RANDOM NOTES

Jumper J18 remaps the second 256K bytes of memory from the system board into the I/O channel, so that non-IBM expansion boards can be used. Thus, you don't need to buy the 256K-byte motherboard RAM option if you buy a minimum system. We used a Sigma Designs 512K-byte (with 384K bytes enabled) multifunction card, scavenged from a PC, in our AT until we could get a 16-bit AT version.

You can't use 64K- or 256K-byte RAMs in the AT motherboard because the pin-outs of the 128K-byte DRAMs are different from industry standards (see table B).

You can add ROM to the AT motherboard in sockets U17 and U37. It appears at address E0000 to EFFFF hexadecimal. The ROM must have the same header as an I/O channel ROM except that byte 2 (ROM length) is not used.

The two 8-bit slots on the AT are wired for the addition of the 36-pin extended connectors.

Table B: *Pin-outs for standard* DRAMs *versus* IBM 128K-*byte* DRAMs.

Standard 64K/256K signal name	PC AT 128K*	Pin number
N/C	Din	1
Din	WE	2
WE	*RAS1	3
RAS	*RAS0	4
A0	A0	5
A2	A2	6
A1	A1	7
PWR	PWR	8
A7	A7	9
A5	A5	10
A4	A4	11
A3	A3	12
A6	A6	13
Dout	Dout	14
CAS	CAS	15
GND	GND	16

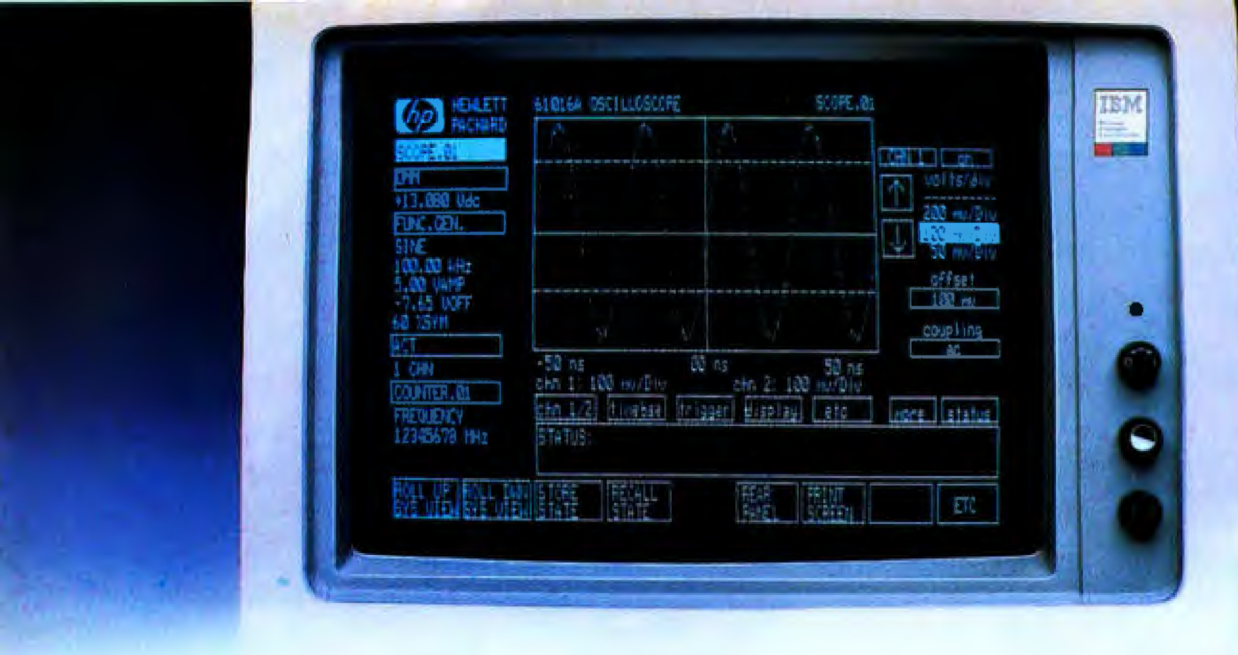
* IBM 128K-byte DRAMs are actually two 64K-byte dies encapsulated in piggyback fashion. One die is connected to RAS0, and the other is connected to RAS1. Otherwise, they are the same.

shows the relative performance characteristics of some popular drives.

You can define your own drive types; for example, you may want to add your old 5-megabyte drive, left over from your PC, as the second hard disk (drive D, fixed disk 1).

You must first build a drive-type entry like the sample shown in table 3. Place the address of the drive-type entry in INT 46 hexadecimal (at address 0:118 hexadecimal).

Using the setup program on your PC AT diagnostic disk, change the fixed-disk-type nybble in the configuration RAM to 15 hexadecimal. This tells the system that your disk is a type 15. See figure 1 for a complete definition of the configuration-RAM contents. Figure 2 shows the format of a drive-type entry. ■



HP PC Instruments
Soft Front Panel

See how you can automate testing with your IBM PC or HP Touchscreen PC.

Now you can perform test and measurement tasks easily, quickly and cost-effectively with affordable, rackable PC Instruments from Hewlett-Packard.

Soft front panels make HP PC Instruments easy to use. By simply touching the HP Touchscreen or using a mouse with the IBM PC, you can set functions, ranges and values, and take measurements.

You can also develop programs faster. A few easy-to-remember commands, like OUTPUT and MEASURE, control your PC Instruments from Microsoft® BASIC. And you can use the soft front panel to enter many of the instrument parameters that have been traditionally typed into a system. In addition, with optional HP Data Acquisition Software, you can be doing voltage scanning and temperature measurement in no time at all.

Add-on HP-IB libraries can also turn your PC into a versatile HP-IB instrument controller. And you can use them to control both PC Instruments and HP-IB instruments from the same BASIC program.

PC Instruments now available include:

- 4½ Digit Digital Multimeter
- 50 MHz Digital Oscilloscope
- 5 MHz Function Generator
- 100 MHz Universal Counter
- Relay Multiplexer
- 12 bit Dual Voltage Digital-to-Analog Converter
- 16 bit Digital Input/Output
- Relay Actuator

See a live demonstration of PC Instruments.

Call 1-800-344-2525 (in the Continental U.S.) or 1-800-348-8181 (in NJ) for details.

Microsoft® and Microsoft BASIC® are U.S. registered trademarks of Microsoft Corporation.
AD 2101552

HEWLETT PACKARD

FIXED DISKS AND THE PC AT

A comparison of speed and performance that matches the PC AT against the PC XT

BY JON SHIELL AND JOHN MARKOFF

THE IBM PC AT provides more flexibility and computing power than its popular predecessor, the IBM PC. The AT is based on the Intel 80286 microprocessor, which executes instructions faster and more efficiently than the PC's 8088.

The AT provides a memory space potentially 16 times larger than the PC's. This is an unexplored realm for personal computer software developers that is certain to prove tempting.

The introduction of the AT set a new standard for fixed-disk capacity and performance for microcomputer systems. The AT provides double the disk capacity of the PC XT and offers a simple method for the user to install up to two hard disks with capacities of almost 140 megabytes each.

WHY THE AT IS FASTER THAN THE XT

There are a number of factors, some obvious, some more subtle, why the AT exceeds the XT's performance. First, the raw clock rate of the AT's 80286 chip is 12 MHz. (The 80286 divides this by 2 internally.) The XT's 8088 clock rate is 4.77 MHz. Therefore, even if everything else were comparable between the two systems, the AT would still be a little more than 26 percent faster than the PC.

However, everything is not that clearly defined. For example, the AT's math coprocessor, the 80287, runs at only one-third the frequency of the AT's raw system microprocessor clock (12 MHz)—that is, 4 MHz. This compares unfavorably to the IBM PC's numeric processor, the 8087, which runs at the full speed of the PC—4.77 MHz. The math coprocessor doesn't help the AT's performance as much as might be expected.

Despite this obvious imbalance, however, most comparisons lean markedly in favor of the AT. A comparison of each computer's system bus reveals that the AT has a 16-bit data bus; the PC has an 8-bit data bus. Using an 8-bit bus for a 16-bit microprocessor (as is done in the IBM PC) costs 15 to 20 percent in system performance. This, combined with the approximately 25 percent increase due to the difference in clock rates, still gives a speedup of less than 50 percent, so there must still be other factors that contribute to the AT's speed advantage.

Significantly, the 80286-powered AT runs with a single wait state; if it ran no wait states, RAM (random-access read/write memory) with a maximum access time of less than 120 nanoseconds (ns) would be necessary. By adding the wait state it is possible for the AT to use standard RAM (150-ns access). The use of the wait state cuts the effective processing rate by less than 25 per-

(continued)

Jon Shiell is a system architect and microprogrammer. He can be reached at POB 61195, *Sunnyvale,* CA 94088.

John Markoff, a technology writer for the San Francisco Examiner, *is a former* BYTE *editor. He can be reached at* 110 *Fifth St., San Francisco,* CA 94123.

ILLUSTRATED BY SUSAN STILLMAN

cent on the average, according to Intel benchmarks.

At the microprocessor level, the 80286 is dramatically more efficient than the 8086. The 8086/8088 merely overlaps the instruction fetch (IF) of the next instruction with the execution of the current one.

The 80286 architecture is "pipelined" and adds an additional level of parallelism by overlapping the fetch of a third instruction with the decoding of a second instruction and execution of the current instruction. In fact, the amount of instruction overlap in the 80286 is roughly comparable to that in some of IBM's mainframe computers.

When combined with the additional hardware speedup of some instructions and internal bus and clock speed enhancements, the 80286 runs between two and six times faster than a 5-MHz 8086 or the PC XT's 4.77-MHz 8088, which is slower and uses an 8-bit data bus.

Another interesting point of comparison is in the speed of DMA (direct memory access), the device-to-memory transfer that doesn't involve the central processing unit (CPU). Here, the AT's performance is significantly slower than that of the IBM PC.

The PC AT's system DMA rate is 1.66 microseconds (μs) per transfer

FACTORS AFFECTING DISK PERFORMANCE

Four major physical factors determine overall disk performance: access time, cylinder size, transfer rate, and average latency.

Access time is the amount of time it takes to move the read/write heads over the desired tracks (cylinders). Once the heads are over the desired tracks, they must settle down from the moving height to the read/write height. This is called the settling time and is normally included in the access time. Specifications for AT and XT disk-drive options are shown in table A.

A cylinder is composed of all tracks that are under the read/write heads at one time. Thus, tracks per cylinder is the same as the number of data heads in the drive. Cylinder size is defined as tracks/cylinder × sectors/track × bytes/sector.

The Quantum Q540, for example, has four platters and eight data heads, while the Vertex V170 has four platters, seven data heads, and one servo head. The difference is that the Quantum drive uses an embedded (or wedge) servo, where the servo signal is embedded on the data tracks, preceding the data portion of each sector on the disk. The Vertex drive uses a dedicated servo that requires its own surface. This difference means that the Quantum drive has 8.5K bytes more data available to it before it must seek the next track; if all other factors were equal (which they aren't), the Quantum would be slightly faster in those cases that required reading that "extra" 8.5K bytes.

Transfer rate is the rate at which data comes off the disk. It depends on rotation rate, bit density, and sector interleaving. The first two factors are practically the same for all AT-compatible 5¼-inch hard disks, but not for all floppy disks (the AT's spin 20 percent faster than the other PC floppies).

Sector interleaving is used to cut down the effective transfer rate. The interleave factor of 6 used on the XT cuts the effective transfer rate from 5 megabits per second to 0.833 megabit per second. Note that embedded servo disks, such as those used in the XT and the AT, actually spin about 1 percent slower than 3600 revolutions per minute (rpm) to allow for the increased density due to the servo.

Average latency is the time required for a disk to spin one-half of a revolution. For hard disks, which spin at 3600 rpm, the average latency is 8.33 ms (1/3600 rpm × 60 seconds/minute × 0.5 = 8.33 ms per half revolution). This is due to the fact that after the heads finish seeking and settling, you must wait for the required sector to come under the heads.

Table A: *Sample transfer rates and a comparison of specifications for the disk-drive options available for the* IBM PC AT *and* PC XT. *Note that the interleave factor is the only difference between the last two entries in the table; the drive is the same for both entries.*

	PC XT Floppy Disk	AT 360K-byte Floppy Disk	AT 1.2-megabyte Floppy Disk	PC XT Hard Disk	PC AT Hard Disk	PC AT Hard Disk
Rotation	300 rpm	360 rpm	360 rpm	3600 rpm	3600 rpm	3600 rpm
Base rate	250K bits/second	300K bits/second	500K bits/second	5M bits/second	5M bits/second	5M bits/second
Interleave	none	none	none	6	3*	2*
Actual rate in bytes/second	31,250	37,500	62,500	104,167	208,333	312,500

* The AT technical reference manual shows an interleave factor of 2, but the advanced diagnostics and the "IBM PC Seminar Proceedings" claim that the interleave factor is 3.

(five AT cycles per transfer at 3 MHz). This yields the channel bandwidths shown in table 1.

The DMA rate of the PC XT is 1 byte per five cycles (1.05 μs) for a DMA channel bandwidth of 0.952 megabyte per second. This bandwidth is the result of IBM designing the AT with 5-MHz DMA controllers.

The AT's DMA controllers cannot run at the full 6-MHz rate of the AT system clock, and it is difficult to attain clock rates that are not simply an integer division of the system timing. The design decision was to set the DMA rate at 3 MHz. This is not as critical a shortcoming for the AT as it may first appear, because few devices (hard disks excluded) can saturate the AT's DMA capability.

It is interesting that the AT does not use DMA for hard-disk input/output (I/O). Instead, the AT's hard-disk controller has a 512-byte sector buffer that is accessible by the 80286 as a 16-bit device.

When the buffer is full or empty, the controller interrupts the 80286 (using INT14 hexadecimal), which then moves the data via programmed I/O (that is, Rep Insw for reads and Rep Outsw for writes) to or from memory at a rate of 2 megabytes per second. This transfer rate is approximately twice that of the XT.

The 2-megabyte transfer rate of the AT for string-move operations is the result of using three cycles (including one wait state) to bring the data (16 bits) into the processor and another three cycles (including another wait state) to move the data into memory. This process uses six cycles per 2-byte transfer. At a clock rate of 6 MHz, a single clock cycle takes 167 ns. Six cycles therefore require 1002 ns for a 2-byte transfer, which corresponds to an effective rate of 2 megabytes per second.

Zero Wait State

IBM has provided a special line called the zero wait state (0WS) signal on the bus to suppress the wait states. This is useful if you have high-speed memory and an expansion card that supports this signal.

For example, in the preceding case performance could be improved by one-sixth because the memory card used in the transfer could suppress the wait state. However, the AT's disk controller won't suppress the wait state when transferring data from its sector buffer. Therefore, a savings of only one cycle per transfer is possible. By using the 0WS signal we can speed up the transfer rate from 2 megabytes per second to 2.5 megabytes per second.

In addition to bus factors, the AT's 1.2-megabyte floppy disk holds more than three times as much data as the XT's 360K-byte floppy disk. It also has a data-transfer rate that is twice that of the XT's and an average latency that is 20 percent less than that of the XT's. Furthermore, the AT's floppy disk has somewhat faster access times.

Of course floppy-disk speeds are not that significant, as the floppy disk plays a much less important role in

(continued)

Table 1: *Channel bandwidth data for the* IBM PC AT.

DMA Channel	Transfer Width	Channel Bandwidth
0–3	1 byte	0.6 megabyte/second
4	used to cascade channels 0–3	
5–7	2 bytes	1.2 megabytes/second

Can the AT Support Noninterleaved Hard Disks?

[*Editor's note: An interleave factor of* 1:1 *implies a noninterleaved disk.*]

Let's do some quick back-of-the-napkin calculations to answer this question. At 3600 rpm one rotation takes 16.67 ms, and one rotation per track of 17 sectors results in less than 1 ms per sector. There are about 10,416 bytes per unformatted track, again at one track of 17 sectors; this gives us about 612 bytes per unformatted sector.

There are 512 bytes of data in a formatted sector, thus about one-sixth of the sector is not data (512 data bytes out of 612 possible). If we assume that during this period the AT's track buffer is available to the 80286, then we have one-sixth of 1 millisecond or about 150 μs to fill or empty the buffer. This assumes that the buffer is single-ported and doesn't support interleaved access, which seems to be the case for the AT.

Programmed I/O normally transfers one word every six cycles, which means a data rate of 2 megabytes per second. To move our 512 bytes of data will take about 250 μs, assuming INT14 hexadecimal (the buffer full/empty interrupt) overhead is 0, which it isn't.

We expect that without interleaving we have less than 150 μs to fill or empty the buffer, while the 80286 would need at least 250 μs. Consequently, the sector buffer will not support 1:1 interleaving but will probably support 2:1 interleaving without any major problems. This is due to the fact that we have almost 1 millisecond to fill or empty the buffer when 2:1 interleaving is used.

the system design of the AT than it does in the PC.

Comparing the AT and XT Hard Disks

Let's compare the hard disks that are standard equipment on the XT and the AT. We will also discuss a 140-megabyte disk, the Maxtor XT-1140.

Looking at the relative access times (including latency) for the XT, the AT, and the Maxtor shows the Maxtor to be by far the fastest drive; however, we also find that the AT's disk is about 40 percent faster on the average than the XT's (see table 2).

But access time is not everything. Table 3 shows the comparative times for a 127,000-byte read (assume all sectors are contiguous and start with sector 1, head 0).

Although the Maxtor's access time for this operation is about 350 percent faster than the AT's, which is about 15 percent faster than the XT's, its access time gets swamped by the transfer times. The apparent differences are shown in table 4.

The XT appears to be about 85 percent slower than the AT, which is only about 14 percent slower than the Maxtor. The Maxtor's faster access time is swamped by the 609 milliseconds (ms) it takes to transfer the data, and it has no advantage in transfer rates. However, for a smaller (more normal size) file, say 34K bytes, we would get the times shown in table 5.

The XT appears to be about 80 percent slower than the AT, which is

Table 2: *A comparison of disk performance for the* AT *and the* XT *with standard disks and the* AT *with the Maxtor disk.*

Track-to-track access time + latency		Index
Maxtor	5 ms + 8.3 ms = 13.3 / 22.3 =	0.60
PC AT	14 ms + 8.3 ms = 22.3 / 22.3 =	1.00
PC XT	16 ms + 8.3 ms = 24.3 / 22.3 =	1.09
Mean access time + latency		**Index**
Maxtor	30 ms + 8.3 ms = 38.3 / 60.3 =	0.64
PC AT	52 ms + 8.3 ms = 60.3 / 60.3 =	1.00
PC XT	85 ms + 8.3 ms = 93.3 / 60.3 =	1.55
Maximum access time + latency		**Index**
Maxtor	48 ms + 8.3 ms = 56.3 / 105.3 =	0.53
PC AT	97 ms + 8.3 ms = 105.3 / 105.3 =	1.00
PC XT	205 ms + 8.3 ms = 213.3 / 105.3 =	2.03

Table 3: *Time required for a* 127,000-*byte read operation.*

Drive Type	Bytes/ Cylinder	Cylinders Required	Average Access Time (note 1)	Cylinder-to-Cylinder Access Time (note 2)	Interleave Factor	Average Latency/ Track (note 3)	Aggregate Access Time (note 4)	Transfer Rate (bytes/second)	Transfer Time (note 5)
PC XT	34K	four	85	(3) × 16	6	8.3	166.2	104,200	1219
PC AT	34K	four	65	(3) × 14	3	8.3	140.2	208,333	609
Maxtor	127K	one	30	n.a. (one cylinder used)	3	8.3	38.3	208,333	609

Notes

1. Average access time is the average time required for the disk heads to move from any cylinder to any other cylinder on the disk (including settling time before the heads can read data).
2. Cylinder-to-cylinder access time is the time required to move the heads from one cylinder on the disk to an adjacent cylinder on the disk (including settling time).
3. Average latency is the time required for a disk to spin one-half revolution. For hard disks, which spin at 3600 rpm, the average latency is 8.33 milliseconds

   (1/3600 rpm × 60 seconds/minute × 0.5 = 8.33 ms per half revolution).

   This is due to the fact that after the heads finish seeking and settling, you must wait for the required sector to come under the heads before reading data.
4. Aggregate access time is determined by

   average access + [number of cylinder-to-cylinder changes × cylinder-to-cylinder time] + [number of cylinders × average latency].
5. Transfer time is determined by

   [127K bytes divided by transfer rate in bytes/second] expressed in milliseconds.

about 40 percent slower than the Maxtor. If the file were spread over two cylinders on the XT and the AT (a more normal case due to their significantly smaller cylinder size), the ratios would be as shown in table 6. As expected, the Maxtor looks even better in this more typical case because it doesn't need the additional track-to-track access and associated latency. Also note that the AT's relative performance slipped a bit due to the small difference in AT and XT track-to-track access times.

Access time is most important when doing a lot of disk accesses in a short period of time, for instance, when searching a large database or working on a file that, due to fragmentation, is spread out over the entire disk. For example, to read a 34K-byte file (17 clusters on the AT's 20-megabyte disk) on a very fragmented disk could take up to 17 average accesses for a total of more than 1 second (1025 ms = 17 × (52 ms + 8.3 ms)).

It takes 163 ms to transfer the data. So our effective I/O time is 1188 ms as opposed to 236 ms for an unfragmented disk. Even if we assume less fragmentation—say the file is split into four parts—we get total access time of about 241 ms (4 × (52 ms + 8.3 ms)); adding the transfer time gives us 404 ms, or about a 50 percent decrease in the disk's effective "speed."

This points out the importance of not letting your disk get very fragmented, as this results in much longer effective access times.

Let's look at another example using our big file of 127K bytes. If it's all in three continuous cylinders, it will take about 117.9 ms, as we've shown, for the access portion of the read (assuming that the file allocation table [FAT] and the directory are already in memory). But the worst case is where the file is spread out over the entire disk, with only one cluster per cylinder and the cylinders in random order.

It could take approximately one mean access per track over 64 tracks (127K bytes divided into 64 clusters of 2K bytes each; one cluster per track) for an effective access time of almost 4 seconds (3859 ms = 64 × 60.3 ms including latency). So we get disk I/O times of about 749 ms versus 4468 ms (almost 4.5 seconds); thus, the fragmented disk appears to be about six times slower than the unfragmented one.

If you want to get an idea of how fragmented your hard disk is, the DOS CHKDSK command, when given a filename, displays the number of discontinuous areas occupied by the file. While this is not as useful as it could be (what we really want to know is how many cylinders the file is spread across), it gives us a fair idea of the amount of fragmentation.

Table 4: *A comparison of access time, transfer time, and performance index for the AT and the XT with standard disks and the AT with the Maxtor disk.*

Drive Type	Access Time	Transfer Time	Total	Performance Index
PC XT	166.2 ms +	1219 ms =	1385 ms / 749 ms =	1.85
PC AT	140.2 ms +	609 ms =	749 ms / 749 ms =	1.00
Maxtor	38.3 ms +	609 ms =	647 ms / 749 ms =	.86

Table 5: *Transfer time and performance index for a 34K-byte transfer.*

PC XT drive: one average access
85 ms + 8.3 ms = 93.3 ms
34,000 bytes at 104,200 bytes/second = 326 ms (interleave of 6)

PC AT drive: one average access
65 ms + 8.3 = 73.3 ms
34,000 bytes at 208,333 bytes/second = 163 ms (interleave of 3)

Maxtor drive: one average access
30 ms + 8.3 = 38.3 ms
34,000 bytes at 208,333 bytes/second = 163 ms (interleave of 3)

Results in:

Drive Type	Access Time	Transfer Time	Total	Performance Index
PC XT	93.3 ms +	326 ms =	419 ms / 236 ms =	1.78
PC AT	73.3 ms +	163 ms =	236 ms / 236 ms =	1.00
Maxtor	38.3 ms +	163 ms =	146 ms / 236 ms =	.62

Table 6: *Transfer time and performance index for a 34K-byte file contained on two cylinders.*

Drive Type	Access Time	Transfer Time*	Total	Performance Index
PC XT	93.3 ms +	326 ms + 24.3 ms =	444 ms / 259 ms =	1.71
PC AT	73.3 ms +	163 ms + 22.3 ms =	259 ms / 259 ms =	1.00
Maxtor	38.3 ms +	163 ms + 0.0 ms =	146 ms / 259 ms =	.56

* Track-to-track access time plus average latency

CLUSTER SIZE

An additional factor, cluster size, is purely an operating-system function.

(continued)

but it does affect disk performance. A cluster is the number of contiguous sectors that DOS allocates each time disk space is needed. The FAT is what DOS uses to keep track of which clusters are allocated to which files and which are available for use. Beginning with DOS 3.0 there are now two types of FATs, depending on the size of the partition. An FAT that uses 12-bit entries is used for disks holding less than 20 megabytes, and one with 16-bit entries is used for disks holding 20 megabytes or more.

On a 10-megabyte disk or partition, each cluster is 4K bytes or eight sectors, and the FAT takes up 4K bytes. Using a 12-bit FAT entry results in 4096 possible clusters. DOS uses a 4K-byte cluster with a 12-bit FAT. This results in a maximum disk size of 16 megabytes (4096 clusters/disk × 4096 bytes/cluster) without using a larger cluster size. All floppy disks use this scheme.

On disks or partitions of 20 megabytes and more, the cluster size is 2K bytes or four sectors. For a 20-megabyte disk, this means that the FAT is 20K bytes, and for a 32-megabyte disk or partition, the FAT occupies 32K bytes.

(continued)

USING DEBUG TO DETERMINE THE CLUSTER SIZE OF A DISK

The first and last register dumps are for the C drive, while dump 2 is a 1.2-megabyte floppy disk.

The register meanings are as follows: If the AX register contains FFFF hexadecimal on return, then the drive number was invalid and the rest of the registers are meaningless. Otherwise the registers contain the following:

AX number of sectors per cluster, four for the hard disk and one for the floppy disk

BX number of free clusters

CX number of bytes per sector, normally 200 hexadecimal or 512 bytes

DX total number of clusters on the disk

Therefore, the hard disk has a DOS partition of 15,630 (3D0E hexadecimal) clusters at four sectors per cluster, for a total of 62,520 sectors or 31.26 megabytes (using a Quantum Q540, Type 7 drive).

```
>debug
-a
XXXX:0100  ; Remember that debug works only with hexadecimal numbers
XXXX:0100  mov ax,3600    ; Load AH with the function code (36 hexadecimal)
XXXX:0103  mov dx,0000    ; DL = drive number (0=default, 1=A, 2=B, 3=C, . . .)
XXXX:0106  int 21         ; Call DOS to do the function
XXXX:0108  nop            ; Space saver to stop at
XXXX:0109
-g =100 108               ; This is for Drive C

AX=0004  BX=2F6C  CX=0200  DX=3D0E  SP=XXXX  BP=XXXX  SI=XXXX  DI=XXXX
DS=XXXX  ES=XXXX  SS=XXXX  CS=XXXX  IP=0108   XX XX XX XX XX XX XX XX
XXXX:0108 90              NOP
-a 103
XXXX:0103 mov dx,0001     ; Now let's look at the 1.2-megabyte floppy
XXXX:0106
-g =100 108

AX=0001  BX=045B  CX=0200  DX=0943  SP=XXXX  BP=XXXX  SI=XXXX  DI=XXXX
DS=XXXX  ES=XXXX  SS=XXXX  CS=XXXX  IP=0108   XX XX XX XX XX XX XX XX
XXXX:0108 90              NOP
-a 103
XXXX:0103 mov dx,0003       ; Now let's look at Drive C
XXXX:0106
-g =100 108

AX=0004  BX=2F6C  CX=0200  DX=3D0E  SP=XXXX  BP=XXXX  SI=XXXX  DI=XXXX
DS=XXXX  ES=XXXX  SS=XXXX  CS=XXXX  IP=0108   XX XX XX XX XX XX XX XX
XXXX:0108 90                NOP
-q
>
```

The actual calculation for FAT size, assuming a 2K-byte cluster, is: disk or partition size/2K bytes × 2 bytes/FAT entry, which reduces to 1K byte of FAT per 1 megabyte of disk or partition. The reason for increasing the number of clusters on the disk instead of just increasing the size of each cluster is that, while large clusters are good for access time, most files fit in just a few clusters, thereby minimizing the fragmenting of the disk.

A large cluster is inefficient in terms of space. Assume that we use an 8K-byte (16 sectors) cluster size for our 20-megabyte disk. This lets us map the disk into an FAT with 12-bit entries (20 megabytes at 1 cluster; 8K bytes is 2560 clusters). But because files are allocated space in units of 8K bytes, on the average each file wastes 4K bytes (half of its last cluster).

This 4K bytes of waste per file can really add up. Let's say we have 512 files on a disk. This means that 2 megabytes, or 10 percent of the total space on the disk, is wasted (512 files × 4K bytes per file). By keeping the cluster size at 2K bytes (four sectors), the amount of wasted space is greatly reduced, but a penalty is paid both in the size of the FAT and in the speed with which the disk gets fragmented.

The reason we care about the size of the FAT is not because it uses up more disk space but because DOS keeps a copy of the FAT of the active disk(s) in memory. And while 20K bytes for a 20-megabyte disk (or 32K bytes for the biggest disk or partition DOS supports) is not so bad in a 256K-byte system, it can add up.

One last relevant feature of the AT is that its hard-disk controller supports overlapped (buffered) seeks. Also, the controller board supports one floppy-disk and one hard-disk data transfer at the same time. The fixed-disk interface is ST412. Overlapped seeks allow a system with two or more hard disks to overlap operations. This is done by sending multiple step (seek) pulses to a drive that then disconnects and does the stepping without the controller. The controller is then free to work with its remaining drive(s). ■

A FEW GOOD REASONS TO BUY A NEW CANON PERSONAL COMPUTER.

Canon Personal Computers run all popular IBM PC-compatible software.

If you're in the market for a smart, fast, modestly priced personal computer that's IBM-compatible, look no further. Canon has just created the ideal series.

The Canon Personal Computer A-200 Series can run hundreds of programs designed for the IBM PC family. Without modification. Even the layout of the keyboard is compatible with the IBM PC's. And with its 16-bit microprocessor, memory and display adapter, each can process complex data in record time.

They expand as you expand.

As your needs change, so can Canon Personal Computers. Each comes with built-in parallel and serial ports. And with expansion capability in most models, you can modify the system according to your own needs.

You can add optional IBM-compatible hardware, too. Like the Canon memory expansion board. Or one of Canon's full line of quality printers, including the advanced laser beam printer.

Canon's three-for-all.

The Canon Personal Computer A-200 Series offers three ways to enter the world of personal computers.

The standard floppy disk model is a ready-to-run, compact system with 256K bytes of memory, expandable to 512K bytes. It offers a choice of monochrome or color graphic display and two 5¼" 360K byte drives.

The powerful hard disk model runs complex business software with even more speed and greater storage memory capacity. It comes with a 5¼" 360K byte floppy disk drive and a 10M byte hard disk drive.

And the transportable model weighs in at a low 19 pounds, so it's perfect for the business professional on the move. It features a 512K byte memory, full-sized graphic LCD screen and two 360K byte floppy disk drives. Plus, the RGB color monitor interface lets you use the transportable model with a color graphic display. And the transportable model has a built-in 300/1200 bps modem.

What's in a name?

Plenty, when that name is Canon. Every Canon Personal Computer is the product of a company world-renowned for quality and innovation. So when you enter the computer age with Canon, you know you're in good company.

For more information, see your local computer dealer or call 1-800-323-1717, x325. (In Illinois, 1-800-942-8881, x325.)

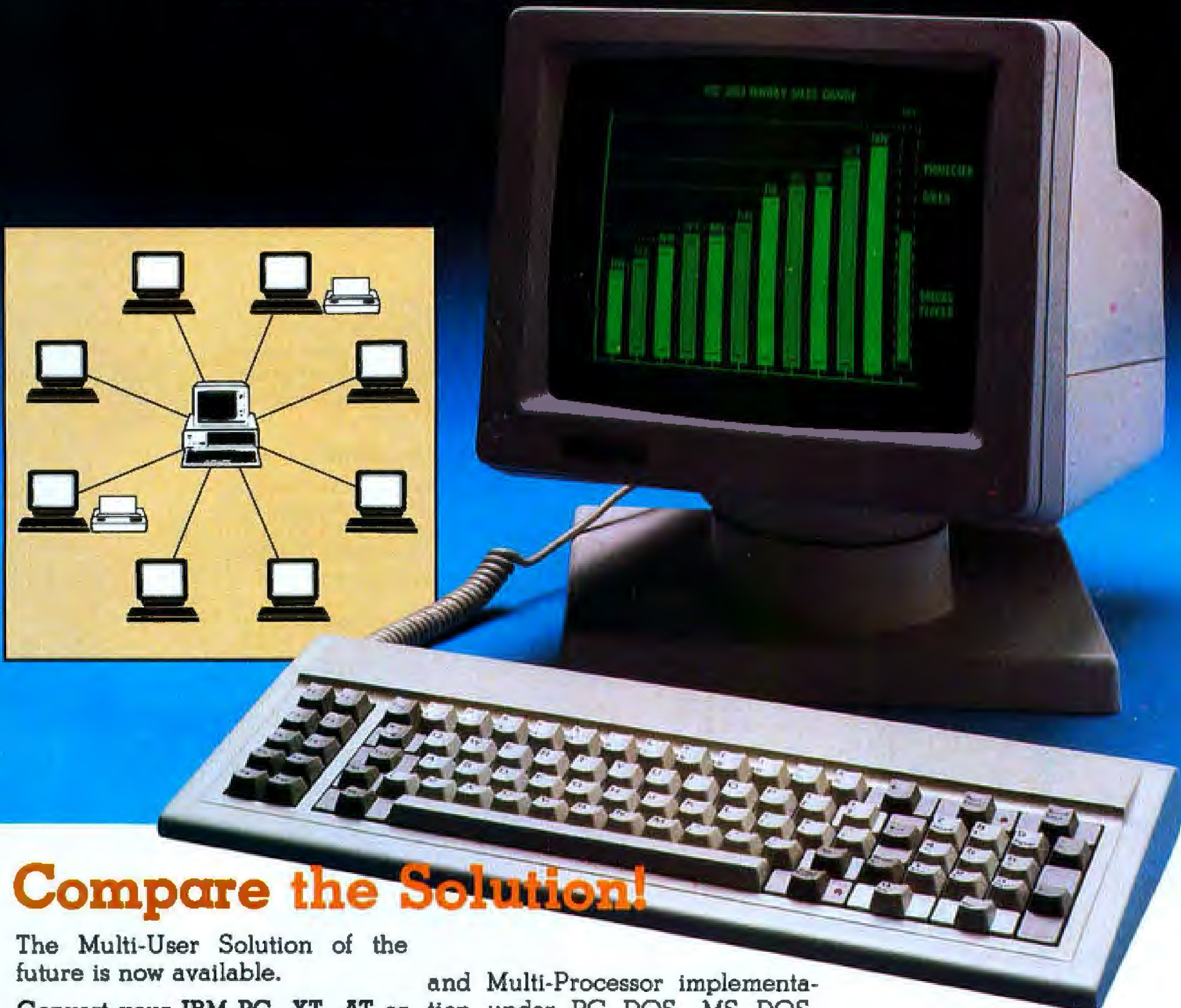
Canon®

Canon U.S.A., Inc., One Canon Plaza, Lake Success, NY 11042.

Inquiry 20

Kimtron

MULTI-USER SOLUTION for IBM PC, XT, AT



Compare the Solution!

The Multi-User Solution of the future is now available.

Convert your IBM PC, XT, AT or Compatibles to a true multi-user system while maintaining display, keyboard and software compatibility.

Since the KT-7/PC display is the same as your PC monochrome monitor, with its look-alike keyboard, operators will feel they're using an IBM PC and can also use the same software manual. Kimtron's multi-user solution includes file and record locking, shared data access, and communication between users. It is **the** intelligent alternative.

The KT-7/PC supports Time Sharing, Enhanced Time Sharing and Multi-Processor implementation under PC DOS, MS DOS, UNIX, XENIX, CPM 86, Multilink, Concurrent PC DOS, and other compatible multi-user operating systems.

Kimtron's multi-user solution may be tailored for cost effectiveness; as low as $1095 for an additional user, and for speeds more than ten times faster than LAN. You can add one or as many as 31 additional users per PC. Kimtron delivers the future now by allowing an ever-widening network of multi-user PC's.

The KT-7/PC may be complemented with one (or more) I/O Card, Memory Card, 8086 Speed Enhancer Card, 80286 AT Card, 8088 Multi-Processor Card, 80186 Speed Enhancer Card, 68000 Card, and related software.

For more information about Kimtron's Multi-User Solution, or general video data terminals for other mini or micro multi-user systems, call your local computer dealers, distributors or Kimtron Corporation Today!

(408) 286-8790
TWX 910-338-0237

1705 Junction Court
Building #160
San Jose, CA 95112

Kimtron

NOTE: IBM PC, XT, AT, PC DOS, MS DOS, UNIX, XENIX, CPM 86, Multi-Link, Concurrent PC DOS are registered trademarks of IBM Corporation, Microsoft Corp., Bell Labs., Digital Research Inc., Software Link Inc. respectively.

Inquiry 104

A SURVEY OF DEBUGGERS

A detailed look at the features you should look for in a debugger

BY FRANK DRAKE JR., ARTHUR MCCAFFREY, AND JOHN SADOWSKY

A WIDE RANGE of people use debuggers. The novice programmer curious about IBM PC architecture, the end user making an occasional patch to software, the hacker exploring existing code, and the software developer wanting to make the best use of his time all need some way to see inside the IBM PC and its software. (See the text box "How Debuggers Work," page 180, for a discussion of debugger techniques.)

Recent developments have brought many new debugging features to the IBM PC. We plan to showcase some of these features and describe some tips for getting the most out of Microsoft's Debug, a program that is included with MS-DOS. Debug is an extremely reliable program. Its more common uses include tracing through and disassembling object code, dumping data, and reading absolute disk sectors. Since anyone who owns MS-DOS owns Debug, it is the cheapest and most common debugger for the IBM PC.

One of the important needs that Debug fills is for a standard debugger. When you want to illustrate some technical feature of the IBM PC, you can be sure that everyone has Debug. Using input redirection, you can simplify the distribution of a patch for a program that is already on the market. The best way to distribute this patch is to create a file that contains the keystrokes of a Debug session that would fix the program. You could send it to software distributors, who would then be free to give a copy to anyone who needed it. You would only have to execute Debug, using input redirection to get its input from the debugging session file. If the file containing these keystrokes is named PATCHES.TXT and the program to be patched is BADPROG.COM, the command line would be

```
A>DEBUG BADPROG.COM <PATCHES.TXT
```

Debug is lacking in several areas. It does not support symbols (in this article, symbols refer to mnemonics that represent memory locations within your program). Its tracing capabilities are weak. It has only 10 breakpoints and as soon as program execution stops, all breakpoints are cleared.

When Debug and the program being debugged share the same output device, Debug overwrites the program's output. One way around the problem of overwritten screens is to redirect Debug's output to a printer. Then the printer acts as a terminal and the screen is free. To accomplish this, enter

```
A>DEBUG >PRN
```

(continued)

Frank Drake Jr. (34 Farrwood Dr., Andover, MA 01810) is a student and employee of the University of Lowell. Arthur McCaffrey (98 Neponset Ave., Hyde Park, MA 02136) has been involved in various aspects of computers, programming, and analysis since the early 1960s. John Sadowsky (Atlas Distributing Corp., 44 Southbridge St., Auburn, MA 01501) is a self-taught programmer who uses microcomputers extensively in small-business management.

ILLUSTRATED BY ALEXA GRACE

just look to your printer for Debug's output. You should remember that most printers are line-buffered and will not show Debug's prompt or what you type until you hit the carriage return. Also, note that this technique does not work for programs that write their output to the standard output device; in this case, the program's output will also go to the printer.

OPERATING MODES

Because a programmer might spend many hours of his day with a debugger, the user interface is an important factor. Two approaches are terminal and windowed output. Terminal output, like that used by Debug, is more common. Usually programs with terminal output have command-driven input, where you type a request and the debugger either completes the given task or supplies the requested information. As the screen fills, the oldest data scrolls off it. An advantage of the terminal interface is that you can look back and see what you have done over the last few commands.

A windowed interface makes it easy to view several kinds of information at once. You might have a window with a memory display, a second window with a disassembly of the code, and a third window for the registers. (Pfix86 from Phoenix Software Associates Ltd. uses this kind of output interface, along with a Lotus 1-2-3–style menu-driven input interface.) You might even have a stack window that grows down or up, depending on how you like to picture the stack in your mind.

PC Probe (Atron Corp.) mixes the features of both worlds. It has a terminal I/O (input/output) interface like Debug, but it also allows definition of macros and windows that can be linked together. For example, you can create a macro that will display the registers and specific memory locations. Once linked to a window, the registers and values at the memory locations are displayed in that window after each debugger command has been completed. This interface lets you make the best use of your screen, displaying exactly what you want when you want it.

SYMBOLIC DEBUGGING

Among the most important debugger features is support for symbols. It is usually a good idea to break down large programs into small modules so you can work on portions of the program, then link them all together. When you have multiple segments, you can no longer use the offsets found in the .LST file to determine the location of the program counter or data. Symbols let you refer to memory locations by the name used in the source file.

Since debuggers use executable code (the .EXE or .COM file being debugged) as their main input, they do not have direct access to the symbols used in the source file and must obtain the symbols in other ways. A common way to obtain symbols is by reading the .MAP file produced by the linker (although you might have to convert it to the debugger's own readable format). In order to obtain symbols this way, you must meet two conditions. You must define the symbols as public (global) symbols when the source is assembled or compiled, and you must link the program using the linker's /MAP option to place all the symbols it encounters into the .MAP file. You can use this process to your advantage in that you can enter symbols into your .MAP file after linking. Unfortunately, generating a new .MAP file will overwrite the old one, and any symbols entered this way will be lost if you link again.

One debugger, SYMD from D & V Systems, can get its symbols from the .LST file (if you are debugging a .COM program) so that public and local symbols will be available while debugging. The keyboard is another source of symbols. Advanced Trace86 from Morgan Computing Co. Inc., CodeSmith-86 from Visual Age, PDT-PC from Answer Software Corp., and Pfix86 Plus from Phoenix Software Associates Ltd. all let you define symbols during the debugging session. CodeSmith-86 lets you take a "snapshot" of the session and write it to disk but does not save the new symbols in the main symbol file. You can reload this snapshot later so you can resume a past session and restore all the symbols entered at the keyboard during that session. On the other hand, Advanced Trace86 saves symbols in the executable file. Pfix86 Plus saves them in a symbol table file.

Symbols make it much easier to follow the flow of the program, especially in a debugger that presents a window on the disassembled code. When you look at disassembled code, it is much easier to understand a JMP ERROR_EXIT than a JMP 127f:1045.

Several debuggers allow symbolic debugging in overlays. Pfix86 Plus gives the symbolic output of overlays linked with Phoenix Software's Plink86 program linker. PDT-PC and PC Probe also support symbolic debugging of overlays with Plink86.

BREAKPOINTS

All debuggers support the ability to stop execution when the CS:IP (code segment:instruction pointer) combination reaches a specified address. In Debug, for example, you can set such a breakpoint with the G ADDR command, where ADDR is either an offset in the code segment or a segment/offset combination. You can set up to 10 such breakpoints.

Other debuggers go far beyond Debug's rather rudimentary treatment of breakpoints. Symbolic debuggers let you set a breakpoint at a user-defined symbol rather than at a numeric address. This allows you to concentrate on program flow rather than on the hexadecimal addresses in a program listing. SYMDEB, the symbolic debugger included with Microsoft's Macro Assembler version 3.0, is a symbolic superset of Debug. In the example

```
-G = START_OF_PROG SUB1
```

SYMDEB would start execution at the location represented by START_OF_PROG and would execute until it reached either the point represented by SUB1 or the end of the program.

Some debuggers not only allow

(continued)

EVEREX
EVER FOR EXCELLENCE

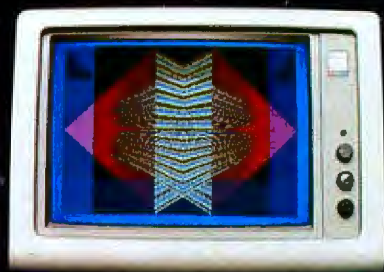
LOTUS 1-2-3 132 COLUMNS

PC PAINT

PC PAINTBRUSH 720x348

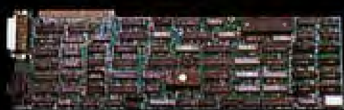
SYMPHONY HIGH RESOLUTION

PC PAINT



16 COLORS, 320x200

The Edge



The Plain Facts:

The EVEREX EDGE	The EVEREX EDGE	Paradise Modular Card	Genoa Spectrum	STB Chauffeur	Hercules Graphics Card	Tseng Labs Ultrapak
Runs an IBM or Compatible Color Monitor	✓	✓	✓			
IBM Monochrome Compatible and High Resolution, 720x348 graphics	✓		✓		✓	✓
Runs Color Software on a Monochrome Monitor, Full Screen:	✓	✓	✓	✓		
—in 16 shades of green on the IBM Monochrome monitor	✓	✓	✓	✓		
—Runs Flight Simulator, PC Paintbrush, IBM BASICA commands and more	✓	✓	✓	✓		✓
—No patches or modifications needed	✓		✓	✓		✓
—Displayed in the IBM standard 9x14 character set	✓	✓				✓
Extended Text Modes:	✓		✓			✓
—132 columns by 25 rows in monochrome	✓		✓			✓
—132 columns by 44 rows in monochrome	✓		✓			✓
—132 columns by 25 rows in color	✓		✓			
Runs Lotus 1-2-3™ and Symphony™ in high resolution monochrome graphics:	✓		✓		✓	✓
—in 132 columns by 25 rows	✓		1-2-3 only			✓
—in 132 columns by 44 rows	✓					✓
Runs Lotus 1-2-3™ and Symphony™ in High Resolution color:	✓		✓			
—16 colors, 320x200	✓		✓			
— 4 colors, 640x200	✓		✓			
Software switchable for all display modes and monitors	✓		✓			
Software switchable display memory, 16K, 32K and 64K	✓					
Disable Screen Memory	✓					
Printer Port (standard)	✓		✓	✓	✓	✓
Light Pen Connector (standard)	✓	✓	✓		✓	
Works in any PC-AT slot	✓			✓	✓	
List Price	$399	$395	$459	$395	$499	$680*

*Includes one serial port and clock/calendar

Excellence is the standard at Everex—it's in our name, our products, in everything we do. When you look for the best for your computer—you'll find Everex products.

Visit your local Everex dealer today and ask to see Everex products in action. For the name of your nearest Everex dealer, please call (415) 498-1111. TELEX: 5101000590 EVEREX FAX: 415-651-0728

Registered Trademarks: Paradise Modular Graphics Card—Paradise Systems, Inc.; Hercules Graphics Card—Hercules Computer Technology; Lotus 1-2-3—Lotus Development Corporation; PC Paint—Mouse Systems Corporation; Tecmar Graphics Master—Tecmar Inc.; Persyst Bob—Persyst Products; Ultra Pak—Tseng Lab.; The Edge—Everex Systems Inc.; IBM, PC Tutorial—International Business Machines Corporation; Flight Simulator—Microsoft Corporation

Dealer Hotline: (800) 821-0806. In CA (800) 821-0807.

Imagineering Ultimo, Australia TLX: 74349 IMAGIN AA
Microage Distribution Ltd. London, England TLX: 881 3241 WONGS G
Feeder Paris, France TLX: 4413241 FEEDER
Automated Office Systems Hout Bay, South Africa 2721-70-8091
Survex, 1027 Speers Road, Oakville, Ontario Canada L6L-2X5, 416-842-6093
Pride Computers, 102-8167 Main Street, Vancouver, British Columbia, V5X 3L2, 604-321-5690○

EVEREX
EVER for EXcellence

Address: 47777 Warm Springs Blvd., Fremont, CA 94539 (415) 498-1111

How Debuggers Work

by Jordan Lee Wagner

An assembly-language debugger provides at least two major functions. First, it runs your program one instruction at a time. After each step, your program stops so you can examine the effects of every statement in your code. A debugger also runs your program at full speed until a particular instruction is reached. These capabilities are called single-step trace and breakpoints.

Since the need to debug programs is well known, Intel designed the 8088 with special features to support tracing and breakpoints. This makes it easy to write a simple debugger.

Interrupts

To help you understand how the 8088 architecture incorporates support for debuggers, I will first review interrupts. People process interrupts all the time. Suppose that you are filling in your income tax form. You suddenly realize that you can't go on without pausing to calculate your deductions. While you are adding receipts, the phone rings. You must complete the phone call, then complete the addition, and finally continue filling in the form.

Because the 8088 must react to the real world, it too must process interrupts. Its interrupt mechanism must let the processor save what it is doing, tend to the important process that generated the interrupt, and then resume its original task. Interrupts can be generated by hardware devices, such as keyboards and modems, or by instructions in a program. In the human example above, the telephone is analogous to a hardware interrupt, and the calculation is like a software interrupt.

There can be many different reasons for interrupting the 8088. Each hardware device or interrupt instruction has its own interrupt type, which is a number from 0 to 255. This number tells the 8088 what caused the interrupt and is used to calculate what needs to be done. For every interrupt type, a program somewhere in memory tells the 8088 how to respond. Such a program is called an interrupt handler.

The 8088 presumes that there is a list of addresses beginning at memory address 0. Each address in the list is 4 bytes long. The first 2 bytes of each address are used as an offset, and the second 2 as a segment. This list of addresses is called the interrupt-vector table. The first address in the list tells the 8088 where to jump to when an interrupt of type 0 occurs, the next 4 bytes contain the address of the program that handles interrupts of type 1, and so on. These are the addresses of the interrupt handlers.

When an interrupt occurs, the 8088 pushes the current flags on the stack. This saves them so that they can be reinstated later. Then the IF (interrupt flag) and TF (trap flag) are cleared. I will discuss the significance of clearing TF later. The IF determines whether or not the 8088 will respond to interrupts. Once IF is cleared, the 8088 can't react to further interrupts until IF is set with an STI (set interrupt-enable flag) command. This ensures that an interrupt handler will not itself be interrupted prematurely. Next, the current contents of the CS (code segment) register and then the IP (instruction pointer) are pushed onto the stack. This enables the 8088 to find its way back after tending to the interruption. Finally, the 8088 uses the interrupt-type number to find the appropriate entry in the interrupt-vector table and loads the values it finds there into CS and IP.

Interrupt handlers return control to the interrupted process with an IRET (interrupt return) instruction. This instruction restores the CS, the IP, and the flags from the stack. Note that no other registers are automatically saved and restored. The interrupt handler is responsible for pushing and popping anything else that it disturbs.

Single-Step Tracing

Normally the 8088 in your PC executes instructions sequentially until it finds a loop, interrupt, return, jump, or call statement in your program. However, when the TF is set, the processor executes one instruction and then generates an INT 1. If the debugger has put its own address in the interrupt-vector table, it regains control after each instruction. Since the INT instruction clears TF after saving the flags, the debugger's instructions will not generate any further INT 1s.

The part of the debugger that handles INT 1s saves your program's display, shows you the contents of registers and memory, and processes your commands. When control is to return to your program, the debugger reinstates your program's display and executes an IRET instruction. This transfers control to your program and sets the trap flag when the flags are popped. Your program will still be in single-step mode and the process will be repeated.

An interesting problem arises when you consider how a debugger can begin single-stepping. If it transfers control to your program before setting the trap flag, single-stepping will not occur. But once it sets the flag, it will single-step its own next instruction before control can go to your program. What you need is a single instruction that both transfers control and sets flags.

The IRET op code solves the problem. If the debugger is passing control to the program for the first time, it can push the flags and the program's CS:IP onto the stack before an IRET. If the debugger has been reached by an interrupt, it can modify the flags on the stack to turn single-stepping on or off. This example shows how to begin single-step trace mode.

```
  ; point to the top of stack
MOV BP, SP
  ; get pushed flags in ax
MOV AX, SS:[BP+4]
  ; turn on single-stepping
OR AX,0000000100000000B
  ; put back modified flags
MOV SS:[BP+4], AX
  ; start tracing the program
IRET
```

The OR statement sets the trap bit and leaves the others unaffected.

BREAKPOINTS

A debugger can execute your program at full speed until it reaches a particular address. It does this by replacing the instruction at that address with an INT instruction. The debugger must put its own address at the appropriate place in the interrupt-vector table in order to regain control. The debugger must also save the instruction that was replaced so that it can restore and execute the instruction later.

Most interrupt op codes on the 8088 are 2 bytes long. One of the bytes identifies the instruction as an interrupt, and the second byte is the interrupt's type number. However, there must be at least one 1-byte interrupt instruction if breakpoints are going to be possible. Otherwise, the debugger would have to replace 2 bytes in order to set a breakpoint. This would work fine for setting breakpoints at 2- or 3-byte instructions, but it would not work for 1-byte instructions. The reason is that, if a program had a jump to the byte following the 1-byte instruction at which the breakpoint was set, the second half of the 2-byte interrupt would be executed as an op code. On the 8088, INT 3 is a special interrupt that generates a 1-byte instruction and is the instruction inserted by debuggers to implement breakpoints.

The program segment in listing A shows why it is necessary to have a 1-byte interrupt. If a debugger used a 2-byte interrupt (say INT 50, which is CD50 in machine language) to set a breakpoint at address CS:0007, then the bytes at address CS:0007 and CS:0008 would be replaced with CD and 50, respectively. The result is shown in listing B. This program will crash because the jump at position CS:0003 will transfer control to the middle of the breakpoint interrupt and execute the instruction at CS:0008. The instruction at CS:0008, 50 hexadecimal, is the machine code for a PUSH AX. The next instruction executed would be E074. E074 will cause the program to decrement CX and then jump forward 74 bytes if the zero flag is not set. But if the zero flag is set, the next instruction will be fetched. This is FB, which is an STI instruction. At this point, the debugger is executing nonsense rather than the program you were trying to debug. Even if execution proceeds normally from CONTINUE, at the very least the stack and CX are ruined.

Some debuggers let you set many breakpoints at once. In this case, the debugger's INT 3 handler can examine the stack to figure out which breakpoint was reached.

In summary, a simple debugger consists of a mainline program and two resident interrupt handlers. A debugger can also include an INT 2 handler. This allows recovery from crashed systems by using a hardware accessory that generates INT 2s.

The mainline program must place the addresses of the INT 1 and INT 3 handlers into the interrupt-vector table; call SETBLOCK (DOS function call 4A hexadecimal) to free memory for the program to be debugged; build an ASCIIZ string (string terminated by zero), file control blocks, and a parameter block for an EXEC call (DOS function 4B hexadecimal); and use the EXEC function call to start the program to be debugged.

The two resident interrupt handlers should display registers and process debugger commands, such as turning tracing on or off and setting or clearing breakpoints. This simple debugger model presumes that an INT 3 has been placed at the beginning of the program being debugged.

Listing A: *Here is a program segment that illustrates an instance where using a 2-byte interrupt instruction to set a breakpoint at a 1-byte instruction would cause the program to crash.*

```
CS:0000   A0B917   MOV  AX, DATA  ; address of DATA is 17B9 hexadecimal
  0003    EB05     JMP  COMPARE
LOOP_1:
  0005    01C3     ADD  BX, AX
  0007    48       DEC  AX
COMPARE:
  0008    38E0     CMP  AL,AH
  000A    74FB     JE   LOOP_1
CONTINUE:
  000C
```

Listing B: *If you use the 2-byte interrupt instruction* INT 50 *to set a breakpoint at address* CS:0007, *the jump at position* CS:0003 *will transfer control to the middle of the breakpoint interrupt and begin executing nonsense rather than the program you are debugging.*

```
CS:0000   A0B917   MOV  AX, DATA  ; address of DATA is 17B9 hexadecimal
  0003    EB05     JMP  COMPARE
LOOP_1:
  0005    01C3     ADD  BX, AX
  0007    CD                      ; first byte of INT 50
                                  ; If reached, control returns
                                  ; to the debugger.
COMPARE:
  0008    50       PUSH AX        ; second byte of INT 50
  0009    E074     LOOPNZ +74     ; What's left of CMP AL, AH
                                  ; combined with beginning
                                  ; of JE LOOP1.
  000B    FB       STI            ; second byte of JE LOOP1
CONTINUE:
  000C
```

temporary breakpoints that are cleared when program execution halts on a breakpoint, but also provide for fixed breakpoints that remain set until program execution ends or you specifically clear them. Advanced Trace86 sets temporary breakpoints with the GO command and uses a separate command to set up to 10 fixed breakpoints. In Phoenix Software's Pfix86, you can scroll through the text of code and mark instructions as temporary breakpoints. Fixed breakpoints are handled with a different command at the menu level.

A good number of debuggers let you enable and disable breakpoints without clearing them. This facility is even more useful if you can enable and disable the breakpoints in sets. For example, one set of breakpoints might be turned on for following one subroutine, then turned off when you are debugging another area of the program. Advanced Trace86 and CodeSmith-86, in particular, are quite thorough in their support of this facility.

Setting a breakpoint at an address is nice, but the ability to stop execution when a given condition is met allows you much greater control. For example, it might be useful to see exactly when one of the registers reaches a certain value or find out when a memory location is changed. Pfix86 is outstanding in this area. It lets you perform quite complex tests using 22 operators (arithmetic, logical, and bitwise) in the expression. Another debugger that is extremely flexible in this regard is Advanced Trace86, where you can define breakpoint conditions that look like assembly-language instructions (e.g., CMP DI,600).

You might not always want to stop execution the first time an instruction is reached, but on the third, twelfth, or six-hundredth time through a loop. Several debuggers on the market can break on a given iteration, including SYMDEB, Advanced Trace86, Pfix86, and CodeSmith-86. With CodeSmith-86, you can also set a counter to see the number of times a given instruction is executed.

Some debuggers let you define a specific action to be taken at the breakpoint. Pfix86 allows a choice of four possible actions when either an address or a conditional breakpoint is reached. You may elect to stop execution, call a subroutine, or enable/disable a subsequent breakpoint.

BACKTRACING

What do you do when you are tracing through a program and would like to back up to see how you got to the present point? One option is to echo the output from your debugging session to a printer so that you can study previous screen displays at any time. Another possibility is to use a debugger that saves the state of the registers after each instruction and lets you read it back at a later time. Reading back through previous states of the machine to see how you got where you are is called backtracing.

Several debuggers offer backtracing capability, and each one approaches the problem differently. IBM's Professional Debug Facility saves its display in a buffer after each instruction. Scrolling through this buffer lets you see what the debugger displayed after each of the last 2340 instructions. SYMD has a command that lets you see what the state of the machine was after each of the last 255 instructions.

A breakthrough in backtracing is the UNDO function of Advanced Trace86. This command does more than save a display of the machine's prior state; it actually steps backward through the code. You can run the code in reverse, one instruction at a time, for up to 20 instructions. There are some limitations, since certain types of instructions cannot be undone (e.g., interrupts or writing to disk). However, in many situations this backtracing capability is an invaluable tool.

MODIFYING CODE

If you want to modify the program you are debugging, you have to enter the machine-language equivalent of the instructions you want. Many debuggers have an in-line assembly feature where you enter new instructions in their mnemonic form, then the debugger translates them to machine language and overwrites old code with the new instructions. In-line assembly lets you concentrate on the problem at hand (finding the bugs) rather than figure out the machine code of the instructions to be inserted.

Code insertion (inserting new code between two instructions of existing code) is another breakthrough in debugging. Embedding code in a program can be a problem because once the new code is inserted, address references in existing code might no longer point to the correct locations in memory. Advanced Trace86 provides many ways to do code insertion in its trace and disassemble modes. Once new instructions are embedded in the program, the debugger adjusts any affected code. You can also delete existing code. However, both of these features are restricted to .COM files.

CodeSmith-86 takes a different approach. The source code is inserted in sequence on the screen (where you can write it out later to disk for reassembly) but is not really embedded in the program. The new machine code is placed elsewhere in memory and reached by an INT 3. The display shows the true CS:IP address of the "inserted" code on the screen. This is for temporary patches only, as access to the new code is quite slow and it might not perform properly in a time-dependent program.

PERFORMANCE ANALYSIS

For programmers who lose sleep wondering if they have chosen the best algorithm or fastest code, the program performance analyzer is a nice feature available with several debuggers. PC Probe, PDT-PC, and SYMD all have performance analyzers. You set up partitions in your program and can then get several kinds of outputs that show where time is being spent and where you should be optimizing.

Screen displays show the percentage of execution time that is spent in each partition. PC Probe and PDT-PC come with hardware that lets you time execution (in milliseconds) within a range. Although these two hardware models can retrieve information at close to full speed, they are expensive, and performance analysis is an add-on option in PC Probe. PDT-PC and SYMD include analyzers with their debuggers.

DEBUGGER ISOLATION

An ideal debugger should have complete independence from the program being tested. The debugger should not interfere with the program's execution nor should the program's unanticipated behavior cause the debugger to crash. If the program were to enter an infinite loop, for instance, the debugger should be able to regain control rather than force you to reboot. The feature of not affecting or being affected by the program being debugged is referred to as "debugger isolation."

The debugger-isolation problem can be solved partially with software. Some debuggers resolve screen conflicts by letting you toggle between program and debugger output screens. Microsoft's SYMDEB lets you access an ASCII terminal via a communication (COM) port in order to debug a program without disturbing its screen activity.

Sharing memory is a more delicate problem. Normally the debugger resides in RAM (random-access read/write memory) ahead of the program to be tested. But a runaway program can overwrite the debugger. This problem led to the development of debuggers that use add-on boards with RAM that can be protected. Debuggers such as Periscope (Data Base Decisions) and PC Probe use add-on hardware that aids isolation in two ways. Once you load the debugger into RAM, the RAM is write-protected and cannot be overwritten. They also provide a button that you can push to generate an NMI (nonmaskable interrupt), letting you recover from the hung system and similar disasters without rebooting the system. But even the NMI switch won't work if the interrupt-vector table is overwritten.

Debuggers that provide software plus a hardware (NMI switch or protected RAM) add-on board can best be categorized as hardware-assisted debuggers. PC Probe and PDT-PC are hardware debuggers. They include a 40-pin socket that you insert between the 8088 and the socket in the motherboard. Now the debugger has the power to monitor the PC's hardware. You have the power to set breakpoints in ROM (read-only memory), break on a read or write from RAM, break on a read or write from an I/O port, and even monitor DMA (direct memory access) lines. Although these units are more expensive than software debuggers, they give you new power for system monitoring. Since PC Probe can redirect input and output to a COM port, it is

(continued)

ITEMS DISCUSSED

ADVANCED TRACE86
Morgan Computing Co., Inc.
POB 112730
Carrollton, TX 75011
(214) 245-4763

CODESMITH-86
Visual Age
642 North Larchmont Blvd.
Los Angeles, CA 90004
(213) 439-2414

DEBUG (included with MS-DOS)
SYMDEB
(included with Macro Assembler version 3.0)
Microsoft Inc.
10700 Northup Way
Bellevue, WA 98004
(206) 828-8080

I²ICE
Intel Corp.
3065 Bowers Ave.
Santa Clara, CA 95051
(408) 987-8080

IN-CIRCUIT EMULATOR
Microcosm Inc.
14355 Southwest Allen
Beaverton, OR 97005
(503) 626-6100

PC PROBE
SOFTWARE SOURCE PROBE
SOURCE PROBE
Atron Corp.
20665 Fourth St.
Saratoga, CA 95070
(408) 741-5900

PDT-PC
Answer Software Corp.
20863 Stevens Creek Blvd.
Cupertino, CA 95014
(408) 253-7515

PERISCOPE
Data Base Decisions
14 Bonnie Lane
Atlanta, GA 30328
(404) 256-3860

PFIX86
PFIX86 PLUS
PLINK86
Phoenix Software Associates Ltd.
Suite 101
1420 Providence Highway
Norwood, MA 02062
(617) 769-7020

PROFESSIONAL DEBUG FACILITY
IBM Corp.
Entry Systems Division
POB 1328
Boca Raton, FL 33432
(305) 998-2000

RTCS/UDI
Real-Time Computer Science Corp.
1390 Flynn Rd.
Camarillo, CA 93010
(805) 987-9781

SOFT-SCOPE
Concurrent Sciences Inc.
POB 9666
Moscow, ID 83843
(208) 882-0445

SYMD
D & V Systems
22 Fox Den Rd.
Hollis, NH 03049
(603) 465-7857

even more isolated from system hazards.

One way to ensure total debugger isolation is to give the debugger its own hardware: a CPU (central processing unit), an input device, an output device, and its own memory. The in-circuit emulator (ICE) is hardware that is independent of the PC. It requires a second computer to run and can fully monitor processes going on in the machine being debugged. Microcosm's In-Circuit Emulator and Intel's I^{2}ICE are two such debuggers. They are several times more expensive than the two previously mentioned hardware debuggers and so are appropriate only for a very small group of developers.

SOURCE-LEVEL DEBUGGING

The concept of source-level debugging is already in use in interpreters. When using an interpreted high-level language, such as Microsoft BASIC, you can stop execution of a program, monitor progress through a program, view data, and make changes to the program without waiting for compiling and linking. In languages that are more commonly compiled, such as C, Pascal, and FORTRAN, this is not possible. The debugging process can be tedious and extremely aggravating. Recently, several source-level debuggers have appeared on the market. Some basic features found in most of these debuggers include symbolic referencing of variables, displaying lines of source code, single-stepping through source code, and setting breakpoints in the code. Single-stepping through source code prevents you from spending most of your time deciphering assembly language in an effort to determine which line of source code you are on.

A useful feature is the ability to view data while stepping through a program. Many symbolic debuggers allow referencing a data item by symbol rather than memory address. The problem with this feature is that the debugger does not know the data type of the memory location. One source-level debugger, Soft-Scope from Concurrent Sciences Inc., gets this information from the object code, which must be in Intel's Object Module Format (OMF). It is easy to ask the debugger what data type a symbol represents, regardless of how complex this type might be. For example, if employee_record is a memory location that points to a data record, figure 1 shows what the debugger displays if you ask what data type employee_record is.

A drawback of Soft-Scope is that on the IBM PC it runs only under the RTCS/UDI (Real-Time Computer Science Universal Development Interface) shell. [*Editor's note: We have been informed that a version of Soft-Scope that will run directly under* MS-DOS *is under development.*] It also works only with languages that use the Intel object-code format, which are slightly more expensive than MS-DOS languages. Information regarding the data types of variables can be found in object modules using the true Intel OMF. Microsoft object files are a subset of the Intel OMF and do not contain this information. Maybe someday MS-DOS compilers will have a switch that lets you produce Intel object files or a separate file that contains this information.

Atron's Source Probe, which needs the PC Probe hardware, and Software Source Probe are source-level debuggers that run under MS-DOS. Although they don't have access to data-type information as Soft-Scope does, they support macros that allow viewing data in a user-defined format. The Source Probe and Software Source Probe macro language even includes conditional and looping commands for complex macro design.

If you have ever debugged a subroutine and wished you could see the calling sequence by which it was reached, you will appreciate the ability to trace that path back to the top level of a program. It is found in both Source Probe products and in Soft-Scope. After tracing down into a subroutine with Source Probe, you can use the NEST command to see the path that got you there. Source Probe will then display the module name and line number where each call was made. Soft-Scope will give that information along with the depth of each call in the program and, optionally, expand this display to show the line of code that did the calling.

```
* type .employee_record -
record
NAME.................................array [15] of char
ADDRESS..............................record
  STREET ............................array [25] of char
  CITY...............................array [15] of char
  STATE..............................array [2] of char
  ZIP_CODE ..........................longint
SEX..................................enumerated (MALE, FEMALE)
SALARY ..............................real
AGE .................................integer
```

Figure 1: *Soft-Scope is a symbolic debugger that allows referencing a data item by a symbol rather than a memory address. It can tell you what type of data a symbol represents, as shown in this sample printout.*

CONCLUSION

After seeing these debuggers in action, we realize that the designers of each debugger had a specific problem to address. Some programs were designed to be an enhancement to Debug at a minimal cost, while others were designed to be absolutely bombproof, no matter what the cost.

If you are looking for a debugger, decide what features are important to you. Remember that spending a little more on a debugger might pay for itself in time saved with the first problem it solves. And before you make your investment, ask a dealer what new features have become available. ■

Introducing!
Unsurpassed Surge Suppression...

...only from GTE.

Finally. AC power protection that offers the best technology. And from GTE . . . a name you can trust.

The SmoothLine™. A surge protector with a strong heritage and known performance. Fully proven on major telephone installations and other sophisticated communications systems around the world.

Flexibility and Convenience? You bet. SmoothLine has a six-foot extension cord with four grounded and switched outlets. Plus, there's a lighted on/off rocker switch for circuit status and control. And, for added protection, a built-in 15 Amp circuit breaker.

Ultra-fast response time is assured with SmoothLine. Even for voltage surges as short as one nanosecond. And, for maximum protection against EMI, RFI and other common interference, there's multi-stage solid-state circuitry.

So relax. Use SmoothLine and get maximum protection from lightning storms, power fluctuations, power surges or voltage transients. Your computers and other electronic equipment will be glad you did.

Call GTE sales today at (800) 392-3908 or (207) 642-4535 for your nearest dealer.

Ask for the SmoothLine at your local computer dealer.

SYLVANIA Control Devices

GTE

Inquiry 76

IBM COMPATIBILITY ISSUES

Making life difficult for independent software developers

BY MARK DAHMKE

OF LATE, IBM has seemed to adopt a "let's keep changing it" philosophy. This philosophy can be one of the most powerful tools a large company has to keep its customers from purchasing third-party plug-compatible computers, add-on boards, and software, but this same philosophy can force a giant like IBM into a difficult position.

I refer mainly to the IBM PC AT and IBM's TopView. Because of a poor operating-system design and little concern on the part of software authors for writing well-behaved applications programs (see the text box "The Levels of Software Behavior" on page 188), both of these products have features that make life miserable for anyone trying to write programs that will run correctly on the entire line of IBM Personal Computers and compatibles.

TopView is an add-on operating-system product that turns an IBM PC into a multitasking system capable of supporting windows and concurrency. Because of poor planning back in the days of PC-DOS 1.1, it has several major performance problems.

First, device ownership is rarely checked by off-the-shelf software. This means that programs are written with the assumption that all hardware devices are owned implicitly by the program and are at its disposal at all times. Some application programs even reprogram the serial ports, counter-timer chips, DMA (direct memory access) channels, and video-display adapters without bothering to see if any other program (or the operating system) is using them.

A large number of the most popular application packages read and write characters directly to or from the video-display buffer. In bypassing the BIOS (basic input/output system) in the ROM (read-only memory), the program gains speed but severely limits the ability of the operating system to retain control of all physical hardware devices.

BASICA, the Microsoft BASIC interpreter, reprograms the serial ports because the BIOS doesn't support all the error-checking and time-out functions the language requires. This means that if another program is using the serial ports, or if one or both serial ports are in use as multiuser consoles, BASICA can crash the system.

In terms of hardware, IBM has stated that it will maintain compatibility with the following features in future products:

1. The sound channel of the counter-timer. Channels 0 and 2 control the time-of-day interrupt and the speaker

(continued)

Mark Dahmke, a contributing editor for BYTE, *owns a computer consulting business in Lincoln, Nebraska. His interests include operating systems, computer graphics, videodiscs, and authoring languages. His new book* Introducing Concurrent PC DOS *has just been published by* McGraw-Hill Book *Company. He can be reached c/o* BYTE, POB 372, *Hancock*, NH 03449.

ILLUSTRATED BY K. MERCEDES McDONALD

and cassette ports.
2. The game-control adapter, port 201 (hexadecimal).
3. The interrupt-mask register.
4. Absolute addresses of the video-display buffers for the color- and monochrome-adapter cards.
5. The ROM BIOS data area.

What this means is that IBM will, in all future products, support these absolute hardware memory and I/O (input/output) port addresses.

WINDOWS

TopView and Digital Research's Concurrent PC DOS and GEM (Graphics Environment Manager) are the three main contenders vying to become the standard in window technology. All three must deal with the problems created by programs that directly access the video buffer. IBM gets around this problem by adding a software interrupt vector that returns the address of the video buffer for the window the application is running in. This lets the application modify the buffer as if it were the real video-display card and then call another interrupt vector to update the video display from it. If this procedure is not followed, a program may rewrite the screen or, even worse, reprogram the display (such as switching from 80-column by 25-row text mode to medium-resolution graphics mode) without telling the operating system about it.

SOLUTIONS

IBM's choice of PC-/MS-DOS 1.1 and its original BIOS have practically eliminated any hope of standardization and have made it very difficult to write well-behaved programs. (It is interesting to note that choosing TopView as a multitasking/windows add-on to PC-DOS doesn't necessarily give you more compatibility than buying Concurrent PC DOS. In fact, in my experience, Concurrent PC DOS may be the better choice.)

Several things should have been done (and could still be done) to solve these problems and make the IBM standard a viable one (as viable as the S-100 or the Apple II) for the next 10 years. First, there should be an easy way to do equipment determination through the DOS (disk operating system) or BIOS calls. There is no excuse for writing programs that don't look to see how many disk drives there are and whether there is a B: drive installed. Similarly, well-behaved programs should at least try to check out device ownership. For example, suppose you load a program that installs itself into the system like the PRINT command or Borland's SideKick. The program replaces interrupt vectors and reprograms a counter-timer channel for its own purposes. You then load a second application program that also tries to reprogram the counter-timer channel and modify interrupt vectors. The result is a crashed system and no clues as to why.

Copy-protection schemes are notorious for reprogramming the hardware, especially the disk-controller board. IBM points out in its documentation that certain schemes may not work on the PC AT with its new high-capacity disk drives. Either copy-protection schemes should be abandoned, or the BIOS should have a "return serial number" interrupt call (I prefer the former). Whatever method is used, the copy-protection scheme shouldn't reprogram the hardware without telling the operating

(continued)

THE LEVELS OF SOFTWARE BEHAVIOR

Level 1: Ill-behaved programs
Programs at this level override all or most operating-system (DOS and BIOS) services and deal directly with the hardware. They do not check device ownership, and they do not perform equipment determination.

Level 2: Somewhat well-behaved programs
At this level programs use the disk I/O services of the DOS but modify the contents of the video-display buffer directly to improve performance. They do not check device ownership or perform equipment determination.

Level 3: Moderately well-behaved programs
These programs use DOS and BIOS function calls for most services but still deal directly with the hardware in some cases (e.g., video buffer, counter-timer chips). They do check ownership first where possible.

Level 4: Well-behaved programs
These programs check ownership of all devices and use system calls for all functions including writing to and reading from the video buffer.

REQUESTS TO OPERATING-SYSTEM AND BIOS DESIGNERS

Please supply function calls to determine what equipment is installed on the computer. Also supply a function to request the serial number of the computer or at least the serial number of the DOS.

Provide function calls to formally open and close all hardware devices and to check for ownership of the device. Treat counter-timer chips and DMA controllers as devices with their own device names.

Please include function calls to perform hardware speed-independent timing loops so the program doesn't have to reprogram a counter-timer to accomplish this.

Save

- Low Low Prices
- No Surcharge for MasterCard and VISA

1-800-231-9842
WESTERN REGIONAL ORDERS

1-800-237-4048
EASTERN REGIONAL ORDERS

• COMPUTERS •

IBM ENHANCED AT	5355
IBM ENTRY LEVEL AT	3298
IBM PC, 1 DSDD DR, 256K	1501
IBM PC, 2 DSDD DR, 256K	1616
IBM PC, NO DRIVES, 64K	1293
IBM PC, 2 HALF/HYTES, 256K	1631
IBM XT, 1DSDD DR/10MB, 256K	3355
IBM XT, 1DSDD DR/ TANDON HD, 256K	2491
COMPAQ, 2 DSDD, 256K	2153
COMPAQ PORTABLE 286 MODEL 2	5292
COMPAQ DESK PRO MODEL 3	3304
COMPAQ DESKPRO MODEL 2	2156

• MONITORS •

AMDEK 300G	116
AMDEK 300A	132
AMDEK 310A	140
AMDEK 600 RGB COLOR	409
AMDEK 710 SUPER HI RES COLOR	447
IBM COLOR	616
IBM MONOCHROME	235
PRINCETON MAX-12 E	162
PRINCETON HX-12 RGB COLOR	446
PRINCETON SR-12	575
PRINCETON HX-12E COLOR	510
QUADRAM AMBERCHROME	150
QUADRAM QUADCHROME II	426

• DISPLAY CARDS •

STB SUPER HI RES 400	402
HERCULES COLOR	145
IBM MONOCHROME	215
IBM COLOR	226
EVEREX GRAPHICS EDGE	328
GENOA SPECTRUM	289
TECMAR GRAPHICS MASTER	459
PARADISE MODULAR GRAPHICS	261

• BOARDS •

AST SIX PACK PLUS, 64K	223
AST MEGAPLUS, 64K	281
AST MP-2, 64K	228
AST ADVANTAGE, 128K	384
AST I/O PLUS	122
JRAM-2	169
QUADRAM QUADBOARD ØK	186
QUADSPRINT	418
TECMAR CAPTAIN, ØK	162
TECMAR FIRST MATE, ØK	200
TECMAR WAVE XT, 64K	187
TECMAR JR CAPTAIN, 128K	295

• MODEMS •

HAYES 2400	627
HAYES 300	138
HAYES 300/1200	407
HAYES 1200B w/ SOFTWARE	380
HAYES 1200B	351
VEN-TEL HALF CARD	374
VEN-TEL 300/1200 INTERNAL	341
AST REACH HALF CARD 1200	362

• PRINTERS •

OKIDATA 83A, WIDE CARRIAGE	545
OKIDATA 84P, WIDE CARRIAGE	660
OKIDATA 92P	392
OKIDATA 93P, WIDE CARRIAGE	597
OKIDATA 2410P	1782
OKIDATA 182	215
OKIDATA 192	363
OKIDATA 182S	259
OKIDATA 184	660
EPSON FX-80, TRACTOR	371
EPSON LX-80	248
EPSON FX-185, WIDE CARRIAGE	517
EPSON FX-100, WIDE CARRIAGE	531
EPSON JX-80 COLOR PRINTER	531
EPSON LQ-1500 PARALLEL	1002
NEC 2050	682
NEC 3550	1045
NEC 8850	1458
JUKI 6100	362
JUKI 6300	695

• ACCESSORIES •

OKIDATA 82/92 TRACTOR	44
NEC BI-DIRECTIONAL TRACTOR	151
NEC CUT SHEET GUIDE	63
JUKI 6100 BI-DIRECTIONAL TRACTOR	105
JUKI 6300 BI-DIRECTIONAL TRACTOR	124
MEMORY, 64K CHIPS	8
MEMORY, 256K CHIPS	39
PRINTER CABLE	10
MICROFAZER BUFFER,8K	131

COMPUTER MART

1901 S. TAMIAMI TRAIL, VENICE, FL 33595

2820 W. CHARLESTON, LAS VEGAS, NV 89102

Call for non-advertised and lower current prices. IBM is a registered Trademark of International Business Machines. Prices subject to change. We accept all major credit cards, money orders, and personal checks. 2% surcharge for shipping, handling and COD shipments, and purchase orders $100 minimum. Returns are subject to a 15% restocking fee. All sales final.

IN FLORIDA CALL 813-493-2736
IN NEVADA CALL 702-877-3988

SUGGESTIONS TO SOFTWARE DEVELOPERS

If you have to write directly to the screen buffer, try to do some equipment determination and check to see if you are operating in a windowed or concurrent environment.

If you must write to the screen buffer, don't change the video mode (e.g., from medium-resolution graphics to text) by writing to the video-controller chip. Use a BIOS call for this function.

Allow files to be assigned to physical disk drives. Don't assume that everyone has two floppy-disk drives.

Don't use undocumented 8088 instructions in programs. The 80186 and 80286 probably won't support them, and any redesign to the 8088 itself could change or eliminate undocumented machine instructions.

Try not to use the scan-code values returned by the keyboard, as they may not be supported in future products.

Don't write code that loads at absolute memory addresses.

Allow print files to be sent to disk instead of directly to the printer. Future operating systems will probably provide print-spooler function calls that allow your program to write a print file to disk and issue a spool command to send it to the physical printer.

system about it first. (See the text box "Requests to Operating-System and BIOS Designers" on page 188.)

All software developers inevitably want to squeeze as much processing speed out of a microcomputer as possible. The old cliché is that programs expand to use the available resources. Because of inherent limitations in the hardware and the operating systems of current microcomputers, it is sometimes necessary to reprogram the hardware to gain maximum efficiency. (See the text box "Suggestions to Software Developers" at left.) Regardless of what programming methods software developers use, it is in their best interest to at least allocate devices through the operating system and avoid device contention. This technique will save them a great deal of recoding when new hardware and software products are introduced by the manufacturers. ■

The Project Scheduler NETWORK™

As a manager, you know the key to a successful project is good planning. You can't afford to overlook critical costs and deadlines.

COMPREHENSIVE PROJECT MANAGEMENT

The Project Scheduler Network will help you meet deadlines and beat cost targets using effective project scheduling, resource planning, costing, and status tracking.

Advanced screen graphics. Produce Network diagrams, Gantt charts, and color graphs to provide the concise information you need for timely decision making.

Work Breakdown Structure (WBS). Organize project activities to produce summary level schedules and reports. Multiple projects can be integrated to present project level schedules and total resource utilization.

Baseline schedules. Save them to compare with current schedules for schedule tracking. Actual, original plan, and current plan costs can be graphically compared for cost tracking.

Comprehensive report generator. Simplify communications with management by synthesizing project information into presentation-quality hardcopy reports and graphs.

SCHEDULE WITH THE CLICK OF A BUTTON

The Project Scheduler Network provides the most advanced software technology available. Start with our mouse-controlled menus to select spreadsheet-style input templates to enter your project data.

Graphically build your dependency Network with only a "click of a button." Zoom to view the Network at selectable resolution levels. Now use the mouse to edit the Network to your satisfaction.

Select a Gantt chart and obtain schedules at any WBS level with a single click. Window resource graphs with the schedule to analyze resource allocations. And finally, select our integrated graphics to view color histograms, bar charts, and line plots of project resources and costs.

FEATURES

- Critical Path Method
- Network Analysis
- Gantt Chart
- Work Breakdown Structure (subtasks)
- Multiple Project Integration
- Tracking with Baseline Schedule and Complete Status
- Resource Scheduling
- Financial Planning and Tracking
- Presentation-quality Graphics

REQUIREMENTS

- IBM® PC, XT, or AT
- PC-DOS 2.0, 2.1, or 3.0
- Monitor plus IBM graphics adaptor
- IBM or Epson graphics printer
- 320K RAM (minimum)
- Mouse (Microsoft compatible)
- Two disk drives (one double-sided)
- IBM or HP plotter optional

*Project Scheduler Network is a trademark of Scitor Corporation.
*IBM is a registered trademark of International Business Machines Corporation

(415) 570-7700

SCITOR

SCITOR CORPORATION

250 Lincoln Centre Drive, Foster City, CA 94404

AT Powe

High performance, high capacity hard disk subsystems

A Single DOS Volume up to 240 formatted MBytes in size on your PC is only the beginning of Emerald subsystems' features. However, it is a very important one to many of our customers.

Most serious micro computer users are familiar with the infamous *32 MByte DOS Barrier*. Some of you have only read about it; others have run up against it head on when trying to run PC Focus or a downloaded mainframe program.

If you are one of those who have hit it head on you'll be pleased to know that *for almost a year Emerald has been shipping fixed disk subsystems that solve your problem*. Subsystems. Plural.

All Emerald subsystems can be operated as a single volume up to their formatted capacity. That includes our 36, 50, 70 and 140 MByte subsystems, in addition to the 280.

```
C>chkdsk

237633536 bytes total disk space
    65536 bytes in 2 hidden files
    49152 bytes in 2 user files
237518848 bytes available on disk

   262144 bytes total memory
   205152 bytes free
```

Actual printout of CHKDSK on 240 MByte volume.

You Determine how many volumes exist in your PC, and what size they are. You can have as many as *24 volumes*, and make each one exactly the size it needs to be.

Set-up is menu driven and as simple as *"How many do you want"* and *"How big should this one be?"*

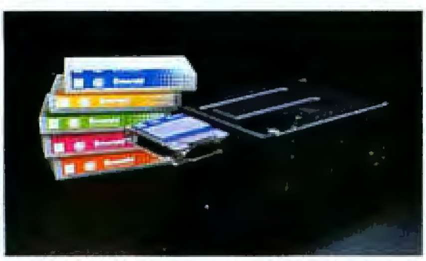
Integrate your existing hard drive into your new subsystem. Emerald's Disk-Meld technology makes it possible for your XT's 10 MByte, or AT's 20 MByte drive to become part of a single large volume. For example, if you have an AT with a 20 MByte drive and an Emerald subsystem with a 70, you can use *all* your storage as a single 90 MByte volume. Disk-Meld makes it easy to combine Emerald drives with each other or your existing drive to get the storage capacity you need.

Operating System and Network flexibility is yours for the asking. DOS 2.X, and 3.X will get you started. But, your Emerald subsystem can also support multi-user operating systems such as XENIX, VENIX, QNX and PC/IX and networks such as Novell, IBM-PC Net, Ungermann-Bass, 3 COM, X-Net, 10 Net, DNA Systems...

Additional Features include password security, 30 millisecond average access time, *automatic retreat to a safe landing zone* in the event of power failure or shutdown, and a long list of PC and AT compatible micro computers.

High performance ¼" tape backup

60 MBytes in 12 minutes is *FAST* backup, but there's more. If your files are larger than 60 MBytes, Emerald's Backup and Restore Utility (BRU) software will automatically break your file into 60 MByte sections and prompt you for a new cartridge. Of course, restoring is just as easy.



Compact tape drive fits in the AT's front panel expansion space. 60 MByte cartridges are certified for high performance and supplied with color coded labels.

Emerald subsystems were designed for the PC, AT and compatibles such as the AT&T 6300 and Compaq DeskPro.

Menu driven software makes it simple, even for novices, to backup or restore exactly what is needed, and no more. Choose one or more files that were modified after a *Specified Date and Time*, one or more *Specific Files or Directories*, or *All Files and Directories on a DOS Logical Volume*.

Restore data on a different micro if you like. Backup up your company's Emerald subsystem in Portland, Maine and ship the tape to your office in Los Angeles. Because of the defect mapping technique used on Emerald hard disks, the subsystem in L.A. will import your data error free.

Special defect mapping technique allows data to be restored on subsystems other than the original source.

The BRU software *automatically checks, and adjusts to, the defect map* of every Emerald subsystem before restoring data to it.

· is Yours.

Configured for today's Micros

The Physical Design of the Emerald subsystems lets you determine the configuration that will best serve your needs. Many of today's high performance micros have sacrificed expansion space in favor of compact size; others offer plenty of room for additional drives, tape units and expansion cards. Emerald subsystems provide you with the expansion ability you need. And, they're designed to allow you to continue to expand as your needs continue to grow. No matter which PC you have, there is an Emerald subsystem that will meet your needs.

Subsystems are available for the PC, AT and true compatibles in a variety of configurations.

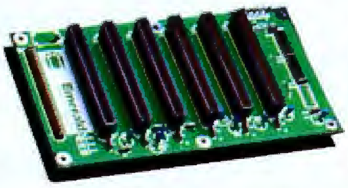
Internal Expansion is easy on the IBM AT and XT. Emerald subsystems are pre-initialized and pre-formatted—just slide the tape or hard drive you have selected into one of the existing expansion areas, plug in a couple of cables, tighten a few screws and replace the system cover. Elapsed time: 10/15 minutes.

The AT accepts 280 MBytes *or* any single drive up to 140 MBytes *and* a 1/4" tape drive. The XT accepts any Emerald hard drive up to 140 MBytes in size, *or* the 1/4" tape drive.

External and Portable expansion is easy and practical with the Emerald Portable Subsystems. These IBM color-matched subsystems have their own power supply and are available with hard drives up to 140 MBytes in capacity *or* with a 1/4" tape drive.

Ordering a tape host adapter card for each of your PCs and physically moving the 1/4" portable tape sybsystem to where it is needed will save you *thousands of dollars* over the cost of individual portables for each micro.

The portable hard drive configuration is ideal for security sensitive environments. The hard disk is password protected and the entire subsystem is small enough to be locked in a standard safe or filing cabinet.



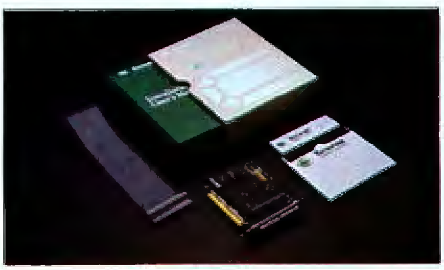
"Expansion chassis" subsystem also provides 6 additional spaces to add cards to your PC.

The 6 Expansion Slots in the Emerald expansion subsystems will be a welcome addition to many micros. If you don't have enough expansion room in your micro, or, if you've used every available slot, then one of these subsystems is just right for you. They are closely matched to the IBM PC in size and color, and have their own, built-in, power supply.

Expansion subsystems are available with drive sizes to 280 MBytes in capacity, with, or without, a built-in 1/4" tape drive. When coupled with an AT a truly powerful computer system results.

The *real* backup procedure

Standing Behind You all the way is the company that broke the 32 MByte DOS barrier.



Documentation, host adapter card, software and cables are included with each subsystem.

First, we provide everything needed to get you up-and-running quickly. Then, we back you up with specialists, to help with network implementation, applications support, XENIX and technical questions, and to provide *48 hour in-and-out repair service.*

Delivery and Warranty are important considerations. Your Emerald subsystem will be shipped within 14 days of order, via UPS, and comes with a 30 day money back guarantee and a 120 day warranty. Warranty extensions are available for 1 and 2 year periods.

Fast service and fast shipping are company policies.

Users with large database applications in factory automation, CAD/CAM, accounting, medical, R&D, Point-of-Sale and many other areas are already telling their friends about Emerald. A large percentage of them are in Fortune 500 companies, and many others are on their way to Fortune status.

If your company fits one of those categories, or, if you're a smaller company with an application you thought could only be done on a mainframe or mini, pick up the phone. Call Emerald. An Applications Engineer is standing by to answer your questions, send you literature and refer you to an installed site in your area.

Let one of our customers tell you, first hand, how Emerald helped give *his* AT Power.

Emerald
Systems Corporation

Mainframe Storage for Micros

4757 Morena Blvd.
San Diego, CA 92117
(619) 270-1994
U.S.A.
Telex: 353458 EMERSYS
Easylink: 62853804

ENIX™ Microsoft Corp.—VENIX™ Ventur Com—QNX™ Quantum Software—Novell™ Novell, Inc.—X-Net™ XCOMP, Inc.—10 Net™ Fox Research, Inc.
ilders.

HOW DOES ONE MEASURE PERFORMANCE?

. . . AT CONQUEST COMPUTER INDUSTRIES, WE MEASURE PERFORMANCE BY WHAT IS MOST IMPORTANT: QUALITY, CAPABILITY, FLEXIBILITY AND DURABILITY WITHOUT THE HIGH COST . . .

INTRODUCING CONQUEST PC AT . . . A COMPLETE AND POWERFUL COMPUTER SYSTEM

HIGHLIGHTS INCLUDE:

- User memory 640K expandable to 1 Mb RAM on board.
- INTEL 6MHZ 80286 with optional 80287 coprocessor.
- 8 I/O expansion slots.
- On board clock/calendar with battery back up.
- 1.2 Mb high capacity floppy disk drive.
- 20 Mb internal winchester disk.
- 20 Mb internal tape backup capability.
- Monochrome/graphic card (720h × 348v) with optional HI-RES monitor.
- PC AT compatible keyboard.

CONQUEST PC TURBO . . . POWERFUL COMPUTER SYSTEM

PRICE: $1225.00

FEATURES INCLUDE:

- On board 640K RAM.
- Built-in turbo with 4.7/8.0 MHhz dual clock operation.
- Parallel and serial I/O's on board (IBM standard).
- 135W power supply (110-220VAC convertible).
- Two 360K slimline drives.
- Floppy disk controller able to operate 4 drives or the tape back up internally.
- Enhanced keyboard with separate numeric pad and cursor keys.

OPTIONAL ENHANCEMENTS:

- On board memory expansion to 1 Megabyte accessable with RAM DISK.
- High speed 8MHz 8087-2 math coprocessor.
- 10-20 or 40 megabytes internal hard disk subsystem.
- Internal 10-20 megabytes tape back up.
- High density 1.2 megabyte floppy disk drive.
- Winchester/Floppy disk combination controller.
- Monochrome/graphic card (720h × 348v)

CONQUEST HARD DISK/ TAPE BACKUP SERIES

MODEL: HD/TAPE-1010 PRICE: $1250.00

- The CONQUEST HD/TAPE series subsystem provides mass storage and backup capability in a single unit.
- All necessary controller cards, cables, and software are provided for operation under DOS 2.0 or 2.1.
- It is compactly designed with a total dimension of 13.00" length × 14.00" width × 2.25" height.

Available CONQUEST HD/TAPE series

MODEL	DESCRIPTION
HD/TAPE-1010	10 MB hard disk & 10 MB tape backup
HD/TAPE-2010	20 MB hard disk & 10 MB tape backup
HD/TAPE-4010	40 MB hard disk & 10 MB tape backup
HD/TAPE-4020	40 MB hard disk & 20 MB tape backup

Other configurations are also available

These features enhance performance and ease of operation without altering the compatibility to the worlds largest software based microcomputer—the IBM PC/XT/AT.

The CONQUEST computer system supports standard industry operating systems: PC-DOS, MS-DOS, XENIX, CP/M86 and concurrent CP/M . . . plus utilizes all of the applications software such as: LOTUS 123, SYMPHONY, FRAMEWORK, dBASE III, VISICAL, WORDSTAR, MULTIPLAN, AUTO-CAD, P-CAD, FLIGHT SIMULATOR, and most of the popular IBM PC software products.

CONQUEST

COMPUTER INDUSTRIES
2630 Walnut Avenue,
Unit C,
Tustin, California 92680
(714) 838-9583

DEALER INQUIRIES WELCOME
TELEX 5106004210 Answer Back: CONQUEST

Inquiry 34 for End-Users. Inquiry 35 for DEALERS ONLY.

IBM PC, is a registered trademark of IBM Corp.

BENCHMARKING THE CLONES

BYTE *ran a variety of* IBM PCs *and work-alikes through benchmark tests*

BY JON R. EDWARDS AND GLENN HARTWIG

BENCHMARKS ARE objective, reproducible measures of performance. Engineers use them to measure precision and to identify poorly performing components or bottlenecks in a system's design. Many users, on the other hand, use benchmarks as a means to determine the speed with which specific systems perform a general class of functions.

The BYTE benchmarks were devised to exercise a computer under review by running it through a series of task-oriented tests. The benchmarks provide reproducible results, they are relatively easy to run and time, and they measure system elements upon which users commonly depend. The Sieve and Calculations tests, for example, are a measure of processor speed, while others time disk access.

Table 1 and table 2 present the results of BYTE's system benchmark tests for the IBM family of personal computers, including the IBM PC, XT, AT, and PCjr, as well as a wide range of MS-DOS work-alikes.

Use the benchmark information to give yourself an introduction to the relative capabilities of a broad selection of equipment. Just as important, do not use this information as the sole basis of your evaluations.

Before running the benchmarks, reviewers adopted a standard configuration with 256K bytes of memory, two 360K-byte disk drives, and PC-DOS 2.0 or MS-DOS 2.0. All tests were run at least three times; the reported time is the average of the trials. When configurations could not be duplicated, reviewers set up the system to be as close as possible to the standard machine. Exceptions are noted below.

We are not attempting to rate the computers competitively, nor can we provide assurances that a computer will be suitable for a specific application. Benchmarks are useful in measuring the time required for specific hardware/software combinations—especially if these combinations reflect the intended use of the system—but benchmarks are imperfect indicators. They provide little or no evidence, for example, regarding ease of use, reliability, compatibility, maintainability, and support. Jerry Houston gave some good advice ("Don't Bench Me In," February 1984 BYTE, page 160) on how to research a product: read reviews, talk with friends and fellow users, and find a reputable dealer. Still, the benchmarks do distinguish time-savers from time-enslavers.

We used readers and national users groups to help perform benchmark tests for those computers we did not have in-house; we could not obtain results for a few machines, including the MAD-1, the Pronto Transportable Solution, and the Sharp PC 5000. No manufacturers or retailers were involved in any

(continued)

Jon R. Edwards and Glenn Hartwig are BYTE *technical editors. Both may be reached at* BYTE, POB 372, *Hancock*, NH 03449.

ILLUSTRATED BY DIANA BRYAN

of the testing. We selected the computers that are listed in table 2 on the basis of figures from Future Computing (8111 LBJ Freeway, Dallas, TX 75251) on the installed base of the computers through the end of 1984.

The Benchmark Tests

The first four tests to follow are BASIC tests and are written in Microsoft BASIC. "Compatibles" were tested with bundled or recommended versions of GW-BASIC. Certain BASIC interpreters and compilers could not run the programs exactly as written. In those cases, the programs were modified slightly, but with the "spirit" of the test preserved. Disk writing and reading were performed with a blank formatted disk. The last two benchmarks test the most commonly used system functions.

Since June 1984, every system reviewer for BYTE has performed the following benchmark tests.

Writing to Disk

The Writing to Disk test measures how long it takes the system/interpreter to

(continued)

Table 1: *Comparative benchmark results.*

Computer Name	BASIC TESTS Writing to Disk	Reading from Disk	Sieve of Eratosthenes	Calculations	Format and Copy Disk	File Copy
AT&T PC 6300	32	30	87	27	11	10
Canon A-200	57	29	132	41	11	13
Columbia	31	30	194	59	10	8.8
Compaq Deskpro	57	53	186	59	12	8.4
Compaq Plus	54	51	168	56		7.8
Corona Desktop PC	57	55	201	61	17	11
Data General/One	56	55	229	69	7.8	12
Epson QX-16	58	30	179	54	12	6.9
Ericsson PC	57	31	182	56	12	9.3
Hewlett-Packard HP 150	35	34	148	49		11
Hewlett-Packard HP 110	42	28	114	38		
IBM PC	56	46	191	69	9.6	5.8
IBM PC AT	26	24	80	27		3.9
IBM PCjr	82	55	236	85		8.5
IBM PC XT	59	41	209	70		5.1
ITT XTRA	33	32	185	56	11	8.8
Kaypro 16	56	30	184	56	11	7.3
Leading Edge PC	32	29	153	46	13	9.2
Micromint MPX-16	58	54	216	72	7.7	9.4
Mindset Personal Computer	58	55	301	54	12	12
Morrow Pivot	82	56	313	96	14	11
NCR Plus 4	57	30	182	56	25	13
NEC APC III	30	29	86	29	4.0	6.5
Osborne 3	59	56	273	83	14	15
Otrona Attache	31	30	78	24	11	10
Panasonic Sr. Partner	30	29	184	56	11	5.8
Polo	31	56	448	72	17	9.9
Sanyo MBC-775	31	29	113	35	4.1	8.0
Sanyo MBC-550	32	29	267	93	13	7.7
Scottsdale Systems Inc. Color Fox	59	56	241	73	11	11
Seequa Chameleon Plus	32	29	215	65	24	9.6
Stearns	31	29	76	24	7.2	6.2
STM PC	32	30	79	24	12	7.7
Tandy 1000	56	55	226	68	12	7.9
Tandy 1200 HD	59	55	223	69		7.3
Tandy Model 2000	30	29	79	24	10	9.8
TeleVideo TS 1605	60	57	184	56	15	11
Texas Instruments Professional	31	31	171	52	8.8	9.5
Texas Instruments Pro-Lite	34	33	155	51		11
Visual Commuter	57	45	182	56	9.3	7.0
Zenith Z-150	32	30	193	57	10	8.4

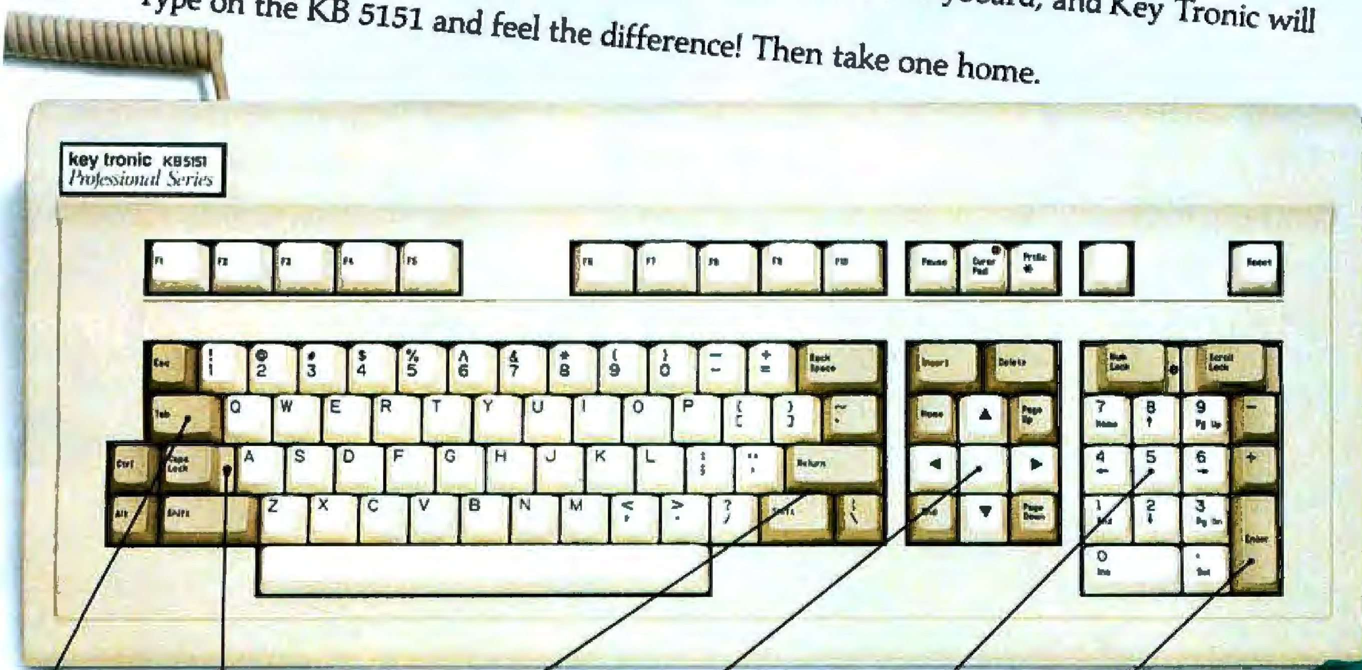
"HURRY! OFFER ENDS DECEMBER 31, 1985."

Key Tronic is Offering $50 for Your New IBM* PC or XT* Keyboard.

The reason why Key Tronic is having a special $50 trade-in offer is because we want you to get the most from your IBM PC or XT!

Buy a KB 5151 and send us your unused IBM PC or XT keyboard, and Key Tronic will send you $50!

Type on the KB 5151 and feel the difference! Then take one home.



Easy to Read
Regular Key Legends Rather than Symbols.

Easy to Identify
Status Lights on "Lock" Keys.

Easy to Use
"Shift" & "Return" Keys are in familiar Typewriter locations.

Easy to Edit
Separate Cursor and Editing Keys.

Easy for Numbers
Separate Numeric Pad is Great for Spreadsheets.

Easy to Enter
Convenient Enter Key.

Key Tronic is the world's largest ndependent manufacturer of full ravel keyboards, and offers a omplete line of integrated input evices.

Take advantage of this limited me offer. Contact your local ealer or call 1-800-262-6006 today!

uiry 103

M and XT are registered trademarks of IBM Corporation

"Feel the Key Tronic Touch"

key tronic
The Responsive Input Company
Spokane, Washington
509-928-8000

"HURRY! OFFER ENDS DECEMBER 31, 1985."

Key Tronic Corporation
Rebate Program
Building 14
Spokane Indus. Park
Spokane, WA
99216

Name ____
Address ____
City/State/Zip ____
Phone ____

To the best of my knowledge the above information is correct and the enclosed keyboard is an unused IBM PC or XT keyboard. I am aware that the conditions for receiving a $50.00 personal rebate check for the IBM PC or XT keyboard include: replacing each such IBM PC or XT keyboard with a Key Tronic keyboard and including a copy of the sales receipt when mailing the keyboard to Key Tronic. I understand that credit will only be given for those keyboards which are new and pass a functional inspection by Key Tronic or its assignee. IBM* PC or XT* keyboards must be shipped in the original boxes with all packaging and documentation intact. Offer good for any KB5151 keyboard purchased after January 1, 1985. *Please allow 4 to 6 weeks for delivery of rebate

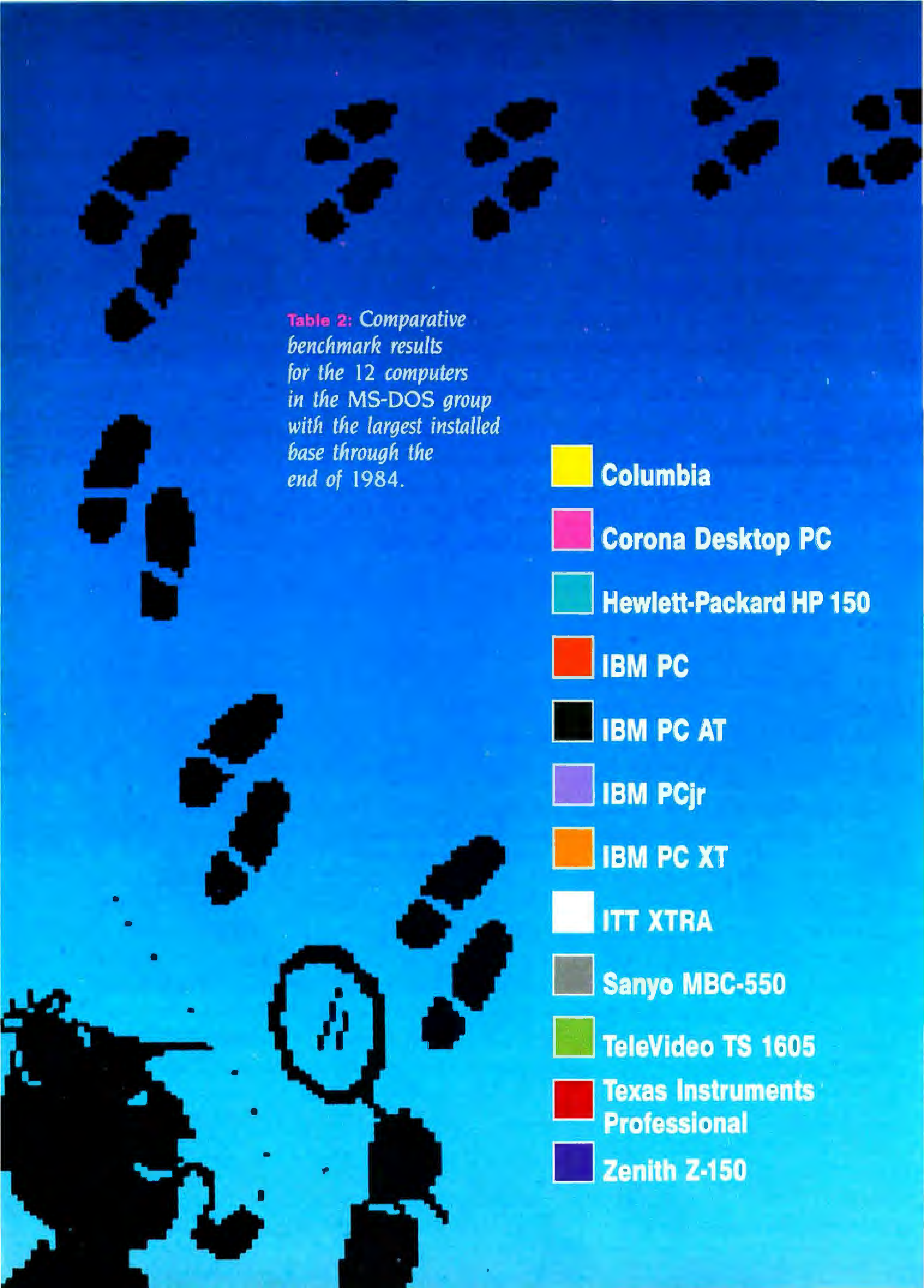
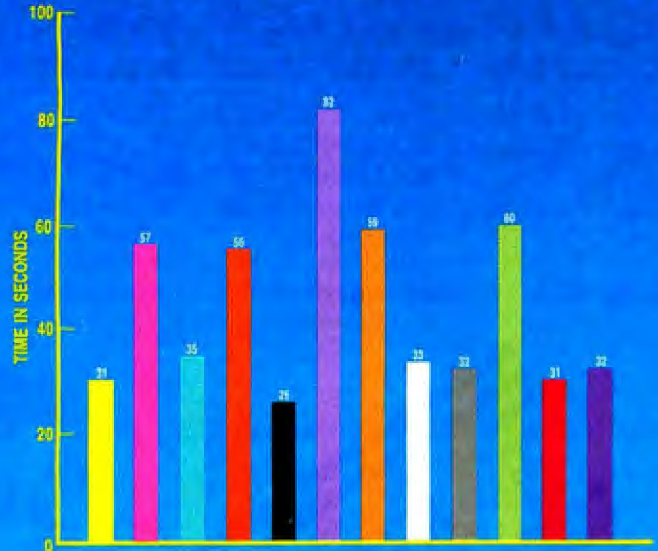
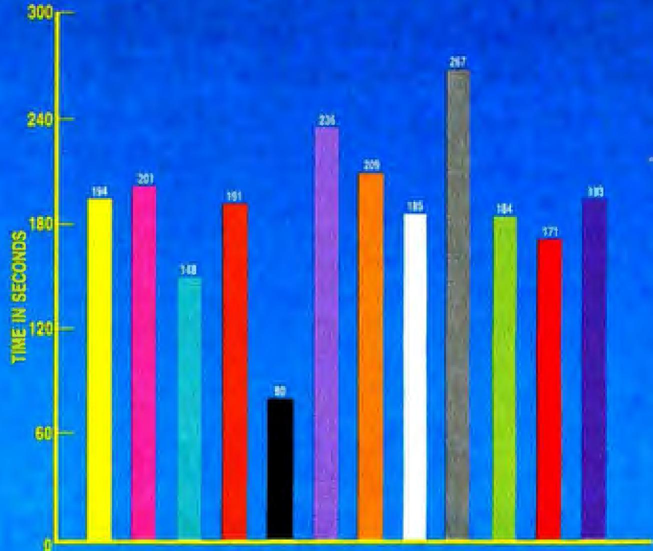


Table 2: *Comparative benchmark results for the 12 computers in the MS-DOS group with the largest installed base through the end of 1984.*

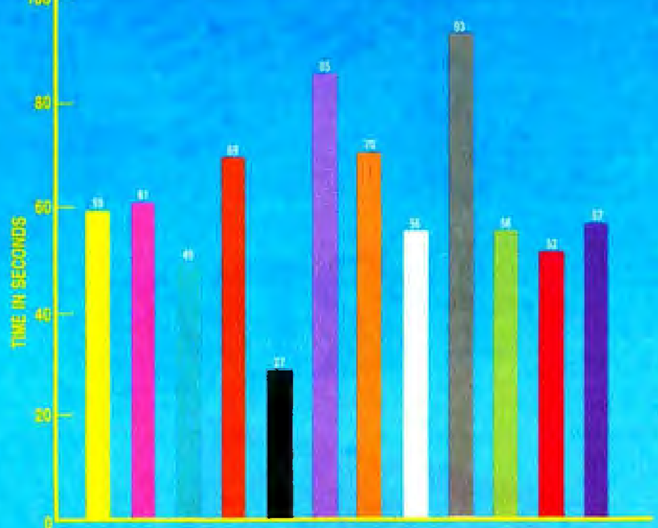
- Columbia
- Corona Desktop PC
- Hewlett-Packard HP 150
- IBM PC
- IBM PC AT
- IBM PCjr
- IBM PC XT
- ITT XTRA
- Sanyo MBC-550
- TeleVideo TS 1605
- Texas Instruments Professional
- Zenith Z-150



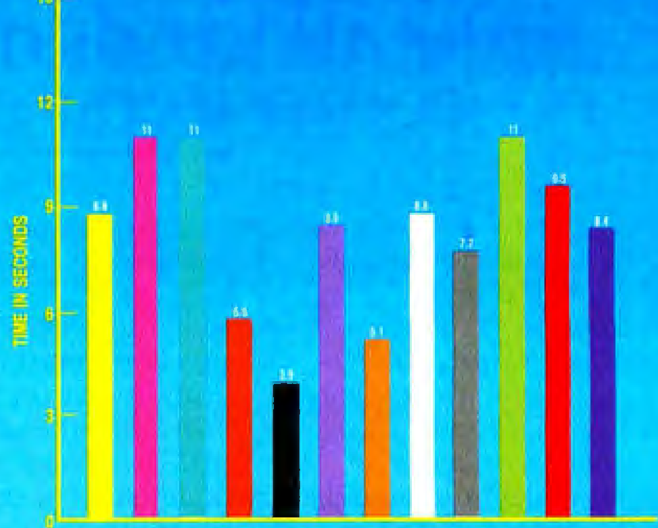
WRITE A 64K-BYTE FILE TO DISK



SIEVE OF ERATOSTHENES PROGRAM



CALCULATING WITH SINGLE-PRECISION NUMBERS



40K-BYTE FILE COPY

write a 64K-byte sequential file onto a disk in 128-byte chunks.

Note that in the tests of BASIC disk access for the Sanyo, the test file was written to and read from the default drive. Sanyo BASIC apparently cannot access other disk drives. The same procedure was used in one-drive systems, like the IBM PCjr.

```
100 A$ = "123456781234567812345678
     12345678"
120 B$ = A$ + A$ + A$ + A$
140 NR = 512
160 OPEN "B:TEST" FOR OUTPUT AS #1
180 FOR I = 1 TO NR
200 PRINT #1, B$
220 NEXT I
240 CLOSE
260 PRINT "DONE"
```

Reading from Disk

The Reading from Disk test that follows measures how long it takes to read the above sequential 64K-byte file in 128-byte segments.

```
300 NR = 512
320 OPEN "B:TEST" FOR INPUT AS #1
340 FOR I = 1 TO NR
360 B$ = INPUT$(128,1)
380 NEXT I
400 CLOSE
420 PRINT "DONE"
```

The Sieve of Eratosthenes Program

Eratosthenes, the head of the Alexandria library around 200 B.C., discovered this technique for finding prime numbers. The program tests how long it takes to do one iteration of the Sieve of Eratosthenes prime-number program.

```
800 SIZE = 7000
820 DIM FLAGS (7001)
830 PRINT "START ONE ITERATION"
840 COUNT = 0
850 FOR I = 0 TO SIZE
860 FLAGS (I) = 1
870 NEXT I
880 FOR I = 0 TO SIZE
890 IF FLAGS (I) = 0 THEN 970
900 PRIME = I + I + 3
910 K = I + PRIME
920 IF K > SIZE THEN 960
930 FLAGS (K) = 0
940 K = K + PRIME
950 GOTO 920
960 COUNT = COUNT + 1
970 NEXT I
980 PRINT "DONE: ";COUNT;"PRIMES
     FOUND"
990 END
```

Calculating with Single-Precision Numbers

The Calculations test measures how long it takes to perform 10,000 multiplication and 10,000 division operations using single-precision numbers and also gives a rough indication of the error involved.

```
500 NR = 5000
520 DEFSNG A-Z
540 A = 2.71828
560 B = 3.14159
580 C = 1
600 FOR I = 1 TO NR
620 C = C*A
640 C = C*B
660 C = C/A
680 C = C/B
700 NEXT I
720 PRINT "DONE"
740 PRINT "ERROR = "; C-1
```

FORMAT AND COPY DISK

The Format and Copy Disk test measures how long it takes to format and copy a disk using the diskcopy command. The test is done only on floppy disks. Copies are made onto previously unformatted disks. Since disk size can vary, the time is adjusted to reflect the number of seconds per 40K bytes of disk space.

The tests on several of the units (STM, Morrow Pivot, Tandy 2000) timed format and diskcopy separately and combined the results. In addition, the STM time includes automatic verification after formatting.

Units with hard-disk drives in their standard configuration (or in the case of the Stearns, for which the review system contained an optional hard disk) will not have results for this test.

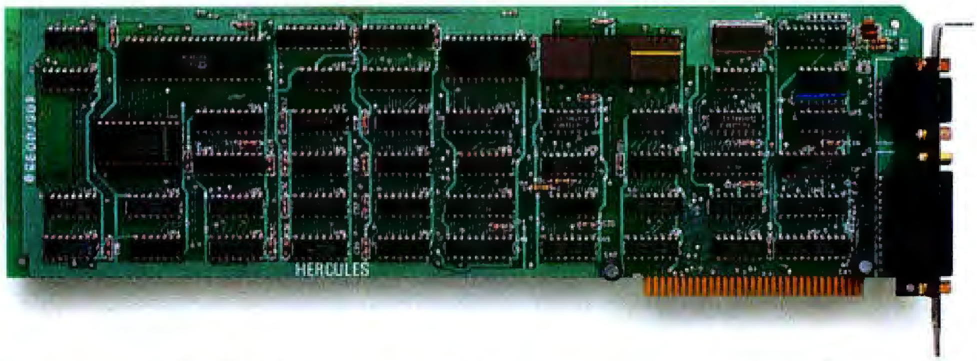
Results on this test could not be obtained for the IBM PCjr and the HP 110, which have only one disk drive, or for the HP 150, because the utility diskcopy is not included with the operating system.

FILE COPY

The File Copy test measures how long it takes to copy a 40K-byte disk file to a blank formatted floppy disk. The tests on units with hard-disk drives measure how long it takes to copy the 40K-byte file from the hard disk to a blank formatted floppy disk. Otherwise, reviewers copied the file to another part of the same disk. There are no results for the HP 110, which had only one disk drive. The results for the IBM PCjr reflect copying to another part of the same disk. ■

The Format and Copy Disk test is done only on floppy disks.

ACKNOWLEDGMENTS
We would particularly like to thank Michael Bamberg, Woody Bell, Neil Rosen, and Avram Tetewsky for their help in running the benchmarks.



The world's best selling monochrome graphics card for the IBM PC.

There are more Hercules™ Graphics Cards in more IBM® PCs, XTs™ and ATs™ than any other monochrome graphics card in the world.

Over a quarter of a million demanding users around the world use a Hercules Graphics Card to improve the performance of their software.

The Lotus® 1-2-3® Booster.

Consider Lotus 1-2-3. Hercules gives an off-the-shelf copy of 1-2-3 the highest resolution possible on an IBM PC, XT or AT. More 1-2-3 users choose the Hercules Graphics Card to get crisper text and sharper graphics than any other monochrome graphics card.

And we bring the same performance to other integrated programs like Symphony™, Framework™, and SuperCalc®3.

But we don't stop there. The Hercules Graphics Card improves all kinds of software.

Like Microsoft® Word, a word processor that enables you to display text with subscripts, superscripts and italics.

Or pfs:® Graph, an easy-to-use business graphics program that converts your data into presentation quality graphs.

Or Microsoft® Flight Simulator, the high flying game for the overworked executive.

Or our own Graph X™, a library of graphics subroutines that eases graphics programming.

Or AutoCAD™, a computer aided design program that offers features normally associated with expensive CAD systems.

And we supply free software with each card to do hi-res graphics using the PC's BASIC interpreter.

State-Of-The-Art Hardware.

The Hercules Graphics Card gives you graphics resolution of 720h x 348v and a parallel printer port. Our unique static RAM buffer provides sharp 9 x 14 characters and flicker-free scrolling. Our exclusive safety features help prevent damage to your monitor.

State-of-the-art custom IC technology delivers unsurpassed reliability. Ordinary graphics cards use up to 30 ICs to do what one Hercules IC does. By using fewer parts, we reduce the possibility of component failure.

Which is one reason we warranty the Hercules Graphics Card for two years.

Unbeatable reliability. Advanced technology. Proven by over a quarter of a million users. Why settle for anything less than Hercules?

Call 1-800-532-0600 Ext 408 for the name of a Hercules dealer nearest you and we'll rush you our free info kit.

Hercules.

We're strong on graphics.

Address: 2550 Ninth St., Berkeley, CA 94710 Ph: 415 540-6000 Telex: 754063. **Trademarks/Owners:** Hercules, Graph X/Hercules; 1-2-3, Symphony/Lotus; IBM, XT, AT/IBM; Framework/Ashton-Tate; SuperCalc/Sorcim-IUS; Microsoft/Microsoft; pfs:/Software Publ; AutoCAD/AutoDesk.

Inquiry 85

FOUR HARD DISKS FOR UNDER $1000

Inexpensive help for your disk storage space woes

BY RICHARD GREHAN

IF YOU ARE a peruser of the back pages of BYTE like most of us, you cannot have failed to notice the plummeting prices of hard-disk systems, particularly those available for the IBM Personal Computer. It is commonplace to find a complete subsystem, including hard disk, controller card, and software, for under $1000.

The advantages of a hard disk should be obvious: Its speed, convenience, and storage space eliminate most of the agonies involved with managing a large pile of floppy disks. If you're interested in setting up a personal bulletin-board system, the purchase of a hard-disk system should be your top priority.

I selected four hard-disk systems from the pages of BYTE and other computer periodicals. My only criterion was that the complete system must cost less than $1000. This article by no means exhausts all the under-$1000 hard disks advertised, but it should give you an idea of some possible trade-offs and troubles if you decide that a hard disk is your PC's next peripheral. Performance and price information is provided in table 1.

Photo 1: *From left to right, the Sider, Rodime, Syquest, and Kamerman Labs hard disks.*

THE SIDER

The Sider is from First Class Peripherals, a Carson City, Nevada, company. An external drive, it is consequently the easiest of the four to install. This also means that the drive has its own power supply; the only added power burden to your PC is the interface card. Additionally, since the Sider does not replace one of your system's floppy-disk drives (all of the internal drives reviewed install in place of one floppy-disk drive), you lose no functionality when you need to, say, copy one floppy disk to another. Best of all, you are spared the trouble of digging through the technical manuals to discover which switches on the PC's motherboard you have to flip to configure the IBM as a one-floppy system.

The Sider comes in a rather large (7½ inches tall, 16½ inches long, and 3½ inches wide) cream-white molded-plastic housing. The hard disk is mounted on its side, and the mechanism is convection-cooled via the case's slotted top. (This slotted top warrants caution: Small objects and certainly fluids could be unwittingly dropped into the inner workings of the unit, inflicting heaven knows what damage.) Since the unit is taller than it is wide, I experienced a not-unjustified fear of knocking it over. A rather stiff but comfortably long cable connects the drive to the interface card.

The installation and operation guide that comes with the Sider is a small 31-page booklet. It is clear and easy to read, obviously written for people with an absolute minimum of hardware knowledge. It includes numerous illustrations of what goes where and

(continued)

Richard Grehan is a BYTE technical editor. He can be contacted at POB 372, Hancock, NH 03449.

PHOTOGRAPHED BY PAUL AVIS

even a diagram of the on-board jumpers and their proper configuration—handy in case one of the jumpers jumped off during shipping. I had no trouble following the instructions or getting the Sider operational. No special software is required. I simply set up a DOS partition using the FDISK program, and then I executed a FORMAT C:/S to format the partition and write an image of the system onto it. For more details on the software provided with the hard disks, see the "Software" text box on page 205.

When you power up a PC with a Sider attached, you must turn on the Sider 30 seconds prior to the computer or (according to the manual) the computer won't recognize the hard disk's existence. The Sider hums pleasantly while operating; head movement is accompanied by muffled thumpings, not the grating squeak of some hard disks I've heard. The benchmarks showed the Sider's speed to be slightly faster than some of the other drives, especially when copying a file from one spot on the hard disk to another.

All in all, the Sider is quite a value; it is easy to install, you get to keep both floppy-disk drives operational, it comes with a full year's warranty, and the price is right. I should also point out that if you find yourself outgrowing the 10 megabytes provided by one Sider, you can daisy-chain a second onto the first—no additional interface card is required. (You cannot daisy-chain an extra Sider on an IBM PC XT, however.)

THE RODIME HARD DISK

I received two of the hard disks from Micro Design. The first is an amazing little drive manufactured by Rodime Inc. I say "amazing" and "little" because it is a full 10 megabytes in a package measuring less than 4 inches wide, 6 inches long, and 2 inches thick. Not surprisingly, it is a low-power drive—the manufacturer promised me that it required no more power than an IBM floppy disk.

The drive itself is mounted in a metallic frame that has extended sides and a large front bezel so that it will fill the hole left by the 5¼-inch floppy-disk drive. There is more empty space to it than hardware.

Installing an internal, as opposed to an external, drive adds several degrees of difficulty. First, since the drive will be replacing your right-hand floppy disk, you must remove that drive and find some dead space in your PC to stuff the now-unused extra length of floppy-interface cable. Next, you have to alter the setting of DIP (dual in-line package) switches on the PC's motherboard to tell it that you now have only one floppy-disk drive. Finally, you install the hard disk and the interface board, hook all the cables up, and turn the system on to observe your success or failure.

I took my time removing the floppy and installing the Rodime, managing to get through the whole procedure without killing anything. The instructions provided by Micro Design are much less polished than those provided by First Class Peripherals, but if you follow them diligently you should survive the installation without a hitch. The documentation consists of 40 photocopied (on one side) pages and includes numerous drawings that are adequate and understandable. Apparently, Micro Design sells several different controller boards as well as drives, and I had some difficulty at first determining which drawings related to the particular hardware I was working with.

When I finally got the Rodime bolted in, I stepped back and noticed that the front bezel was lopsided. Ultimately, I discovered that a socketed 40-pin PROM (programmable read-only memory) on the hard disk's circuit board (which was mounted underneath the unit) was pressing on the metal plate on which the floppy disk had originally rested. This was unsightly and potentially dangerous; Micro Design should look at mounting this hard disk a little higher behind the bezel. Fortunately, since the ceramic body of the PROM was nonconducting, it caused no problems.

The Rodime performed well. It certainly performed quietly. It also lived up to the claims of low power; the PC I used it on held the floppy-disk controller card, a monochrome adapter card, a 256K-byte Quadboard, and a hard-disk controller card. I encountered no problems running everything full tilt. Since it ran so smoothly, I was unable to verify one of the Rodime's more interesting features. If the drive's on-board microprocessor detects an error condition, it flashes the binary code of the error on one of the drive's front-panel LEDs (light-emitting diodes): long flash = 1, short flash = 0.

THE SYQUEST HARD DISK

Easily as amazing as the Rodime hard disk is the Syquest drive, a whopping 30 megabytes in a 5¼-inch half-height package. Also from Micro Design, the Syquest uses the same interface card as the Rodime, except with different PROMs and jumper settings. Installation, too, is almost identical. (In fact, the same documentations serves for both.) Amazingly, this drive also requires no modifications to the PC's power supply.

The Syquest fit neatly in place of the removed floppy; there was no PROM tilting the front plate sideways. The only problem was that when I slid the

Table 1: A *comparison of the four hard-disk systems. Write, read, and copy times were measured using the* BYTE *benchmarks. See* June 1984 BYTE, *pages* 334 *and* 336, *for details.*

	Sider	Rodime	Syquest	Kamerman Labs
Write (seconds)	40.0	41.0	41.0	27.0
Read (seconds)	29.0	29.0	29.0	27.0
Copy (seconds)	2.5	4.7	4.7	2.0
Price	$795	$520	$995	$995

drive all the way into place, the power connector (a large 4-pin Molex connector) jammed against the IBM's power supply, placing a noticeable strain on the connector itself and the hard disk's printed-circuit board to which it was attached. I tried to flex the supply wires some, but it was still a tight fit—the manufacturer should have mounted the connector vertically.

Understandably, formatting the Syquest took a little longer than formatting the other drives, but the end result was worth the wait: 31,819,776 bytes of formatted disk space. In operation it was the noisiest of any of the drives, although not nearly loud enough to be annoying. In the benchmark figures it turned out to be practically identical to the Rodime hard disk.

One irritating side effect of all that space is how long a DIR (directory) command took. When you execute a DIR, it whisks out the filenames and then calculates and displays how many bytes remain free on the disk. A DIR command on the Syquest took a full 6 seconds to calculate and display the bytes-remaining figure, and this was after the filenames had been listed. (Of course, I had configured the hard disk into only one 30-megabyte partition. One solution would be to divide the disk into, say, two 15-megabyte partitions.)

Although the Syquest barely makes it into the under-$1000 bracket, the amount of storage that fits into that little package is frightening. And if 30 megabytes isn't enough, remember that it's only a half-height drive, so you could easily mount two in the place of one floppy. (The controller card is equipped to handle two hard-disk drives, but you'll want to be careful about power requirements.)

THE MEGAFLIGHT 100

The last hard disk I installed was the Kamerman Labs Megaflight 100, a 10-megabyte internal hard disk with a Kamerman CT-100 controller card. This half-height 5¼-inch drive comes with a full-height replacement front bezel. It isn't a low-power drive; the documentation clearly states that the Megaflight may require more power than your system can supply. (This warning doesn't appear until page 6, after you've opened your PC and inserted the CT-100 card.) Consequently, I tested this drive on a Zenith Z-150, which has the power-supply capabilities of a PC XT. (Kamerman Labs points out in the documentation that its system will not work on all PC

(continued)

SOFTWARE

Every one of the systems reviewed was shipped with disks carrying various amounts of software to ease the job of bringing the hard disk on line. In many cases, however, you don't need any more than is already provided with the PC-DOS operating system.

FDISK is a PC-DOS program that IBM supplies for managing what are known as "partitions" on a hard disk. You can think of a partition as an area on the hard-disk drive that acts like an independent "logical disk."

For example, a 10-megabyte hard disk could be partitioned into two 5-megabyte logical disks that the operating system would see as two independent disk drives. Basically, FDISK writes information onto the hard disk that tells DOS where the partitions are located.

Many systems provide a hard-disk low-level formatting and verification program. The Sider's version is called XUTIL; the Micro Design program is called WUTIL. The Megaflight system incorporated the program into its INSTALL program. Such a program typically performs exhaustive write and read operations using selected bit patterns designed to locate failed sectors on the hard disk. If bad sectors are found, the program will then tag them so that the operating system is certain not to use them during normal operation.

Finally, some companies simply throw in additional programs as part of the package. For instance, the Micro Design International hard disks include a CACHE program. CACHE allows you to set aside a designated portion of your computer's memory as a disk cache, 8K bytes minimum. CACHE monitors all disk accesses and keeps the most recently accessed sectors in the allotted memory buffers. New sectors are read only as required, and swapping disk and memory sectors is controlled by the least recently used access technique.

ITEMS DISCUSSED

THE SIDER (IBM PC version)
10-megabyte external hard disk with case, power supply, interface board, and cables

First Class Peripherals
3579 Highway 50 East
Carson City, NV 89701
(800) 538-1307

RODIME
Internal 10-megabyte 3½-inch hard disk with controller board and cables

Micro Design International Inc.
6566 University Blvd.
Winter Park, FL 32792
(305) 677-8333

SYQUEST
Internal 30-megabyte half-height hard disk with controller board and cables

Micro Design International Inc.
6566 University Blvd.
Winter Park, FL 32792
(305) 677-8333

MEGAFLIGHT 100
Internal 10-megabyte half-height hard disk with controller board and cables

Kamerman Labs Inc.
8054 Southwest Nimbus Dr.
Beaverton, OR 97005
(503) 626-6877

clones. In fact, the manual lists a couple of the uncooperative ones. The Z-150 was not among them.)

The Megaflight's documentation is undoubtedly the worst of the lot. It consists of 14 photocopied (on one side) sheets stapled in the upper left-hand corner. The illustrations included are barely adequate, and the text is an absolute maze. Attached to the front of the manual are 5½ pages listing changes made to the upcoming 14, and some programs are mentioned that I never found on the enclosed disk. It never became clear just where to fit some of the errata in the original manual. Endurance on my part finally paid off, and in spite of all the confusion I got the drive going.

A point for owners of PC clones to remember—not brought out in any of the installation documentation I saw—is that all dual-floppy systems consist of a controller card operating the two drives via a single cable; one floppy-disk drive attaches to the cable somewhere along its length, and the other attaches at the cable's end. If, when removing one of the floppies, you remove the one at the end of the cable, the odds are good that you are removing the one that has a necessary terminating resistor pack installed. This is an integrated circuit usually located near the cable connector, and it must be moved to the empty socket at the same spot on the remaining floppy. You should consult the hardware manual for your floppy drive concerning the resistor pack's location, or, if you are at all unsure of the hardware, have a trained technician check it out. Owners of IBM PCs will find the resistor pack on the rear right of the left-hand floppy's printed-circuit board as a blue 16-pin IC—it does not have to be moved when you replace the right-hand floppy disk with a hard disk.

What the Megaflight lacked in documentation, it made up for in performance. In the benchmarks, it scored the swiftest figures. It was also the quietest of the drives; head movement was inaudible over the faint rushing noise that it made.

CONCLUSION

Considering the variety and number of hard-disk systems available, there certainly seems to be something for everyone. With patience and care, you should have little trouble following the instructions provided with most hard disks. Or if you want to keep your tampering inside your PC to an absolute minimum, an external system like the Sider reduces your job to not much more than a board installation.

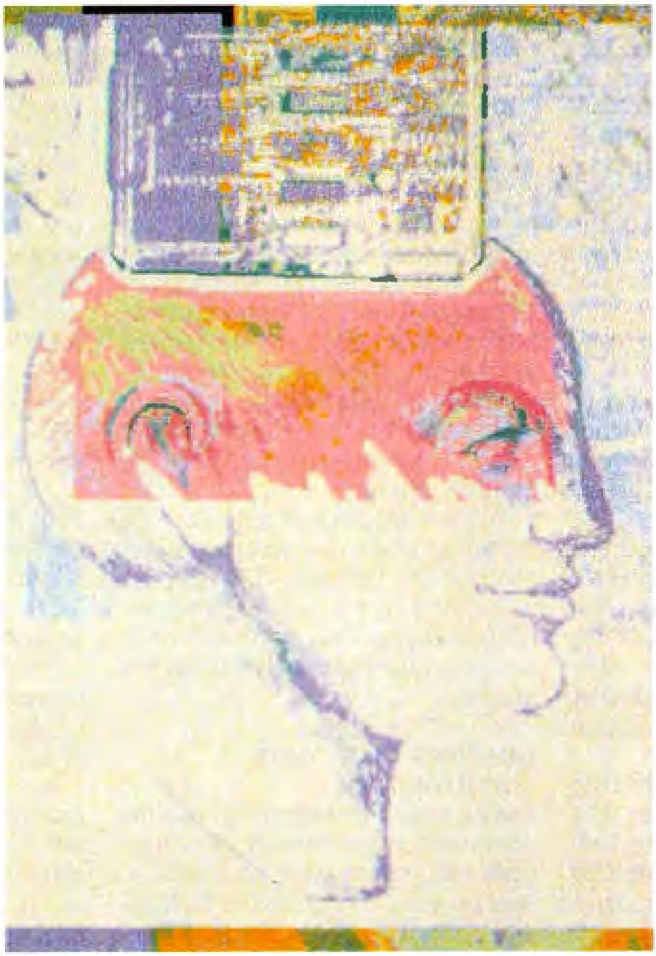
If you can afford an IBM PC, cost is now no excuse for not upgrading it with a hard disk. ■

Inquiry 52

PROGRAMMING THE ENHANCED GRAPHICS ADAPTER

Writing assembly-language drivers for the new graphics card

BY RICHARD WILTON



IBM's Enhanced Graphics Adapter (EGA) for the IBM PC is what many people thought the older Color/Graphics Monitor Adapter (CGA) ought to have been. The EGA can generate 16-color bit-mapped raster graphics with resolution that is good enough to display text without causing eyestrain. It provides for dynamically modifiable character sets, allows horizontal and vertical pixel-by-pixel panning, supports several types of video monitors, and does a good job of emulating the Color Graphics Adapter's function.

Unfortunately, programming the EGA is not a trivial task. The hardware has a complex, sophisticated, and somewhat idiosyncratic architecture, but it is rarely obvious from the technical documentation just what all that complex hardware does. The on-board ROM (read-only memory) BIOS (basic input/output system) does a good job of performing basic video I/O (input/output) functions, but BIOS support for bit-mapped graphics is rudimentary and slow. Despite these obstacles, the EGA can be rewarding to program at the "bits and bytes" level.

In this article, I will stick to the basics of graphics programming on the EGA. Once you are familiar with the EGA's graphics architecture and the capabilities of its ROM BIOS, programming other EGA functions becomes much easier.

CGA COMPATIBILITY

Programmers who have spent some time working with the IBM PC's Color Graphics Adapter will be relieved to find that the EGA can emulate virtually 100 percent of the CGA's function. All the CGA's alphanumeric and all-points-addressable (APA) graphics modes are supported by the hardware and in ROM, with the exception of the 320- by 200-pixel "black and white" APA graphics mode intended for composite video displays. EGA users will probably find this to be a minor shortcoming.

If you know how to program the CGA, you should have no trouble writing programs that run on both the CGA and the EGA. The major source of incompatibility between the EGA and the CGA is related to the video-controller circuitry.

Although the EGA's custom video-controller chips can emulate the function of the Motorola 6845 video controller on the CGA, the I/O port and register assignments on the EGA video controller do not correspond exactly to the registers on the CGA's 6845. Because of this, programs that write directly to the CGA's video controller (for example, to modify video-display

(continued)

Richard Wilton is a software developer with Laboratory Microsystems Inc. (the creators of PC/FORTH), 3007 Washington Blvd., Marina Del Rey, CA 90295. He recently completed the development of the Enhanced Graphics Support Package, which provides an interface between the EGA and FORTH programs.

ILLUSTRATED BY LINDA GOTTFRIED/LEXICA CORPORATION modes or to control the cursor) perform quite unpredictably when they are executed on the EGA.

Another difference between the EGA and the CGA is in the ROM BIOS. The EGA BIOS supports several new function calls in addition to the ones in the CGA BIOS. These functions support character sets loadable from RAM (random-access read/write memory) and return information about the EGA's configuration.

HARDWARE CONFIGURATIONS

If you want to write graphics software for the EGA, you must take into consideration two important aspects of the adapter's hardware configuration: the type of video-display monitor attached to the adapter and the amount of RAM used on the adapter. The available graphics modes and graphics memory mapping vary with the hardware configuration (see table 1).

You configure the EGA via a set of DIP (dual in-line package) switches and jumpers to support one of three different types of RGB (red-green-blue) video-display monitors. The adapter can provide alphanumeric and bit-mapped graphics support for both the IBM PC monochrome display (5151) and the IBM PC color display (5153) or their equivalents. Also, IBM offers an enhanced color display (5154) that provides better resolution than the 5153 does.

The graphics resolution on the IBM monochrome display is 640 by 350 pixels, somewhat less than the 720- by 348-pixel resolution provided by the Hercules monochrome graphics card. The best graphics resolution on the IBM color display is 640 by 200 pixels. You can use the enhanced color display in both 640- by 350-pixel and 640- by 200-pixel modes.

Only one monitor can be attached to the EGA at a time. However, the EGA can be placed in the same system as a Monochrome Display Adapter or a Color Graphics Adapter, so it is possible to use the EGA as part of a dual-display configuration.

The EGA comes with 64K bytes of RAM reserved for use as a video-display refresh buffer. The piggyback Graphics Extender Card adds up to three more banks of EGA RAM in 64K-byte increments, up to a total of 256K bytes.

The additional RAM provides for dynamically loadable character sets and for video paging similar to that performed by the CGA. Also, with an Enhanced Graphics Display but only 64K bytes of EGA RAM, the highest resolution (640 by 350 pixels) graphics mode is limited to four colors. At least 128K bytes are needed to display 16 colors at the same time in this graphics mode.

An EGA graphics program can be written so that it executes properly in each of the different video modes demanded by the various hardware configurations. I will discuss some of the programming details later in this article.

GRAPHICS CHIP PORTS AND REGISTERS

The programmer directly controls the operation of the custom LSI (large-scale integration) logic on the EGA. Your program configures the EGA display control logic to perform the various bit-manipulation and video-display functions you need to display bit-mapped graphics. The configuration is controlled by the contents of various special-purpose registers defined on the EGA chips themselves.

You program the EGA registers by writing data bytes to a set of predefined I/O ports. At the assembly-language level, you do this via the OUT instruction. The IBM PC BASIC's OUT statement also performs this function.

Often, several EGA registers are mapped onto the same I/O port. In this circumstance you generally need to specify a register *number* at one port address and the *contents* of that register at another port address. For example, the Graphics Controller port at address 3CE hexadecimal maps nine different registers. To store a byte of data into one of those registers, you first OUT the register number to port 3CE. Then you OUT the data byte itself to port 3CF.

You can find examples of how this is done in source listings 2 to 5.

THE ARCHITECTURE OF EGA RAM

Like the Color Graphics Adapter, the EGA contains its own on-board RAM. This RAM is used primarily as a refresh buffer by the EGA's video-controller circuitry. Individual pixels on the screen correspond to groups of bits in the EGA's RAM.

As on the CGA, it is up to the programmer to set the individual bits in the EGA's display RAM. This can place quite a computational burden on the PC's main microprocessor (the Intel 8088 in the PC or the 80286 in the AT), especially when many pixels must be modified over a large area of the screen.

Display RAM on the EGA differs from its counterpart on the CGA in several important respects. Unlike CGA RAM, which always starts at segment address B800 hexadecimal, the starting location of EGA RAM can be modified. The address of the first byte in EGA RAM can be programmed to appear at any of three segment addresses: B800 hexadecimal for ROM

Table 1: *Enhanced graphics modes.*

ROM BIOS mode (hexadecimal)	Pixel resolution	Number of colors	Type of display
0D	320 by 200	16	color, enhanced color
0E	640 by 200	16	color, enhanced color
0F	640 by 350	4	monochrome
10	640 by 350	4	enhanced color (EGA RAM = 64K bytes)
10	640 by 350	16	enhanced color (EGA RAM > 64K bytes)

BIOS modes 0 through 6, which emulate the Color Graphics Adapter; B000 hexadecimal for ROM BIOS mode 7, which emulates the Monochrome Adapter; and A000 hexadecimal for enhanced graphics modes 0D, 0E, 0F, and 10 hexadecimal.

The memory map of pixels in EGA RAM is different than on the CGA. On the CGA, alternate rows of pixels are mapped into two separate halves of the display buffer. In contrast, all EGA enhanced graphics modes map pixels in linear sequence, from left to right and top to bottom across the display.

In enhanced graphics modes, the first displayable pixel is represented by bit 7 (the high-order bit) of the byte at address A000:0000. Thus, one row of 640 pixels requires an address space of 80 (50 hexadecimal) bytes. With this addressing scheme, the address of the first pixel in the second row of the screen is at bit 7 of the byte at A000:0050, the second pixel in the row is at bit 6 of that byte, and so forth.

Finally—and this is a major architectural difference from the CGA—the EGA display RAM consists of four parallel bit planes. For example, an EGA with 64K bytes of RAM is actually configured as four 16K-byte bit planes, all four of which share the same address space (e.g., starting at A000:0000 for enhanced graphics modes). Since four banks of RAM share the same address space, the EGA contains special logic for accessing each of the bit planes individually and in tandem.

Bit Planes and Pixels

It helps to visualize the logical configuration of EGA RAM as two-dimensional. Imagine one row of 640 pixels. This row of pixels is 640 bits (80 bytes) "across" and 4 bits (1 bit per plane) "deep." You find the address of an individual pixel by counting "across" the row. The value of that pixel is represented by the bits set in each of the four bit planes at that address.

For instance, a pixel whose value is 5 (binary 0101) in the upper left corner of the display is addressed by the high-order bit (bit 7) of the zeroth byte in EGA RAM. Its value is determined by setting the corresponding bits in bit planes 0 and 2 to 1, and the corresponding bits in planes 1 and 3 to 0.

The EGA's Graphics Controller permits all four bit planes to be addressed at the same time. When you read 1 byte at an address in EGA RAM (say, with an 8086 MOV instruction or a BASIC PEEK), the Graphics Controller can read 4 bytes, 1 byte from each bit plane at that address. The Graphics Controller latches all 4 bytes during a graphics read. That is, the 4 bytes of data are stored in registers internal to the Graphics Controller, where they can be modified and subsequently rewritten—again, all 4 bytes at the same time—to the bit planes in EGA RAM.

The way in which bits in the four bit planes are arranged to specify the value of a pixel depends upon the amount of EGA RAM present and upon the graphics mode being displayed. For example, consider the configuration for a 640- by 350-pixel four-color graphics mode on an EGA with only 64K bytes of RAM.

In this case, to display 224,000 (640 × 350) pixels, each bit plane must contain 28,000 (224,000 ÷ 8) bytes, which is more than the 16K bytes available per bit plane. To solve this problem, the EGA's video logic is set up to chain the bit planes together (plane 2 is chained to plane 0; plane 3 is chained to plane 1).

With chained bit planes in this graphics mode, the value of a pixel is determined by bit planes 0 and 2 if the byte containing the pixel is at an even address, and by planes 1 and 3 if the byte is at an odd address. If you are reading or writing pixels in this particular configuration, you must be very careful to use the correct bit

(continued)

planes for each pixel.

In general, a program that manipulates pixels in EGA RAM should be "aware" of the EGA's hardware configuration as well as the current video mode. ROM BIOS function 12 hexadecimal of interrupt 10 hexadecimal returns this information. A graphics program that does not test the hardware configuration properly will get into trouble manipulating pixel bits in different EGA graphics modes. In fact, the EGA's ROM BIOS read-dot function contains exactly this type of bug: it appears in monochrome graphics mode 0F hexadecimal on an EGA with 64K bytes of RAM (see listing 1 and table 2).

HOW TO WRITE A PIXEL

The conceptually simple task of setting a pixel's value requires that you become familiar with several key functions of the EGA's Graphics Controller. For this reason it is worth examining the sequence of events carefully. A comparison of the sample subroutine listings with the following discussion should be helpful.

On the EGA, you can set the value of individual pixels in three different "write modes." These modes have nothing to do with the video-display modes discussed earlier. They describe different methods of programming the EGA Graphics Controller chip.

The EGA's ROM BIOS uses Write Mode 0 as its default, so I will describe its use in detail. To set the value of a pixel in Write Mode 0, you perform a five-step sequence of operations (see listing 2).

First, you must calculate a byte address and bit mask for the pixel. With integer arithmetic and 80 bytes per row, the byte offset of a pixel at location (x,y) is at $(y \times 80) + x/8$ and the bit number to be set (where bit 7 is the left-most or high-order bit) is $7 - (x \bmod 8)$.

You generally need to program the EGA Graphics Controller with a bit mask rather than a bit number. The corresponding bit mask can be calculated by $2^{(7 - (x \bmod 8))}$ or, more efficiently, with a simple lookup table.

Listing 1: *A FORTH routine that demonstrates a bug in the EGA's ROM BIOS read-dot routine. The pixels are written properly by the write-dot routine, but the read-dot procedure doesn't return the correct value (see table 2).*

```
HEX
: WRITE_DOT ( x y value - )       \ use ROM BIOS to store pel
    >R SWAP R>                     \ stack: - y x value
    0 SWAP 0C00 +                  \ stack: - y x 0 0C+value
    video-io 4DROP;                \ do INT 10h, discard stack

: READ_DOT ( x y - value )        \ use ROM BIOS to fetch pel
    SWAP 0 0D00                    \ stack: - y x 0 0D00
    video-io                       \ stack: - DX CX BX AX
    >R 3DROP R> 0FF AND ;          \ stack: - value

DECIMAL
: BUGDEMO (-)
    640X350 VMODE B/W CLS      \ select mode, clear the screen

    0 0 15 WRITE_DOT 0 0 READ_DOT
    CR ." WRITE dot in even byte, READ dot in even byte: ".

    8 1 15 WRITE_DOT 8 1 READ_DOT
    CR ." WRITE dot in odd byte, READ dot in odd byte  : ".

    0 2 15 WRITE_DOT 8 2 READ_DOT
    CR ." WRITE dot in even byte, READ dot in odd byte      : ".

    CR ." End of demo." CR ;
```

Table 2: *Sample output from listing 1. The first value is correct; the second value should be 15, and the third value should be 0.*

```
WRITE dot in even byte, READ dot in even byte: 15
WRITE dot in odd byte, READ dot in odd byte  :  0
WRITE dot in even byte, READ dot in odd byte : 15
End of demo.
ok
```

Listing 2: *Sample subroutine using Write Mode 0.*

```
wp0     proc    near        ; call with     AX = y-coordinate
                            ;               BX = x-coordinate
                            ;               CX = pixel value

        push    cx          ; Preserve the pixel value on the stack.
                            ; (It gets popped into AX later on . . .)

; Calculate byte address (segment & offset) and bit mask
        mov     dx,0A000h
        mov     ds,dx       ; DS := EGA buffer segment address

        mov     dx,80
        mul     dx          ; AX := (y * 80)

        mov     cx,bx       ; CX := x
        shr     bx,1
        shr     bx,1
        shr     bx,1        ; BX := x/8
        add     bx,ax       ; BX := (y * 80) + x/8 (offset)

        and     cl,7        ; CL := (x mod 8)
```

(continued)

```
        xor     cl,7            ; CL := 7 - (x mod 8)
        mov     ch,1
        shl     ch,cl           ; CH := 2 ^ (7 - x mod 8) (bit mask)

; Set the Bit Mask register
        mov     dx,3CEh         ; Graphics Controller port address
        mov     al,8
        out     dx,al           ; select register 8

        mov     dx,3CFh
        mov     al,ch
        out     dx,al           ; load the bit mask into register 8

; Set all 4 bits in the pixel to 0
        mov     al,[bx]         ; Read at address A000:offset
                                ; This latches all 4 bit planes.
                                ; (The byte "read" is ignored.)

        mov     al,0
        mov     [bx],al         ; Write at address A000:offset
                                ; This sets the bit-masked bits to 0
                                ; and stores the latched bytes to the
                                ; bit planes

; Set bits in the appropriate bit planes to 1
        mov     dx,3C4h         ; Sequencer/Map Mode port address
        mov     al,2
        out     dx,al           ; Select Map Mask register 2

        mov     dx,3C5h
        pop     ax              ; AL := map mask (i.e., pixel value)
        out     dx,al           ; Load the map mask into SMM register 2.
                                ; This enables the appropriate bit planes.

        mov     al,[bx]         ; Latch the bit plane data.
        mov     al,11111111b
        mov     [bx],al         ; Set bits to 1 in appropriate planes.

; Restore default EGA graphics status
        mov     dx,3C4h
        mov     al,2            ; Again, select ...
        out     dx,al           ; ... Sequencer/Map Mask register 2.

        mov     dx,3C5h
        mov     al,1111b        ; Default map mask
        out     dx,al           ; Enable all four bit planes

        mov     dx,3CEh
        mov     al,8            ; Again, select ...
        out     dx,al           ; ... Graphics Controller register 8

        mov     dx,3CFh
        mov     al,11111111b    ; Default bit mask
        out     dx,al           ; Restore default bit mask

        ret
wp0     endp
```

Second, configure the EGA Graphics Controller to perform the write operation. You need to specify the write mode, the bit planes to be modified, and the bit mask. The ROM BIOS always leaves the Graphics Controller in Write Mode 0 with all four bit planes enabled, so all that you need to specify is the bit mask for the pixel.

Load the Graphics Controller's Bit Mask register (register number 8) by storing 8 to I/O port 3CE hexadecimal. (Use the 8086 OUT instruction.) Then store the bit mask to the data port 3CF.

Third, reset the value of the pixel to zero by setting the bits in all bit planes corresponding to the pixel to 0. To do

(continued)

this, read 1 byte at the byte address you calculated and then write a data byte consisting of 8 zero bits to that address.

When you perform the byte read, the Graphics Controller stores 4 bytes from the bit planes into its internal latch registers. (You can ignore the byte you read.) When you write the data byte, the Graphics Controller uses the bit mask to copy bits from the data byte to the bit planes. Only masked bits are copied. The result is that a zero bit is copied to each of the four bit planes, and the value of the pixel is zero.

Fourth, set the bits to 1 in the appropriate bit planes for the pixel. Do *this* by enabling only those bit planes that are to contain a 1 bit. Bit planes are enabled by storing a "map mask" in the Graphics Controller's Sequencer/Map Mode register (register 2).

The map mask is a 4-bit pattern (the low-order nybble of 1 byte) with 1 bits that correspond to the bit planes to be enabled. Since the value of a pixel likewise corresponds to the bits set in each bit plane at a given address, the map mask is the same as the value of the pixel. Thus, when you OUT a 2 to I/O port 3C4 and then the value of the pixel to I/O port 3C5, you enable the appropriate bit planes for the 1 bits in the pixel.

To store 1 bits in the enabled bit planes, you must again latch bytes from the bit planes by doing a byte read. Then, write a data byte of all 1 bits. As before, the Graphics Controller uses the bit mask to determine which bits in the data byte to copy into the bit planes. However, this time only the enabled bit planes are updated. The result is that a 1 bit is copied to each of the enabled bit planes, and the pixel has its new value.

Fifth, you must reconfigure the Graphics Controller to the default status assumed by the ROM BIOS. Reenable all four bit planes. (OUT a 2 to port 3C4 and then binary 1111 to port 3C5.) Finally, set the Graphics Controller's bit mask to binary

(continued)

11111111. (OUT an 8 to port 3CE and then 11111111 to port 3CF.)

In Write Mode 1 the Graphics Controller simply copies the contents of its latch registers (presumably loaded by a previous latched read) into the bit planes. This function is particularly useful for copying one region of EGA RAM to another, since all four bit planes can be updated in a single memory cycle.

You can also use Write Mode 2 to set the value of an individual pixel. The technique differs from that of Write Mode 0 in that you do not need to selectively enable the bit planes in order to set bits. Rather, bits in the appropriate planes are set according to a data byte you write to the pixel's byte address.

The programming sequence is as follows (see listing 3). Calculate the byte address and bit mask for the pixel. Then set Write Mode 2 by writing a 2 to the Graphics Controller register 5, the Mode register. (OUT a 5 to port 3CE, then OUT a 2 to port 3CF.) Set the bit mask for the pixel. (OUT an 8 to port 3CE, then OUT the bit mask to port 3CF.) Next, latch the bytes at the calculated address in the bit planes. Write a byte containing the value of the pixel to the calculated address. The bits in this byte are copied into the corresponding bit planes (bit 0 to plane 0, bit 1 to plane 1, and so on) for the bit specified in the bit mask. Finally, restore the default Graphics Controller write mode and bit mask.

In addition to providing three different ways of writing pixels, the Graphics Controller can also be programmed to perform certain useful (and maybe not so useful) bit manipulations. You can set any or all of the 8 bits in the bit mask so that up to 8 pixels at a time can be modified in a single write operation. This is especially handy in setting groups of pixels to the same color (to draw a horizontal line, for example). Bitwise AND, OR, or XOR operations on the data in the bit planes can be specified by modifying bits 3 and 4 of the Graphics Controller's Data Rotate/Function Select register (register 3). In Write

Listing 3: *Sample subroutine using* Write Mode 2.

```
wp2     proc    near            ; call with     AX = y-coordinate
                                ;               BX = x-coordinate
                                ;               CX = pixel value

        push    cx              ; Preserve the pixel value on the stack.
                                ; (It gets popped into AX later on . . .)

; Calculate byte address (segment & offset) and bit mask
        mov     dx,0A000h
        mov     ds,dx           ; DS := EGA buffer segment address

        mov     dx,80
        mul     dx              ; AX := (y * 80)

        mov     cx,bx           ; CX := x
        shr     bx,1
        shr     bx,1
        shr     bx,1            ; BX := x/8
        add     bx,ax           ; BX := (y * 80) + x/8 (offset)

        and     cl,7            ; CL := (x mod 8)
        xor     cl,7            ; CL := 7 - (x mod 8)
        mov     ch,1
        shl     ch,cl           ; CH := 2 ^ (7 - x mod 8) (bit mask)

; Select Write Mode 2
        mov     dx,3CEh         ; Graphics Controller port address
        mov     al,5
        out     dx,al           ; Select Mode register (register 5)

        mov     dx,3CFh
        mov     al,2
        out     dx,al           ; Set Write Mode 2

; Set the Bit Mask register
        mov     dx,3CEh         ; Graphics Controller port address
        mov     al,8
        out     dx,al           ; select register 8

        mov     dx,3CFh
        mov     al,ch
        out     dx,al           ; load the bit mask into register 8

; Latch all four bit planes
        mov     al,[bx]         ; "Graphics read" at address A000:offset
                                ; This loads 1 byte from each bit plane
                                ; into the latch registers.
                                ; (The byte "read" is ignored.)

; Write the pixel
        pop     ax              ; AL := pixel value
        mov     [bx],al         ; Write at address A000:offset
                                ; This sets the appropriate bit-masked bits
                                ; in the latch registers and stores
                                ; the latched bytes to the bit planes.

; Restore default EGA graphics status
        mov     dx,3CEh         ; Again, select . . .
        mov     al,5
        out     dx,al           ; . . . Graphics Controller "Mode Register"

        mov     dx,3CFh
        mov     al,0
        out     dx,al           ; Restore Write Mode 0 (the default)

        mov     dx,3CEh
        mov     al,8            ; Again, select . . .
```

(continued)

```
        out     dx,al           ; . . . Graphics Controller register 8

        mov     dx,3CFh
        mov     al,11111111b    ; Default bit mask
        out     dx,al           ; Restore default bit mask

        ret
wp2     endp
```

Listing 4: *Sample subroutine using Read Mode* 0.

```
rp0     proc    near            ; call with     AX = y-coordinate
                                ;               BX = x-coordinate
                                ; returns       CX = pixel value

; Calculate byte address (segment & offset) and bit mask
        mov     dx,0A000h
        mov     ds,dx           ; DS := EGA buffer segment address

        mov     dx,80
        mul     dx              ; AX := (y * 80)

        mov     cx,bx           ; CX := x
        shr     bx,1
        shr     bx,1
        shr     bx,1            ; BX := x/8
        add     bx,ax           ; BX := (y * 80) + x/8 (offset)

        and     cl,7            ; CL := (x mod 8)
        xor     cl,7            ; CL := 7 - (x mod 8)
        mov     ch,1
        shl     ch,cl           ; CH := 2 ^ (7 - x mod 8) (bit mask)

; Read each bit plane
        mov     ah,3            ; AH := bit plane number
L1:     mov     dx,3CEh
        mov     al,4            ; Select Graphics Controller . . .
        out     dx,al           ; . . . Read Map Select register 4

        mov     dx,3CFh
        mov     al,ah           ; Select bit plane 3, 2, 1, 0
        out     dx,al

        mov     al,[bx]         ; AL := byte from bit plane

; Gather the bits together in CL
        shl     cl,1            ; bit0 of CL := 0
        and     al,ch           ; Apply bit mask to byte read
        jz      L2              ; Jump if bit read was 0
        or      cl,1            ; bit0 of CL := 1
L2:     dec     ah              ; AL := next bit plane number
        jge     L1              ; Loop through all four bit planes

        and     cx,000Fh        ; CX := pixel value
        ret
rp0     endp
```

Mode 0, the data byte written to the bit planes can be rotated by the Graphics Controller before it is written to the bit planes. The number of bit positions to rotate to the left is specified in bits 0 through 2 of the Data Rotate/Function Select register.

How To Read A Pixel

Some graphics algorithms, particularly scan-conversion and region-fill algorithms, require that you determine the value of an individual pixel. The EGA facilitates accurate programming of such algorithms by providing two different ways of reading the value of pixels.

In Read Mode 0, the EGA BIOS default, the value of the bits in a given byte in any one bit plane can be determined. As in Write Mode 0, you must specify a bit mask and select a set of bit planes.

The programming steps are as follows (see listing 4). Calculate the address and bit mask for the pixel. Select a bit plane to be read by loading a bit number (not a bit mask) into the Graphics Controller's Read Map Select register (register 4). Use OUT to write a 4 to port 3CE, then OUT the bit number (0, 1, 2, or 3) to port 3CF. Then read the byte at the address you calculated. This byte is the actual data stored in the bit plane you selected at the address you read. Then, AND the byte you read with the bit mask. If you are reading one pixel, the bit you mask is one of the 4 bits that make up the value of the pixel. Repeat these steps for all bit planes. Put the bits you read from each of the bit planes together in the low nybble of 1 byte. This is the value of the pixel.

Read Mode 1 is conceptually quite different from Read Mode 0. In Read Mode 1 you compare pixels to a prespecified value to see if they match rather than actually determining the value of a pixel. This technique is practical when you need to scan a row for the occurrence of a pixel of a given value since you can test 8 pixels at a time.

Here are the steps involved (see listing 5). As usual, calculate the address of the pixel(s) in question. Select Read Mode 1. (OUT a 5 to port 3CE, then OUT an 8 to port 3CF to set bit 3 of this register to 1.) Load the value to which you want the EGA to compare pixels into Graphics Controller register 2, the Color Compare register. (OUT a 2 to port 3CE, then OUT the comparison value to port 3CF.) Read the byte at the calculated address. This byte will contain a 1 bit in bit

(continued)

Listing 5: *Sample subroutine using Read Mode 1.*

```
rp1     proc    near            ; call with     AX = y-coordinate
                                ;               BX = x-coordinate
                                ;               CX = compare value
                                ; returns       CX = flag

        push    cx              ; Push the compare value on the stack.
                                ; (It gets popped into AX later on ...)

; Calculate byte address (segment & offset) and bit mask
        mov     dx,0A000h
        mov     ds,dx           ; DS := EGA buffer segment address

        mov     dx,80
        mul     dx              ; AX := (y * 80)

        mov     cx,bx           ; CX := x
        shr     bx,1
        shr     bx,1
        shr     bx,1            ; BX := x/8
        add     bx,ax           ; BX := (y * 80) + x/8 (offset)

        and     cl,7            ; CL := (x mod 8)
        xor     cl,7            ; CL := 7 - (x mod 8)
        mov     ch,1
        shl     ch,cl           ; CH := 2 ^ (7 - x mod 8) (bit mask)

; Select Read Mode 1
        mov     dx,3CEh
        mov     al,5            ; Select Graphics Controller ...
        out     dx,al           ; ... Mode register (register 5)

        mov     dx,3CFh
        mov     al,8            ; Set bit 3 to indicate ...
        out     dx,al           ; ... Read Mode 1

; Specify the comparison value
        mov     dx,3CEh
        mov     al,2            ; Select Graphics Controller ...
        out     dx,al           ; ... Color Compare register

        mov     dx,3CFh
        pop     ax              ; AL := comparison value
        out     dx,al           ; Load Color Compare register

; Read and test the pixel value
        mov     al,[bx]         ; Read at address A000:offset.
                                ; If a bit in AL is 1,
                                ; the corresponding pixel's
                                ; value matches the
                                ; comparison value.

        and     al,ch           ; Apply the bit mask.
        jz      L1              ; Jump if pixel didn't match.

        mov     cx,1            ; CX := "true"
        ret

L1:     mov     cx,0            ; CX := "false"
        ret
rp1     endp
```

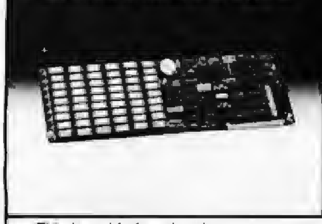
positions corresponding to pixels that match the comparison value. If a pixel does not match, the corresponding bit position is 0. Then restore the default Read Mode 0. (OUT a 5 to port 3CE, then OUT a 0 to port 3CF.)

By default, the EGA compares bits in all four bit planes to the value you load into the Color Compare register.

(continued)

MICROTEK INTRODUCES NEW PC BOARD LINE BY OFFERING 10,000 20% OFF DISCOUNT COUPONS* TO IBM PC USERS.

1. GL MULTIFUNCTION BOARD

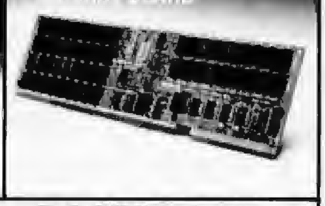


This board is functionally equivalent to the IBM Printer Adapter, Asynchronous Communications Adapter, enough memory expansion to equal 384K, and a clock card. Plus comes with software that allows an extra 640K of Virtual Memory (can be used with fixed or RAM disk and Lotus/Intel/Microsoft specification software). For IBM PC/XT and compatibles.

2. GL 640 + 2MB BOARD

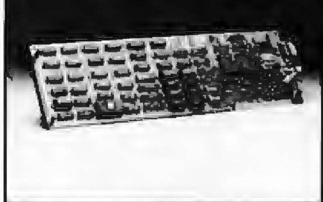
This board provides up to 2 megabytes of paged RAM memory for use with Symphony 1.1, Lotus 1-2-3 (2.0 version), and new versions of Framework, SuperCalc and other programs that use the Lotus/Intel/Microsoft expanded memory specifications. It is shipped with Disk Emulation and Printer Spooler software.

3. GL 0-384K ULTRAFLEX MEMORY BOARD

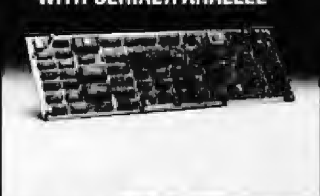


The GL 0-384K Ultraflex Memory Board does more than just provide up to 384K of extra memory. In addition to print spooling, this board comes with software that allows an extra 640K of Virtual Memory (can be used with fixed or RAM disk and Lotus/Intel/Microsoft specification software). For IBM PC/XT and compatibles.

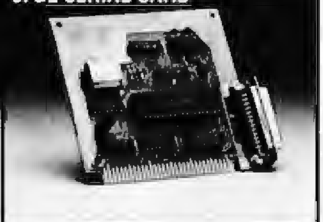
4. GL COLOR GRAPHICS BOARD



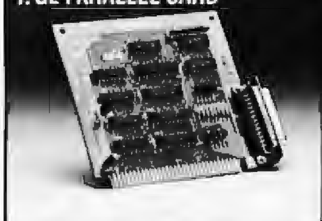
5. GL MONOCHROME DISPLAY WITH SERIAL/PARALLEL



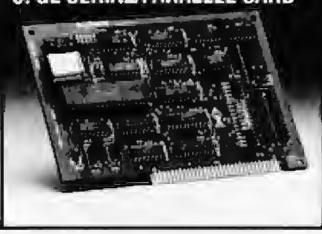
6. GL SERIAL CARD



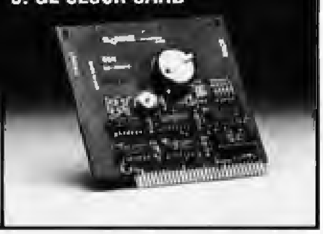
7. GL PARALLEL CARD



8. GL SERIAL/PARALLEL CARD



9. GL CLOCK CARD



20% Off Coupons will be honored at your local dealer.

Now you can get the discounts you're looking for plus full product support from your local dealer. Here's all you do. Simply fill out the Coupon Request Form at the bottom of this ad and mail it back to Microtek. We'll send you your 20% OFF COUPON, product information and the name of your nearest participating Microtek dealer.

More than 1,000 dealers and over 300,000 customers know the value of Microtek's:

- 7–24 Customer Service (7 days, 24 hours per day)
- Wide Selection of Products
- Fully-Tested Products
- 2 Year Warranty
- Manufacturing Quality (Built in USA)

*20% Off suggested list price

COUPON REQUEST FORM

MICROTEK inc.

Please send me the following:

☐ 20% OFF COUPON and the name of my nearest participating dealer

☐ Information on products circled 1 2 3 4 5 6 7 8 9

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Telephone Number ______________________

Dealer and distributor inquiries welcome. Call (619) 569-0900

MAIL TO: Microtek, Inc., P.O. Box 23817, San Diego, CA 92123

IBM PC is a registered trademark of International Business Machines Corporation. Framework is a registered trademark of Ashton-Tate. SuperCalc is a registered trademark of Sorcim/IUS Micro Software. Lotus is a registered trademark of Lotus Development Company.

Inquiry 149 for End-Users. Inquiry 150 for DEALERS ONLY.

However, in some video modes only two bit planes are used for a given pixel. (These are 640- by 350-pixel four-color graphics modes on the IBM monochrome display or on an EGA with only 64K bytes of RAM.) In these instances you must specify which bit planes are used in the comparison by loading a bit pattern into the Graphics Controller's Color Don't Care register (register 7).

PIXELS AND PALETTES

The value of a pixel corresponds to one of 16 palette registers internal to the EGA's video attribute controller circuitry. In turn, the content of each of these registers corresponds to the colors displayed. Each bit in the palette registers corresponds to the IRGB (input red-green-blue) signals that drive the video display. Since a pixel can have any one of 16 values, and any of 64 colors can be specified in each of the palette registers, 16 of 64 colors can be displayed.

When a video mode is selected using the ROM BIOS, the palette registers are loaded with "reasonable" color values. For example, in 16-color video modes, the colors correspond to those available on the CGA. However, you can alter the contents of the palette registers to achieve color-mixing, masking, and animation effects.

You can load the palette color registers directly by writing I/O port 3C0. However, you should probably use the ROM BIOS function 10 hexadecimal of interrupt 10 hexadecimal, which programs the palette registers either individually or from a list of values. This is the same BIOS software interface used in the PCjr.

SUMMARY

I have covered the fundamentals of graphics programming on the EGA. If you have followed the sample listings, you will have an idea of the type of low-level programming required.

Understanding the graphics architecture of the EGA is the essential first step in programming other interesting functions, including panning, split-screen display, and RAM-loadable character sets. IBM's technical documentation provides programming examples for these EGA functions.

At this writing, the EGA is still fairly new on the market. Few programs make use of its enhanced graphics capabilities. But as the use of the EGA becomes more widespread and more ingenious programmers begin working with it, you should see some remarkable graphics programs. ■

BIBLIOGRAPHY

Hoffman, Thomas V. "Graphic Enhancement." *PC Tech Journal*, April 1985, volume 3, number 4, page 58.

IBM PC Technical Reference: Options and Adapters. Reference Number 6322509.

IBM Personal Computer Seminar Proceedings. May 1985, volume 2, number 1-11.

Powerful MS™-DOS Software.

For the IBM®-PC, XT, AT & compatibles with MS-DOS/PC-DOS 2.0 or higher.

Only $39.95 Each

UTAH COBOL™

ALL NEW! We believe this is the fastest COBOL compiler on any micro; for example, we have compiled 4500 statements per minute on an IBM-AT. Utah COBOL is based on ANSI-74 standards with powerful level 2 features, including compound conditionals & full CALL CANCEL. Speed and simplicity are certain to make it a favorite in the classroom as well as with professional programmers. Package includes diskette, 165-page manual, many examples & 16 complete COBOL source code programs. **$39.95.**

COBOL Application Package, Book 1: $9.95.

UTAH PASCAL™

Has many advanced features including: 14 digit precision, BCD math (no round-off errors), floating point +63 -64, TRACE debugging, Arrays up to 8 dimensions, 64K strings, External procedures & Dynamic Module loading. Package includes diskette & 134-page mannual. **$39.95.**

UTAH PILOT™

Written by Prof. J. Starkweather, the language's creator, Utah PILOT exceeds all PILOT-73 standards. And it has an integrated full-screen text editor for easy program development. Ideal for classroom instruction, business training & home study. Package includes diskette, 129-page manual & 10 sample programs. **$39.95.**

UTAH EDIT™

A full-screen, video-display text editor designed specifically for computer program text preparation. Completely user-changable, it can be configured to almost any terminal & takes only 12K disk space. Package includes diskette & 58-page manual. **$39.95.**

UTAH **BASIC**™ $39.95
Package includes diskette & 239-page manual.

BIGPRINT™ DISKETTE $19.95
Great for making signs. Requires 132-print position printer.

UTAH software requires 128K RAM (i.e., 90K user RAM) an IBM-PC, XT, AT, PCjr, or compatible (16-bit) microcomputer with MS-DOS or PC-DOS Operating System, Rev. 2.0 or higher. For 8-bit CP/M-80 systems, ask about our **NEVADA** Software Series.

SATISFACTION GUARANTEED. If for any reason you're not completely satisfied, just return the Utah package within 15 days–in good condition, with the sealed diskette unopened–and we'll refund your money. **There's absolutely no risk to you, so why wait?–order today.**

Please send me these UTAH Software packages (IBM-PC diskette):

☐ COBOL ☐ BASIC ☐ EDIT
☐ PASCAL ☐ PILOT ☐ BIGPRINT

Send me ___ (how many) software packages: TOTAL $______
Utah COBOL Application Book 1; BIGPRINT: ______
California residents add sales tax (6% or 6½%) ______
HANDLING/SHIPPING: no shipping charge within U.S.; *Overseas* add $10 for first package, $5 each additional. ______

☐ Check ☐ VISA ☐ MasterCard

Checks must be in U.S. Dollars, drawn on a U.S. bank.

(Sorry, no C.O.D.s.)

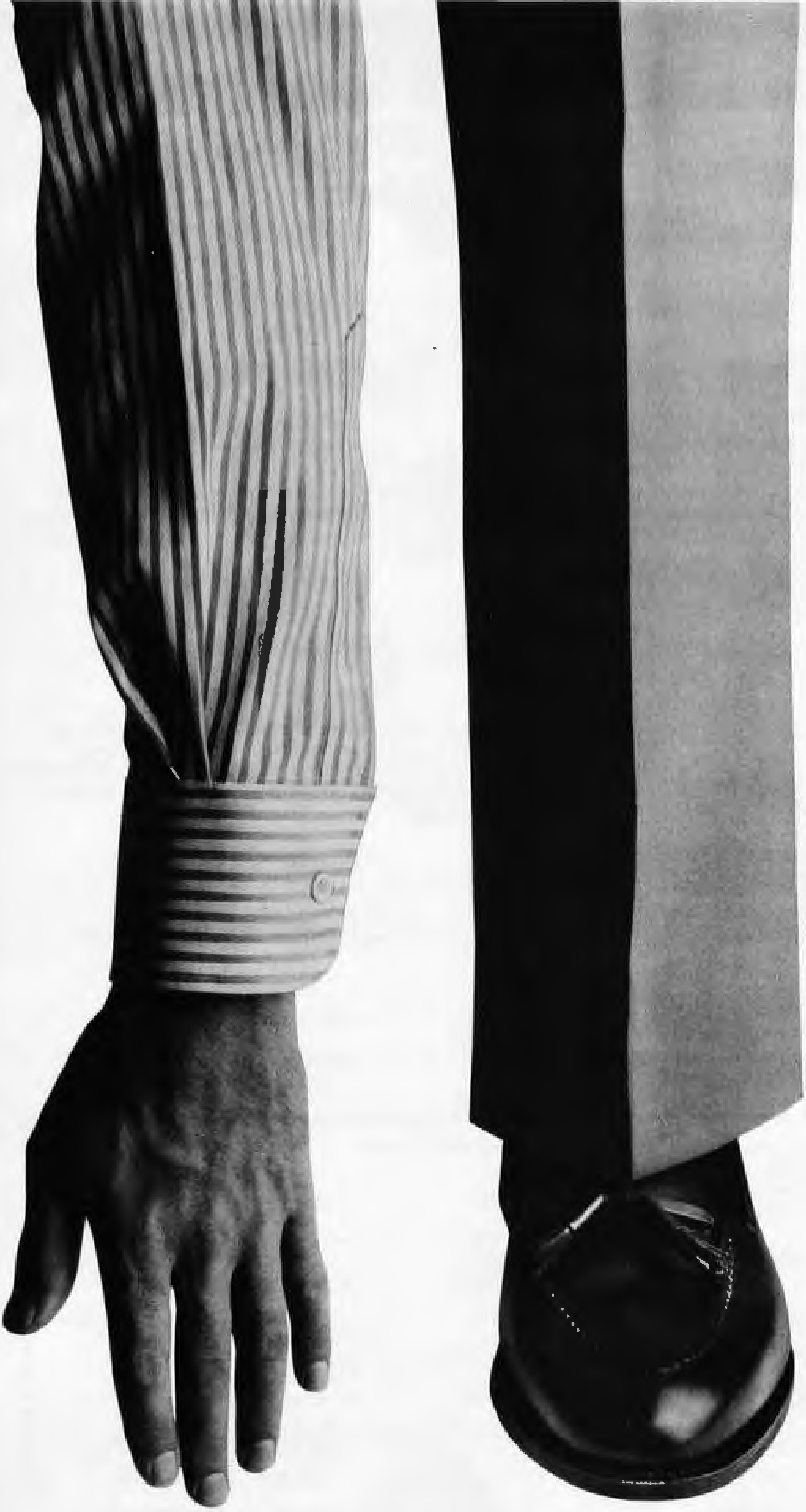
Enclosed: TOTAL $______

Send my UTAH order to:

CARD NO.__________ EXP.______
SIGNATURE__________
SHIP TO NAME__________
STREET__________
CITY/STATE/ZIP__________
COUNTRY__________

Order now from: **ELLIS COMPUTING, INC.**
3917 Noriega Street
San Francisco, CA 94122
Phone (415) 753-0186

SINCE 1977
ELLIS COMPUTING™



IF YOU'RE TIRED OF PAYING OUTRAGEOUS LONG DISTANCE CHARGES EVERY TIME YOUR COMPUTER TALKS ON THE PHONE, GTE TELENET OFFERS AN ALTERNATIVE. IT'S CALLED PC PURSUIT.[SM]

NOW FOR ONLY $25 A MONTH (PLUS A ONE-TIME INSTALLATION FEE OF $25) YOU'LL BE ABLE TO REACH OTHER PC SUBSCRIBERS, FREE LOCAL DATA BASES AND BULLETIN BOARDS IN ATLANTA, BOSTON, CHICAGO, DALLAS, DENVER, DETROIT, HOUSTON, LOS ANGELES, NEW YORK (AREA CODE 212), PHILADELPHIA, SAN FRANCISCO AND WASHINGTON D.C. AND TAKE ADVANTAGE OF LOCAL SOFTWARE DOWNLOADING CAPABILITIES, AS WELL.

YOU CAN USE THE PC PURSUIT SERVICE BETWEEN 6PM AND 7AM WEEKDAYS. AND ALL DAY ON WEEKENDS. ALL YOU NEED IS AN AUTO-ANSWER MODEM AND COMMUNICATIONS SOFTWARE.

AS PART OF GTE TELENET'S DATA NETWORK, THE PC PURSUIT SERVICE OFFERS THE QUALITY, RELIABILITY AND ACCURACY NEEDED IN THE PURSUIT OF INFORMATION.

FOR MORE INFORMATION CALL 800-368-4215 BETWEEN 9AM AND 5PM. AFTER HOURS CALL 800-835-3001 ON YOUR TERMINAL TO SIGN UP AND START USING THE SERVICE. OR USE THIS SECOND NUMBER TO GET OUR ON-LINE USERS GUIDE.

WITH PC PURSUIT, YOU CAN FINALLY GET YOUR HANDS ON THE INFORMATION YOU NEED, WITHOUT PAYING AN ARM AND A LEG.

GTE Telenet

Inquiry 77

NOW YOU DON'T HAVE TO PAY THIS MUCH
TO ACCESS INFORMATION WITH YOUR HOME COMPUTER.

IBM PC INTERRUPT SERVICE ROUTINES

Guidelines for writing an interrupt handler in a high-level language

BY PAUL M. DUNPHY

ONE OF THE most important concepts in computer technology is the idea of hardware-generated interrupts. Interrupts increase the overall efficiency of a computer system because the external devices request the attention of the processor as needed. If a system had no interrupts, the processor would have to poll every device in the system periodically to see if any of them required attention.

Unfortunately, both the hardware and software aspects of interrupt systems are poorly documented and tend to be reserved for operating-system functions such as peripheral drivers, disk handlers, etc. Programmers wishing to make use of the interrupt system for customized applications are frequently forced to resort to complex assembly-language routines that are difficult to write and understand. In this article, I will describe how to use the interrupt system on the IBM Personal Computer and demonstrate that you can write an interrupt service routine (ISR) in a high-level language (in this case, Borland International's Turbo Pascal). (Interested readers should consult "Interrupts and the IBM PC," Parts 1 and 2, by Chris Dunford, *PC Tech Journal*, November/December 1983, page 173, and January 1984, page 144.) You should be able to use these guidelines to write ISRs in other languages.

A BIT OF BACKGROUND ON INTERRUPTS

Peripheral devices can request the processor's attention for many different reasons. They may require control signals, have data available, or simply need to tell the processor that they have finished a task. Whatever the reason, an interrupt is the event that makes a processor suspend execution of its current program to perform some requested activity.

An "interrupt line" is a pin on the microprocessor chip that, when activated, causes the processor to save its current location (the processor usually does this by placing the contents of its instruction pointer on the stack) and transfer control to a fixed address. This fixed address is the beginning of the ISR. The ISR performs its tasks and returns control to the exact location at the time of the interrupt so that the interrupted routine is never aware that it was disturbed. Notice that the ISR must save any of the processor's registers before it modifies them and then must restore them before it returns. Otherwise, information that the interrupted routine had stored in those registers might be lost.

If a processor has only one interrupt line with several devices attached, then the ISR's first job is to determine

(continued)

Paul M. Dunphy is a microcomputer specialist at an oceanographic research institute. He can be reached at the Bedford Institute of Oceanography, AOL Software Systems, POB 1006, Dartmouth, Nova Scotia, Canada.

ILLUSTRATED BY JAMIE HOGAN

Most processors provide several interrupt lines and each is associated with a different ISR *start address.*

which device caused the interrupt. Most processors, however, provide several interrupt lines and each line is associated with a different ISR start address. In this arrangement, each device is serviced by its own ISR. Often, as on the IBM PC, the ISR start addresses are stored contiguously in memory in an "interrupt-vector table." This is commonly referred to as a "vectored interrupt" system.

Whether a system has one or several interrupt sources, some or all of these may be ignored or "masked out" at times when the processor is executing critical code and cannot be interrupted. In addition, when several interrupt lines are available, they are often prioritized so that the system can decide which one should be serviced first should more than one device request attention simultaneously.

THE INTERRUPT SYSTEM ON THE IBM PC

The 8088 microprocessor in the IBM PC supports a special instruction, INT, which allows programs to perform synchronous transfers to subroutines through the interrupt-vector table in a manner much like a hardware interrupt. The operating system takes advantage of these software interrupts to perform virtually all of its I/O (input/output). You should be aware of the fact that most of the documentation on IBM PC interrupts is actually referring to the INT instruction.

As for true hardware interrupts, the architecture of the IBM PC provides eight lines, designated IRQ0 to IRQ7. As delivered, the IBM PC uses three of these lines. IRQ0 is reserved for the system timer, IRQ1 for the keyboard, and IRQ6 services the NEC PD765 floppy-disk controller.

The remaining lines are not used by the system; they can be used by properly wired cards in the peripheral slots. The IBM PC interrupt lines are ordered in priority; IRQ0 is the highest and IRQ7 is the lowest. Any or all lines can be masked. To resolve priority- and control-masking operations, the IBM PC uses the Intel 8259 programmable interrupt controller (PIC). The 8088 processor can only handle one interrupt at a time, so the 8259 PIC evaluates each request and determines whether it should pass the interrupt on to the processor. An interrupt routine can be interrupted if the 8259 PIC receives a request of higher priority. On an off-the-shelf IBM PC, the clock can interrupt the disk controller but not vice versa.

Suppose the 8259 PIC receives an interrupt request, the line is not masked, and no higher-priority interrupt is being serviced. The 8259 PIC signals the processor that it has a valid interrupt request pending. The processor finishes any instruction that it is currently executing and then acknowledges the request by signaling the 8259 PIC that it is ready to service the interrupt. The 8259 PIC then adds an offset of 8 to the interrupt-request number to obtain an 8-bit interrupt-type number. For example, if the disk is requesting attention, the interrupt-type number is 14 (8 + IRQ6). This interrupt-type number is placed on the system data bus. The processor reads the data bus and uses this number as an index to access the interrupt-vector table stored in memory.

Each entry in the interrupt-vector table consists of 4 bytes. The first 2 bytes specify the ISR's segment address, and the next 2 bytes contain its offset within the segment. For IRQ0, IRQ1, and IRQ6, these addresses point to routines in the BIOS (basic input/output system) that service the corresponding system device. The entries for the remaining interrupts point to a dummy routine in the BIOS that returns control to the executing program. To add a new ISR to the table, you simply overwrite the appropriate dummy-routine pointer with a pointer to your new service routine.

ISR HOUSEKEEPING

Recall that the ISR's first job is to save all the registers it will modify in the course of executing so that the original machine state can be restored upon completion. This is best done by pushing the registers onto the stack. If you write your ISR in assembly language, you only need to preserve the registers that are actually used. However, if you write your ISR in a high-level language—Pascal, for example—you cannot make assumptions about which registers the compiler will use. Therefore, your ISR must save all registers just in case. Turbo Pascal has an inline statement that allows you to insert machine code directly into the program. You can use inline to generate the PUSH instructions at the beginning of the ISR, and again at the end to generate the POP instructions.

Now that you know how to preserve the registers, can you proceed to write the actual code that will service the interrupt? Not yet. First, there is a subtle problem associated with the 8088's segmented architecture that you must take into account. Remember, an ISR is invoked asynchronously. This causes no problems as long as the interrupted code is somewhere in the user program. However, most high-level languages (Turbo Pascal included) make use of the system BIOS to perform I/O. Suppose the program is using the BIOS when it is interrupted. Control will be transferred to the user ISR, but the data segment (DS) register will contain the BIOS data segment. If the code in the ISR attempts to access any Pascal variable, the DS register will be incorrect and the actual referenced memory locations will be somewhere in the BIOS rather than the Pascal program. Needless to say, this will cause the program to go haywire, particularly if the routine attempts to modify these variables.

How do you overcome this problem? You must restore the Turbo Pascal data segment after you push the registers. There are probably several ways to do this, but I have found the following method adequate. A Borland representative has assured me that words 5 to 15 of the code segment contain nonexecutable code, and the program will not be corrupted if data is stored in this area. Therefore, in the main program you create an absolute variable in this area, and, using Turbo Pascal's Dseg function, your first executable statement saves the DS address in this variable.

Now that the DS segment has been determined and saved in a known absolute location, our ISR can restore it each time it is invoked. First, copy the code segment into the DS register: this allows you to read the contents of the absolute variable into the AX register. Then transfer the AX to the DS register to complete the operation.

There is one more thing to do before the fundamental "shell" of the ISR is complete. Just before it pops the registers and returns, it must tell the 8259 PIC that it has finished. The 8259 PIC maintains an 8-bit register in which it records which ISRs are in service. The associated bit for the ISR must be set to zero when the routine completes. To do this on the IBM PC, send a hexadecimal 20 to the control port of the 8259 PIC. This is called a nonspecific end of interrupt (EOI). When the 8259 PIC receives a nonspecific EOI, it clears the bit belonging to the highest-priority interrupt in service. This will always be the routine that sent the command.

On the IBM PC, the port addresses of the 8259 PIC are 20 and 21 (hexadecimal). You can access I/O ports directly from the high-level code using Turbo Pascal's predefined array port. You now have constructed the basic framework for the ISR on the IBM PC using Turbo Pascal (see listing 1).

Initializing the System to Handle User Interrupts

Now that you know how to construct an ISR, you must prepare the system to process the interrupts. First, you have to tell the 8259 PIC that your IRQ line is not to be masked. The 8259 PIC maintains an 8-bit register called the interrupt-mask register (IMR). Each bit from 0 to 7 corresponds to an interrupt-request line. If the corresponding bit is set to 1, the associated IRQ line is masked and subsequent interrupt requests on that line will not be passed on to the processor. When MS-DOS initializes the 8259 PIC at boot-up, it sets all the bits in the IMR to 1 except those corresponding to IRQ0, IRQ1, and IRQ6. To activate your IRQ, you must clear the appropriate bit. Listing 2 contains the Turbo Pascal procedure for doing this.

Your next step is to modify the interrupt-vector table so that the appropriate entry will point to your ISR. You can find the address of the ISR by

(continued)

Listing 1: *The basic structure of a Turbo Pascal ISR. The order and number of registers saved differs slightly from what is suggested in the Turbo Pascal reference manual, but I have found that this method is necessary to ensure proper operation.*

```
Program Test(input,output);

  var
    dsave       : integer absolute Cseg:$0006;

  Procedure Interrupt_Service_Routine;

    begin
      inline($FB/          { STI       enable further interrupts     }
             $1E/          { PUSH DS                                 }
             $50/          { PUSH AX                                 }
             $53/          { PUSH BX                                 }
             $51/          { PUSH CX                                 }
             $52/          { PUSH DX                                 }
             $57/          { PUSH DI                                 }
             $56/          { PUSH SI                                 }
             $06);         { PUSH ES                                 }

      inline($8C/$C8/      { MOV AX,CS  restore data segment         }
             $8E/$D8/      { MOV DS,AX                               }
             $A1/dsave/    { MOV AX,dsave                            }
             $8E/$D8);     { MOV DS,AX                               }
                 .
                 .

      { Pascal code for servicing the interrupt                      }
                 .
                 .

      port[$0020] := $20;  { nonspecific EOI to 8259 PIC             }

      inline ($07/         { POP ES                                  }
              $5E/         { POP SI                                  }
              $5F/         { POP DI                                  }
              $5A/         { POP DX                                  }
              $59/         { POP CX                                  }
              $5B/         { POP BX                                  }
              $58/         { POP AX                                  }
              $1F/         { POP DS                                  }
              $CF);        { IRET      return from interrupt         }
  end

begin
  dsave := Dseg;
end.
```

With the exception of a few in-line machine instructions, an entire ISR can be written in Pascal.

using the Turbo Pascal functions Cseg and Ofs. However, the compiler adds a few instructions of overhead to the start of every procedure. Consequently, the first instruction of the compiled ISR is not the STI instruction that is inserted by the inline statement. The actual start of the ISR is 7 bytes prior to the first statement of source code. Since the instructions added by the compiler use the registers before you have a chance to save them, you must add 7 to the offset returned by Ofs so they will not be executed. The vector table is then modified using the predefined Turbo Pascal array memW. Listing 3 contains the procedure for doing this.

PUTTING IT ALL TOGETHER

To demonstrate the use of the IBM PC interrupt system using Turbo Pascal, I wrote a program called Acquire. [*Editor's note: The source code is available via BYTEnet Listings. Call* (617) 861-9774 *before November 1; thereafter, call* (617) 861-9764.] This program reads data from a cassette-tape reader transmitting 16-bit words at 2000 hertz and writes it to disk. No handshaking is possible because the tape reader simply places the data on the input lines and strobes the receiving device once every 500 microseconds.

To read the data, I chose the Intel

Listing 2: *The Turbo Pascal procedure for enabling an interrupt. Byte variable IRQ selects which interrupt line is to be "unmasked."*

```
Procedure Enable_IRQx(IRQ : byte);

  var
    imr, mask     : byte;

  begin
    mask        := not ( 1 shl IRQ );
    imr         := port[$21];          { Get IMR from 8259       }
    imr         := imr and mask;       { clear mask bit          }
    port[$21]   := imr;                { and return to controller }
  end;
```

Listing 3: *The Turbo Pascal procedure for setting the interrupt-vector table. The variable entry selects which entry in the table is to be set. Note that entry 0 corresponds to interrupt line 0, etc.*

```
Procedure Set_IVT(entry : integer);

  var
    offset, segment, first_word, second_word : integer;

  begin
    offset := Ofs(Interrupt_Service_Routine) + 7;
    segment                          := Cseg;
    first_word                       := (entry + 8) * 4;
    second_word                      := first_word + 2;
    memW[$0000:first_word]           := offset;
    memW[$0000:second_word]          := segment;
  end;
```

8255 programmable parallel port on the Tecmar Lab Master board. My program reads two 8-bit quantities from ports A and B of the 8255 to form the 16-bit data word. I programmed the 8255 to generate an interrupt when it has latched the data and jumpered the board to use IRQ7.

The program Acquire concurrently collects data from the Tecmar board and writes it to disk. Because the capacity of the floppy disk is limited to 360K bytes, Acquire terminates when the disk is full.

To collect data and simultaneously transfer it to disk, Acquire maintains two large buffers of 12,800 16-bit words each. The program actually incorporates two interrupt service routines: one fills the first buffer, the other fills the second. The interrupt-vector table is initially set so that it points to ISR 1. When the first buffer is full, the vector table is altered to point to ISR 2. While the second buffer is filling, data from the first buffer is flushed to disk.

CONCLUSION

I have shown that, with the exception of a few in-line machine instructions, an entire interrupt service routine can be written in Pascal. The examples in the listings should serve as templates for programmers who wish to develop their own interrupt handlers. ■

"INSIDE IBM" SPORT-SHIRT!

The classic November '83 Byte cover—and boy, does it look great on this ¾ sleeve "baseball shirt"! The vivid royal blue sleeves and neckline really make the design jump out. And don't mistake this for one of those rubbery patches that cracks and peels off after a few washings. This is true four-color process: the inks are silk-screened into the fabric of the shirt, resulting in a beautiful, full-color image that lasts.

You'll also appreciate the shirt itself: a real heavy-weight made of 50% cotton, 50% polyester. You'll enjoy cotton comfort in a tough, sporty shirt that keeps its crisp, fresh look through many washings—with almost no shrinking! The price for each "Inside IBM" Sport-Shirt is only $12.50 ($11.50 each for 3 or more). Your order will be shipped within a week.

Please send me the following shirt(s) at$12.50 each, or $11.50 each for 3 or more. I have included $2.00 for shipping and handing.

Qty.	Size	Amount
______	Adult—Extra Large	$______
______	Adult—Large	$______
______	Adult—Medium	$______
______	Adult—Small	$______
______	Child—(sizes 10-12)	$______
	Shipping and Handling	$2.00
	(Overseas add $3.00)	$______
	TOTAL	$______

☐ I have enclosed check or money order.

☐ VISA ☐ MasterCard ☐ Free Color Brochure

Card #: ______________________

Exp. Date: ______________________

Ship my shirt(s) to:

Name: ______________________

(Business): ______________________

Address: ______________________

City: ______________________

State: ____________ Zip: ____________

Mail this coupon to:

Robert Tinney Graphics
1864 North Pamela Dr.
Baton Rouge, Louisiana
70815

For VISA or MasterCard Orders
or for more information
Call 1-504-272-7266

Your System/Controller should fit you like a good suit.

LET MICROMINT CUSTOM FIT YOU

Whether it's suits or system controllers, you can't buy off the rack when you need a custom fit.

That's why Micromint individualizes its system controllers to meet your particular needs and budget. What's most important to you? Software compatibility? Speed? Everything on one board? Economical computing power?

#1: *"I want software compatibility."*

Solution:

THE SB180 COMPUTER/CONTROLLER

The SB180, only 4" by 7½", offers a Z80 compatible CPU running at 6MHz, 256K bytes of RAM, up to 32K bytes of ROM, two serial ports, a parallel port, Z80/6800 I/O expansion bus, and an industry standard 765A-compatible disk controller for up to four disk drives — any combinations of 3½", 5¼" or 8" drives. The SB180 is based on the Hitachi HD64180 CPU, a microcoded CMOS chip which provides high performance, reduced system cost, and low power operation while maintaining complete compatibility with the large base of standard CP/M software.

SB180-1 *w/8K ROM monitor* . . *$369.00*
SB180-5 *w/8K ROM monitor, BIOS source and Z-system* *$418.00*

#2: *"I need speed."*

Solution:

THE Z8 FORTH SYSTEM/CONTROLLER

The Z8 FORTH System/Controller is only 4" by 4½" and includes a custom masked Z8 version of the FORTH language with a full screen editor, cassette I/O driver primitives, EPROM programmer primitives, and other utility words. It also contains up to 4K bytes of RAM or EPROM, an RS-232 serial port with selectable baud rates, and two parallel ports. Additional Z8 peripheral boards include memory expansion, a smart terminal board, serial and parallel I/O, real time clock an A/D converter, and an EPROM programmer. It's perfect for data reduction and high speed control applications.

BCC21 *w/utilities . . . $225.00*

#3: *"Let me have an entire development system on one board."*

Solution:

THE BCC52 SYSTEM/CONTROLLER

The BCC52 is a new stand alone single board microcomputer which is bus compatible with the Micromint BCC11 /BCC21 Z8 System/Controllers and expansion boards. The BCC52 features the Intel 8052AH-BASIC microprocessor which includes a ROM resident 8K byte floating point BASIC interpreter with extensions for process control work. It contains sockets for up to 48K bytes of RAM/EPROM, an "intelligent" 2764/128 EPROM programmer, 3 parallel ports, a serial terminal port, and a serial printer port.

BCC52 . . . *$239.00*

For a System Controller suited to your needs, give us a call.

#4: *"Give me lots of economical computing power."*

Solution:

THE BCC11 BASIC SYSTEM/CONTROLLER

The Z8 BASIC System/Controller is nearly identical to the FORTH System/Controller but contains a tiny BASIC interpreter, up to 6K bytes of RAM and EPROM, an RS-232 serial port with switch selectable baud rates, and two parallel ports. Add a power supply and terminal to start programming in BASIC or machine language. Programs can be transferred to 2732 EPROMS with the optional EPROM programmer for auto-start applications. It can also use any of the expansion boards mentioned under the Z8 FORTH System/Controller.

BCC11 . . . *$149.00*

Additional information on peripheral boards and OEM pricing is available.

Order Toll Free
1-800-635-3355
In Connecticut call: 1-871-6170

MICRO MINT

MICROMINT, INC.
25 Terrace Drive
Vernon, CT 06066

Actually, we give you two things free.
Our source code. And your freedom.

Just buy part or all of our excellent integrated business accounting system, the SBT Accounting Software Library.

We'll give you our source code absolutely free.

Which, in turn, gives you the freedom to customize our software to fit your business needs.

Say, for instance, you want to change the way a management report is formatted. Our free source code enables you to change it.

What's more, the change will be quick and simple because our software is written in easy-to-use dBASE.

In fact, the entire SBT Accounting Software Library runs with dBASE III or dBASE II,* so you get the power and flexibility of those best-selling programs. Plus the freedom to use any computer that runs dBASE.

The SBT Accounting Software Library. Great software and freedom. All in the same box.

Call today for our demo disk and brochure. (415) 331-9900.

THE SBT ACCOUNTING SOFTWARE LIBRARY.		
dProfessional	Time & Billing	$395
dOrder	Sales Order Processing	$195
dInvoice	Billing/Inventory Control	$195
dStatement	Accounts Receivable	$ 95
dPurchase	Purchase Order	$195
dPayable	Accounts Payable	$295
dPayroll	Payroll/Labor	$395
dLedger	General Ledger/Finance	$395
dAssets	Asset/Depreciation	$195
dProject	Project/Job Accounting	$395
dBackup	Menu/Backup	$ 65

sbt CORPORATION

Three Harbor Drive
Sausalito, CA 94965
(415) 331-9900

Call today for the name of the SBT consultant in your area.

Free source code in every box.

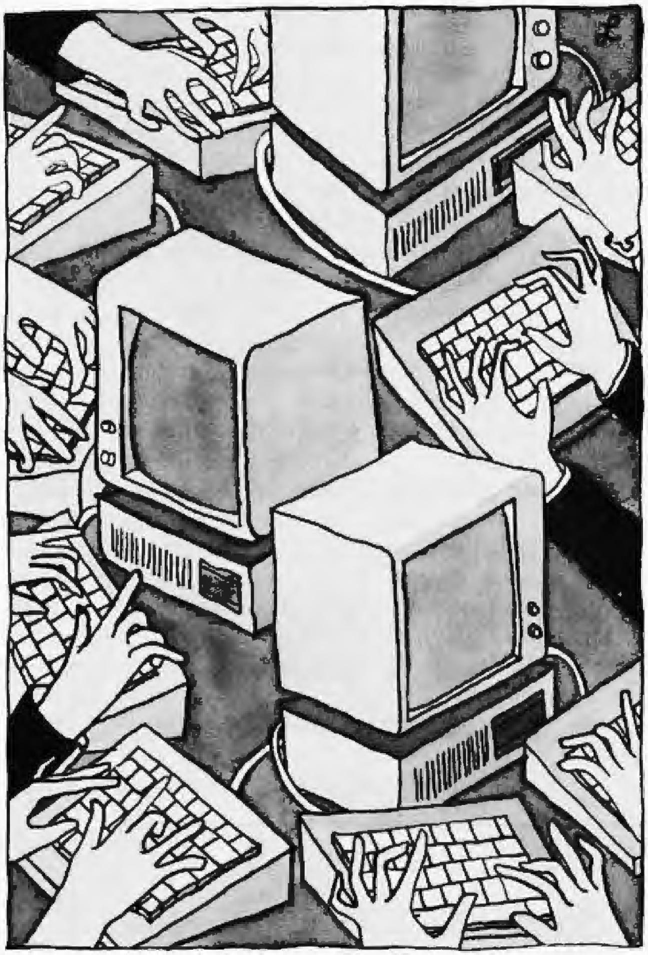
*Compiled versions are also available. dBASE III and dBASE II are registered trademarks of Ashton-Tate, Inc. Copyright 1985, SBT Corporation.

Inquiry 176 for End-Users. Inquiry 177 for DEALERS ONLY.

PICK, COHERENT, AND THEOS

Multiuser systems on the IBM PC XT

BY MARC J. ROCHKIND



PICK, COHERENT, AND THEOS are all multiuser, multitasking operating systems for the IBM PC XT. [*Editor's note*: THEOS *is the new name for the* Oasis *operating system. In this article,* THEOS *refers to* Oasis86. *The name was changed this year.*] None of them can run MS-DOS programs, so you can't use them with the familiar spreadsheet, word-processing, and other packages for which most people buy XTs. These operating systems are intended for more specialized purposes. Pick and THEOS are designed as a basis for multiuser database applications. Coherent is more general in principle (it's a UNIX clone), but its most appropriate use is for program development in C.

Because these operating systems are so different from each other and are designed for different purposes, I won't compare them feature for feature. Instead, I'll discuss them separately and evaluate them only in terms of their intended uses.

I tested release 1.3 of Pick (from Pick Systems, 1691 Browning, Irvine, CA 92714, (714) 261-7425), release 2.3.43 of Coherent (from Mark Williams Co., 1430 West Wrightwood Ave., Chicago, IL 60614, (312) 472-6659), and release 7.0 of THEOS (from THEOS Software Corp., Suite 100, 201 Lafayette Circle, Lafayette, CA 94549, (415) 283-4290). Pick and Coherent sell for $495 (including programming language). THEOS sells for $595 without a language; C or BASIC costs about $400 more. None of these operating systems are copy-protected.

I tested the systems on my IBM PC XT, which consists entirely of IBM parts (even the memory board). It has a 10-megabyte hard disk, a 360K-byte floppy, 512K bytes of RAM (random-access read/write memory), and a monochrome display. I used a Zenith Z-19 terminal (at 9600 bps) to exercise the multiuser features of each operating system. My printer is connected to the parallel port.

All three systems supposedly work on many other hardware configurations besides a plain-vanilla XT, but I didn't try them on any IBM-compatibles or with other kinds of hard disks.

These operating systems are too complex for me to claim to have tested them thoroughly. In the few days I spent with each, I tested about 10 percent of their functions. With such a sampling, a defect gets emphasized more than it should in view of the overall capabilities of the product.

OVERVIEW OF PICK

Pick is a special-purpose operating system designed to run the Pick database. In fact, when you use Pick you do not encounter the usual boundary

(continued)

Marc J. Rochkind is president of Rochkind Software Corp. (3080 *Valmont Rd., Boulder,* CO 80301), *which markets the business programming language* RIDE. *He is the author of* Advanced UNIX Programming (*Prentice-Hall,* 1985).

ILLUSTRATED BY LYNNE BUSCHMAN

between operating system and database system. For example, Pick doesn't have an operating-system command to list the files in a directory. Instead, you use a data-management facility to list the entries in your account dictionary. Pick terminology is different, too: Commands are called verbs, records are called items, fields are called attributes, and so on.

For anyone used to more traditional operating systems, the Pick approach seems strange. But after a day or so, I got used to it.

To me, the most remarkable aspect of Pick is that it tends to have just one of each function or feature rather than several. It has only one text editor that edits source files, memos, data records, and dictionary entries. There's only one file type for data; an item can be a data record, in which case the fields have their usual meaning, or it can be a source program or document, in which case each field corresponds to a text line. Its uniformity of treatment and simplicity of design make the Pick system easy to understand once you master the basic concepts. They undoubtedly contribute to the efficiency and reliability of Pick as well.

Internally, Pick implements a virtual computer in two senses. First, data and programs are accessed via a virtual-memory system that divides real memory into a large number of 512-byte frames. Data and pieces of program are moved between disk and real memory as needed. Second, programs are compiled into Pick assembly language, which is not 8088 code but the code for a virtual computer implemented by an interpreter deep within the Pick kernel. This interpretation is necessary for the virtual memory to work because the XT lacks virtual-memory hardware. In addition, the interpreter protects users from interfering with one another. This is a problem with other multiuser operating systems that run on the XT because the machine has no memory protection of its own.

The Pick database is structured hierarchically as shown in table 1. For further information on Pick file structures, see "The Pick Operating System, Part I: Information Management" by Rick Cook and John Brandon, October 1984 BYTE, page 177.

The flexibility of items, attributes, values, and subvalues lets you organize data differently with Pick than in many other database systems. For example, in a medical office database you might have a patient record and an additional detail record for each visit. However, Pick can keep all the patient's data in a single item. It might have an attribute called, for example, Visit, and you could add an additional value with each visit. This organization is particularly easy to handle from within Pick programs because the entire item is read and written as a unit.

You don't have to specify sizes when you allocate a file or define an attribute. Attributes have a length that is used for printing reports, but that length does not limit the amount of data that you can insert. In fact, you can insert data into records without even defining the attributes. All good database systems allow some flexibility in handling data, but few go to the extremes of Pick.

The Pick database supports record locking but not transactions (a transaction is a sequence of database updates that are to be kept atomic). If a program aborts before a transaction is completed, the partial changes remain intact. There's also no transaction log to restore the database to its current state if a disk fails. You must make backup copies of the database at regular intervals, and if the database is lost, you will lose all updates since the last backup. This naive approach to database recovery is standard for microcomputers, but I expected more from Pick.

The Pick system comes with several major subsystems. The command processor (shell) is called TCL (terminal-control language). It's nothing special—it just processes commands a line at a time.

ED is a line editor, not a screen editor. You edit using line numbers, and you can't see your changes in context unless you issue a list command. After inserting or deleting a line, you can't do much else without getting a SEQN? message from the editor. You have to issue an F command to make your updates active before you can proceed. It's bad enough that the Pick user has to struggle with a line editor, but this one is the worst I have ever used.

The PROC processor is used to collect command lines, including those for interactive commands like ED, into stored procedures. PROC can be quite elaborate; it has features for I/O (input/output), branching, terminal prompting, and so on. You can think of the PROC processor as a command-level programming language, similar in concept but not in design to the UNIX shell.

The query and report-generation

Table 1: *The Pick database hierarchy.*

System dictionary	One per system; contains account names, passwords, and other administrative information.
Master dictionary	One per account (user); contains a user-specific vocabulary of commands and the user's filenames, so it acts as a log-in directory.
File dictionary	Many per account; contains definitions of fields and relationships among them.
Data file	One or more per data dictionary; contains data.
Item	Many per file; contains data for one record, up to 32,767 bytes. Might be a data record, source program, etc. Each item has a unique ID. All Pick files are accessed by hashing.
Attribute	Many per item; corresponds to a field of a record, a line of text, etc.
Value	One or more per attribute; can be used to record multiple instances of an attribute (e.g., names of dependents).
Subvalue	One or more per value; similar to value but one level deeper.

language is called ACCESS. It has facilities for selecting, sorting, adding and counting (with control breaks), and listing—the usual abilities that you expect to find in a sophisticated database system. You can operate on only one file at a time, so the system can't do a relational join. For example, if you have an employee-organization file and a sales file, in most cases you can't use ACCESS to answer queries like, "Who are the supervisors of the salespeople who did not meet their quotas?" An exception occurs when the item IDs in one file are the same as those in another; for example, if both the employee-organization and sales files used the employee ID as their item ID, one query can create an item list and the second query can use it for selection. If you use such queries, you must take considerable care in designing the database to ensure that the item IDs are strategically chosen.

Pick has a rudimentary text formatter called RUNOFF. When used with ED, it makes a good 1960s-style word-processing system.

Finally, the one and only programming language for Pick is PICK/BASIC. This language bears little resemblance to the awkward Microsoft BASIC. It's actually a fine little language.

Among the major features in PICK/BASIC that aren't in Microsoft BASIC are typeless variables, strings up to 32,767 bytes long, subroutines with arguments, built-in database functions, multiline structured control statements, and no line numbers.

You handle a database item by reading it into or writing it from a string variable. You can reference attributes, values, and subvalues by treating the string as a three-dimensional array. For example, if R holds a data item, then R<4,3,2> is a reference to the second subvalue of the third value of the fourth attribute. Such an expression can appear on either side of an assignment statement.

Installing and Using Pick

Installation of Pick went smoothly. I booted the XT with the first disk, then followed the on-screen directions to load the remaining four disks. I didn't even have to create a partition on my hard disk; Pick automatically found an unused stretch of cylinders and made it the Pick partition. Therefore, if you also want MS-DOS or any other operating system, you have to install Pick last.

Pick is the only multiuser system I have ever seen that tells you how to wire the cable for remote terminals. Not only that, but the wiring diagram was correct. However, the list of supported terminals is short and strange (e.g., the VT-100 isn't listed).

The problem with the installation instructions is that they stop after telling you to insert the last disk. I was presented with a log-in prompt and had no idea how to respond. (I tried MARC and it didn't work.) After searching the manual for 20 minutes, I learned that I should log in as SYSPROG (Pick takes only uppercase). Then I was able to create an account for MARC with the CREATE-ACCOUNT verb.

Next I created a file named PGM and wrote a small PICK/BASIC program (painfully, with ED) into the item TST1. When I tried to compile it with the BASIC command, I received a complaint about an invalid source format. After more searching in the manual, I found a note in the file-structure section to the effect that in order to create a program file, I must use the editor to "change the D-pointer in the master dictionary to a DC-pointer." I had only a fuzzy idea of what this meant, but I was able to use ED to edit the master dictionary, find a D, and change it to a DC. I still don't know what this was all about, but it worked. I think the Pick people should have used their PROC language to supply a procedure called CREATE-PROGRAM-FILE.

In the long run, problems like these don't matter. You get used to them and you forget how silly they are. But they frighten you when you're getting started and need all the confidence you can muster.

You can boot Pick off the XT's hard disk. When it first comes up, it spends several minutes doing something called "verifying system nodes," during which time you can't use the XT's keyboard. However, the remote terminals are activated and you can use them right away. I don't know if modifying the database while it is being checked is safe.

Pick can't read or write MS-DOS floppies, nor can it access the MS-DOS partition on the hard disk. To get to MS-DOS, you have to shut down Pick and boot MS-DOS. To get back, you have to reboot Pick and wait for it to verify those system nodes.

I only used Pick for a few days, and I never tried anything really sophisticated. But it didn't crash, and I didn't find any bugs. My feeling is that it is very solid.

Pick Documentation

The Pick documentation is well written and full of helpful examples. From what I could tell, it's also accurate and complete. The manual has a good index, but it's hidden in the system maintenance section where novices are sure not to find it. A tiny tutorial section covers so little of Pick that I didn't find it useful.

The Pick Systems people teach courses that show you how to use Pick effectively. The courses last a week and cost about $900 each. Since Pick is so different from traditional operating systems, these courses might be worthwhile.

Pick Performance

To get some feel for Pick's performance, I compared it to Revelation running under MS-DOS. Revelation (a product of Cosmos Inc., 19530 Pacific Highway S, Seattle, WA 98188, (206) 824-9942) is a single-user database system patterned on Pick.

I ran a program that writes 3000 records of 150 bytes each to a file that is initially empty. [*Editor's note: The benchmark programs used in this article are available for downloading from* BYTEnet *Listings. Call* (617) 861-9774 *before November* 1. *Thereafter, call* (617) 861-9764.] Each record has a 20-byte ID. When I created the file, I followed the recommendations in the Pick

(continued)

The important features of Pick are very well designed.

manual and determined that there should be 503 hash buckets. It took Pick 8 minutes and 20 seconds to run this program, and the resulting file consisted of 531,000 bytes.

Revelation ran the identical program (except for a small syntax change to the OPEN statement) in 21 minutes and 58 seconds and created a file of 1,016,832 bytes. This file size includes lots of space for additional records, so it can't be compared to the file size for Pick.

The times don't tell the complete story. During the Pick run, the disk light came on only now and then at the beginning, and then it blinked on and off steadily. During the Revelation run, the light stayed mostly on, and I could hear the disk seeking constantly. My disk had never had such a sustained workout.

Don't interpret these results to mean that Pick is three times faster than Revelation/MS-DOS. They're based on only one program. I'm sure Pick is faster, but exactly how much I don't know.

Also, bear in mind that Revelation has advantages over Pick: It's well integrated into the MS-DOS environment (you can read and write MS-DOS files, and you can execute MS-DOS commands from within Revelation), and many of its subsystems are better designed (there's a screen editor, for instance). It might make sense to develop your application under Revelation and then move it to Pick when it's ready for production use.

Conclusions on Pick

Many of the Pick operating system's less important features are badly designed, but its important features are very well designed, particularly its file structures and the PICK/BASIC language. Pick is simple and powerful, and it seems to be efficient and reliable, too. It does exactly what it was designed to do.

Pick merits careful consideration if you are planning to use XTs to run dedicated database applications. Because it works well as a multiuser system, it's probably the most cost-effective way to use an XT.

Overview of Coherent

Coherent appears to be nearly a clone of UNIX Version 7, an older release of UNIX that has since been replaced by System III and System V. I write "appears to be" because the Coherent manual doesn't say it is based on UNIX. The failure to mention UNIX has a practical disadvantage: No advice is given on how to port Coherent programs to the various UNIX versions, something that many Coherent programmers will want to do.

As a UNIX clone, Coherent is amazingly complete. It includes even advanced features like yacc (a parser generator) and awk (a report-generation language), but it lacks many commands that are part of Version 7. Some major commands that are missing include f77 (FORTRAN), bas (BASIC), troff (typesetter formatter), eqn (equation processor), tbl (table formatter), lint (C checker), uucp (file-transfer program), and plot (plotting program). There are also 19 other missing commands.

On the other hand, Coherent includes about 20 commands not present in Version 7, including kermit, which substitutes for cu and uucp, and dos, which allows reading and writing of MS-DOS floppies (but not the MS-DOS hard-disk partition).

Two screen editors, trout and elle, are based on EMACS. The difference between them wasn't clear to me (I used trout), but I was told by a technical-support person at Mark Williams that trout is easier to use and elle is more robust. In my opinion, either editor is far superior to the UNIX editor vi because they avoid the command-mode/insert-mode problems that make vi a pain to use. For UNIX old-timers who want to get started in a hurry, the ed line editor is there, too.

Some Coherent commands have the same name as their UNIX counterparts, but they are not equivalent. For example, the Coherent nroff is much less powerful than the real thing. Of the 77 requests in the Version 7 nroff, only 31 are present in Coherent (the most useful 31, however).

Coherent has all the Version 7 system calls except nice (which sets a process's priority), and they seem to be used in the same way. It should be easy to port C programs between Coherent and UNIX Version 7.

Installing and Using Coherent

Coherent's installation procedure is much less automated than Pick's. I was asked to make lots of decisions about the sizes of file systems without knowing exactly what the impact of my decisions would be. I was warned that the root partition of my hard disk would be overwritten, but the manual didn't define that term.

Information about leaving disk space for an MS-DOS partition is at the end of a rather long installation section in the manual, where you might not see it until too late. You must install Coherent first, leaving some space for MS-DOS, and then use the MS-DOS FDISK command to create the MS-DOS partition. If you already have MS-DOS installed, you have to calculate cylinder numbers carefully when you install Coherent. You can't boot Coherent off the hard disk; you need to use a boot floppy.

I doubt that most people who aren't XT and MS-DOS experts will be able to successfully install Coherent along with MS-DOS. But if you don't care about MS-DOS, you can just do the installation blindly and Coherent will take over the whole disk with suitable default values for the various file systems. The installation instructions don't tell you how to wire a cable for a terminal, but three wires seem to be enough (pin 1 straight through and 2 crossed with 3). Only VT-52 and Z-19 terminals are supported. There is no terminal-capabilities facility to support more terminals.

Although Coherent is a multiuser system, you might not want to use it that way, at least for program development. The XT has no memory protection, and it's easy for one user to bring down the whole system. However, if you're running only debugged programs, multiuser access should be safe enough.

You have to be careful when shutting down Coherent. You must kill the init process and then issue a sync command to flush the buffers to disk. Coherent has no shutdown command, as do many other UNIX implementations, but you can write your own.

I could read MS-DOS files off a floppy disk easily enough once I figured out the proper name for the floppy device file. There are different names for single- and double-sided disks, and for 8 and 9 sectors per track. These names are not given in the dos command write-up, but in a separate write-up for fd.

If you get stuck—and you probably will—you can call a toll-free number to get help. The support people were too busy to talk to me when I called, but they called back within a few hours and the person who called knew what he was talking about.

After I installed Coherent, I played around with it awhile and became convinced that it's close enough to UNIX to qualify as a clone. If you sat a UNIX expert down at the keyboard without telling him or her that Coherent was running, he or she would think it was the real thing.

Coherent didn't crash during the few days I used it, but I did have trouble accessing it from a terminal. The terminal would lock up after a while. From the console, I logged in as a superuser and killed the program I was running at the terminal; then the terminal came alive again. However, one time the shell was running at the terminal and I couldn't kill it (I sent it a true kill signal, not a software termination signal). When you can't kill a process, it means the kernel has a bug. I don't think the problem was with my hardware, because I used the same terminal with Pick for many hours with no problems at all.

Coherent Documentation

Coherent comes with two fat binders full of beautifully typeset UNIX-style documentation. There are individual manual pages for commands, subroutines, system calls, and device files. Several major subsystems (e.g., nroff and trout) also have their own manuals.

However, there is no manual on C and hence no information about sizes of types, signed/unsigned arithmetic, register variables, and assembler interfacing. A separate MWC86 user's manual is referenced, but it isn't supplied with Coherent.

Most manual sections have separate indexes, and an index also covers the individual manual pages. The manual seems reasonably complete, accurate, and well written, but it would be unintelligible to anyone who doesn't already know UNIX. Fortunately, many textbooks on UNIX can fill that gap.

Coherent Performance

I ran five benchmarks to compare the efficiency of Coherent to PC/IX (a UNIX System III product from IBM). Some of these benchmarks were also run on MS-DOS and THEOS (see table 2).

I had a problem with the program I used for benchmarks 4 and 5. This is a 2000-line, three-file C program that implements a B-tree access method. The Coherent C compiler failed to compile two of the three source files. These files had previously compiled successfully on five different C compilers or computers ranging from the XT to the VAX-11/780. One file caused a fatal compiler error with the message "no match, op = 65" and a dozen or so lines of debugging information that looked like a parse tree, one of the more interesting error messages I've seen. The other file caused the fatal error "more than 20 stores."

I called the technical-support number and was told that in the first case a type cast I was using wasn't handled by the compiler, and in the second case the compiler ran out of registers. After changing the source a little, I was able to compile the files, and the resulting program ran correctly without further incident.

Since most users of Coherent are likely to want it for C program development, the compiler's inability to handle perfectly legal C programs is

(continued)

Table 2: *Benchmarks comparing* THEOS, *Coherent*, PC/IX (UNIX *System* III), *and* MS-DOS. *Times are in minutes and seconds. User time is the* CPU *(central processing unit) time spent executing instructions in the program itself; system time is the* CPU *time spent executing instructions in the operating system kernel on behalf of the program; real time is the total elapsed (wall clock) time.*

Test		THEOS	Coherent	PC/IX	MS-DOS
Shell program	User	—	19:24	3:58	—
(benchmark 1)	System	—	3:25	1:24	—
	Real	—	28:33	9:22	—
Random I/O	User	—	1:44	0:39	—
(benchmark 2)	System	—	0:20	1:28	—
	Real	2:55	2:35	2:28	3:24
Sequential I/O	User	—	4:42	1:40	—
(benchmark 3)	System	—	1:05	0:59	—
	Real	8:57	7:22	3:20	7:07
C compile and	User	—	3:52	6:32	—
link (benchmark 4)	System	—	0:32	0:33	—
	Real	—	5:20	8:05	7:18
B-tree install	User	—	0:52	0:43	—
and fetch	System	—	0:19	0:20	—
(benchmark 5)	Real	—	1:43	1:27	2:47

Coherent is based on an obsolete version of UNIX, *but the concepts are the same.*

disturbing. You can expect to have to mess around a little with C programs to get them to run. The technical-support people were familiar with the problems I was having, so perhaps the compiler will be improved in a later release.

For the shell program (benchmark 1), Coherent is much slower than PC/IX. I don't know why this is. The Coherent C compiler and linker was much faster than either the PC/IX compiler and linker or the Lattice C compiler and the MS-DOS linker, but since these compilers are different programs that produce different output, the time differences don't necessarily say anything about the speeds of the operating systems. Coherent is nearly as fast as PC/IX for the random I/O and B-tree tests (benchmarks 2 and 5), but these times are governed more by the hard disk than by the operating system, which can't help much on random I/O.

Clearly, Coherent is slower than PC/IX, although the difference varies and might be insignificant for your application. If you're doing program development, the fast Coherent C compiler will be a boon.

CONCLUSIONS ON COHERENT

If you want to learn UNIX and you have some space on your XT, you should consider buying Coherent. The price of $495 is a bargain. Many of the details are different and it's based on an obsolete version of UNIX, but the concepts—the hardest part to learn—are the same.

Coherent is probably a better C program-development environment than MS-DOS, but its C compiler needs improvement. Coherent is a weaker development environment than a complete UNIX system because it lacks tools such as lint and the Source Code Control System (SCCS).

Coherent costs half as much as PC/IX and requires less memory. Although it is less complete and runs more slowly, it's a good buy.

OVERVIEW OF THEOS

Oasis8 was probably the most sophisticated operating system for Z80-based microcomputers, light-years ahead of the much better known CP/M. THEOS is an extension of Oasis8 for 8088/8086-based machines. It supports more users, more tasks, more RAM, and more disks, and it has more commands.

Unfortunately, while THEOS offers many advantages over MS-DOS, it also has to compete with operating systems moved to the PC from the other direction: minicomputers. Examples are Pick, Coherent, and, of course, UNIX. Roughly speaking, the functionality of THEOS is somewhere between MS-DOS and UNIX.

The THEOS core system consists only of what can be properly called the operating system: the kernel and utilities for configuring the system, managing the printer, manipulating files, editing text, and executing commands. Programming languages, word processors, and database systems are extra.

Each THEOS user is assigned a fixed memory partition up to 64K bytes in size. A program's data space is limited to what the partition can hold, but the instructions are located elsewhere. If two or more users are running the same program—a common occurrence for multiuser applications—they share the instructions.

A user might be running a multitasking program that consists of a main task and one or more subtasks. The tasks can communicate through shared variables and they can coordinate their access via semaphores. There are no pipes or messages.

Multitasking in THEOS is fairly restricted. A single user can't run a background task that is unrelated to a foreground task. For example, you can't compile some programs in the background while you write a report in the foreground. To do that, you need two terminals so you can pretend to be two users. Another restriction is that one user's task can't communicate with another user's task. And the command interpreter doesn't provide for multitasking at all (via a pipe operator like ¦); you set up multiple tasks only from BASIC or C.

The multitasking printer spooling system seems to be an exception, but it isn't really. It actually runs as a separate user.

The THEOS file system is a three-level hierarchy, but it's less general than that of MS-DOS or UNIX. A filename can consist of three components separated by periods: the filename, the file type, and the library member name. Each component can be up to eight characters long. Although there is nothing that corresponds to current directory, you can set various default libraries, such as link library and macro library. If you edit a file named, for example, Memo, the editor treats it as a member of the default macro library. If you want it to be a file in its own right, you have to use periods in the name.

Each file has an owner. The owner can set the file to be accessible by everyone or only by the owner. Since you need a password to log in, this system is reasonably secure. There are separate permissions for erase, read, write, and execute.

In addition to sequential and random I/O, the file system also supports a keyed access method. It has a mechanism for record locking, but a user can lock only one record at a time in a given file. This is inappropriate for transaction processing, where you want to keep all your records locked until the transaction completes. Also, THEOS doesn't provide other facilities needed for serious database management, such as automatic rollback of aborted transactions. Deadlock is possible, but I couldn't find any mention of it in the manual. I assume that the operating system doesn't detect it, let alone resolve it.

(continued)

THEOS is easier to use than UNIX or Coherent once an expert has set up the system.

THEOS can't access MS-DOS files, although it can access Oasis8 files. The text editor is a combination screen editor and line editor, similar in concept to vi and ex. You spend most of your time in the screen editor part, but you do operations such as global changes in the line editor part. You can use many of the special keys on the IBM PC XT keyboard (such as the arrow keys), so I had an easy time learning and using the editor.

The C compiler is called Definitive C and is described in its manual as compatible with UNIX Systems III and V. However, I didn't find this to be true. The compiler failed to handle the types enum and void, and the library functions worked differently (for example, there is no fseek function). Also, you can't include a file named, for example, stdio.h, because a member of a macro library can't have a period in it. You have to use stdio instead.

Aside from these irritating incompatibilities, the compiler worked. One nice touch is that it pauses after displaying an error message so you can read the message when you get back from the refrigerator. Then you can kill the compilation if you think you've seen enough. The linker is also special: It flashes the object module names on the screen as it finds them in the library. Niceties like these might not be important, but they conveyed a feeling of quality and attention to detail that enhanced my confidence in the system.

A regular BASIC and a graphics VBASIC are available at extra cost. VBASIC uses a virtual-device interface to support a variety of output devices. I did not evaluate either BASIC in detail.

INSTALLING AND USING THEOS

I made a mistake in installing THEOS: I read the one-page installation instructions, followed them, and then got stuck. Although the system was installed, I didn't have the foggiest idea how to use it.

What I should have done is spend several hours studying the 58-page introductory manual first. Nuggets of essential information are spread throughout this manual, and you really need to know about them before you can get THEOS up and running.

For example, to get the printer working you have to know that its physical name is centlp. The manual doesn't tell you this, but it does tell you where to find a file that lists the physical names. Then you need to know how the ATTACH command works. There's a section titled "Attaching Printer Devices," but it only discusses general principles. You are directed instead to the chapter titled "Using the Printer Spooler." There isn't any chapter with that title, but there is one called "Using a Printer." After more chasing around, you finally realize that it would have been much easier to read the manual sequentially from the beginning, memorizing as much as possible. Then at least you would know where to find information.

You can boot THEOS from the hard disk, but it was difficult to figure out from the manual how to do it. You use a disk utility with an option titled "write boot sectors."

It was also very difficult to figure out how to install a second terminal, but once I did, the actual installation went smoothly. Again, if I were an expert on physical and logical devices, attaches, and class codes, it would have been a breeze. There was no wiring diagram for the serial cable, but the same cable I used for Coherent worked.

Ignoring the documentation problems, I would say that THEOS is easier to use than UNIX or Coherent once an expert has set up the system. This is partially because THEOS has fewer commands, with fewer options on each command, and partially because of more use of menus and prompts. THEOS doesn't try to be as clever as UNIX does.

THEOS DOCUMENTATION

The sentences and paragraphs in the documentation are written well enough, but there are major problems with the manual's organization. First, it has too little redundancy. For example, instead of listing the terminal class codes in the subsection titled "Console Class Codes" in the section titled "Attaching Console Terminals," the manual refers you to the chapter titled "Console Terminal Class Codes." No such chapter exists. I experienced far too many wild-goose chases to track down cross-references.

The manual is also too verbose. Long treatises on the philosophy of multitasking are mixed in with essential installation information. Since the cross-references are messed up and there's no index, you have to read the manual sequentially. This takes twice as long as it should because you have to wade through lots of stuff that you could have read later at your leisure.

Finally, the target audience for the manual is ill-defined. One paragraph discusses head load delay for a hard disk, while another explains what is meant by the word "abbreviation," complete with an example. Not only that, but the explanation of head load delay precedes the explanation of how to install a printer.

In summary, the manual does not present information in a logical order, has too many cross-references (many of them erroneous), mixes nonessential and essential information, and has no index. But after reading the whole manual, I was able to understand how THEOS works and I had very little trouble installing and using it.

THEOS PERFORMANCE

I didn't run the B-tree program (benchmark 5) on THEOS, because it couldn't read the source code from my MS-DOS floppy, and I didn't have time to work out a serial port connection to another computer. Instead, I tested just the random I/O and sequential I/O programs.

Based on these tests, it seems that random I/O is faster than MS-DOS but slightly slower than Coherent and PC/IX. In sequential I/O, THEOS was the slowest of the four operating systems by a wide margin. It's difficult to extrapolate from just two programs, but it's safe to say that the THEOS file system is in the same ballpark as MS-DOS, which is slow. Since THEOS has indexed access methods built in, they are likely to be fast, although I ran no test to prove this.

CONCLUSIONS ON THEOS

THEOS is well designed and worked the way it should. It's easier to use and understand than UNIX, but it also has many fewer features. Of course, if you are implementing a specific application, you don't care about the number of features; you care about whether the features you need are present. You might well find that THEOS is a perfect match for your requirements.

Compared to other operating systems for the XT, the price for THEOS seems too high. It costs twice as much as Pick or Coherent, and, with the C compiler, it costs even more than PC/IX. ■

Filling your needs m

Today's PC buyer is no pushover. Not only are low prices and prompt delivery demanded, but professional support, sound advice and innovative servicing of accounts is expected. It's no coincidence then that PC buyers all across the nation call PROGRESSIVE MICRO DISTRIBUTORS first. Here's why.

LOW COST, HIGH TECH SOLUTIONS.

PROGRESSIVE MICRO DISTRIBUTORS has the best prices on popular IBM PC hardware and software in the country (call and compare). Most orders are shipped the same or next day from our 25,000 sq. ft. warehouse. Although we ship literally hundreds of orders daily, all PC's are burned in overnight to insure that your system arrives in excellent condition.

We believe it takes a professional to provide professional service. That's why all sales and service staff members, from our factory authorized technical experts to our highly experienced account representatives, are rigorously trained on all current and new hardware and software. And if the need ever arises, direct access to our on-line customer service and technical support centers is available and convenient to use.

HOW TO KEEP UP WITH PROGRESSIVE.

As a PROGRESSIVE MICRO DISTRIBUTORS' customer, you'll also be privy to new product information and special money saving incentives through the Microgram,™ our monthly customer newsservice. You'll also receive monthly price lists showing current pricing on our complete line of hardware, software and services. Lists are shipped automatically with all system orders and are available on disk when requested. Plus, **FREE** UPS shipping is guaranteed on all prepaid orders with confirmation of current pricing.

Call PROGRESSIVE MICRO DISTRIBUTORS today, **TOLL FREE** at **1-800-446-7995**, and we'll mail a current price list to you immediately. And remember, only PROGRESSIVE MICRO DISTRIBUTORS gives you all of these benefits: low prices, sound advice, prompt delivery, professional support and direct customer service.

POLICIES

1. We accept VISA, MC, AMEX; No surcharge for VISA or MC.
2. COD requires cash or certified check; Company check when approved.
3. Allow 1 week for personal and company checks to clear.
4. Wiring information available upon request.
5. Corporate, Government and Institutional volume purchase agreements available.
6. Call for exact shipping charges; air shipments take priority.
7. If we must split shipment, you incur no additional shipping charges.
8. All orders shipped insured — No additional charge.
9. All products fully warranteed; Some up to 5 years.
10. All items subject to availability; Prices subject to change without notice.

FOR ORDERS
1-800-446-7995

ORDERING INFORMATION
1-404-446-7995

CUSTOMER SERVICE
1-404-446-7996

TECH ASSISTANCE
1-404-446-7997

HRS: 8AM-8PM EST. SAT. 12N-5PM EST.

akes us Progressive.

PC SYSTEMS

AT&T PC 6300
640K, 2 drives, monochrome monitor/adapter, DOS/BASIC $1995
640K, 2 drives, color monitor/adapter, DOS/BASIC $2495
640K, 1 drive, 10MB hard disk, DOS/BASIC $2895

AT&T PC 7300 & 8300
Call for your best price on your specific configuration

IBM PC
256K, 2 drives, monochrome/printer adapter & monitor $2195
256K, 2 drives, color/graphics adapter & monitor $2495
256K, 2 drives, 10MB internal hard disk $2895

IBM PC XT
256K, 2 drives, monochrome/printer adapter & monitor $2295
256K, 2 drives, color/graphics adapter & monitor $2595
256K, 2 drives, 10MB internal hard disk $2895
256K, 2 drives, 20MB internal hard disk $3395

IBM PC COMPATIBLE
256K, 2 drives, 8 slots, 135w power supply $995
256K, 1 drive, 10MB, 8 slots, 135w power supply $1595

SANYO MBC 555-2
MS/DOS w/128K, 2 drives, color monitor $995

These systems are brand new, shipped fully tested and burned in, fully warranted for ninety days AND ARE ALWAYS IN STOCK!

PRINTERS

EPSON
FX-85 (160 cps narrow carriage NLQ dot matrix) $425
FX-185 (160 cps wide carriage NLQ dot matrix) ... $625
LX-80 (100 cps narrow carriage NLQ dot matrix) $249
JX-80 (160 cps narrow carriage color dot matrix) $525
LQ-1500 (LQ dot matrix w/parallel interface) $995

OKIDATA
ML 192 (160 cps narrow carriage dot matrix) $389
ML 193 (160 cps wide carriage dot matrix) $550
PACEMARK 2410 (350 cps wide carriage dot matrix) $1995

CANON
LBP-8A1 (8 ppm, better than HP) $2595
PW-1080A (160 cps dot matrix) $329
PJ-1080A (32 cps color printer) $549

TOSHIBA
P-351 (192 cps wide carriage LQ dot matrix) $1295
P-1340 (120 cps narrow carriage LQ dot matrix) .. $595

TEXAS INSTRUMENTS
TI-855 (150 cps narrow carriage LQ dot matrix) .. $649

BROTHER
HR-15 + (17 cps narrow carriage letter quality) .. $365
HR-25 (23 cps wide carriage letter quality) $550
HR-35 (32 cps wide carriage letter quality) $795

C. ITOH
Starwriter (40 cps letter quality) $895
Printmaster (55 cps letter quality) $1195

NEC
Pinwriter P-2 & P-3 .... CALL
Spinwriter 2050, 3550, 8850 .... CALL

Ask about our full line of tractors and cut sheet feeders!

PLOTTERS

We carry all HOUSTON INSTRUMENTS and HEWLETT-PACKARD plotters

FLOPPY DISK DRIVES

TANDON TM 100-2 (DS/DD 360K) $149
CONTROL DATA (DS/DD 360 K) $149
HALF-HEIGHT 360K (PC/XT/AT compatible) $119
HALF-HEIGHT 1.2MB (PC/XT/AT compatible) $159

MISC. HARDWARE

POLAROID Palette (Makes color slides from your PC's screen) $1595
ORCHID PC Turbo (adds 8MHz processor, multitasking to PC) .... from $649
KEYTRONIC KB 5151 & 5153 keyboards .... CALL
MOUSE SYSTEMS Optical PC MOUSE w/PAINT $155
MICROSOFT Serial or Bus mouse w/Mouse Menu $155
GIS Power Back-ups (SPS' and UPS') .... CALL
QUADRAM Microfazer Printer Buffer .... from $135
CURTIS System accessories .... CALL

HARD DISK DRIVES

Internal			
PC/XT 10MB	from $429	AT 105MB	$2995
AT 20MB	$695	AT 140MB	$3595
PC/XT 20MB	$645	AT 190MB	$4495
AT 30MB	$895	AT 260MB	$5995
PC/XT 30MB	$995	**External**	
AT 65MB	$1995	PC/XT 10MB	$695
		PC/XT 20MB	$945

We proudly offer what we think are the best price/performance hard disk and streaming tape sub-systems available. We will gladly test and format any hard disk we sell at a nominal charge. Our line includes: BERNOULLI TECHNOLOGY, CIPHER DATA, EMERALD SYSTEMS, DMA, IRWIN, PEACHTREE TECHNOLOGIES, QUADRAM, SYSGEN, TALLGRASS, and WANGTEK

MULTIFUNCTION BOARDS

AST RESEARCH
Six Pak (384K, serial, parallel, C/C, opt. game) $169
Advantage (AT RAM expansion with options) from $399

QUADRAM
Quadboard (384K, serial, parallel, game, C/C, bracket) $239
QuadMeg (AT memory expansion up to 4MB) from $329
QuadPort (AT serial and parallel, opt. 4 more serial) $109

MICROLOG
Baby Blue II w/64K (64-384K, Z80, 2 ser, par, C/C) $429
Little RedRam (256K memory expansion) $139

CHIPS

64K (nine chips)	$10	128K (AT RAM)	CALL
8087 (coprocessor)	$119	256K (jr & AT boards)	CALL
80287 (coprocessor)	$199		

CALL FOR QUANTITY DISCOUNTS

GRAPHICS CARDS

We carry a variety of display adapters. So, call us if you don't see the one you want or need help choosing.

HERCULES
Graphics Card (Lotus compatible monochrome graphics) $329
Color Graphics Card (Short-slot color card w/par) $179

PARADISE
Modular Graphics Card & Multi-Display Card from $299

TECMAR
Graphics Master (640x400 16 colors and monographics) $469

PERSYST
BoB Board (clear text on a color monitor) $489

TSENG LABS
UltraPAK (132 col. mono graphics, ser, par, C/C) ... $450

MONITORS

AMDEK
300G & 300A (Green or Amber, composite video monitors) .... $129/$159
310A (Amber TTL input, non-glare tube) $165
COLOR 600 (13" 640x240 RGB w/text button, audio) $435
COLOR 700 (13" 720x480 RGB w/.31mm dot) $475
COLOR 710 (13" 720x480 RGB, non-glare/flicker) $575

PRINCETON GRAPHICS
PGS MAX-12 (Amber TTL input, non-glare tube) .. $189
PGS HX-12 (690x480 RGB w/.31mm dot) $449
PGS SR-12 (690x480 non-interlaced RGB w/.31 dot) $595

TAXAN
RGBvision 420, 420L, 425 & 440 .... CALL

We carry WYSE and KIMTRON terminals, call for prices.

PROTOCOL CONVERSION

We carry all AST and DCA products, call for prices.

MODEMS

HAYES
Hayes 1200 (external 300/1200, auto answer/dial) .. $419
Hayes 1200B (internal 300/1200, w/Smartcom II) .. $379
Hayes 2400 (New external 2400 baud modem) $675

DCA
New Fastlink 10KB modem .... CALL

ANCHOR AUTOMATION
Signalman Mark XII (1200 baud at 300 baud price) $299

We also carry modems by AST RESEARCH, BIZCOMP, PENRIL, QUADRAM, RIXON, VENTEL — please call for current pricing.

NETWORKING

We carry all the best names including 3COM, AST RESEARCH, FOX, IBM, NOVELL, and ORCHID TECHNOLOGIES. Call our Networking Applications experts to design a cost-effective device-sharing networked layout for your office!

DISKETTES

VERBATIM		
Single sided/Single density	10/$24	100/$199
Double sided/Double density	10/$29	100/$249
DYSAN		
Double sided/Double density	10/$29	100/$249
High density AT diskettes	10/$59	100/$549
MAXELL		
High density AT diskettes	10/$59	100/$549
PMD		
Double sided/Double density	10/$15	100/$129

SOFTWARE

LOTUS Symphony $449
ASHTON-TATE Framework $399
ENABLE by Software Group .... CALL
SOFTWARE PUBLISHING Pfs Write, File, Graph, Report .... from $79
SSI WordPerfect $239
MULTIMATE w/spelling checker and tutorial $279
MICROSOFT Word 4.0 .... $299
LIFETREE Volkswriter Deluxe w/Textmerge $189
LOTUS 1-2-3 .... $309
ASHTON-TATE dBase III $399
MICRORIM 5000 .... $359
MICROSTUF Crosstalk XVI $109
PERSOFT SmarTerm 100, 125, 400 .... CALL
RELAY .... CALL
DOW JONES All products .... CALL
SORCIM/IUS Complete accounting modules .... from $369
BPI ACCOUNTING Complete accounting modules ... CALL
OPEN SYSTEMS Accounting .... CALL
SYSTEMS PLUS Books! accounting system .... CALL
MECA/TOBIAS Managing Your Money $129
MONOGRAM Dollars and Sense $119
DECISION RESOURCES Signmaster/Chartmaster .... $179/$349
AUTODESK AutoCAD with or without extensions 1, 2, & 3 .... CALL
ZSOFT PC Paint Brush $95
HARVARD Harvard Project Manager .... CALL
MICROSOFT Project .... CALL
WORDTECH dBase compilers .... CALL
BORLAND Turbo Pascal & Toolbox .... ea. $39
MICROSOFT All products .... CALL
DIGITAL RESEARCH All products .... CALL
BORLAND All products .... from $39
SOFTSTYLE Printworks printer utilities $35
NORTON Utilities $59
QUAID Copywrite, Disk Explorer & Zero Disk .... CALL
CENTRAL POINT Copy II PC copies protected software $35
ATI Training $59

PROGRESSIVE MICRO DISTRIBUTORS
3039 Amwiler Road, Atlanta, Georgia 30360

Inquiry 158

PC Paintbrush.™ Because life is too short for monochrome pie charts.

Fun is the best thing to have.

With PC Paintbrush, you can add color, flair, dimension and creativity to a chart, a presentation, or an otherwise dull day. From charts and graphs to serious computer art, our newest generation 3.0 PC Paintbrush will cheer you on with features no other graphics package can match.

Best of all, it's easy to use. You don't have to learn up to sixty commands, like you do with some products. If you can understand icons as simple as scissors, paintbrush, spray can and paint roller, you're ready to start using PC Paintbrush.

The pen is mightier than the keyboard.

None of history's great artists drew with a keyboard, and you shouldn't have to either. So PC Paintbrush is now available with a cordless PenMouse, to give you complete freedom of expression. Of course, it also supports regular mice, joysticks, graphics tablets, and is compatible with most graphics cards.

PC Paintbrush also has a beautiful way with words. The text icon lets you write in any of eleven fonts, in nine sizes, with italics, outline, shadow and boldface variations.

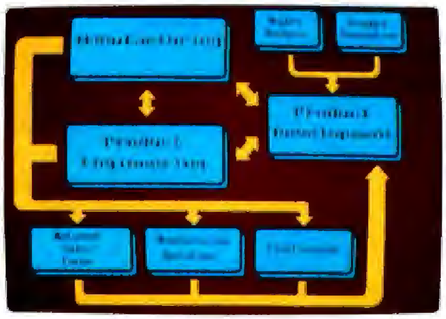
What's more, with the new 3.0 PC Paintbrush, you can draw rounded boxes, rubber band curves and circles, and edit pictures many times larger than the screen.

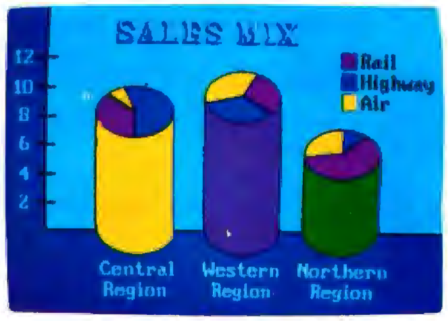
Are we making fun of 1-2-3®? Why not?

For Lotus™ users, PC Paintbrush's new PIC interpreter loads 1-2-3™ and Symphony™ charts and graphs at your equipment's best resolution, from an IBM EGA™ (640 X 350 X 16 colors) to a Number Nine Revolution™ (512 X 512 X 256 colors).

With our FRIEZE™ frame grabber you can pull graphics created by any program right off the screen into PC Paintbrush. So you can take your Paintbrush and pallette anywhere, improving the looks of things as you go. And having a lot of fun on the way. In addition, our optional presentation package, SLIDESHOW, allows you to program your graphics into a first class presentation with fades, zooms, quick cuts and animation.

Go on, live a little.





PC Paintbrush supports 19 video graphics cards and 30 printers and plotters.

For more information on PC Paintbrush, call or write us at the address below, or ask your computer dealer for a demonstration.

Z·SOFT
PC Paintbrush

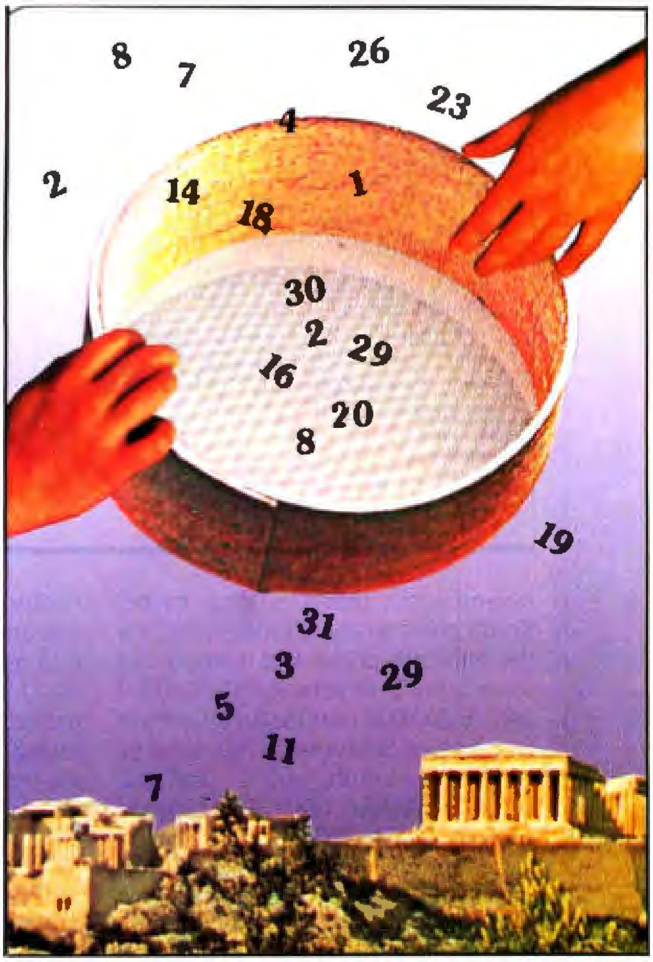
Corporate Headquarters:
ZSoft Corporation, 1950 Spectrum Circle, Suite A 495, Marietta, GA 30067, 404/980-1950
West Coast Sales Office:
160 Sir Francis Drake Blvd., San Anselmo, CA 94960, 415/456-0955

Inquiry 219 for End-Users. Inquiry 220 for DEALERS ONLY.

ONE MILLION PRIMES THROUGH THE SIEVE

Generate a million primes on your IBM PC *without running out of memory*

BY T. A. PENG



A POPULAR WAY to benchmark microcomputers is with the Sieve of Eratosthenes. It is a simple and effective method for generating prime numbers. However, if you try to use the Sieve to obtain more than a few thousand primes on your IBM PC, you will soon encounter the dreaded phrase, "Out of memory." You would think, then, that as far as microcomputers are concerned, the Sieve of Eratosthenes would be an impractical way to generate a large number of primes. This is not so. Let me show you how to use the Sieve to generate a million primes on your microcomputer.

Listing 1 (written in Microsoft BASIC) illustrates how, with very little memory, you can put 500,000 numbers through the Sieve to obtain all the primes less than 1,000,000. The idea is quite simple. Use an array of flags to represent the first 1000 odd numbers. After the nonprimes among them have been sieved out, reinitialize the array to represent the next 1000 odd numbers. Lines 120 through 140 initialize the array and lines 340 through 360 reinitialize it before you use it for the next 1000 numbers. The largest prime whose square is less than 1,000,000 is 997 and it is the 168th prime—starting with the prime 2. To generate all the primes less than 1,000,000, you don't have to use primes larger than 997. This is the reason for line 220 and for the size of two of the arrays in line 110. The loop in lines 240 through 270 flags all numbers less than 1000 that do not yield primes. (We have K = I + nP, so that K + K + 1 = (I + I + 1) + 2nP = P(2n + 1), which is not a prime.) After each loop is executed, the value of K will be greater than 1000 (and K would flag the next number if the size of the array were larger) and this is remembered as K(C). The variable C keeps count of the primes generated with C − 1 as the actual number of primes generated at the end of each loop. Line 390 assures that the value of K lies between 1 and 1000. You need line 460 to give the correct value for the prime Q in line 490. All the variables except C, Q, and R are integer-valued. There is a reason for this. If the program executes correctly, the output of line 540 should read, "999,983 is the 78,498th prime and the largest less than 1,000,000."

It is clear how to modify listing 1 to generate all the primes less than 2,000,000 or even 10,000,000, but to get a predetermined number of primes, we need to know a little about their distribution. Specifically, what we need to know is the size of the arrays K and P and the largest prime to be used in the Sieve. And in order to know this, we must have a rough idea of how large the

(continued)

Dr. T. A. Peng *is a faculty member of the* Department of Mathematics *of the* National University of Singapore *and may be contacted at* Lower Kent Ridge Rd., Singapore, 0511, Republic of Singapore.

ILLUSTRATED BY JOAN HALL

Listing 1: *This program illustrates how you can get* 500,000 *numbers through the Sieve to obtain all the primes less than* 1,000,000.

```
100 DEFINT F-P
110 DIM F(1000),K(168),P(168)
120 FOR I=1 TO 1000
130 F(I)=1
140 NEXT I
150 P=2
160 PRINT P;
170 C=2
180 FOR I=1 TO 1000
190 IF F(I)=0 THEN 310
200 P=I+I+1
210 PRINT P;
220 IF P>997 THEN 300
230 K=I+P
240 WHILE K<=1000
250 F(K)=0
260 K=K+P
270 WEND
280 P(C)=P
290 K(C)=K
300 C=C+1
310 NEXT I
320 R=1
330 FOR N=1 TO 499
340 FOR I=1 TO 1000
350 F(I)=1
360 NEXT I
370 FOR J=2 TO 168
380 P=P(J)
390 K=K(J)-1000
400 WHILE K<=1000
410 F(K)=0
420 K=K+P
430 WEND
440 K(J)=K
450 NEXT J
460 R=R+2000
470 FOR I=1 TO 1000
480 IF F(I)=0 THEN 520
490 Q=R+I+I
500 PRINT Q;
510 C=C+1
520 NEXT I
530 NEXT N
540 PRINT Q;"is the";C-1;"th prime
and the largest less than 1,000,000"
550 END
```

Listing 2: A *modified listing, this program generates a million primes without overtaxing the microcomputer's memory.*

```
100 DEFINT F-P:DEFDBL X-Z
110 DIM F(10000),K(550),P(550)
120 P=2:PRINT P;:C=2
130 FOR I=1 TO 10000:F(I)=1:NEXT I
140 FOR I=1 TO 10000
150 IF F(I)=0 THEN F(I)=1:GOTO 220
160 P=I+I+1:PRINT P;
170 IF P>4000 THEN 210
180 K=I+P
190 WHILE K<=10000:F(K)=0:K=K+P:WEND
200 P(C)=P:K(C)=K
210 C=C+1
220 NEXT I
230 X=1:Z=C
240 FOR N=1 TO 799
250 FOR J=2 TO 550
260 P=P(J):K=K(J)-10000
270 WHILE K<=10000:F(K)=0:K=K+P:WEND
280 K(J)=K
290 NEXT J
300 X=X+20000
310 FOR I=1 TO 10000
320 IF F(I)=0 THEN F(I)=1:GOTO 360
330 Y=X+I+I:PRINT Y;
340 IF Z=1000000# THEN 380
350 Z=Z+1
360 NEXT I
370 NEXT N
380 PRINT Y;"is the";Z;"th prime"
390 END
```

one-millionth prime is likely to be. Knuth gives an approximate value for the millionth prime and it works out to be a number between 15,000,000 and 16,000,000 (see *The Art of Computer Programming: Seminumerical Algorithms* by Donald E. Knuth, vol. 2, 2nd ed. Addison-Wesley, 1981). The square root of 16,000,000 is 4000, so the largest prime should not exceed this. The 550th prime is 3989, the largest prime less than 4000. (You can verify this result with listing 1.) With this knowledge, I modified listing 1 to obtain listing 2, which generates a million primes as I promised.

A few remarks about listing 2 may be in order. We now sieve 10,000 numbers at a time and the execution time for each loop is much longer. Also, listing 2 will need more memory to run, but this can be handled comfortably by most 64K-byte microcomputers. I used multiple statements in some lines to make the program look neater. One improvement is in lines 150 and 320, where the array variable F is reinitialized immediately after it has done its work. This will save some execution time. The efficiency of listing 2 could be further improved if we replace K = I + P with K = (P * P − 1)/2, but this would involve other changes and the program would no longer be as simple. While you can run listing 1 in interpreted mode, you'll be better off if you compile listing 2 (using the Microsoft BASIC compiler) to increase its execution speed. To test whether or not listing 2 runs correctly, you can delete the PRINT statements in lines 160 and 330 (and change the number 1000000 in line 340 to 78498) to see if it produces the same final result as listing 1. It may not be very important to know what the one-millionth prime number is, but it is certainly satisfying to find out for yourself that it is 15,485,863. ■

a message to our subscribers

From time to time we make the BYTE subscriber list available to other companies who wish to send our subscribers material about their products. We take great care to screen these companies, choosing only those who are reputable, and whose products, services, or information we feel would be of interest to you. Direct mail is an efficient medium for presenting the latest personal computer goods and services to our subscribers.

Many BYTE subscribers appreciate this controlled use of our mailing list, and look forward to finding information of interest to them in the mail. Used are our subscribers' names and addresses only (no other information we may have is ever given).

While we believe the distribution of this information is of benefit to our subscribers, we firmly respect the wishes of any subscriber who does not want to receive such promotional literature. Should you wish to restrict the use of your name, simply send your request to the following address.

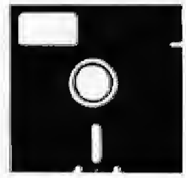
BYTE Publications Inc.
Attn: Circulation Department,
70 Main St., Peterborough, NH 03458

Express Computer Presents:

a DISKETTE SALE

You CAN buy quality diskettes at low PRICES including a LIFETIME WARRANTY.

SS DD .81
DS DD .90
minimum quantity of 50 diskettes



Includes:
* Box
* Labels
* Envelope
- 100% Error Free
- Lifetime Guarantee

TO ORDER CALL TOLL FREE:
800-437-5200
ask for operator 213
MD. residents call:
1-800-638-8890
(24 hours a day)
Dealer Inquiries Invited

LIFETIME WARRANTY!
Send defective diskette back and we'll replace it (if there is a defect in original material and workmanship).
C.O.D. and prepaid orders accepted

EXPRESS COMPUTER

1101 Valley Glen Rd.
Elkins Park, PA 19117

9-TRACK MAG. TAPE SUBSYSTEM FOR THE IBM PC/XT/AT

$4757

For information interchange, backup and archival storage, IBEX offers a 9-track, IBM format-compatible ½" magnetic tape subsystem for the IBM PC, featuring:

- **42 M-Bytes on a single reel.**
- **IBM format 1600 cpi.**
- **Software for PC-DOS, MS-DOS.**

Write, phone or TWX for information

I B E X

IBEX COMPUTER CORP.
20741 Marilla St.
Chatsworth, CA 91311
(818) 709-8100
TWX: 910-493-2071

"KING OF THE ELECTRONICS JUNGLE"

LEO ELECTRONICS

Now that you have the best computer money can buy . . .
UPGRADE WITH THE BEST!

Dynamic Rams	Eproms	Math Co-processors
41256-150ns (256K)	2708	8087 80287
4128-150ns (128K-AT)	2716	Disk Drives
4164-150ns (64K)	2732	Tandon, Qume
Static Rams	2764	Mitsubishi, Teac
6116P-3 (2K x 8)	27128	Power Supplies
6264P-3 (8K x 8)	27256	135V 150V

OUR QUALITY IS THE HIGHEST!
OUR PRICES ARE THE LOWEST!

Please **CALL** or **WRITE** for pricing/brochure.

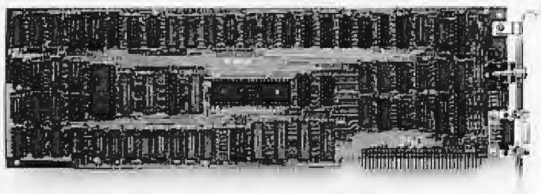
LEO ELECTRONICS, INC.
P.O. Box 11307
Torrance, CA 90510-1307
Calif. residents 213/212-6133
outside Calif.
800/421-9565
Telex: 291 985 LEO UR Fax: 213/212-6106

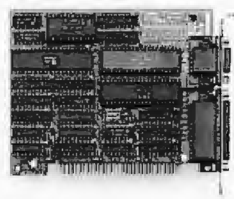
TRY US ON ANY "HARD TO FIND" COMPUTER RELATED ITEMS.

All major manufacturers. All parts prime.. All 100% guaranteed.

Compare the Hercules™ Color Card to IBM®'s.

Five reasons why the Hercules Color Card is better.





	IBM Color Adapter $244	Hercules Color Card $245
1. Compatibility	Runs hundreds of graphics programs.	Runs the same hundreds of graphics programs. "The Hercules Color Card is so nearly identical to the IBM Color/ Graphics Card that it's almost uncanny." PC Mag.
2. Printer port.	None.	Standard. Our parallel port allows you to hook up to any IBM compatible printer.
3. Size.	13.25 inches. Limited to long slots.	5.25 inches. Fits in a long or short slot in a PC, XT, AT or *Portable.*
4. Flexibility.	Can't always work with a Hercules Graphics Card.	Always works with a Hercules Graphics Card by means of a software switch.
5. Warranty.	90 days.	Two years.

Any one of these five features is enough reason to buy a Hercules Color Card. But perhaps the most convincing reason of all is just how easy the Hercules Color Card is to use: "Right out of the box, the Hercules Color Card goes into an empty expansion slot, ready for you to plug in... and go to work—no jumpers, no software. For most applications, it's just that easy." PC Magazine.

Call **1-800-532-0600 Ext. 432** for the name of the Hercules dealer nearest you and we'll rush you our free info kit.

Hercules. Inquiry 84

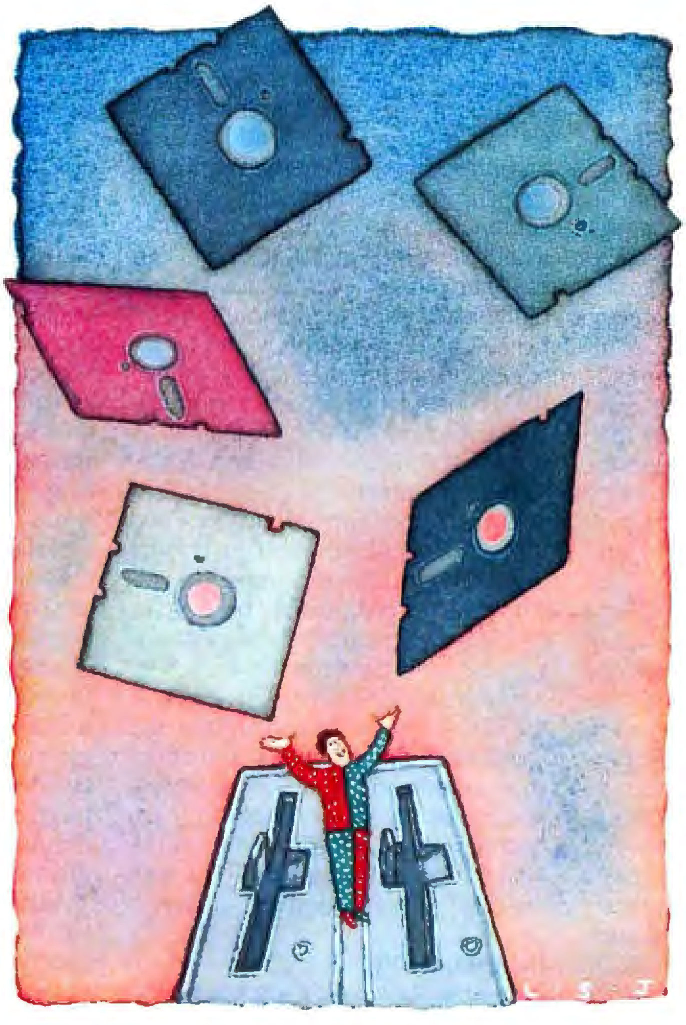
We're strong on graphics.

Address: Hercules, 2550 Ninth St, Berkeley, CA 94710 Ph: 415 540-6000 Telex: 754063 Trademarks/Owners: Hercules/Hercules Computer Technology; IBM/IBM.

TopView

IBM's long-awaited multitasking program makes its debut

BY TJ BYERS



TOPVIEW is a multitasking program that, for $149, enables your IBM Personal Computer to install more than one program in the system. This is different from the window programs that presently claim to accomplish the same thing. When working with windows, you must quit a program before you can begin another. With TopView, however, you don't have to quit either one of them. Both can be resident on the screen—and, more important, in the microprocessor—at the same time.

MULTITASKING

TopView's multitasking capabilities allow several programs to run simultaneously (see photo 1). This isn't the same thing as switching between programs without quitting them; it means that you can actually have one program running in the background while using another. Let's say, for example, that you need to calculate a large spreadsheet, and the job will take several minutes. Instead of staring idly at the screen while the computer crunches away, you can banish the spreadsheet to TopView's background mode and proceed to work on another program—the computer will handle both tasks at the same time. While one program is making calculations in the background, the other can be receiving data from the keyboard. You lose no time waiting for one program to finish before you start the other.

Multitasking is not a new concept. Mainframe computers have used multitasking for many years to enhance their performance. What is new, however, is putting multitasking capabilities into a personal computer.

TopView brings multitasking to the IBM PC using a multiplexing technique known as time slicing. Basically, TopView divides the microprocessor's time into slots during which it switches rapidly from one program to another. The time slices are very short, on the order of milliseconds, and the switching action is not apparent to either the application program or the user, so the programs appear to be running concurrently on the machine. In actuality, they are processed consecutively in very quick order. The procedure gives a single computer the ability to run more than one program at a time.

Multitasking is not without its faults, however. While one program is being processed, the others are held in suspension. Consequently, the programs tend to run more slowly. The more programs you have running at the same time, the slower each apparently becomes. A quick benchmark test using TopView to conduct a simple word search of Writing Assistant on an IBM PC AT showed that it took a full 14 seconds to search a typical 3000-word file as

(continued)

TJ Byers is a freelance author of numerous articles and books, including Guide to Local Area Networks (Prentice-Hall). *He may be contacted at* POB 372, Hancock, NH 03449.

ILLUSTRATED BY LONNIE SUE JOHNSON

TopView is very memory intensive. The minimum 256K RAM *leaves you about* 80K *bytes for use by applications.*

opposed to 3+ seconds without TopView.

TopView is also very memory intensive, as are all multitasking routines. While programs are in suspension between microprocessor access cycles, their parameters must be held in RAM (random-access read/write memory). This takes a lot of memory. IBM specifies a minimum of 256K bytes for TopView and recommends more for most applications. Installing TopView under PC-DOS 2.1 using minimum memory (256K bytes) leaves approximately 80K bytes of working memory for use by application programs; PC-DOS 3.0 reduces that number to 68K bytes. This is not much memory when you start talking multitasking. To give you an idea of how memory intensive TopView can be, let me say that my IBM AT contains the full memory complement of 640K bytes. After installing Writing Assistant, a very hungry program requiring 232K bytes, and Filing Assistant into TopView, I have a mere 96K bytes of memory left, barely enough to take on another multitasking chore. Moreover, very large programs, such as Writing Assistant and dBASE III, can't be run at the same time; there just isn't enough room for both.

IBM solves the memory problem by modifying some application programs. Writing Assistant, for example, can be reduced to 104K bytes, less than half its previous value, by using a routine that is outlined in the *TopView Application Guide*. Unfortunately, it also reduces the capabilities of the word-processing program. Automatic save and retrieve are no longer available, and you have to enter these functions by hand. The program also runs considerably slower, which is a real nuisance if you happen to be a good typist. But then, that's a tradeoff you have to make when using a multitasking program, whether it be TopView or any other.

USING TOPVIEW

TopView is not extremely difficult to use, but it does take some training and a bit of expertise, which you can pick up from the IBM *TopView User's Guide*. True to IBM form, however, the guide is not exactly the epitome of clarity. Although the guide is well organized, it sometimes glosses over very important points with a simple sentence or two. Unless you know what you're looking for, the point can easily be missed. For instance, I found difficulty in releasing the cursor keys from TopView control when I first started using the program. A call to IBM resolved the problem. Oh, the instructions were there in the manual all right, just somewhat obscured by surrounding material.

To help you learn about TopView, IBM has included an excellent tutorial disk. In fact, IBM recommends you study the tutorial carefully before proceeding with the *User's Guide*—wise advice, because the tutorial gives you invaluable hands-on experience with the various attributes and functions of TopView, something that would otherwise take many hours of intensive reading before the concepts and procedures became clear in your mind. Plan on spending at least an hour with the tutorial; it's loaded with material.

INSTALLATION

Installing TopView is probably the simplest chore the program lets you do. You just type G and hit Enter; a batch file boots the TopView command program into the system. This is consistent with IBM's policy of using the same command to activate all its newly released programs.

The TopView batch command is different from other IBM programs, however, in that it can include optional extenders. If you wish to bypass the welcome window, for example, and go straight into the program, you use the entry code g /a to initiate TopView. Altogether, there are 18 optional extenders that can be used to modify TopView operations. Most of these extenders either specify or modify the monitor and/or its screen format. A couple suppress certain routines, like the welcome window, and one (k) assists with the installation of undisciplined non-IBM programs into the TopView environment.

Once inside, you control TopView using a combination of windows and menus. This programming concept combines the advantages of menu-driven programs, which are normally keyboard controlled, with the convenience of icons. Although TopView has no icons, its menu entries act like icons. Unlike real icons, though, which are rigidly defined entities, the menu entries can be changed. Items can be added or deleted at will, and there is no limit to the type of application programs you can include, provided they are TopView-compatible. More on that later.

You install application programs into the TopView environment by manipulating a pointer inside the window until it overlaps the desired program title on the main menu. The pointer can be moved using the cursor keys or with a mouse. Once you have selected a program, a single keystroke (or mouse click) loads the program into TopView. You can load as many programs into TopView as your memory will support, and all active programs may run simultaneously.

Application programs are installed into TopView using a strict set of rules, and defining these parameters is no easy chore. You must specify no less than eight factors and up to a dozen more for certain applications. To facilitate the process, IBM provides you with a program menu containing 38 predefined programs (see photo 2). These programs, which include DisplayWriter and VisiCalc, have their values already established, and you can place them on the TopView program menu by simply placing the pointer over the desired program and completing a simple TopView menu routine. You are prompted every step of the way, and the table makes program additions very convenient. Once installed, the program remains in the menu until you remove it, so you don't have to go through the routine every time you start up TopView.

When including other application programs for which IBM has not provided predefined parameters, however, you have to enter the values yourself. IBM assists you in this by including with TopView a rather thick *TopView Application Guide*, which lists the parameters for 88 prominent software manufacturers and 350 software packages. The listing includes such popular programs as Framework, CrossTalk, and Lotus 1-2-3, among others. To install one of these programs you simply look at the printed information table and duplicate it.

For all programs, you need to specify both a pathway to the program and the program command that activates its files. When doing this you must be aware that TopView does not support batch files and therefore does not recognize batch-file commands. If you try to use the g command to start an IBM program, for instance, you will be in for a surprise. Much to your dismay, the machine simply locks up and goes bye-bye. Only a machine restart can correct the problem, which is too bad if you have files in progress that haven't been saved before the incident occurs.

I suspect IBM did this on purpose to avoid DOS (disk operating system) conflicts. You see, TopView does not support all PC-DOS commands and supports none of the enhanced PC-DOS 3.0 commands. If you were to initiate a batch file that contained a command that TopView couldn't execute, it's hard to say what might happen. To avoid the issue altogether, TopView simply doesn't support batch files.

TopView Attributes

The DOS commands that TopView does support, however, are available in a DOS Services file (see photo 3). This window lets you execute PC-DOS commands from within TopView, and although the command list is abridged, what remains can accomplish quite a lot. Altogether, TopView supports 23 DOS commands, 6 as selectable-word "icons" and 17 in a regular menu (you have to type these in). They include COPY and ERASE, among others, and a rather compre-

(continued)

Photo 1: *TopView's multitasking features let you run simultaneous applications. This photo shows* BASIC *and TopView's Calculator program running at the same time.*

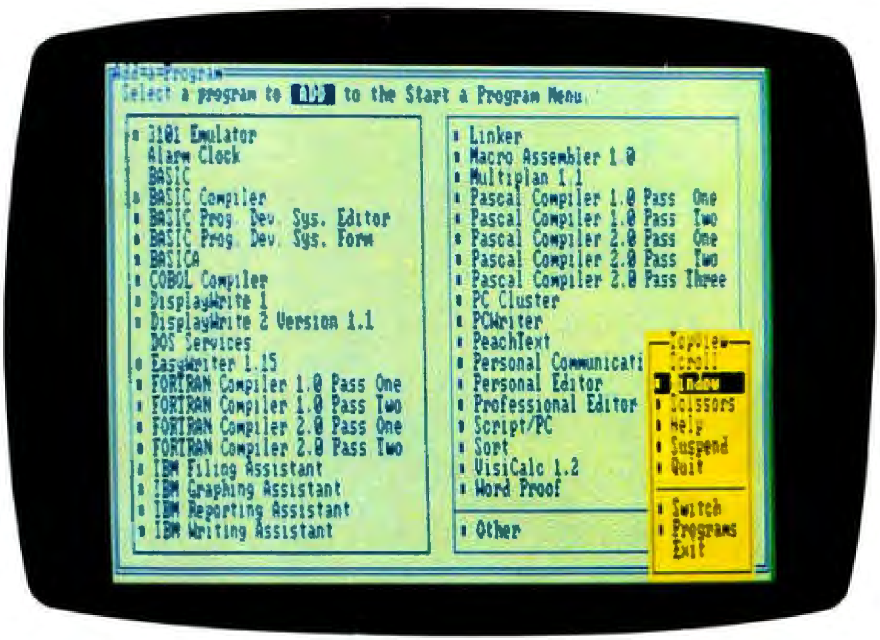


Photo 2: *TopView's Add a Program utility contains the installation parameters for 38 programs. Installing these programs involves simply placing the cursor over a name and hitting a key.*

hensive SORT command.

The purpose of DOS Services, of course, is to let you step in and out of PC-DOS without having to leave TopView. IBM also claims that the icon-generated commands make PC-DOS easier to use by eliminating the lengthy DOS equations. I don't find this to be true. In some cases, it would be faster to just enter the equation, rather than having to dole out the instructions through three windows.

The most useful TopView DOS command is undoubtedly PRINT. In its new enhanced form, PRINT allows you to print files in the background while you exit DOS and proceed with other TopView routines. Moreover, PRINT contains a file buffer that permits you to specify the printing of more than one file at a time; filenames are queued one after another, and wild-card entries are legal. If you request a PRINT C:*.*, for instance, TopView would type out the entire contents of your hard disk, which may contain hundreds of files, while you busy yourself with other duties.

The full impact of this accomplishment is somewhat marred by TopView's lack of support for a serial printer. Unless your application program contains a serial-printer function and you are working from that program, you are limited to parallel printer interfaces. Perhaps this deficiency will be corrected in later versions of TopView, but I doubt it. With fewer and fewer new serial printers appearing on the market, IBM may have simply decided the extra programming and memory buffering required is not worthwhile.

Among the attributes particular to TopView are SCROLL, COPY, CUT, and PASTE. Although these are features commonly found in most word processors, TopView has extended their capacity to include interaction between programs. TopView lets you work between the programs, transferring data and files as you wish. If, for instance, you want to transfer an address from a database file into a word-processing program, you can do it using the SCISSORS command, provided the two programs contain the TopView SCISSORS subroutine. Presently, only IBM software contains these routines, and not all IBM programs at that. But IBM has already released a TopView Programmer's ToolKit that contains most of the TopView subroutines on two disks, so I suspect it won't be long before we begin seeing TopView-compatible software from third-party vendors.

Photo 3: *Under TopView, you use the Start a Program menu to begin applications and utilities. One utility, DOS Services, lets you use DOS commands by choosing from a menu (6 commands) or entering from the keyboard (17 commands).*

SOFTWARE COMPATIBILITY

Installing a couple of TopView subroutines into a program doesn't necessarily make it TopView-compatible. Programs that manipulate DOS or the BIOS (basic input/output system) routines won't work with TopView. Neither will programs that rely on precise timing cycles for their operation (because of the way TopView slices up the microprocessor's time) or software that loads at a specific memory location. Memory control must be the absolute domain of TopView if it is to function at all. Programs that directly control computer hardware (other than the keyboard or communications port) are strictly taboo. And certain interrupt calls are also not allowed. CopyIIPC, for example, is not TopView-compatible, because of the way it modifies the internal workings of the computer. And although I haven't had a chance to test Flight Simulator on TopView, I doubt it will work for many of the above reasons.

Some copy-protected programs also have a hard time with TopView. Often the problem is getting the program to load into the TopView environment, a problem caused by the copy-protection scheme itself. This is especially true of programs like SideKick that use "fingerprints" to prevent unauthorized use. Even when such a program is copied to the TopView program, the disk must remain resident in the A: drive for access to the fingerprint. Writing Assistant also works this way. Unless you have the correct disk in the drive at exactly the moment the program expects it (or if the program's timing cycle is a little bit off), the computer shuts itself down. And despite IBM's claim that the effects of this action are nondestructive, it isn't necessarily so. Sometimes a warm boot (hitting Ctrl-Alt-Del) can recover the system, minus the files, but in many cases a complete machine restart is the only recourse. The difficulty in loading copy-protected material can sometimes be resolved by suspending all TopView application programs while the new program is loading.

Communications programs are another tough nut to crack. Many of the public-domain communications programs simply won't work due to the time-slicing effects of TopView. Only communication protocols with a pacing mechanism are recommended for asynchronous communications. This will ensure that no data is lost because of unavailable time slots. A few examples of pacing-mechanism protocols are XON/XOFF, XMODEM, and Kermit. Kermit, by the way, can be found in the public domain.

What *is* TopView-compatible, then? Just about any program that runs in PC-DOS and is not limited by the conditions mentioned above. There are many such programs. Nearly all IBM software is TopView-compatible, as are many of the most popular non-IBM software packages, such as dBASE II, Electric Desk, ProKey, and WordStar. A rather notable exception among this elite group is IBM's Debug, which is not TopView-compatible. Overall, though, you will find that most software written for the IBM PC is TopView-compatible. IBM has gone to great lengths to guarantee that whatever runs on PC-DOS should run in TopView, though some of TopView's attributes may not be available to all programs. Even networking programs, like Telmar's MicroNetwork Service 5.0, can be used within TopView's environment.

You have to be a little careful, however, of how you set up your applications. Programs that run under interpreted or compiled languages, for instance, often change the software interrupt vectors. Running more than one such program under TopView can cause both to look into the same memory space for program information. The result is competition between the programs. The same condition can occur with two programs that open files using the same filename. You can resolve the conflict, in many cases, by installing the programs in different directories so that each program has its own unique file path.

AT A GLANCE

Name
TopView

Type
Multitasking environment

Manufacturer
IBM
POB 1328
Boca Raton, FL 33432
(800) 447-4700

Computer
IBM PC, XT, AT (and compatibles) with 256K minimum, 640K and a hard disk preferred; requires PC-DOS 2.0 or higher

Documentation
Loose-leaf manual in a standard IBM binder along with two application guides and a Quick Reference foldout

Warranty
90-day limited warranty on the program itself; the warranty states that if a customer finds a program defect, IBM will correct it or refund the customer's money

Price
$149

TopView Utilities

IBM has installed its own utility programs in TopView. Version 1.00 incorporates four of them, including DOS Services, a functional alarm clock (which lets you set up to five alarms within a 24-hour period), and a Color Change Program. This last utility allows you to specify colors for the background, foreground, and window borders.

My favorite utility, though, is the TopView Calculator. The calculator itself is nothing special—you can get the same thing at the supermarket for about five bucks. But the TopView Calculator is always handy and its batteries never go dead, which always seems to be the case with the half dozen or so calculators I have scattered about.

The calculator does all of the standard mathematical functions like add, subtract, multiply, divide, percentage, squares, and square roots. It even has a memory. You can enter data into the calculator in any of several ways. If you use a mouse, you can roll it around until the cursor is over the desired "key," then click. You can also move the cursor with the cursor keys, which is quite a chore—it takes at least three keystrokes to move the cursor one calculator button. Fortunately, the calculator will accept input from the regular keys and the numeric keypad. You can execute math functions from either the regular keyboard or the function keys. Unfortunately, IBM forgot to put a function-key template into the TopView package, so unless you memorize the functions, you will always be looking up their definitions. This is not hard, since the calculator includes a very comprehensive help menu (as do many TopView programs) that fully outlines all calculator operations and options.

Like so many other IBM introductions, TopView may very well set new standards for the computer community. Beyond its computer-based implications, IBM has introduced a unique feature: It sells TopView with a limited 90-day warranty. This means that if within 90 days of purchasing it you find a defect in the program, IBM will correct it or refund your money.

Conclusion

TopView is a very sophisticated and potentially useful program. It has its problems, such as its voracious appetite for memory and the difficulty or inability of incorporating some application programs, but TopView's innovative multitasking features, its very attractive price tag of $149, and the limited 90-day program warranty will attract a lot of takers. ■

To 27 million Americans, this scene is a fairy tale.

That's because 27 million American adults can't read a child's bedtime story, can't read a street sign, can't read...period.

Functional illiteracy has become an epidemic, an epidemic that has reached 1 out of 5 American adults. It robs them of a decent living, it robs them of self-respect, it robs them of the simplest of human pleasures...like reading a letter from a friend or a fairy-tale to their children.

Tragic as that is, it's not the worst part. Because people like this could be taught to read, if we had more tutors to teach them. Sadly, we don't. Today, the waiting period for a tutor can be up to a year.

You can change that by joining the fight against illiteracy yourself. It takes no special qualifications. If you can read, you can tutor or help us in countless other ways. You'll be trained to work in programs right in your own community. And you'll experience the greatest satisfaction of all...the satisfaction of helping people discover whole new lives.

So join the effort. Call the Coalition for Literacy at toll-free **1-800-228-8813.** Helping takes so little. And illiteracy robs people of so much.

**Volunteer Against Illiteracy.
The only degree you need
is a degree of caring.**

Ad Council   Coalition for Literacy

WHEN YOUR PC DOESN'T WORK

You can *find and fix most malfunctions yourself*

BY GENE B. WILLIAMS

WHEN YOU BUY an IBM Personal Computer, you get two diagnostic aids to help you spot many of the possible causes of malfunctions. First, each time you turn on the power, the PC goes through a self-testing cycle called POST (power-on self-test). And, second, there is a diagnostics program in the back of the *Guide to Operations* manual that provides more extensive testing of the various systems, circuit boards, and peripherals on your computer.

The major difference between these two diagnostic aids lies in the extent of the testing performed. The IBM folks did not include sufficient instructions on how to use the aids, nor did they provide simple error messages. There are no messages like "I think your RAM [random-access read/write memory] is bad." Instead you see a string of numbers like "0340201." Once you understand how to decode the string, however, you'll find that it tells you not just *what* is malfunctioning, but *where*. Both POST and the diagnostics program generate the same codes. So, once you know the decoding procedure for one, you know it for both.

These two methods of testing, coupled with some simple steps and common sense, tell you exactly what has gone wrong and where, for about 95 percent of all malfunctions. Chances are good that you can find and fix what appears to be a major failure with little more than your fingers. Instead of spending hours of downtime, and perhaps hundreds of dollars, you can usually do the job yourself in a few minutes. This article covers all of the basic steps. For more information, see my book *How to Repair and Maintain Your IBM PC* (Chilton Book Co., 1984).

POST

Each time you flip on the power switch to the IBM PC, POST runs a quick check on the major systems inside. This testing takes from a few seconds to about three minutes, depending on how much RAM you have. During this time, POST conducts quick tests on the system board, RAM, power supply, keyboard, external chassis (if you have one), and some of the adapter cards. During this testing cycle, nothing shows on the screen, nor is there any indication that anything is happening (except for the delay) unless POST finds a malfunction. Then, it signals you with error codes and/or audio signals.

A part of any diagnostics aid is knowing how things are *supposed* to work. After you become accustomed to using the PC, you can easily forget the exact sequence of beeps and grinds it makes as it boots up. This is important to

(continued)

Gene B. Williams is a professional writer and the author of the series, How to Repair and Maintain Your . . . (IBM PC, Apple, VCR, Kaypro, Macintosh). *His other interests include the martial arts and ham radio. He can be reached c/o* BYTE, POB 372, *Hancock*, NH 03449.

ILLUSTRATED BY DIANE TESKE HARRIS

know; however, without these sounds you are less likely to realize when something malfunctions.

For example, one day when you sit down at your PC, just before the program begins to load, you hear a long beep and then a short one. You might recognize this as being something different from the norm or you might not. If the program loads successfully, you're quite likely to ignore the warning—only to lose all your work a few hours later when the PC locks up or refuses to save your work due to a system-board malfunction.

The next time you begin working at your PC, pay attention to the sounds of the normal boot sequence. Then, make paying attention a part of your normal routine. At the very least, let yourself become aware that something other than the norm *can* happen.

When everything works correctly, the cursor begins blinking on the screen a few seconds after you turn on the power. It continues to blink during POST. The light on drive A comes on, there is a single short beep, and the LED (light-emitting diode) goes out momentarily. When it comes on again, the program loads. Anything other than this sequence indicates that there is a problem.

The original plans for the IBM PC included a cassette-tape machine to load and store programs. Although the idea was abandoned almost immediately, the cassette port remained. So did a form of BASIC resident in ROM (read-only memory) to operate the cassette. So, if you're attempting to load a program and you suddenly find yourself staring at a BASIC screen, drive A isn't loading the program. ROM has taken over and has automatically booted up cassette BASIC.

Try loading another program, and carefully observe the sequence. As the program loads, you should hear the drive's stepper motor moving the read/write heads across the disk. If the drive is trying to load but can't for some reason, the cassette BASIC will load from ROM.

POST Errors

Table 1 contains all the primary error codes, signals, and symptoms generated by POST. An *x* in the code represents any number. For example, "*xxxx*201" won't appear on the screen with the *x*s, but with numbers, such as "0340201." This code indicates a problem with the RAM (the "201" ending), and it also tells you that the program found an error in the second RAM module from the right on the front row (the "0340" prefix).

A few of the indications given are not strictly a part of POST but are common enough symptoms at power-up to warrant mention. The "nothing happens" symptom, for example, is not a generated error code, nor is the cassette BASIC display. Yet both are strong indications that something is wrong. Some of the error signals are audible beeps. These may or may not be accompanied by actual error codes.

If you have either an error code or a symptom, you should use the diagnostics disk, which checks the system more thoroughly. If the PC won't load the diagnostics program, there are some initial steps that you can take. Most of these are self-evident. If nothing at all happens when you flip on the power switch, most likely no power is getting to the PC. The actual problem could be anywhere. The wall outlet might be dead. The cord might have been accidentally kicked out. The power supply inside the PC could be faulty. Or something deeper inside the PC could be shorting out, causing the power supply to drag down and shut itself off.

Diagnostics Disk

One mistake many people make is to use the diagnostics program only when a malfunction occurs. This is somewhat like going to the doctor only after you're seriously ill. You should use the diagnostics program as a preventive check. If something is starting to fail, the program might spot it before it becomes a serious problem. If you use the diagnostics program on a regular basis, you'll have a better idea of how things are operating.

The diagnostics disk that comes with your PC does a more thorough testing of your computer and its systems than POST does. It also offers the option of running multiple testings. This is very important when it comes to testing circuits and IC (integrated circuit) modules. Unlike less complex components that either function or fail, a chip can work fine, then it can fail and then work again. A chip is also more prone to operate correctly while it is cool and then break down after the PC has been going for a while. POST or the single-testing option of the diagnostics program are less likely to spot such malfunctions. The multiple-testing option of the diagnostics program should be used to detect such problems. If you suspect that one of the circuit boards is faulty and if you suspect the system board in particular, choose the multiple-testing option. You should let the test run a minimum of 30 times, preferably more.

To run the diagnostics program, insert the disk in drive A and reset (reboot) the PC (by Ctrl-Alt-Del or by shutting it off, waiting 5 seconds, and turning it on again). The normal sequence of beeps and grinds should occur. After POST completes its cycle, the light on drive A should come on, followed by a single short beep, and then you will be able to hear the grinding of drive A as the program loads.

As testing begins, you may log any errors to disk. The log itself can be kept only on the diagnostics disk; however, if you choose this option you cannot cover the write-protect notch. Obviously, to log errors to disk, your PC and its disk drives must be running well enough to be able to write to the disk.

You can also log the errors to a printer, which is how I prefer to do it. This frees the disk drive and allows you to let the testing run unattended.

As the diagnostics program executes, it tests the disk drives. When it's ready to run the test, it stops and signals you with a long beep and then

If nothing at all happens when you flip on the power switch, most likely no power is getting to the PC. The actual problem, though, could be almost anywhere.

a short one so that you can replace the diagnostics disk with a scratch disk. (The reason a scratch disk is needed is because the testing destroys any data on the disk.) If you've installed additional RAM as one or more RAM drives, you will get errors on these drives. This portion of the diagnostics program tests physical drives only. (The RAM chips are all tested earlier in the program, regardless of their function.)

Diagnostics Errors

Table 2 lists the primary error codes, signals, and symptoms generated by the diagnostics program. The primary

(continued)

Table 1: POST *error signals and codes.*

SIGNAL OR CODE	MEANING
Nothing happens	Power supply or ?
Continuous beep	Power supply
Repeating short beeps	Power supply
1 long beep, 1 short	System board
1 long beep, 2 short	Monitor
No display	Monitor
Cassette BASIC display	Disk drive (usually A)
101,131	System board
201	RAM
*xxxx*201 and Parity Check *x*	RAM
Parity Check *x*	Power supply
301, *xx*301	Keyboard
601	Floppy-disk drive
1701	Hard-disk drive
1801	Expansion chassis

If there's no display and you must resort to the audio signals alone, move cautiously.

difference between these and those given by POST is that the diagnostics program generates a code even when a particular system or circuit is operating properly. POST does not. This successful-completion code ends in all zeros. (For example, when the system board successfully completes testing, a 100 code is generated, while properly operating floppy-disk drives give a 600 code.)

Audio error signals are also provided, making it possible to run complete diagnostics even when the monitor is dead. The order of the tests and screens that are supposed to appear are given in the *Guide to Operations* manual. However, this doesn't include a key to the audible signals. It's a good idea to run the diagnostics program before you suspect any problems and make appropriate notes about the meanings and occurrences of these signals in your manual.

If your monitor is not displaying anything and you must resort to the audio signals alone (see table 3), move slowly and cautiously. Pay extra attention to what happens and when. If you lose track or doubt what is happening for any reason, begin again.

Table 2: *Diagnostics program codes.*

CODE	MEANING
02*x*	Power
1*xx*	System board
20*x*	RAM
xxxx	RAM
*xx*20*x*	RAM
30*x*	Keyboard
*xx*30*x*	Keyboard
4*xx*	Monochrome monitor
5*xx*	Color monitor
6*xx*	Disk drive
7*xx*	8087 math coprocessor
9*xx*	Printer adapter
11*xx*	Asynchronous communications
12*xx*	Asynchronous communications
13*xx*	Game adapter
14*xx*	Printer
15*xx*	SDLC communications
17*xx*	Hard-disk drive
18*xx*	Expansion chassis
20*xx*	BSC adapter
21*xx*	BSC adapter

Table 3: *Diagnostics disk audio run-through.*

1. Power is applied; cursor should blink during POST; *short beep*; program loads.
2. "Select an Option" screen (0—run diagnostics routines, 1—format disk, 2—copy disk, 9—exit to system disk); enter "0" for diagnostics routines.
3. "Installed Devices" screen; *beep*; enter "Y".
4. "Diagnostics Choices" screen (0—run tests one time, 1—run tests multiple times, 2—log utilities, 9—exit diagnostics); enter "0" or "1"; program now proceeds through tests for the system board and RAM, and there will be a delay of up to several minutes; this delay will be followed by a short beep.
5. "Typematic" (keyboard test) screen; press Enter; *short beep*.
6. "Display Attributes" screen; enter "Y"; *short beep*.
7. "Character Set" screen; enter "Y"; *short beep*.
8. "80x25" display; enter "Y"; *short beep*.
9. "40x25" display; enter "Y"; *short beep*.
10. "320x200 Graphics" display; enter "Y"; *short beep*.
11. "640x200 Graphics" display; enter "Y"; press any key eight times, *short beep*; enter "Y".
12. *Long beep–short beep* signals beginning of drive tests. Insert scratch disk in drive A; press Enter; drive LED will light and drive will grind; *two beeps*; enter "Y".
13. Repeat step 12 for each drive installed (*long beep, short beep*, scratch in B, Enter, LED lights, *grind, beep beep*, enter "Y"); ignore results on RAM drives.
14. Short delay and a single short beep signals the end of a completed cycle.

FINDING THE MALFUNCTION

Tracking down a malfunction isn't as difficult as you might think. It requires little more than the process of elimination. Just where you begin your search depends on the symptoms. It is usually fairly obvious where to start looking. There are just so many problems that can occur and just so many causes for a particular malfunction.

The process of elimination begins with the simple and the most obvious and then moves in a step-by-step order to the more complex. For example, if your PC is completely dead, start from the outside and work in. Check the wall outlet; plug something else into that outlet, such as a lamp, or use a volt-ohmmeter (VOM) to test the outlet. And check for the obvious—is the plug properly inserted into the outlet?

Next, eliminate anything else outside the PC, including any switches. If your computer is plugged into a power strip or another external device, bypass it, and plug the machine directly into the outlet to see if the problem lies in that device. (The internal switch on the PC almost tests itself. If the fan is running, power is getting to the computer and the switch is good. If the fan *isn't* running, then power *isn't* getting to it, due to either a lack of power or a faulty switch.)

After eliminating everything up to the power supply as the possible cause, try to find out if the problem is coming from the power supply or from somewhere else in the computer. You can do this best by testing the power supply outputs with a VOM. Or you can disconnect everything inside the PC, including all the circuit boards. If the power returns, you know that one of the connected devices caused the problem. Then connect the devices again, one at a time, until the power fails. (*Be sure to shut off the power before removing or connecting anything!*)

The same basic procedure applies to other malfunctions. Set up an either. . .or sequence. Find out what is *not* malfunctioning and then you'll know what *is*. For example, if a program fails to load, the problem is with either the disk drive or the program. If the drive is failing, the cause is probably either the drive itself or the circuit boards in the computer. (With disk-drive problems, you can swap drive A with drive B. If things begin to work again, you know that something is wrong with drive A. If not, then the problem lies with either the drive card or the system board.)

PROBABLE CAUSES

The single most common cause of computer malfunction is operator error. You may have accidentally kicked out the plug or turned the contrast or other controls on the monitor. Operator error is often the last thing suspected or checked. Checking for the obvious and possible operator errors are always the first steps to take in tracking down a malfunction. This is especially important if you've made some change to your system. Installation instructions and software documentation are notorious for being poorly written. It's easy to make a simple mistake.

For example, one PC owner upgraded his system with a new multifunction card that was fully populated with RAM. Off came the cabinet, and in went the board (after carefully following the instructions). However, when he tried to use it, the PC locked up and refused to do anything but reboot—and then lock up again. His automatic response was to blame the new board, but after removing the board, his computer still didn't operate correctly.

What seemed to be a major computer malfunction turned out to be a simple problem. When he added the new board, he brought the total number of COM ports to three. The PC handles only two. With that third port installed, the PC no longer knew what to do and locked up. During installation he had changed the switch settings on the system board. When

(continued)

the expansion board was removed, the switch settings still indicated the additional RAM. When it wasn't there, the PC generated an error. All he had to do to "fix" his PC was to change a switch block on the new board to disable one of the COM ports.

Once you've eliminated operator error, check the software. Many errors are brought on by not completely understanding how to use your software. The more complex and powerful a program, the more likely you are to make a mistake along the way. Even a complete software crash isn't necessarily a software problem. According to Verbatim Corporation of Sunnyvale, California, at least 80 percent of all software and disk failures are directly due to fingerprints on the disks.

Then there are the bugs that manufacturers almost invariably miss. A program may work just fine in its primary form, but when you try a new feature, it doesn't react the way you expect. This may be a manufacturer's bug, but it may also be another operator error, caused by the operator not fully understanding how to use the new feature.

Once you have eliminated operator and software errors, you can concentrate on the PC itself. Suspect the mechanical devices before the electronics. Anything that moves is prone to wear. A drive failure is more common than a chip failure. (Failure to load programs can be as simple as a broken or improperly closed disk-drive door.)

SUMMARY

Computer malfunctions are relatively rare. Despite the fact that computers seem to be complex, a computer with the power of the IBM PC is actually much less complicated than a portable television set. There just isn't much that *can* malfunction. When something does go wrong, it is most likely to be caused by: first, operator error; second, software error; third, mechanical problems; and last, the electronics. By using the process of elimination, you can usually track down the source of the trouble fairly quickly.

POST and the IBM PC's diagnostics disk can help you pinpoint the causes of many malfunctions. However, there are times when neither of these will run. Normally, this indicates a more serious malfunction. (It can also mean that nothing at all is wrong.) There are also times when these diagnostic aids won't be able to find the cause of the problem. Then, when all else has failed, you may need to call a repairman.

The most important point I can make is this: Realize that you *can* find and fix most malfunctions if you move carefully and thoughtfully—one step at a time. ■

IBM PC FAMILY BIOS COMPARISON

BIOS *vectors for the* PC *and its relatives*

BY JON SHIELL AND JOHN MARKOFF

IBM'S PERSONAL COMPUTER has grown into an impressive family of related machines. In addition to the original system, there are the PC XT, XT/370, 3270 PC, Portable PC, PCjr, PC AT, and AT/370, as well as several industrial and military versions of the XT.

As the PC family has grown there have been additions and deletions to the basic hardware of the machines. This article and the accompanying tables 1, 2, and 3 compare the various BIOS (basic input/output system) features of the PC-family machines. If you wish to program on one specific machine, these tables can assist you in determining which functions can be used across all machines in the family.

The purpose of the BIOS is to present a common interface to programs, be they application programs or operating systems, to minimize the amount of code that must be rewritten between machines. The BIOS allows the programmer to isolate hardware dependency to a single set of primitive routines. The BIOS permits portability and compatibility between different hardware environments. At the same time, almost all of the speed and control of direct hardware access is retained.

The BIOS is made up of the code/programs that provide the device-level control for the major I/O devices in the system. In the IBM PC family the BIOS is contained in ROM (read-only memory) on the system board, along with cassette BASIC and a set of routines (called POST for "power-on self-test) that check out the machine when it is turned on.

The BIOS creates hardware independence by providing a level of indirection and separation from the hardware. For example, when using a BIOS call to send a character to a printer, the programmer doesn't need to know what the I/O address of the printer port is or how to control it.

The BIOS is normally invoked via a set of interrupts that are vectored into various BIOS entry points. Other interrupt vectors are used to service hardware interrupts, such as "disk operation finished." In practical terms, the software invokes the BIOS by loading the appropriate registers in the microprocessor and issuing an INT instruction. For example:

```
MOV AH,0   ; Load AH with the
             BIOS function code
             for "print the charac-
             ter in register AL"
MOV AL,'B' ; Character to be
             printed, in this case a
             "B"
MOV DX,0   ; Print it on LPT1
             (printer number
             minus 1)
INT 17     ; Printer BIOS entry
             interrupt
```

The INT (or software interrupt) instruction transfers control of the microprocessor to the routine whose address is in the 4-byte interrupt vector for this interrupt. There are 256 interrupts in the 808x microprocessor family; the first 128 are used by the BIOS and the operating system, the other 128 by BASIC. These 256 addresses are arranged in table form in the first 1K byte of memory, where bytes 0 to 3 are for INT 0 and bytes 3FC to 3FF are for INT FF (addresses and interrupts are in hexadecimal).

The BIOS is extensible. When the POST routines perform their start-up diagnostics, they scan the ROM address space for "add-on" routines that install themselves. An example of this extension is the IBM Enhanced Graphics Adapter, which extends the video interrupt INT 10 as indicated in table 1.

To install itself as a BIOS extension, a routine

1. Copies the current interrupt pointer into your routine.
2. Disables interrupts.
3. Replaces current interrupt pointer with the entry address of your routine.
4. Reenables interrupts.

If the routine is installed after DOS, the DOS functions 35 (get interrupt vector) and 25 (set interrupt vector) can be used.

The rule for BIOS entries is one software interrupt per device. Additionally, there may be one or more hardware entries and one or more entries that point to tables or blocks of data used by the device driver. The interrupt vectors used as pointers to data instead of code allow easy alteration of the enviroment, such as changing the character set displayed for 80 to FF by the Color Graphics Adapter.

According to IBM, the only time you can safely bypass the BIOS is when you access the following:

- I/O port 21: Interrupt mask registers.
- 61: Sound control.
- 40,41,42: Timer/counter. (Note:

(continued)

Jon Shiell is a systems architect and microprogrammer. He can be reached at POB 61195, *Sunnyvale,* CA 94088.

John Markoff, a technology writer for the San Francisco Examiner, *is a former* BYTE *editor. He can be reached at* 110 *Fifth St., San Francisco,* CA 94123.

Table 1: *The* ROM BIOS *vectors for the* IBM PC *family.*

Interrupt	Function Code	PC	PC XT	PCjr	PC AT	BIOS Ext.	Comments
0	n/a	y	y	y	y		divide-by-zero trap
1	n/a	y	y	y	y		single-step mode (used by Debug)
2 NMI	n/u	y	y		y		parity-check routine
					y		also coprocessor interrupt
	n/u	n	n	y	n		keyboard-interrupt routine (keyboard has data ready)
3	n/a	y	y	y	y		breakpoint (used by Debug)
4	n/a	y	y	y	y		overflow trap
5	n/a	y	y	y	y		print-screen function
		y	y	y	y		uses address 50:0 for status
6	n/u						reserved
7	n/u						reserved
8	n/a	y	y	y	y		timer-interrupt handler
9	n/a	y	y	y	y		keyboard-interrupt handler
A	n/a	n	n	n	n	ega	vertical retrace interrupt
B	n/a	y	y	y	y		communications controller (e.g., serial port) hardware entry
C	n/a	y	y	y	y		communications controller (e.g., serial port) hardware entry
D	n/a	y	y	y	y		alternate printer (PC AT's 80287 interrupts first come in here)
E	n/a	y	y	y	y		disk-controller interrupt entry
F	n/a						reserved
10 video I/O	0	y	y	ye	y	ega	set CRT mode
	1	y	y	y	y		set cursor type
	2	y	y	y	y		set cursor position
	3	y	y	y	y		read cursor position
	4	y	y	y	y		read light-pen position
	5	y	y	ye	y		select active display page
	6	y	y	y	y		scroll active page up
	7	y	y	y	y		scroll active page down
	8	y	y	y	y		read attribute/character at current cursor position
	9	y	y	y	y		write attribute/character at current cursor position
	10	y	y	ye	y		write character only at current cursor position
	11	y	y	ye	y		set color pallette
	12	y	y	y	y		write dot
	13	y	y	y	y		read dot
	14	y	y	y	y		teletypewrite to active page
	15	y	y	y	y		return current video state
	16	n	n	y	n	ega	set palette registers

Interrupt	Function Code	PC	PC XT	PCjr	PC AT	BIOS Ext.	Comments
10 (continued)	17	n	n	n	n	ega	character-generator routine
	18	n	n	n	n	ega	alternate select
	19	n	n	n	y	ega	write string (with optional attributes)
	FE	n	n	n	n	tv	get screen buffer address (text mode only)
	FF	n	n	n	n	tv	update real display (text mode only)
11	n/u	y	y	y	y		equipment determination, returns status of optional equipment
		y	y	y	y		uses BIOS data area: EQUIP_FLAG
12	n/u	y	y	y	y		memory-size determination, returns amount of memory in the system
13 floppy disk	0	y	y	y	y		reset disk system
	1	y	y	y	y		read status of last operation
	2	y	y	y	y		read sectors into memory
	3	y	y	y	y		write sectors from memory
	4	y	y	y	y		verify sectors
	5	y	y	y	y		format a track
	15	n	n	n	y		read disk type (none, disk no change line, disk, hard disk)
	16	n	n	n	y		disk change status
	17	n	n	n	y		set disk type for format
	n/a	y	y	y	y		uses BIOS data area DISK_POINTER
	n/a	n	n	n	y		uses disk 'state machine' at 40:90 to 95
13 hard disk	n/u		y		y		not used by PC or PCjr
	0		y		y		reset disk system
	1		y		y		read status of last disk operation
	2		y		y		read sectors into memory
	3		y		y		write sectors from memory
	4		y		y		verify sectors
	5		y		y		format a track
	6		y		n/u		format a track and set bad-sector flags
	7		y		n/u		format the drive starting at the desired track
	8		y		y		return the current drive parameters
	9		y		y		initialize drive-pair character (INT 41 used by XT, 41 and 46 by AT)
	0A		y		y		read long
	0B		y		y		write long
	0C		y		y		seek to desired track
	0D		y		y		alternate disk reset
	0E		y		n/u		read sector buffer
	0F		y		n/u		write sector buffer
	10		y		y		test drive ready
	11		y		y		recalibrate
	12		y		n/u		controller RAM diagnostic
	13		y		n/u		drive diagnostic
	14		y		y		controller internal diagnostic
	15		n	n	y		read disk type (none, disk no change line, disk, hard disk)
14 RS232 I/O	0	y	y	y	y		initialize communications port
	1	y	y	y	y		send a character
	2	y	y	y	y		receive a character
	3	y	y	y	y		get port status

(continued)

Interrupt	Function Code	PC	PC XT	PCjr	PC AT	BIOS Ext.	Comments
14 (continued)	n/a						uses BIOS data areas
		y	y	y	y		RS232_BASE (0:3)
							space for four ports but BIOS supports only two
		y	y	y	y		RS232_TIM_OUT
15							cassette I/O and advanced functions
	n/u		y				not used on PC XT
	0	y	n	y	n/u		turn cassette motor on
	1	y	n	y	n/u		turn cassette motor off
	2	y	n		n/u		read from cassette
	3	y	n	y	n/u		write to cassette
	20	n	n	n	y		AL = 10, setup of SYSREQ routine*
		n	n	n	y		AL = 11, completion of SYSREQ function*
	80	n	n	n	y		device open*
	81	n	n	n	y		device close*
	82	n	n	n	y		program termination*
	83	n	n	n	y		event wait
	84	n	n	n	y		joystick support
	85	n	n	n	y		AL = 0, system-request key has been pressed*
		n	n	n	y		AL = 1, system-request key has been released*
	86	n	n	n	y		wait (for *N* microseconds)
	87	n	n	n	y		move block of memory
							(can move to and from extended memory)
	88	n	n	n	y		return amount of memory above 1 megabyte
	89	n	n	n	y		switch processor to virtual mode
	90	n	n	n	y		device-busy loop*
	91	n	n	n	y		interrupt complete flag set*
16 keyboard	0	y	y	y	y		read next ASCII character
	1	y	y	y	y		set Z flag if buffer not empty
	2	y	y	y	y		read shift status
	3	n	n	y	n		set Typematic rates
	4	n	n	y	n		turn on/off keyboard click
17 printer	0	y	y	y	y		print a character
	1	y	y	y	y		initialize printer port
	2	y	y	y	y		return printer-port status
	n/a						uses BIOS data areas
		y	y	y	y		PRINTER_BASE (0:2) space for three printers
		y	y	y	y		PRINTER_TIM_OUT
18 ROM BASIC	n/a	y	y		y		points to resident BASIC
19	n/a	y					bootstrap loader, IPL the system from a disk
	n/a		y		y		bootstrap loader, IPL the system from a disk or hard disk
	n/a			y			bootstrap loader, IPL the system from a disk or cartridge
1A time of day	0	y	y	y	y		read current clock setting
	1	y	y	y	y		set the current clock
	2	n	n	n	y		read the real-time clock
	3	n	n	n	y		set date from real-time clock
	4	n	n	n	y		read date from real-time clock
	5	n	n	n	y		set date into real-time clock

Interrupt	Function Code	PC	PC XT	PCjr	PC AT	BIOS Ext.	Comments
1A (continued)	6	n	n	n	y		set the alarm (24-hour maximum, goes off and causes an INT 4A)
	7	n	n	n	y		reset the alarm
& sound select	80	n	n	y	n		set up sound multiplexer
1B	n/u	y	y	y	y		keyboard break address
1C	n/u	y	y	y	y		timer tick
1D	n/u	y	y	y	y		video parameters for 6845 initialization
1E	n/u	y	y	y	y		disk parameters
1F	n/u	y	y	y	y		graphics character extension for 320 (and 640) by 200 mode color graphics adapter
20–3F							DOS functions
40	n/u	n	y	n	y		pointer to disk BIOS entry
41	n/u	n	y	n	y		pointer to first hard disk, parameter block
42	n/u	n	n	n	n	ega	pointer to screen BIOS entry
43	n/u	n	n	n	n	ega	pointer to EGA initializing parameters
44	n/u	n	n	y	n	ega	pointer to EGA graphics-character table (also PCjr)
45	n/u						reserved
46	n/u	n	n	n	y		pointer to second hard disk, parameter block
47	n/u						reserved
48	n/u	n	n	y	n		cordless-keyboard translation
49	n/u	n	n	y	n		nonkeyboard scan-code translation-table address
4A–59	n/u						reserved
5A	n/u					clu	cluster-adapter BIOS entry address
5B	n/u						reserved
5C		n	n	n	n	net	IBM PC Network NETBIOS entry point
5D–5F	n/u						reserved
60–67	n/u						reserved for user program interrupts
68–6F	n/u						not used
70	n/u	n	n	n	y		IRQ 8, real-time clock interrupt

(continued)

Interrupt	Function Code	PC	PC XT	PCjr	PC AT	BIOS Ext.	Comments
71	n/u	n	n	n	y		IRQ 9, redirected to IRQ 2
72	n/u	n	n	n	y		IRQ 10
73	n/u	n	n	n	y		IRQ 11
74	n/u	n	n	n	y		IRQ 12
75	n/u	n	n	n	y		IRQ 13, coprocessor, BIOS redirect to NMI interrupt (INT 2)
76	n/u	n	n	n	y		IRQ 14, hard-disk controller
77	n/u	n	n	n	y		IRQ 15
78–7F							not used
80–85							reserved by BASIC
86–F0							used by BASIC when the BASIC interpreter is running
F1–FF							not used

Notes on the table:

All PC AT interrupts are valid for real mode only.
The XT/370 and 3270 PC both use the PC XT BIOS.
The AT/370 uses the PC AT BIOS.

- tv — TopView function.
- clu — Cluster adapter.
- ega — Enhanced Graphics Adapter function.
- net — IBM PC Network, NETBIOS function.
- n/a — Not applicable.
- n/u — Not used.
- n — Not supported.
- y — Supported.
- ye — Supports a superset.
- * — These INT 15 functions are just operating-system hooks; they perform no BIOS functions.

The Typematic rate of the PC AT keyboard is programmable, but no explicit BIOS support is provided. Also, the AT's keyboard has an internal 16-key buffer.

When a hard disk is present the INT 13 disk interrupt is rerouted to INT 40, and INT 13 points to the hard-disk BIOS.

When the NETBIOS is installed, interrupts 13 and 17 are interrupted by the NETBIOS; interrupt 18 is moved to INT 86 and one of INT 2 or 3 is used by the NETBIOS. Also, the NETBIOS extends the interrupt 15 WAIT and POST functions.

BIOS extension addresses

C0000–C3FFF	16K	EGA BIOS
C4000–C5FFF		
C6000–C63FF	256 bytes	Professional Graphics Display communication area
C6400–C7FFF		
C8000–CBFFF	16K	hard-disk BIOS
CC000–CDFFF	8K	IBM PC Network NETBIOS
CE000–CFFFF		
D0000–D7FFF	32K	cluster-adapter BIOS
D8000–DBFFF		
DC000–DFFFF		
E0000–E3FFF		
E4000–E7FFF		
E8000–EBFFF		
EC000–EFFFF		
F0000–FFFFF	64K	ROM BASIC and "simple" BIOS

INT 15 functions

WAIT function (A = 90): can be used by the operating-system task dispatcher to dispatch another task while the current task waits for its I/O operation to finish. This is the most efficient form of multitasking.

POST function (A = 91): I/O operation complete, which can be used to inform the operating-system task dispatcher that an I/O operation for a waiting task has been completed, and the task should now be moved to the ready queue.

Table 2: *Hardware interrupts for the* IBM PC *family of computers.*

Hardware-Interrupt Request Line	PC and PC XT	PCjr	PC AT
NMI	Parity Errors	Keyboard Interrupt	Parity Errors
IRQ 0	timer	timer-clock interrupt	timer output 0
IRQ 1	keyboard	I/O channel (reserved)	keyboard (buffer full)
IRQ 2	reserved	I/O channel	cascade for 8 to 15
IRQ 3	serial port 2	asynchronous port (RS-232C)	serial port 2
IRQ 4	serial port 1	modem	serial port 1
IRQ 5	hard disk (not on PC)	display vertical retrace	parallel port 2
IRQ 6	floppy-disk control	floppy disk	floppy-disk control
IRQ 7	parallel port 1	I/O channel (parallel printer)	parallel port 1
IRQ 8	Interrupts 8 through 15 are not available on the PC, PC XT, or PCjr		real-time clock
IRQ 9			redirected to IRQ 2
IRQ 10			reserved
IRQ 11			reserved
IRQ 12			reserved
IRQ 13			coprocessor
IRQ 14			hard-disk controller
IRQ 15			reserved

NOTE: IRQ 3 and 4 may be used by SDLC (synchronous data-link control) or bisynchronous communication cards instead of serial ports.

Table 3: IBM PC DOS *interrupts used by the* IBM PC *Network program.*

Interrupt	Code Function	Subfunction	Comments
21	3D	n/u	open file with sharing specified
	44	09	IOCTL, is device redirected?
		0A	IOCTL, is handle local or remote?
		0B	IOCTL, change sharing retry count
	59	n/u	get extended error (additional errors added)
	5A	n/u	create temporary file with unique name
	5B	n/u	create new file
	5C	00	lock byte range
		01	unlock byte range
	5E	00	get machine name
		02	set up printer-control string
	5F	02	get assign-list entry
		03	redirect device to net
		04	cancel redirection
2A	00	n/u	check to see if network BIOS is installed
	01	n/u	execute NETBIOS request
	02	n/u	set net printer mode
	03	n/u	get device-shared status
2F	BB	00	net command installation check
		03	get server POST address
		04	set server POST address

Don't change port 41.) Timer frequency will remain fixed at 1.19 MHz.

• 201: Game control adapter. (Note: Use the timer for delays.)

Concerning absolute memory locations, note the following: For interrupt vectors (0:0 to 3FF), functions will be added but no functions will be redefined. For the video-display buffers (B000:0 and B800:0), the display memory maps will not change for a given video BIOS mode of operation. If the bit map is altered, a new mode will be defined to support it. For ROM BIOS data areas (starting at 40:0), variables will retain their current definitions as long as the corresponding functions are defined—don't count on these! ■

Low power

Complete hard disk kit $395

Comes complete with virtually the identical controller that's in the IBM®XT, and Xerox® warranties the hard disk for one year

Guaranteed 100 percent IBM PC compatible

How can we offer this fantastic price? Simple. We buy in such volume that even the most avaricious hard disk businessmen understand they have to give us the best price possible. We could pocket the difference, but we don't.

Instead, we put the extra profit into our testing facilities. That's why Xerox guarantees our $395 10 megabyte hard disk for one year.

Xerox knows, as our customers know, that we have an extensive testing program. Here is what we contribute toward giving you the maximum hard disk performance.

Best Drives Available

First, we buy the best drives available. Sounds trite, doesn't it? I mean, a drive's a drive—right? Hardly. You should see some of the junk we get in our labs. Some have such high failure rates that we even questioned our own $10,000 hard disk tester. But when we tested other manufacturers' drives we were assured that our equipment was fine, which just confirmed that the bad hard disks were not only bad—they were real bad.

But that's just the weeding out process. We then take each drive that we've put through our tester and test it again with the controller you've requested. We call this a "tested pair."

DOS Doesn't Do It

In case you're thinking that all this is an unnecessary duplication of what DOS does for you, let me explain the disk facts of life.

If DOS did what you may think it is supposed to do when you format the disk, DOS would map around these bad areas. Unfortunately, DOS doesn't do this.

DOS 2.0 and 2.1 can't enter the bad tracks. DOS 3.0 can, but only on the IBM AT. Unfortunately, as the press has so well documentated, the AT's hard disk develops bad tracks later on.

We do what DOS can't

We believe the problem is so bad, we use a software program that performs a powerful test of your disk drive on all of the IBM or IBM compatible computers—PCs, XTs, and ATs. Our format takes hours to analyze the disk. But when we finish, you know that the bad tracks are really mapped out so you won't write good data that will disappear into a black hole. We even send you a printed statement of our test results.

Our software allows you to type in the bad track locations from the list supplied by the manufacturers, so you'll never write good data to them—even if DOS didn't identify them as bad. The software even lets you save the location of these bad sections to a file, so that you can reformat your disk without spending hours retesting.

We even include a program that will give you continuous comments on the status of your hard disk. No more waiting for that catastrophic failure.

Average Access Time

As you might suspect, some hard disks are faster than others in their ability to move from one track of data to another. The time it takes the hard disk to move one-half way between the beginning of the disk to the end is called the "average access time."

The first generation of 10 megabyte hard disks had average access times of 80-85 milliseconds (msec). But computer users love speed, and guess what—the average access time for the new 20 megabyte hard disk in the IBM AT is only 40 msec. (We sell an AT equivalent with only 30 msec access time!)

There are some legitimate reasons for the shorter access time. It's particularly helpful when there are multiple users on the same hard disk. It's also important when running a compiler. But remember, before you get too wrapped up in the access speed, there's always that ST 506 interface which won't let data transfer from the hard disk to the computer any faster than 5 megabits/second. We've bypassed that choke hole, too. If you want the functional equivalent of a Ferrari with a turbocharger, order our 10 Mbit per second 100 megabyte hard disk with 18 msec of average access speed.

Compatibility

To be sure that your hard disk is 100 percent compatible with the IBM XT you don't need to buy the same hard disk that's in the XT. You can't even be sure what brand hard disk it is because IBM, like Express Systems, goes into the marketplace and buys hard disks from several vendors. However, they buy their XT hard disk controller from only one vendor—the same one we do.

You can buy the IBM XT controller from IBM for $495 or you can buy from us, the functional equivalent, manufactured by the same company that makes it for IBM for only $195. Is it the exactly identical IBM XT controller? No, it's better. First, it takes less power, and secondly, it can control from 5 to 32 megabytes—the IBM controller can work with only 10 megabytes. It is 100 percent IBM XT compatible, and 100 percent is 100 percent. If you want to save a slot, we carry a version that lets you operate two hard disks and two floppy disk drives.

More than 32 Megabytes

You can operate with more than 32 megabytes (the limit of DOS) through the use of "device drivers." Express Systems can supply you with device drivers for our hard disks for over 32 megabytes formatted. But, if you don't have individual files, or databases that are large, you might want to consider one of our controllers that can divide our 65 megabyate (formatted) hard disk into two equal volumes of 32 megabytes each.

Reliability

We offer you a choice between iron oxide and plated media—the stuff that covers the hard disk and gives it its magnetic properties. Iron oxide is,—well, it's rust. If you inadvertently joust your disk, you may cause the low flying head to dig out some iron oxide. A little rust flake can ruin your whole day. Plated media is more resistant to damage, and if it happens, less data is lost.

We offer both types of hard disks. The iron oxide is older

technology, and quite frankly, manufacturers understand it better. Their better understanding, combined with some of the special head locking mechanisms, gives us peace of mind when we sell you one.

Power

Hard disks consume power. Our small, half-high hard disks consume so little power that you can use them with your existing IBM PC power supply. If you plan to use lots of slots, you'll want to increase your power supply to be safe. We offer the same amount of power for your PC that comes in the XT.

Our Customers

Some folks just never feel comfortable buying mail order. They forget that Sears began as a mail order house or that IBM is now into mail order. But, if it helps, here is a *partial* list of customers who have felt comfortable to buy from us.

IBM	Sears
American Express	Honeywell
U.S. Army	MIT
AT&T (Bell Labs)	RCA
Bausch & Lomb	Lockheed
Xerox	Sperry

Easy to Install

If you're like most of us, raised on the boob tube rather than the Great Books, you'd rather see the movie than read the book. Well, now you can choose to read our installation manual or for only $9.95 more, you can get a VHS or Beta video cassette showing the simple steps for installation.

Warranty

We offer you a one year warranty on our hard disks—the same as IBM on the AT and 90 days on the tape drives. (It's all the manufacturer gives us.) If anything goes wrong with your tape or disk drive or hard disk, send it back in the box it came in. However, we have found that we can usually solve the problem over the phone. So call first for a return authorization number because we can't accept any returns without it.

Complete Hard Disk Kits

Formatted MB	Height	Plated Media	Average Access	Transfer Rate	PC or PC/XT	AT
10	½	no	85 msec	5 Mbits/s	$ 395	$ N/A
10	½	yes	85 msec	5 Mbits/s	$ 495	$ N/A
21	½	yes	85 msec	5 Mbits/s	$ 795	$ 595
21	Full	no	30 msec	5 Mbits/s	$ 1,535	$ 1,340
32	½	yes	85 msec	5 Mbits/s	$ 995	$ 795
32	Full	no	30 msec	5 Mbits/s	$ 1,775	$ 1,575
65	Full	no	30 msec	5 Mbits/s	$ 2,295	$ 2,070
100	Full	yes	18 msec	10 Mbits/s	$ 4,995	$ 4,995
Removable Hard Disk						
10	½	no	90 msec	5 Mbits/s	$ 1,095	N/A

Tape Systems and Subsystems

Formatted Storage Capacity	Height	Data Transfer Rate (k/sec)	PC or PC/XT	AT
60 Mbytes	½	88	$ 995	$ 995
60 Mbytes Subsystem		88	$ 1,295	$ 1,295
21 Mbytes (unformatted) Start/stop Subsystem		24	$ 595	$ 595
26 Mbytes Floppy Tape® Subsystem		31	$ 749	$ 749

Controllers

All of our hard disk and tape controllers are available separately: Please call for prices.

Subsystem Chassis

Any of our disk or tape units are available in an external subsystem for an additional $250.00. You can mix & match any of our ½ high hard disks or tape drives together or add any single full height hard disk.

Tape Cartridges

Express Certified 555 foot 310 Hci ¼-inch Data Cartridge $35.00

Power Supply

130 Watt Power supply $75.00*
150 Watt Power supply $125.00

*with the purchase of any drive

Comes complete

All Express Systems products come complete with the appropriate software, tape and/or hard disk controllers, and cables where required. Hard disks are formatted and tested with the PC DOS of your choice. All drive sizes are formatted capacities.

If your application requires a stacking kit, power splitter cables, daisy chain cable, or some other variation, we'll supply these items at a nominal charge. We even ship our hard disks with Command Assist™ an on-line DOS-like manual to give you help with your DOS commands.

More questions?

Because we spend so much attention on the front end with ensuring that our disks will arrive in working order, we have a customer service department that, unlike many of our competitors, has little to do. When you need us, you won't get a constant busy signal.

Call our friendly, knowledgable customer service staff to get answers to your questions—before or after the sale. Our people, who know the PC, can talk you through the sticky parts, and they'll respond to you quickly. Just call us.

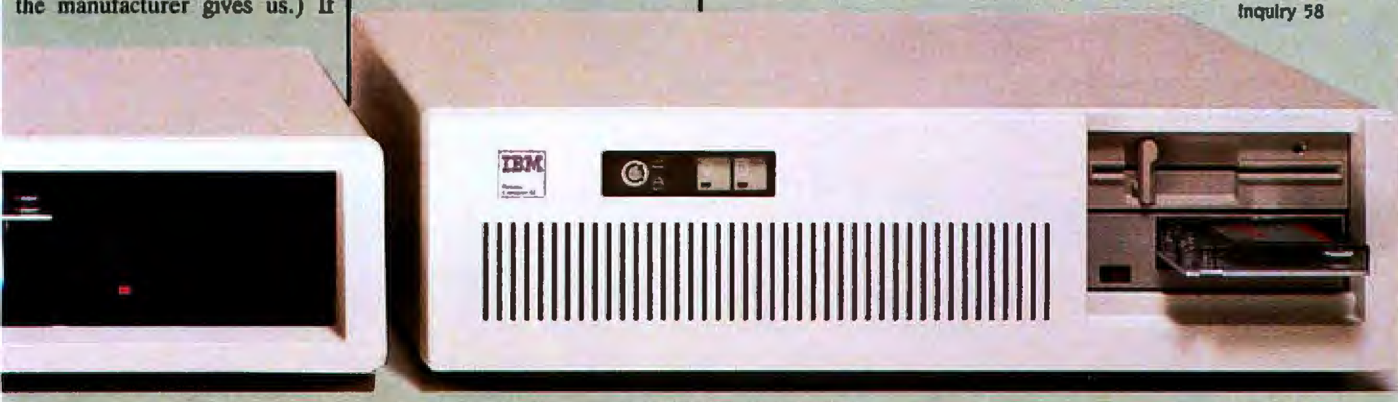
How to order

Pick up the telephone and call 1-800-341-7549, to order. We accept Master Card, VISA, American Express and Diners Club. Or send a cashier's check or money order (We'll take a check, but you'll have to wait for it to clear) and tell us if you want one of our recommended configurations or you want to mix and match yourself. Corporations with a DUNS number may send purchase orders for quantities over five.

EXPRESS SYSTEMS

Call Toll Free 1-800-341-7549 Ext. 1600
In Illinois call (312) 882-7733 Ext. 1600
Express Systems, Inc., 1254 Remington, Schaumburg, IL 60195

Inquiry 58



Command Assist is a trademark of Micro Design International

DEVICE-INDEPENDENT GRAPHICS

Writing software for the IBM *Enhanced Graphics Adapter using the Virtual* Device *Interface*

BY THOMAS B. CLARKSON III

WHEN IBM'S Enhanced Graphics Adapter (EGA) was introduced just short of a year ago, a fair amount of anticipation was generated among IBM PC programmers and end users who relished the prospects of another speedy, feature-bangled accessory for their souped-up PCs and ATs. However, anticipation turned to frustration as programmers settled down to develop applications for the new wonder; the board was extremely complex and quite different from previous IBM graphics-adapter boards such as the Color Graphics Adapter (CGA).

Application developers addressing the EGA have two choices: program the hardware directly or program to the Virtual Device Interface (VDI) provided by IBM in its Personal Computer Graphics Development Toolkit.

Programming the EGA directly has two areas of difficulty—a long development cycle and limited portability of applications—but one perceived advantage: fast execution speed.

Conversely, programming the EGA via the VDI has two big advantages—a very short development cycle and universal portability of applications—but one perceived disadvantage: slower execution speeds, relative to the direct-hardware approach.

This article describes VDI programming techniques and points out the advantages to be gained from programming the EGA with the VDI rather than directly to the hardware with conventional graphics programming techniques.

First, let's take a look at programming the EGA directly.

The Enhanced Graphics Adapter is an advanced graphics controller card that supports resolution as high as 720 by 350 pixels on monochrome monitors and as high as 640 by 350 pixels on color monitors. Text and graphic images are greatly improved over their appearance on the CGA.

While it delivers high image quality, the EGA can cause programming difficulty. Specific challenges presented by the EGA include

- Board complexity
- Mode/RAM/monitor variability
- Indirect access to bit-map memory
- Write-only registers

BOARD COMPLEXITY

The EGA is an extremely complex collection of silicon. It contains 52 VLSI (very-large-scale integration) circuits and a very complex register structure. A large number of instructions must be issued and coordinated for every graphics operation.

For this reason, it's practically impossible to exhaustively document every register combination for the EGA, leaving the programmer no choice but to take a trial-and-error approach. The EGA *Technical Reference Manual* contains descriptions of the registers but tells nothing of how they interact, what combinations produce what effects, etc.

MODE/RAM/MONITOR VARIABILITY

The EGA can be operated in 17 modes and four RAM (random-access read/write memory) configurations on three different IBM monitors. In order to run, a program must be written to a specific mode/RAM/monitor combination and then cannot run on any other mode/RAM/monitor combination without changes.

Faced with this proliferation of boards within a board, in addressing the hardware directly you have two choices: You can invest a tremendous amount of development time and address every possible mode/RAM/monitor configuration, or you can choose a subset of EGA mode/RAM/monitor configurations.

INDIRECT ACCESS TO BIT-MAP MEMORY

Access to the EGA's bit-map memory is indirect, with different access modes provided by different EGA register settings. These various bit-map memory-access modes provide a spectrum of mechanisms for optimally implementing bottom-level graphics primitives. For example, line drawing is best done in DX writing mode (10 in the Write Mode field of the Mode register), but copying rectangles of pixels from off-screen bit maps to the screen bit map is best done in processor data mode (00 in the Write Mode register). Both differ substantially from the direct bit-map access familiar to CGA programmers. DX mode provides a means of writing all planes with particular colors at once, and processor data mode lets you read and write each plane individually. There is no way that a programmer can directly read all bits of a given pixel in the EGA, as you can

(continued)

Thomas B. Clarkson III (GSS, 9590 *Southwest Gemini Dr., Beaverton,* OR 97005) *is chairman and* CEO *of* GSS, *developer of* IBM*'s Personal Computer Graphics Development Toolkit and Professional Graphics Series.*

with the CGA. All references to the bit map are mediated by the EGA hardware.

WRITE-ONLY REGISTERS

Many of the EGA's registers are write-only. This means that a supervisory or background program cannot arbitrarily save and later restore the screen state. For example, when a program is directly accessing the hardware, concurrent applications cannot grab the current state of the screen, switch to a different process, and then restore the screen state when the process is resumed.

Another challenge presented by the EGA is long processor wait states caused by its high refresh rate. The EGA's number-one priority is to keep the screen refreshed. The running program may manipulate the screen buffer whenever it likes, but contention for the buffer is always resolved in favor of screen refresh. The advantage of this priority is the absence of the "hash" familiar to CGA programmers. The disadvantage is that, in some screen modes, the processor may have quite a few wait states before a screen buffer cycle is made available to it.

Also, the EGA hardware only directly supports some Boolean operators: REPLACE, AND, OR, and XOR. To support all possible operators may require an extra write of the bit map.

Most of these programming difficulties are concentrated at the environmental setup level rather than at the graphics functionality level, where draw-line commands occur and pixels are actually being moved. If graphics functionality is complex (such as drawing vectors, copying rectangles, and drawing on-screen bit maps) and the program must run in diverse hardware/software environments (the situation faced by most IBM PC application programmers), the direct-to-the-hardware programming task is very complex. Writing and optimizing the necessary low-level primitives is time-consuming and highly memory-intensive. However, if you are writing a simple graphics program that will never run in an environment with other graphics software (that is, there

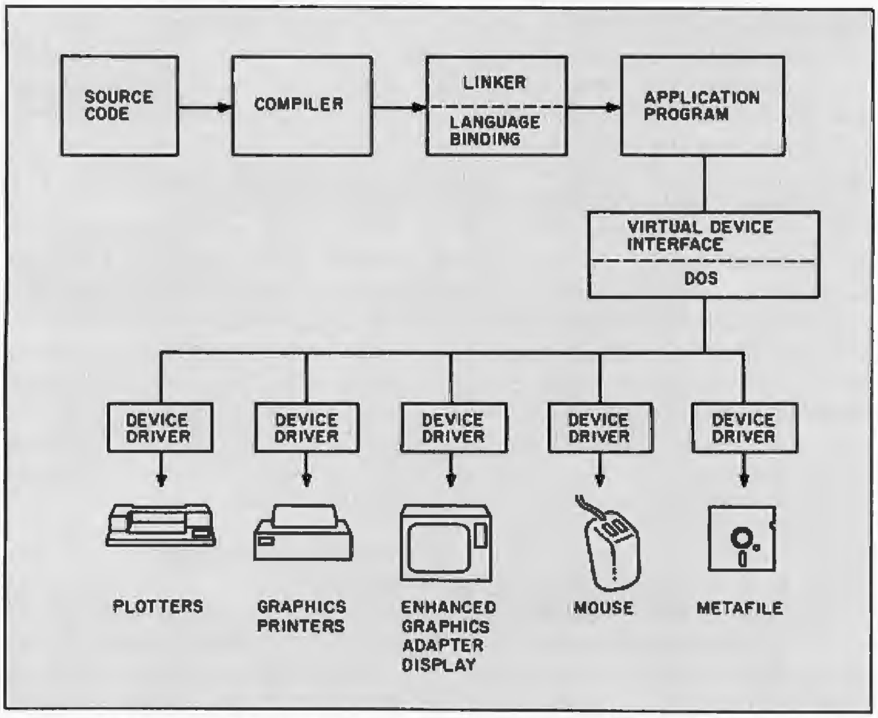


Figure 1: *The Virtual Device Interface stands between an appliction program and the specific devices that the application controls.*

> *A challenge presented by the EGA is long processor wait states caused by its high refresh rate.*

is no environmental code to deal with), then programming the EGA will be a simpler job.

Making programs compatible with different or higher-resolution graphics adapter cards that will be produced is another important issue. For every new card and monitor introduced, the application developer who wrote directly to the EGA hardware will have to rewrite code—sometimes a little, sometimes a lot.

DEVICE INDEPENDENCE: THE VDI

IBM has provided a way to program the EGA in a device-independent manner. The Personal Computer Graphics Development Toolkit allows programming to a Virtual Device Interface, rather than to specific mode/RAM/monitor configurations of the current EGA. IBM's VDI implementation contains optimized device drivers to all IBM PC hardware, from printers and screens to graphics adapter cards, together with language bindings to C, Pascal, FORTRAN, Compiler BASIC, assembly-language, and macro assembler.

The Virtual Device Interface is a layer of software at the operating-system level that serves as a logical graphics interface between the application and the devices being controlled by the application (see figure 1). The role of the VDI is analogous to that of a portable operating system: As a consistent application interface, the BDOS performs all logical system functions while the BIOS handles hardware-dependent operations such as displaying data on the screen.

(continued)

reading from or writing to disk, and accepting keyboard input. The VDI specifies high-level graphics tasks to device drivers that in turn instruct graphics input and output devices to perform those tasks.

The VDI controller receives high-level, device-independent graphics commands (called primitives) from the application and passes them to device-specific drivers. The drivers translate the high-level primitives into lower-level instructions specific to each device. The VDI serves as a standardized interface between multiple and diverse drivers (devices), the application program, and the operating system. The application never talks to specific drivers, only to the VDI. The insulation of application from hardware gives VDI-based software its portability. An application will run on any device (even one developed 10 years from now) as long as the device driver is resident in the system.

In addition to providing device independence and application portability, the VDI offers a much shorter development cycle than device-specific programming. The environmental setup code that takes so much time to write and changes from device to device is already written for the application developer in the VDI, and it never changes. You can immediately begin application design without spending your time on systems-level software development.

The VDI deals in very high-level graphics commands, one of which can replace scores of low-level move-draw commands. Figure 2 shows some of these commands.

The VDI also provides full access to EGA hardware features such as the EGA color-map hardware. This EGA

```
output circle
output arc
output bar
output pie slice

set color
set character height
set fill style
set polyline color map
set polymarker type
```

Figure 2: *Sample* VDI *commands.*

Table 1: *The Virtual Device Interface supports a variety of the Enhanced Graphics Adapter's* 17 *mode/*RAM*/monitor configurations, as shown.*

Display	Mode	RAM	Colors	Palette	Pages	Resolution
mono-chrome	00FH	64	4	mono	1	640 x 350
		128	4	mono	1	640 x 350
		192	4	mono	1	640 x 350
		256	4	mono	2	640 x 350
color or enhanced	00DH	64	16	16	2	320 x 200
		128	16	16	4	320 x 200
		192	16	16	6	320 x 200
		256	16	16	8	320 x 200
	00EH	64	16	16	1	640 x 200
		128	16	16	2	640 x 200
		192	16	16	3	640 x 200
		256	16	16	4	640 x 200
enhanced	010H	64	4	64	1	640 x 350
		128	16	64	1	640 x 350
		192	16	64	1	640 x 350
		256	16	64	2	640 x 350

feature lets you assign any color index to any color displayable on the screen. It can be used for special effects, such as simple animation or rapid color changes. With the VDI, changing the color map is a quick operation; without the VDI, accessing the color map is prohibitively difficult.

Table 1 shows the variety of tradeoffs possible with the four key modes that IBM has chosen to support with the VDI. Note that even the monochrome monitor supports four colors (black, white, bold, and blinking), each of which must be addressed specifically and separately. Page support refers to the EGA's ability to store screen buffers, or background screens, for fast screen switching.

Is the VDI Slow?

Many programmers associate device independence with very low performance. It's true that device-independent programs may run slower than device-specific programs, simply because of the process of "translating" a universal program to specific device requirements. VDI translation is made up of two key processes called transformation and emulation.

Transformation is the process of mapping virtual, or normalized, device coordinates (which are not specific to any particular device) to device-specific coordinates. Transformation enables an application to be moved from device to device without change.

Emulation is the process of instructing devices how to accomplish graphics commands that they are normally incapable of doing. For instance, if the VDI issues a command to "draw a polygon" and the device is not designed to do that, the VDI sees to it that the polygon gets drawn using whatever capabilities the device does have. It might accomplish "draw polygon" by piecing together a number of "draw line" commands. Emulation takes time. A sophisticated application running on low-level hardware would require quite a bit more emulation and thus run much slower than the same application running on

(continued)

BASF
2 HD high density
two sided
sectors soft
FlexyDisk® 5.25
Now that you can store twice as much on one disk, you had better be sure of that one disk.
BASF
BASF QUALIMETRIC™ 1.2 MB FLEXYDISKS.® CERTIFIED 100% ERROR-FREE AND WARRANTED...
FOREVER

Programming the VDI involves opening and closing workstations.

a device with high-level graphics capability.

Even with the need for transformation and emulation, a VDI need not drag down performance. The VDI's device drivers are highly optimized and could give better performance than some hardware-dependent applications. Primarily to blame for poor VDI program performance is inappropriate programming techniques.

PROGRAMMING THE VDI

The VDI programming flow is a repetitive cycle consisting of the following steps: (1) open workstation, (2) set attributes, (3) draw graphics primitives, (4) clear workstation, and (5) close workstation. A workstation is any logical entity from which the application receives input or to which output is directed; it can be a display screen, a keyboard, a mouse, etc.

The VDI graphics primitives are polyline, polymarker, filled areas, arcs, circles, bars, and pie slices. Primitives are assigned attributes such as color, line style, text height, font, and rotation. You don't need to concern yourself with drawing pixel-to-pixel vectors; the VDI does this for you.

The command open workstation generally prepares the device to receive output and to return input. Specific tasks in this step include establishing default attributes for primitives (color, line styles, fill styles, text size, text rotation, etc.), opening any necessary files, clearing the screen, form-feeding paper in the printer, replacing pens on a plotter, and informing the user of device-specific information such as colors, line styles, and fill styles.

Setting attributes involves a variety of commands that determine attributes for graphics primitives (such as color, line style, etc.) and setting the

(continued)

"clipping rectangle," which restricts output to a certain area of the display surface.

Drawing graphics primitives involves commands that display the fundamental units of graphics primitives in the lines, text, filled areas, circles, arcs, pie slices, and rectangles that make up the picture. This is also the step in which pictures are updated interactively based on user actions and application program directives.

The command clear workstation dumps the existing picture to a printer workstation (or other hard-copy device), flushes the printer buffer, and returns a blank screen to the display.

The command close workstation terminates communications with the display device; updates, rasterizes, and dumps the on-screen picture to the printer; closes any open files; and puts the device in a known and stable state, preferably that which existed before the open workstation command was received.

Programming a VDI is quite different, conceptually and semantically, from writing traditional graphics programs. The VDI operates at a much higher level than conventional graphics programs that deal on a pixel-manipulation, move-draw level.

In conventional graphics programming, the programmer must build each function specifically for the particular device(s) on which the application will run. Traditional graphics packages address devices in device-specific coordinate spaces, which differ from device to device (pixels, plotter steps, dots per inch, etc.). The VDI, on the other hand, addresses devices in universal, normalized device coordinates (NDC). The VDI is thus able to guarantee a baseline environment of useful functions, emulating those not present on the device.

VDI functions are much higher-level graphics primitives than those found in traditional graphics programming environments; VDI primitives such as polyline, fill area, and circle replace low-level drawing units like pixel and line. Rather than having to turn on and off pixels or draw thousands of short line segments to form images like circles and arcs, the programmer enters a few (sometimes only one) high-level graphics primitives. For example, the traditional move-draw way to draw a three-segment figure would be:

```
move x1, y1
draw x2, y2
draw x3, y3
draw x4, y4
```

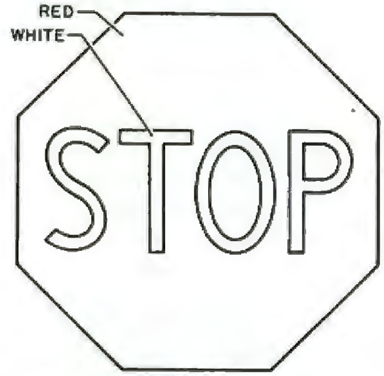
Whereas drawing the same figure using VDI would only require issuing one polyline command:

```
draw polyline (4, xy)
```

For another example of how a programmer might use the VDI primitives and attributes to write graphics code, see figure 3. It shows the VDI code necessary to draw a fairly complex two-color graphic that, using conventional graphics programming methods, could take hundreds of lines of repetitive move-draw commands. Using the VDI, the programmer merely establishes a few high-level parameters: points of the polygon, colors used, fill style, text alignment, and color and text orientation. The VDI interprets these high-level commands to come up with the picture, thinking in terms of graphics primitives (text, color, polygon) rather than pixels or vectors.

VDI also incorporates the notion of line styles (dashed, dotted, etc.), so you don't have to piece together hundreds of short draw line commands to create a dashed line. Similarly, to draw a circle in VDI you issue a circle

(continued)




```
data xy (18) = 14000, 22000, 10000, 18000, 10000
               14000, 14000, 10000, 18000, 10000
               22000, 14000, 22000, 18000, 18000
               22000, 14000, 22000

set color index (display,1,WHITE)        fill area (display,9,xy)
set color index (display,2,RED)          set text alignment (display, center, center)
set fill interior style (display,SOLID)  set text color index (display,1)
set fill color index (display,2)         text (16000, 1600, "STOP")
```

Figure 3: *The* VDI *figures and attributes required to draw a stop sign.*

command with center point and radius. In a non-VDI environment you would have to draw many short vectors and compute their endpoints using sine and cosine tables.

The VDI also assigns colors to each attribute individually, rather than forcing one universal color to be applied to all primitives. This allows a great deal more flexibility.

In a traditional graphics programming environment, you build into the application assumed knowledge about the devices that will run the program. Supporting new devices is thus difficult if not impossible without making extensive program changes.

The VDI, on the other hand, lets you build programs in universal, high-level, device-independent primitives. At run time, the program queries the device to discover its particular capabilities or limitations and modifies its presentation of images to conform to those capabilities or limitations (in the case of limitations, emulation takes place). The VDI program adapts "on the fly," having no preset, hard and fast device-specific data built into it. This enables the VDI program to adapt to new hardware without changing the source code.

The difference between writing traditional graphics programs and writing VDI-based graphics programs is analogous to the difference between writing code in assembly language and writing code in a high-level language. There's an entirely different mindset.

In the VDI, even many complex images (such as a filled star) become high-level primitives that can be invoked with a single command (draw polygon). In a conventional graphics program, reams of move-draw commands are required to draw the horizontal raster lines comprising the image. Also, in an image like a star, where the same raster line forms discontinuous segments of the image, it's tricky (and time-consuming) to specify begin-/endpoints for each raster line.

With VDI, you can think of your pictures as sets of high-level pieces, not as millions of dots (pixels). You predefine data and call a single polyline or polymarker primitive to output all the pieces, instead of issuing many individual calls. By making inquiries of the device to determine its capabilities (colors, text sizes, raster writing modes), the program can tailor output for these device attributes at run time.

VDI primitives and attributes are uncoupled. Instead of having to set attributes for each primitive each time that primitive occurs in the program, attributes are set once and stay set until changed.

Inquiry 57

Inquiry 92

Many programmers make the mistake of trying to use overly complex VDI primitives to do simple things, or, conversely, they may fail to take advantage of VDI commands in doing complex things. For instance, you might use many short polyline statements to draw a dashed line instead of setting the line style to "dashed" and drawing a single polyline. Or you might use four draw line commands to draw a rectangle when one four-point draw polyline would do the same thing in one-fourth the code.

Programmers may have a difficult time with the VDI in the beginning· there are so many different ways to combine the high-level commands to achieve the same picture that inefficiencies can result. For instance, there may be 10 different ways to draw a blue circle inside a green box using VDI commands. While VDI documentation covers the most straightforward uses for each graphics primitive, it cannot possibly list every possible combination. Programmers must draw on their own programming experience and creativity to develop optimum VDI programming techniques.

CGA vs. EGA

Even without the VDI, the performance drop in moving from the CGA to the EGA is dramatic. A high-resolution card will usually have lower performance than even a closely related low-resolution card: there are simply more pixels to manipulate for every visual representation. Thus, programmers moving from a CGA to an EGA environment should be prepared for a significant performance degradation.

Conclusion

The EGA is an extremely capable but complex device. It produces beautiful images for users hungry for high-resolution graphics, but it presents the application developer with special programming challenges that the CGA never did. Complex register structure, multiple modes, multiple RAM configurations, multiple monitors, and indirect access to the bit-map memory make programming the EGA a challenge.

Not only is direct programming of the EGA difficult, but the programs are limited to the specific mode/RAM/monitor combination(s) for which they were written and cannot be moved to other combinations or to future cards and monitors without extensive change. Making a program run on multiple EGA modes, multiple CGA modes, and future graphic cards requires a mammoth programming effort.

The Virtual Device Interface offers a viable means of writing application software for these rapidly changing, increasingly complex devices. ■

Introducing The B

ADVANCED TECHNOLOGY COMPUTER BT/AT

44 Mb Hard Disk Drive.
640K Internal RAM.
8 I/O Expansion Slots.
1.2 Mb Floppy Disk Drive.
2 Asynch Serial Communication Ports.
2 Parallel Printer Ports.
Monochrome Graphics Adapter (720 x 348).
Monochrome Monitor (Green).
Tilt/Swivel Monitor Base.
IBM PC AT Hardware and Software Compatiblity.
GW-BASIC and MS-DOS 3.1.

HIGH RESOLUTION COLOR MONITORS
HR31 200
HR31 350 (Enhanced)

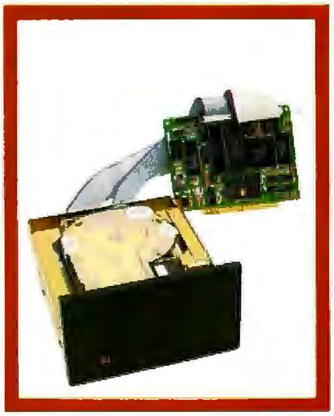
.31mm Dot Pitch.
14" Diagonal Screen.
Black Matrix Picture Tube
Plugs Into IBM Color/Graphics or Compatible Adapter Card.
Resolution (640 x 200).
Enhanced (640 x 350).
Tilt/Swivel Base.

HIGH RESOLUTION MONOCHROME MONITORS
HR 39 (Green)
HR 134 (Amber)

Plugs Into The IBM Monochrome or Compatible Adapter.
720 x 350 Resolution.
12" Diagonal Screen.
Super Crisp Text Capability.
High Resolution TTL.
Tilt/Swivel Base.

PC KEYBOARD
BT5151
AT KEYBOARD
BT5161

Solid State Capacitive Key Switches.
3-Position Height Adjustment.
Keys In Standard Typewriter Positions.
Seperate Cursor Control and Numeric Keypads.
Easy-To-Read Key Legends
LED Indicators For All Lock Keys.



PC HARD DISK SUBSYSTEMS
BT20 (Internal)
BT20E (External)

20 Megabyte Capacity.
Boot From The Hard Disk--No Software Patches or Drivers.
Runs All The Popular Software Designed For The XT.
High Reliability and Durability--Specially Plated Drives.
Includes Idir "Visual Shell" and Xydex Software, Bezels.

BACKUP POWER SOURCE
AT800
XT300
SB200

Battery Powered Up To 30 Minutes.
Surge Suppression/ Noise Filtering.
Brownout Protection.
800, 300, 200 Watt Models.

MONOCHROME CARD
HR720
COLOR CARD
HR640

Complete IBM Compatibility.
High Resolution Graphics Capabilities.

When the World Sells Micros, the Common Language is BYTE.

Have you considered your international prospects already? Good. Then you know that capturing this market will have a tremendous impact on your business. Now is the time to plant the seeds of recognition—the international micro industry is taking off. In fact, in Europe alone, annual micro sales are in the billions of dollars.

So how are you going to do it? Let BYTE offer you a solution: we have over 50,000 paid readers outside the North American continent—plus 127,000 pass-along readers—all of whom comprise the most select audience for your advertising investment. BYTE is the micro medium that communicates your sales message in a language understood all over the world. And BYTE is the only U.S.-based micro publication to earn the reputation as *the* international standard.

BYTE also offers a special International Advertising Section (IAS) which allows you to advertise your products only to BYTE readers outside the North American continent at reduced rates. Call your regional sales representative to find out how BYTE can speak for you. . .all over the world.

For advertising information, call:

Austria, Vienna	Hans Csokor	75 76 84
France, Paris	Jean Christian Acis	720 33 42
Hong Kong	Seavex, Ltd.	260149
Israel, Bat Yam	Gurit Gepner	866 561 321 39
Italy, Milan	Savio Pesavento	86 90 656
Japan, Tokyo	Hiro Morita	581-9811
Singapore	Seavex, Ltd.	734-9790
Spain, Madrid	Maria Sarmiento	45 52 891
Sweden, Nacka	Andrew Karnig	44 0005
U.K., London	Arthur Scheffer	493 1451
W. Germany, Frankfurt	Fritz Krusebecker	72 01 81
U.S.A.	Dennis J. Riley	(603) 924-9281

BYTE. The Official Translator of the Micro World.

BYTE

THE SMALL SYSTEMS JOURNAL

70 Main Street
Peterborough, NH 03458
(603) 924-9281

For subscription information in Europe, call Jean Tobias in England: 628-23431.

IBM PC Disk Performance and the Interleave Factor

How the distribution sequence of sectors on a disk affects read/write performance

BY MARCUS KOLOD

MANY FACTORS AFFECT the performance of fixed and floppy disks, including read/write-head seek time and disk rotational delay, or latency. A lesser known but important and alterable component of the performance curve is the distribution sequence of the sectors on a disk. This sequence is called the interleave pattern, and it defines the sector skewing of the disk.

This article examines some of the factors relating to disk performance and outlines a benchmarking method for testing different interleaving values. We'll discover how interleaving affects the performance of the IBM PC XT and AT.

For a discussion of some of the important factors that determine disk performance and capacity, see the accompanying text boxes "Calculating Disk Capacity" and "Disk-Encoding Schemes."

Yet Another Hard-Disk Benchmark

In the case of a moving-head disk system, two time components are involved in the delay between receiving an address at the disk controller and the beginning of the actual data transfer. The first, called seek time, is the time required to move the read/write head to the proper track. This time obviously depends on the initial position of the head relative to the track specified in the address. Average values in the 30-millisecond (ms) range are typical. The second component is the rotational delay, also called latency time. This is the amount of time that elapses after the head is positioned over the correct track until the starting position of the addressed sector comes under the read/write head. On the average, this is the time for half a rotation of the disk. Assuming a disk spinning at 3600 revolutions per minute (rpm), this is 0.5 (60 sec/3600) = 8.3 ms. The sum of these delays is usually called the disk-access time.

The mechanical components of a disk drive are not alterable; the physical sequencing of sectors on a disk are. That is the crux of the following hard-disk benchmarks.

To define sector skewing, also called interleaving, picture a disk platter spinning in a disk drive. The sectors in the track over which the head is positioned are passing by the head one after another—sector 1, sector 2, and so on—until the disk has made one complete revolution. Then the sequence repeats itself. A standard IBM fixed disk spins at 3600 rpm. One revolution of the disk (using the 3600 from before, with 17 sectors/track) takes 60/3600 second, about 0.0166 second per track, or 0.9 ms a sector.

Now imagine PC-DOS loading a program or accessing a file from such a disk. PC-DOS takes a finite amount of time to read and process each sector. Assume that it has to make repeated reads to load the file or program. By the time PC-DOS has read and loaded sector n, it will be too late to read sector $n + 1$. This sector will have already passed by the head and will not come around to position for another 16.6 ms. Proceeding in this fashion, almost two-tenths of a second are needed to read one complete track (this small amount may seem ludicrous, but these amounts do add up over time).

This problem can be overcome by simply numbering the sectors so that there are several physical sectors between each numerically contiguous sector.

Figures 1 and 2 show disks formatted with interleaves of 1 and 3, respectively. The interleave value is specified when the disk is formatted. PC-DOS does not permit the alteration of the interleave value after the disk is formatted. In older versions of CP/M, a translation table stored in RAM (random-access read/write memory) mapped the logical sectors to the physical sectors on a disk, permitting a crude form of changing the pattern of sector reading at any time.

Depending on the size and type of files, the wrong interleave can strongly affect performance. It is not a gradual effect, either; if you "miss" the interleave, the perceived performance will be very slow. One of the nice features about PC-DOS is its use of physical-sector numbers in addressing media. This feature leaves the interleaving separate and independent of PC-DOS.

The primary storage medium I use is the fixed disk. I use floppies primarily for archival purposes and for program distribution. I had to know if there was a noticeable difference in disk performance depending on the interleave chosen. To that end, I conducted two logically distinct tests.

The first test functions using the following algorithm:

1. Format the entire drive at a given

(continued)

Marcus Kolod is a computer consultant with CNA *of Chicago and the president of Kolod Research, a software-development firm. He can be reached at* 2027 *Woodlawn Rd., Northbrook,* IL 60062.

CALCULATING DISK CAPACITY

The effective data capacity for a floppy disk is calculated in the following manner: (bytes/sector × sectors/track × tracks/side × number of recording surfaces). For a disk with 512 bytes/sector, 8 sectors/track, 40 tracks/side, and 2 sides, the capacity is 512 × 8 × 40 × 2 = 327,680. For the quad-density drives of the PC AT, the capacity is 512 × 15 × 80 × 2 = 1,228,800.

The maximum effective data-transfer rate for a 512 bytes/sector, 8 sectors/track disk is calculated as follows: Assuming a rotational speed of 300 rpm, a complete track can be read or written in 60/300 = 0.2 second. This corresponds to a maximum effective transfer rate of (8 × 512)/0.2 = 20,480 bytes/second. For a 9 sectors/track disk, the byte-transfer rate is (9 × 512)/0.2 = 23,040 bytes/second. The rotational speed for a 15 sectors/track disk (the highest-supported IBM density in the PC AT) is 360 rpm. Consequently, the rate is (15 × 512)/0.166667 = 46,079.9 bytes/second.

The advantages derived from the use of Winchester-technology fixed disks include greater density and a larger capacity for a given physical size. This ultimately results in lower-cost units. Another advantage of Winchester technology is that data integrity tends to be greater in sealed units where the storage medium is not exposed to contaminating elements.

Whereas the different floppy-disk formats supported by IBM are derived primarily from software-formatting parameters, with extensions to 2 sides or 15 sectors/track with 80 tracks as opposed to 40, the IBM PC XT and PC AT support a variety of hard-disk types.

The principles of organization and addressing are the same for both floppy and hard disks. Addressing of data on the fixed disk is accomplished by specifying the surface number, the track number, and the sector number.

The effective data capacity for a given fixed-disk drive is calculated in the same manner as for a floppy-disk drive. For a fixed disk with 512 bytes/sector, 17 sectors/track, 306 tracks/side, and 4 sides, the capacity is 512 × 17 × 306 × 4 = 10,653,696 bytes. For the standard IBM-supplied fixed-disk drive in the PC AT, the capacity is 512 × 17 × 615 × 4 = 21,411,840 bytes.

The maximum effective data-transfer rate for a 512 bytes/sector, 17 sectors/track disk is calculated as follows: Assuming a rotational speed of 3600 rpm, a complete track can be read or written in 60/3600 = 0.016 second. This corresponds to a byte-transfer rate of (17 × 512)/0.0166666 = 522,239 bytes/second, close to the maximum data transfer possible on the PC bus using standard RAMs.

Tables A and B summarize the different disk types and sizes for all disks and the standard fixed-disk drives available from IBM. What will follow here is an overview of that information.

In PC-DOS versions 1.x, we have to give addresses to a total of 640 sectors. The first 320 sectors are on the first side of the disk and have the same addresses for both single- and double-sided disks. The second side of a double-sided disk has addresses 320 to 639 (140 to 27F hexadecimal).

In versions 2.x, the logical-sector numbering scheme has changed. All the sectors on a track (or cylinder) are assigned numbers before moving on to the next track. To get a clear picture of this, I will compare the logical-sector numbering between versions 1.x and versions 2.x on a double-sided disk formatted for 8 sectors per track. This change wrought in 2.x minimizes read/write-head movement and consequently increases effective data throughput since data access from the other side of the disk is limited only

ILLUSTRATED BY CLAUDIA TANTILLO

by the electronic switching speeds of the read/write heads as opposed to mechanical movement time associated with the read/write armature.

Track 0, head 0 has logical-sector numbering 0 through 7. In versions 1.*x*, the sectors accessed by the second head on that track (track 0, head 1) are numbered 320 through 327. In versions 2.*x*, they are numbered 8 through 15. The consequences of this difference in logical-sector numbering are usually hidden from you, but sometimes you need to know that the difference exists.

In versions 3.*x*, the logical-sector numbering scheme is consistent with that implemented in versions 2.*x*. The difference here is that the increase in the number of sectors per track (15 possible with a quad-density drive) causes a corresponding increase in the logical-sector numbering, 0 through 14 for track 0, head 0, and 15 through 29 for track 0, head 1.

Logical-sector numbering on a hard disk follows the pattern established with versions 2.*x*. All the tracks on one cylinder are numbered before moving on to the next cylinder. The sectors read by head 0 are numbered 0 through 16, those read by head 1 are numbered 17 through 33, those read by head 2 are numbered 34 through 50, and those read by head 3 are numbered 51 through 67. The second cylinder begins at logical sector 68.

Tables C and D summarize the overheads and usable capacities of different media as supported by the different releases of PC-DOS. In addition, they show the increase in the size of PC-DOS over the releases.

The format of the PC-DOS overhead with any disk is the same. There is a boot sector allocated, one file-allocation table (FAT), and the root directory. Two copies of the FAT are maintained for integrity. Therefore, the amount of space allocated to the FAT must be doubled when calculating the amount of available space. The formula is total number of sectors (from media-size calculations before) − (one boot sector) − (2 × the number of FAT sectors) − directory-space sectors = total number of usable sectors.

The amount of usable space can be misleading, though. In PC-DOS, there is not necessarily a one-to-one correspondence between physical sector and addressable space. The smallest addressable unit of space in PC-DOS is the cluster. This cluster corresponds to an entry in a vector, the FAT, which is a map of the available space on the medium.

The implications can be seen from table C. Though the total amount of space on a double-sided 9 sectors/track disk is greater than that of a single-sided 8 sectors/track disk, the amount of addressable space in clusters is not that much greater, 354 versus 313. With a cluster size of 1 sector, the smallest amount of space a file can take up is 512 bytes. With a

(continued)

Table A: *Different disk types for the IBM PC family of machines and information on number of directory entries available, number of tracks and sectors/track, and operating-system support.*

FAT byte	Number of sides	Number of tracks	Sectors/ track	Total sectors available	FAT size	Directory sectors	Directory entries	Sectors/ cluster	Operating system supported under
FE	1	40	8	320	1	4	64	1	1.0, 1.1, 2.0, 2.1, 3.0, 3.1
FF	2	40	8	640	1	7	112	2	1.1, 2.0, 2.1, 3.0, 3.1
FC	1	40	9	360	2	4	64	1	2.0, 2.1, 3.0, 3.1
FD	2	40	9	720	2	7	112	2	2.0, 2.1, 3.0, 3.1
F9	2	80	15	2400	7	14	224	1	3.0, 3.1

Table B: *Hard-disk types provided by IBM for the PC XT and AT and associated data.*

FAT byte	Number of sides	Number of tracks	Sectors/ track	Total sectors available	FAT size	Directory sectors	Directory entries	Sectors/ cluster	Operating system supported under
(XT hard disk, PC-DOS 2.0, 2.1)									
F8	4	306	17	20,723	8	32	512	8	2.0, 2.1, 3.0, 3.1
(XT hard disk, PC-DOS 3.0, 3.1)									
F8	4	306	17	20,723	8	32	512	8	2.0, 2.1, 3.0, 3.1
(AT hard disk, PC-DOS 3.0, 3.1)									
F8	4	615	17	41,735	16	32	512	4	3.0, 3.1

cluster size of 2 sectors, the smallest amount of space a file can take up is 1024 bytes.

This limitation in part stems from the original specification for an FAT entry, which called for a 12-bit number. There can be at most 4093 unique sector numbers. A maximum FAT would occupy 6144 bytes in storage and 12,288 bytes on disk (two copies). Such an FAT could track the allocation of only 2,095,616 bytes of data if a sector is 512 bytes. Obviously, a 10-megabyte disk surpasses this limitation. The cluster size under PC-DOS 2.0 for a hard disk became 8 sectors. Here again, though the smallest physical addressable unit is a 512-byte sector, the smallest possible file size is 4096 (8 × 512) bytes.

With the introduction of PC-DOS 3.0—and continued through PC-DOS 3.1—the FAT entry size has been modified to support either 12 or 16 bits. The maximum addressable amount of clusters can now be 65,535. In addition, on the PC AT hard disk, when it is formatted using PC-DOS 3.*x*, the cluster size is 4 sectors (2048 bytes). This provides a dramatic increase in the amount of usable space on the PC AT.

Interestingly enough, if a PC AT's hard disk has been formatted under PC-DOS 3.*x* and PC-DOS 2.*x* is booted on the system, the hard disk will be an invalid drive to PC-DOS. This is because the partitioning specification has also changed under PC-DOS 3.*x*. The FDISK utility distributed as part of PC-DOS 3.*x* can now put an indicator into the partitioning header that indicates whether or not 16-bit clusters are used.

If, however, PC-DOS 3.*x* is used to format a standard PC XT hard disk, the format will be compatible with PC-DOS 2.*x*.

Table C: *Calculated overhead for various releases of* PC-DOS.

Number of sides	Sectors/ track	PC-DOS version	Data sectors available w/o system	Data sectors available w/system	Clusters available w/o system	Clusters available w/system	Bytes available w/o system	Bytes available w/system
1	8	1.0	313	289	313	289	160,256	147,968
1	8	1.1	313	286	313	286	160,256	146,432
2	8	1.1	630	603	315	301	322,560	308,224
1	8	2.0	313	235	313	235	160,256	120,320
2	8	2.0	630	552	315	275	322,560	281,600
1	9	2.0	351	273	351	273	179,712	139,776
2	9	2.0	708	630	354	314	362,496	321,536
1	8	2.1	313	234	313	234	160,256	119,808
2	8	2.1	630	551	315	275	322,560	281,600
1	9	2.1	351	272	351	272	179,712	139,776
2	9	2.1	308	629	354	314	362,496	321,356
1	8	3.0	313	196	313	196	160,256	100,352
2	8	3.0	630	513	315	256	322,560	262,144
1	9	3.0	351	234	351	234	179,712	119,808
2	9	3.0	708	591	354	295	362,496	302,080
2	15	3.0	2371	2254	2371	2254	1,213,952	1,154,048
1	8	3.1	313	193	313	193	160,256	98,816
2	8	3.1	630	510	315	254	322,560	260,096
1	9	3.1	351	231	351	231	179,712	118,272
2	9	3.1	708	588	354	293	362,496	300,032
2	15	3.1	2371	2251	2371	2251	1,213,952	1,152,512

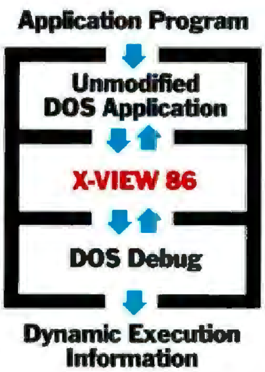
Table D: *Hard-disk overhead for various versions of* PC-DOS.

Fixed-disk type	PC-DOS version	Data sectors available w/o system	Data sectors available w/system	Clusters available w/o system	Clusters available w/system	Bytes available w/o system	Bytes available w/system
XT	2.0	20,674	20,596	2584	2572	10,584,064	10,534,912
XT	2.1	20,674	20,595	2584	2572	10,584,064	10,534,912
XT	3.0	20,674	20,557	2584	2568	10,518,528	10,534,912
XT	3.1	20,674	20,554	2584	2568	10,518,528	10,534,912
AT	3.0	41,670	41,553	10,417	10,387	21,334,016	21,272,576
AT	3.1	41,670	41,550	10,417	10,386	21,334,016	21,270,528

X-RAY VISION

Get it with X-VIEW 86™

X-VIEW 86™ analyzes, profiles, and debugs DOS application software — for improved performance on the IBM PC and its compatibles.



X-VIEW 86 lets you observe the internal operations of DOS application software.

Helps you analyze, debug, test, port, or convert DOS application software — and get faster, more reliable results.

Real solutions to technical challenges.

Looking for code hot spots?
X-VIEW 86 helps you find them fast.

Irritated by bugs that DEBUG can't reach?
X-VIEW 86 captures them.

Sweating over hardware compatibility issues?
X-VIEW 86 pinpoints the trouble spots.

Struggling with a conversion that just won't work?
X-VIEW 86 breaks the deadlock.

What X-VIEW 86 does.

X-VIEW 86 saves you hours of time-consuming, tedious work. It automatically collects data on application programs:

- ☐ memory map references
- ☐ I/O space references
- ☐ INT call usage
- ☐ instruction set usage
- ☐ segment usage data

X-VIEW 86 can interrupt the application program

- ☐ on any processor I/O access
- ☐ on any processor INT instruction
- ☐ on a specified execution path address
- ☐ on a specified memory location reference

X-VIEW 86 can also start the application program execution at a specified address. And it displays the results of the analysis on screen.

Hardware and software requirements.

X-VIEW 86 runs on any member of the IBM PC family — or any operationally-compatible machine that has at least 64 Kbytes of memory. And it's not copy-protected. You use it with PC DOS DEBUG 2.0 or 2.1.

Exciting — not expensive.

X-VIEW 86 is priced at an affordable $59.95.

To order, call 1-800-221-VIEW.

In Texas, call 1-800-233-VIEW.
Then get ready for a whole new outlook on your work.

Inquiry 70

To order X-VIEW 86™ by credit card, call toll free 1-800-221-VIEW. In Texas call 1-800-233-VIEW or send the coupon today.

Customer Service
McGraw-Hill Inc.
8111 LBJ Freeway
Dallas, Texas 75251

X-VIEW 86™ is $59.95. All orders are subject to acceptance by McGraw Hill, Inc. Prices are subject to change without notice.

☐ Check enclosed (Make check payable to McGraw-Hill Inc. Orders paid by check are subject to delay.)
☐ A.E. ☐ M.C. ☐ VISA ☐ D.C.

Name ____________________
Title ____________________
Company ____________________
Address ____________________
City ____________ State ______ Zip ______
Phone ( ) ____________________
Qty ______ $59.95 Amt ______
Shipping ($2.00 per copy) Texas sales tax ($3.67 per copy) ______
Total ______
Signature ____________________
Name on card ____________________
Credit card no. ____________ Expiration date ______

DISK-ENCODING SCHEMES

Encoding schemes are designed to maximize the usable channel capacity of a device. In the 1940s, Claude Shannon of Bell Labs rigorously defined the quantization of information, channel capacity, and their relationship to transmission. His ideas were originally designed in conjunction with the telephone operating system.

If you look at a disk drive as a communication channel, the notions created and defined by Shannon show great applicability. Associated with each channel is a maximum capacity. In a perfect channel, data can be stored at this maximum. In the case of a disk, the ability to generate and detect signals is limited by numerous factors, including the magnetic flux (changes per inch) that the medium can support, the size of the magnetic field created by the read/write head, the error in drive rotational speed, and others. Except for mechanical aberrations, the major physical limitation in magnetic recording has to do with the packing density, that is, the number of bits per inch that can be recorded linearly and still be distinguishable. The fewer overall changes in state, the closer the bits can be packed together.

Many recording techniques are used. Figure A illustrates some of these. In the non-return-to-zero (NRZ) format, when a 1 follows a 1 or a 0 follows a 0, no change of state can be detected on output. Otherwise, there is a change. A variation of this is non-return-to-zero inverted (NRZI), where a change of state occurs only when there is a 1. A third format, phase encoding (PE), changes state in the middle of every bit. If it is a 1, it generates a negative pulse; if it is a 0, a positive pulse. This, of course, means that if two 0s or two 1s follow each other, there is a change of state between them.

NRZI, with its fewer changes in state, allows a better packing density than NRZ. However, since it is impossible to tell where some bits start and stop, a clock track must also be recorded with the data. This is not true of PE, which is self-synchronizing and therefore allows higher densities. However, the many reversals PE requires limits the density that it can achieve.

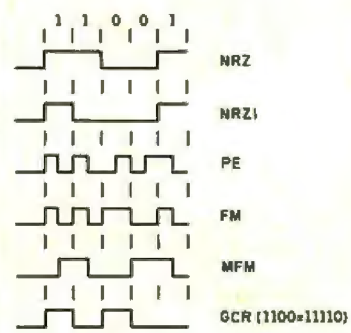


Figure A: *Different schemes for encoding data on magnetic disks.*

A recording format called group-code recording (GCR), based on NRZI, achieves much higher densities by encoding the data bits. In NRZI, a string of 0s will have no change of state, causing synchronization problems. GCR overcomes this by encoding 4 data bits into 5-bit patterns, none of which have more than two 0s adjacent nor have more than one 0 at either end. In the 32 combinations possible with 5 bits, 17 meet this requirement. Sixteen of them are used to encode data, and the seventeenth (11111) is used as a synchronization pattern.

Another type of recording format is frequency modulation (FM), or Manchester code. A change of state occurs in the middle of a 1 but not in a 0. There is also a change of state at the beginning of every bit. This last change makes this format self-synchronizing—there is always a pulse, either positive or negative, when the next piece of information passes under the head. A variation of this is modified FM, or MFM, which also has the change in the middle of a 1, but has a change of state at the beginning of a bit only when two 0s are adjacent. This, in effect, can double the density because the bit size can be cut in half.

IBM uses MFM to define the storing of data in both its floppy-disk drives and fixed disks.

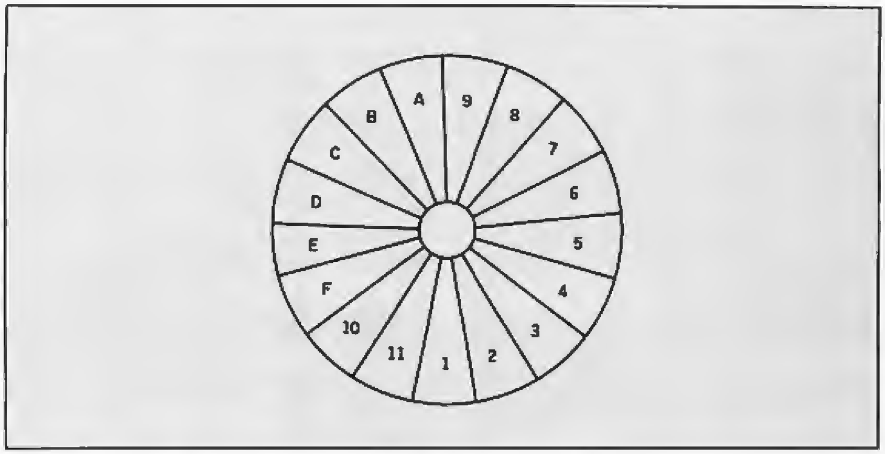


Figure 1: *The interleave pattern for disks formatted with an interleave value of 1. (Disks formatted with 17 tracks, numbered in hexadecimal.)*

Table 1: *Times, in milliseconds, for blocked reads on the* IBM PC XT.

	Number of sectors per block accessed							
Interleave	1	2	3	4	5	6	7	8
1	17.93	35.86	53.79	71.72	89.65	107.58	125.51	143.44
2	18.91	37.82	56.73	75.64	94.55	113.46	132.37	151.28
3	19.89	39.78	59.67	79.56	99.45	119.34	139.23	159.12
4	20.88	41.76	62.64	83.52	104.40	125.28	146.16	167.04
5	21.86	27.02	32.69	37.36	43.52	48.69	54.35	59.02
6	22.84	28.99	35.63	41.29	48.42	54.56	61.23	66.88
7	7.38	14.75	21.64	29.50	35.90	43.28	50.19	58.05
8	8.31	16.63	24.59	33.24	40.80	49.18	57.05	65.91
9	9.34	18.68	27.53	37.36	45.73	55.06	64.13	73.77
10	10.08	20.16	30.24	40.32	50.40	60.48	70.56	80.64
11	11.06	22.12	33.18	44.24	55.30	66.36	77.24	88.48
12	12.04	24.08	36.12	48.16	60.20	72.24	84.28	96.32
13	13.02	26.04	39.06	52.08	65.10	78.12	91.14	104.16
14	14.00	28.01	42.01	56.01	70.01	84.02	98.02	112.02
15	14.98	29.96	44.94	59.92	74.90	89.68	104.86	119.84

Table 2: *Times, in milliseconds, for nonblocked reads on the* PC XT.

	Number of sectors accessed							
Interleave	1	2	3	4	5	8	7	8
1	17.80	34.56	51.71	68.85	86.00	103.15	120.29	137.44
2	18.88	35.54	52.68	69.83	86.98	105.10	122.49	155.07
3	19.86	36.52	53.66	70.81	88.69	121.76	138.91	156.05
4	20.84	37.50	54.65	72.31	105.59	122.74	139.88	157.30
5	21.82	22.06	22.80	24.02	40.93	41.66	43.62	59.82
6	22.80	23.04	23.78	41.17	41.90	42.64	60.03	60.80
7	7.36	24.27	24.27	41.41	41.90	59.05	59.54	61.50
8	8.34	24.76	25.49	42.39	43.13	60.03	60.76	77.66
9	9.32	25.74	26.47	43.37	44.60	61.01	78.15	78.64
10	10.06	26.71	27.45	44.35	61.50	61.99	79.14	81.09
11	11.04	27.69	44.87	45.33	62.48	63.95	80.12	97.26
12	12.02	28.67	45.85	46.31	63.46	80.61	82.36	98.26
13	13.00	29.66	46.81	47.79	64.45	81.59	98.77	100.70
14	13.98	30.64	47.79	64.93	66.16	82.57	99.72	116.86
15	14.96	31.62	48.76	65.91	83.05	84.52	100.69	117.84

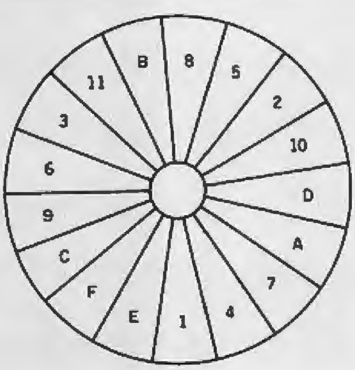


Figure 2: *The interleave pattern for disks formatted with an interleave value of 3. (Disks formatted with 17 tracks, numbered in hexadecimal.)*

Table 3: *Times, in milliseconds, for the interleave benchmark, blocked reads, for the* IBM PC AT.

	Number of sectors per block accessed							
Interleave	1	2	3	4	5	6	7	8
1	17.76	35.52	51.81	68.08	85.85	102.62	119.88	136.18
2	9.13	18.25	27.38	36.51	45.64	54.76	63.89	73.02
3	6.17	12.34	18.51	24.68	30.85	37.02	43.19	49.36
4	12.83	25.66	38.49	51.32	64.15	76.98	89.81	102.64
5	7.15	14.30	21.45	28.60	35.75	42.90	50.05	57.20
6	3.21	6.42	9.63	12.84	16.05	19.26	22.74	25.68
7	5.18	10.36	15.54	20.72	25.90	31.08	36.26	41.44
8	14.80	29.60	44.40	59.20	74.00	88.80	103.60	118.40
9	17.02	19.25	21.46	23.69	25.91	28.13	30.35	32.59
10	11.84	23.68	35.52	47.63	59.20	71.04	82.88	94.72
11	13.81	27.63	41.44	55.26	69.07	82.88	96.70	110.51
12	10.11	20.23	30.34	40.45	50.56	60.68	70.79	80.90
13	4.20	8.39	12.59	16.78	20.98	25.17	29.37	33.56
14	10.94	21.87	32.81	43.74	54.68	65.61	76.55	87.48
15	8.14	16.28	24.42	32.56	40.70	48.84	56.98	65.12

Table 4: *Times, in milliseconds, for the interleave benchmark, nonblocked reads, for the* PC AT.

	Number of sectors accessed							
Interleave	1	2	3	4	5	6	7	8
1	17.80	33.79	49.82	65.85	81.88	97.91	113.94	130.48
2	9.13	25.90	26.39	43.17	44.15	60.44	61.91	77.69
3	6.17	22.99	23.26	23.93	40.45	41.58	42.18	57.97
4	12.83	29.60	29.85	46.86	63.14	79.91	81.15	98.16
5	7.16	23.93	24.47	24.91	41.44	42.68	58.70	60.43
6	3.21	19.98	20.23	20.63	21.47	21.96	36.02	38.24
7	5.18	21.95	22.70	22.99	37.50	39.72	41.20	41.69
8	14.80	31.57	46.37	48.84	65.86	82.88	99.51	115.43
9	17.02	18.99	19.24	19.49	19.98	20.98	21.47	21.97
10	11.84	28.61	28.86	45.78	62.16	63.15	80.17	95.69
11	13.82	30.59	30.83	47.85	64.87	80.89	97.67	114.45
12	10.05	26.64	26.89	43.42	44.15	60.41	76.96	78.44
13	4.20	20.97	21.22	21.95	22.45	38.73	38.98	40.41
14	10.85	27.62	27.87	44.40	61.17	62.15	77.94	92.98
15	8.14	24.91	25.41	42.22	42.98	57.72	59.77	61.42

interleave.
2. Set the clock to zero.
3. Read a number of sectors (1 through 8) for the entire length of the drive—track by track, cylinder by cylinder—until all sectors on the drive have been read. For example, if reading two sectors, read sectors 1 and 2, then 2 and 3, then 3 and 4, etc.—access across tracks and cylinders is done automatically. By doing this, the staggered fashion of the reads will closely approximate a random read.
4. Get the ending time.

This process continues for interleave values of 1 through 15.

The second test performed the same algorithm with this change: If reading two sectors, read sectors 1 and 2, then 3 and 4, then 5 and 6, etc. This process also continues for interleave values of 1 through 15.

Tables 1, 2, 3, and 4 summarize the results of these tests as performed on the PC XT and AT. Looking at the tables, most of the read times are linear with respect to the number of sectors accessed for a given interleave and the required time for that read. With some of the times, noticeably interleave 9 on the PC AT, the average access time decreases dramatically as the number of sectors requested increases.

Did IBM choose the best interleave? There is a discrepancy between the stated interleave (6) in the DOS 2.0 manual and the interleave that both showed the best times in my test and matched the interleave of a brand-new XT from IBM, namely, 7. Considering that, I think the answer is yes for both machines. IBM has optimized its code around single-sector reads. Although DOS may intermittently perform multiple-sector reads, it primarily accesses single sectors.

If multiple-sector reads are performed, it does appear that better interleaves exist. ■

I wish to thank Les Hodges and Tom Fortman for their assistance in preparing the materials comprising this article.

DATA ACQUISITION TO GO
INTERFACE FOR ANY COMPUTER

Connects via RS-232. Built-in BASIC. Stand alone capability. Expandable. Battery Option. Basic system: 16 ch. 12 bit A/D, 2 ch. D/A, 32 bit Digital I/O. Expansion boards available. Direct Bus units for many computers.

SPECIALISTS IN PORTABLE APPLICATIONS

(201) 299-1615

P.O. Box 246, Morris Plains, NJ 07950

ELEXOR

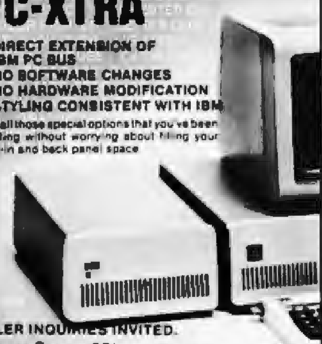
Inquiry 55

DOUBLE THE OPTION CAPACITY OF YOUR IBM PERSONAL COMPUTER

PC-XTRA

- DIRECT EXTENSION OF IBM PC BUS
- NO SOFTWARE CHANGES
- NO HARDWARE MODIFICATION
- STYLING CONSISTENT WITH IBM

Add all those special options that you've been wanting without worrying about filling your plug-in and back panel space



DEALER INQUIRIES INVITED.

$549.00* F.O.B. SANTA ANA

CALIFORNIA RESIDENTS ADD 6% SALES TAX

P C HORIZONS, INC.

1701 E. Edinger, Ste. A6, Santa Ana, CA 92680

(714) 953-5396

Inquiry 145

VT102/TEK 4010

TERMINAL EMULATION

For the IBM PC, XT, AT, PCjr, and Compatibles

PC102 precisely emulates DEC VT102, 101, 100, and VT52 terminals.

PC4010 includes all PC102 features plus Tektronix 4010 graphics.

A few reasons why thousands of customers—including GE, Dow, Raytheon, Westinghouse, and Stanford University—prefer our products

- Complete keyboard and screen emulation w/line graphics (optional 132-columns)
- ANSI color, local printer, bidirectional file transfer support
- Guaranteed compatibility with all DEC applications including EDT, WORD-11, ALL-IN-ONE, DEC-CALC, UNIX vi
- New DOS shell key, ten programmable softkeys, plus full DOS 2.X-3.X path names
- Written in "C" and ASM up to 38.4 KB

PC102-JR	$69.00
PC102 (for PC, XT, AT Only)	$139.00
PC4010	$179.00

Prepaid, C.O.D., Mastercard, VISA

30-DAY money back guarantee! CALL TODAY!

gms GENERAL MICRO SYSTEMS

6440 Flying Cloud Dr.
Suite 205
Minneapolis, MN 55344 **(612) 944-0593**

Inquiry 71

NEW LOCKIT II

- Password Protected Subdirectories
- Hides files-makes files Read-Only
- Runs on any PC or compatible with Hard Disk under DOS 2.0 or Higher—
- Easy to use - only $79.95

LOCKIT I

- Password Enabled Boot-up
- For IBM PC or PC/XT
- Optional Hard-Disk-Only Boot.
- Invulnerable! $129.95

PC RESET/QUICKON

- Reset without turning off power
- Eliminate turn-on & Reboot wait
- Saves time, frustration & damage to PC & PC/XT $89.95

Specify PC or XT, MC/VISA

SECURITY MICROSYSTEMS CONSULTANTS
16 Flagg Place
Suite 102B
S.I., NY 10304
(718) 667-1019

Inquiry 178

PC EXPANSIONS

AST SixPakPlus (64k)	$259
SixPakPlus (384k)	$299
MegaPlus (64k)	$269
Advantage (128k)	$419
Advantage (1.5M)	$639
I/O Plus	$129
PCnet - starter kit	$809
Quadboard (64k)	$199
Quadboard (384k)	$249
Quadmeg-AT (128k)	$319
Quadmeg-AT (2M)	$609
Quadport-AT	$105
HERCULES graphics board	$319
Color Card with PP	$159
HAYES Modems 2400	$call
Smartmodem 1200	$409
Smartmodem 1200B	$379
Set of 9 chips (64k)	$10
256k chips (each)	$4
8087 chip	$119
Meynard Disk Controller	$100
Sandstar Series	$call
Internal Hard Disks from	$609
MaynStream tape backup from	$979
Qume 142A	$109
Teac FD55B	$109
Tandon TM100-2	$109
CDC 9409	$119
Verbatim Datalife disks (10)	$25

VLM Computer Electronics

10 Park Place • Morristown, NJ 07960

(201) 267-3268 Visa, MC, Check or COD

ROSE ELECTRONICS DATA SWITCHES

SHARE computers, printers, any parallel or serial device
ELIMINATE cable swapping
INEXPENSIVE way to network
COMPATIBLE with all computers.

Businesses, Schools, Homes

WE ALSO OFFER
Data Buffers, Line Drivers, Modems, Protocol Converters, Parallel ↔ Serial Converters, Cables, Computers, Printers, Disk Drives, and more

AUTOMATIC - **CARETAKER** is ideal for a business or school to share a printer or modem among many computers. Operation is fully automatic with no software required.
Parallel or Serial 4 channels - $295 8 channels - $395

MANUAL - **HARDSWITCH** is operated with the flip of a switch. 2:2 and 2:4 models allow simultaneous communication.

Serial	1:2 - $99	1:4 - $159	2:2 - $189	2:4 - $279
Parallel	1:2 - $79	1:4 - $139	2:2 - $119	2:4 - $199

LED and spike protection on serial models add $20.

CODE ACTIVATED - **PORTER** connects one computer to multiple peripherals. A software code selects the peripheral.
Parallel or Serial 4 channels - $295 8 channels - $395
Buffer option 64K - $100 256K - $250

REMOTE - **TELEPATH** connects multiple computers to multiple peripherals. A selector at each computer or terminal chooses up to 4 peripherals and displays busy status.
4:4 - $495 4:8 - $795 selector - $39

Give a Rose to your computer

ROSE ELECTRONICS
P.O. BOX 742571
HOUSTON, TX 77274

(713) 933-7673
MC & VISA Accepted
Dealer Inquiries Invited

CALL US FOR ALL YOUR INTERFACE NEEDS

Inquiry 234

ATTENTION MEGABYTERS

Let your PC grow!

Vfeature
for IBM AT, XT and compatibles
supports BIG disks
secures disk data.

Vfeature operates with AT- and XT-compatible hard disk controllers.

Golden Bow Systems

$60.00
Add $3 for shipping/handling
California residents add 6% sales tax

(619) 298-9349

Inquiry 75

68000 Cross Assembler

Motorola VERSAdos + Compatible

Assembler, Linker, Object and Macro Librarian. Absolute and Relocatable Code, Macros, Includes, and Conditional Assembly. Structured Programming. No limit on source file size.

Unix (C) Compatible Source
$700

CP/M-80*	PC/DOS†	CP/M-86*
$200	$250	$250

Manual: $20
(refundable)

farbware

1329 Gregory
Wilmette, IL 60091

(312) 251-5310

*Digital Research trademark †IBM trademark + Motorola trademark

Inquiry 62

5¼" BASF FLEXYDISKS

	10-90	100+
SS / DD Soft	$ .92 ea.	$ .89 ea.
DS / DD Soft	$1.02 ea.	$ .99 ea.
High Density (IBM - AT)	$3.15 ea.	$3.05 ea.
3.5 / SS Micro	$2.10 ea.	$2.05 ea.

In Stock — Immediate Shipment.

Mastercard, VISA, Check or Money Order.
Add $3.00 shipping charges per each 100 or part
Add $2.50 additional for C.O.D. shipments.
N.J. residents add 6% sales tax

Data Exchange
Dept. B, P.O. Box 993
179 Route 206 South
Somerville, N.J. 08876
(201) 874-5050

Inquiry 42

TOLL FREE
800-854-8230
CA Residents **714-558-8813**

OUR POLICY
•Surcharge for VISA or Mastercard.
•Volume purchasing agreements available.
•Orders subject to availability.
•Pricing subject to change without notice.
•ACP Retail Store pricing may vary. Not responsible for typos.

Inquiry 7

MAJOR BRAND LIQUIDATION

Retail Value $5000

Save up to 60%

ACP PRICE NOW ONLY!

$1985.00

Color 14" RGB Included
10 Mb Hard Disk

ACP is proud to make this one-time special offer for a complete computer system that is 110% compatible to IBM™. This is by far the most significant bargain that we at ACP have offered in our 10 year history. This system was successfully designed and manufactured to exceed IBM™'s PC in terms of quality, expansion modularity and capability, aesthetic appearance, and performance.

The system design utilizes the latest in state-of-the-art technology including:

- VLSI - Large Scale Integration Circuit Design
- Ergonomic CRT Design with Tilt Screen
- Professional Molded Packaging and Design
- High Quality 100 Watt Switching Supply
- Complete Integrated System
- Microsoft Compatible Mouse Function

The system is not a Taiwan or Korean knock-off. Each component is specifically designed and specified to meet the highest performance and reliability standards in the industry. It represents the best that Japanese craftsmen have to offer and you will be equally proud to own one of your own. ACP has a limited quantity of these systems in several different configurations. IBM™ PC-DOS™ v1.1/2.1, MS-DOS™ v2.11 and Concurrent v3.1 compatible. We have found no known incompatibility with any IBM™ PC application. Our technical staff has 8.5 Megabytes of various MS-DOS software packages installed including Lotus 1-2-3 and Flight Simulator. Each system comes complete with a 90 day warranty.

ACP Base System Consists of:

- (1) 360K DD/DS Floppy Disk Drive
- Mouse with Software
- 256K Memory Expandable to 640K on the Motherboard
- Deluxe Keyboard with LEDs
- Serial Port and Parallel Port
- Color or Monochrome Controller
- 4.77MHz, 8088 CPU
- 100 Watt Switching Supply w/Fan
- Three Expansion Slots
- Optional 6 Slot Expansion Chassis with Power Supply (add $399)

Base System A (as above) $995.00

	SYSTEM CONFIGURATION	Est IBM List*	Your Price
SYSTEM A	Base System (see left) PC with 360K Floppy, Keyboard & Mouse.	$2100.00	$995.00
SYSTEM B	Base System (see left) plus Add'l 360K Floppy Drive	$2295.00	$1099.00
SYSTEM C	Base System plus 12" Green Monitor with Detachable Tilt/Swival Base.	$2575.00	$1399.00
SYSTEM D	Base System plus 12" Color Monitor with Detachable Tilt/Swival Base.	$2995.00	$1699.00
SYSTEM E	Base System plus Clr Monitor, 10Mb Hard Disk and Boot Diagnostics.	$5000.00	$1985.00
SYSTEM F	Base System plus 80 Col. x 25 Line LCD Screen	N/A	$1299.00

*Assumes required add-in boards to provide same capacity

IBM PC is a trademark of IBM Corp.

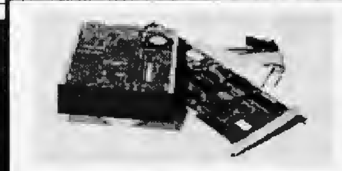
LQP PRINTER SPECIAL

Major Manufacturer Daisy Wheel Model 620 Letter Quality Printer with Serial Interface.

Perfect for IBM PC and Compatibles. Prints 25cps and comes with 90 day warranty.

List $1495...... **$495.00**

With purchase of any system above **$395.00**



10 Mb $449.00
HARD DISK
w/Controller for IBM™ PC and Compatibles.

10 Mb Standalone	**$249.00**
20 Mb Standalone	**$419.00**

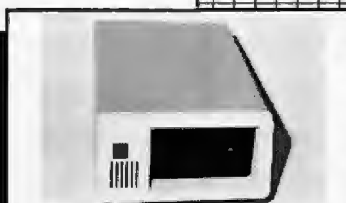
PC UPGRADE SPECIAL

$8.95 SET OF (9) 64K RAMS

$34.95 SET OF (9) 256K RAMS

$5.95 128K PIGGYBACKS

IBM Brand ASYNCH
INTERFACE CARD
List $100 **ACP $49.95**



5 Mb EXTERNAL
Sharp IBM Look-a-like w/contr. for IBM PC. Sub-Sys Price:**$429.00**
External Enclosure Add **$195.00**
Internal w/controller
Sub-System Price: **$299.00**

DELUXE JOYSTICK
$11.95

Compatible w/Atari 2600, 400, 800, VIC-20/64 and Apple. Apple requires optional cable adapter. Add **$2.95**

APPLE DISK DRIVE
$115.00

High quality ½ high drive for Apple II, II+, IIe or IIc. Apple IIc requires optional cable adapter. Add **$10.00**

★Advanced Computer Products Inc.

Mail Order: P.O. Box 17329 Irvine, CA 92713 • Retail: 1310 B E. Edinger, Santa Ana, CA 92705

CAT MASTER®

"THE ULTIMATE TIME-SAVER"

Have you ever **forgotten a File Name?** Or you recall the name but not on which Diskette it is located? CAT MASTER® eliminates this wasted time as it is much more than a Diskette Cataloguer, it is a Diskette **Manager!** CAT MASTER® can extend file names beyond the limits of DOS, so even though you can't remember the actual name, you need only to recall a subject, category, comment, etc., that was associated with the file.

Just present your Diskettes to CAT MASTER®, it does the rest.
For PC's, XT's, Compatables using DOS Ver. 2.

Visa, MC, Check - $69.95 + $2.00 Handling

(813) 546-5672

COMPUTERIZED CONCEPTS
5191 PARK BLVD.
PINELLAS PARK, FL 33565

Inquiry 33

Compute the Comet with Comet Vision!

Track Halley's Comet on your own IBM PC or compatible (with graphics), and follow it among the stars and planets from your own location.

★ **Compute the comet's celestial coordinates with display of comet and planet orbits.**
★ **See horizon views of Halley's Comet in the sky any time or location during the 85/86 apparition.**

Introductory software offer: **$34.95** plus **2.00** postage and handling. New Mexico residents add 4.375 sales tax. Cashiers check or money order only, mailed same day. Check, allow 3 wks. delivery.

Order from: **KLB** SOLAR INC.

201 La Veta Dr. NE, Abq., NM 87108.
For information call (505) 848-9001.

*IBM is a registered trademark of International Business Machines Corporation.

Inquiry 105

3M Diskettes Lifetime Warranty

TIRED OF WAITING FOR SERVICE AND PRICE?
9 out of 10 SURVEYED DISK BUYERS PREFERRED

NORTH HILLS
#1 IN SERVICE AND PRICE
1-800-328-3472

Formatted and hard sectored disks in stock-Dealer inquiries invited.
COD, VISA, MASTERCARD
All orders shipped within 24 hrs.

NH

NORTH HILLS CORP.
INTERNATIONAL
3564 Rolling View Dr.
White Bear Lake, MN. 55110
MN. call collect—612-770-0485

Genie®

Stomps Sidekick

Memory resident windowing utility includes: KEY MACROS, DOS Services, Alarm clock, Two calculators, Notepad/Editor, Rolodex Card file with no limit on # of cards, Calendar, ASCII table, Typewriter, and True cut & paste. Each module loads separately. Not copy protected for PC compats, or Zenith 100.

$49⁹⁵

for more info, or to order contact:

ADVANCED SOFTWARE TECHNOLOGIES
452 West 47 Street
NYC, NY 10036 (212) 247-0150

Inquiry 8

64K ■ 128K ■ 256K

DRAMS

80287-8 ■ 80287-3
8087-3 ■ 8087-2
8087-1

BITTNER ELECTRONICS

899 SOUTH COAST HIGHWAY
LAGUNA BEACH, CA 92651
(714) 497-6200

CALL NOW FOR **FREE** CATALOG

Inquiry 15

FORMAT-1-2-3

* **Loads ANY ASCII data into 1-2-3 and Symphony !**
* **On-line Help System!**
* **Send Spreadsheets using E-Mail!**
* **Runs on PC,XT,AT & compatibles**
* **Also TI Professional & AT&T!**
* **Header feature and more!**
* **Extended Warranty!**
* **Dos 2.0+ with 64K, 1 drive**

**** $150 MC, VISA, Check, COD**

RELIABLE SOFTWARE CO.
P.O. Box 99
Titusville, FL 32781
(305) 267-2043
(800) 235-6646 Ext. 581 (nat)
(800) 235-6647 Ext. 581 (ca)

Inquiry 170

Universal Machine Independent Assembler

MOPI:
The only assembler capable of generating machine code for any 8- or 16-bit micro using; a universal set of instructions, including procedural, the manufacturer's or user defined instructions.
The first assembler to implement a proposed AL Standard, applicable to any computer; simplifies learning AL, reduces development time, offers limited portability.
Simple enough for a beginning programmer, sophisticated enough for experienced computer professionals.

For CP/M or PC-DOS – $195
Proposed AL Standard – $10
Special Club, Site, Dealer Rates

VOCS
P.O. Box 3705
Mpls., MN 55403

Inquiry 209

MCFORMAT $50

Don't let DOS waste your disk space! Add up to 50% more hard disk capacity to your IBM PC, XT, AT or 100% compatible by selecting format parameters, including cluster & directory size. Fully DOS compatible (3.0 or 3.1). Complete with installation program and resident device driver. Replace DOS Format with interactive and informative MCFORMAT. VISA/MC.

MICROCOMPUTER CONCEPTS, INC.
9715 SE 43 Street
Mercer Island, WA 98040
(800) 722-8088 (206) 236-2300

Inquiry 127

OPT-TECH SORT™

SORT/MERGE Program for the IBM-PC, XT, AT & Compatibles
Btrieve Files Now Supported

- Written in assembly language for high performance. Example: 4,000 records of 128 bytes sorted to give key & pointer file in 30 seconds. COMPARE!
- Sort ascending or descending on up to nine fields
- Ten input files may be sorted or merged at one time
- Supports many file structures & data types
- Filesize limited only by your disk space
- Output file can be full records, keys or pointers
- Can be run from keyboard or as a batch command
- Can be called as a subroutine to many languages
- Easy to use — Fully documented
- $99 — VISA, M/C, Check, Money Order, COD, or PO. Quantity discounts and OEM licensing available

To order or to receive additional information write or call

OPT-TECH DATA PROCESSING
P.O. Box 678 Zephyr Cove, Nevada 89448
(702) 588-3737

Inquiry 144

HIGH RESOLUTION
640 x 260, .38 Dot Pitch
TAXAN 415
RGB COLOR
MONITOR
LIST PRICE $699
For cable add $19.00
$295

Printers

EPSON FX-85 & FX-185, OKIDATA 192 & 193, C. ITOH, CITIZEN, STAR

PRICED TOO LOW TO PUBLISH!

Call Us Toll Free For The Best Deal In The Country!

Printer Accessories

	LIST	JADE
IBM PC Style Cable	$54	$19.95
Standard Parallel Cable	$40	$19.95
RS-232 Serial Cable	$30	$24.95
Ribbons	as low as $4.99	
Apple IIc/Macintosh Cable	$39	$19.95

ISO-BAR

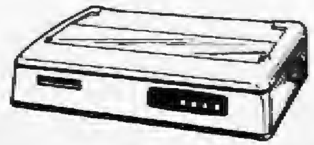
These industrial quality ISO-BARs contain surge suppression circuitry and built-in noise filters plus a 15 amp circuit breaker.

	LIST	JADE
6 Socket, 1 Filter ISO-BAR	$69	$44.95
4 Socket, 2 Filter ISO-BAR	$89	$59.95
8 Socket, 4 Filter ISO-BAR	$99	$69.95

UNINTERRUPTABLE POWER SUPPLY
Emergency back-up power to save your computer system. A must for every computer system.

	LIST	JADE
200 Watt UPS	$359	$279.95
425 Watt UPS	$538	$459.95
1000 Watt UPS	$1179	$995.95

WHY PAY $1149 FOR A C. ITOH
STARWRITER™ F-10
When our 40 cps Letter Quality Daisy-Wheel from the same manufacturer is only
$499.95



™StarWriter is a Trademark of C. Itoh Digital Products, Inc.

Letter Quality Upgrade For Your EPSON RX-80, RX-100, FX-80, FX-100

Near-Letter-Quality upgrade with front panel font controls for your old printer.

	LIST	JADE
LetterWriter Upgrade	$79	$58.95

MICROFAZER Buffers

Expandable to 64K (Parallel model expands to 512K)

		LIST	JADE
8K	Parallel in/Parallel out	$169	$139.95
64K	Parallel in/Parallel out	$225	$164.95
128K	Parallel in/Parallel out	$445	$269.95

Your choice: serial in/serial out, parallel in/serial out, serial in/parallel out.

	LIST	JADE
8K	$199	$169.95
64K	$260	$199.95

Microbuffers

Parallel or serial for printers and modems

		LIST	JADE
32K	Microbuffer	$299	$229.95
64K	Microbuffer	$349	$249.95
64K	Add-on Board	$179	$149.95

CAD Software

Best schematic and mechanical drafting package we've tested! Powerful, accurate, fast, easy to learn and use for:

- Space Engineering
- Interior Decorating
- Electrical Engineering
- Logic Diagrams
- Circuit Board Layouts
- Flow Charts
- Organizational Charts
- Project Schedules
- And More!

	LIST	JADE
CAD Software	$1790	$895

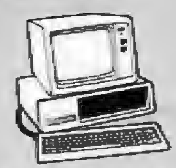
A-B Printer/Data Switch

Fully bi-directional switch allows your computer to run either of two printers, or allows two computers to share one printer, standard A-B switch box.

	LIST	JADE
Parallel Printer A-B Switch	$149	$69.95
Serial A-B Switch	$129	$79.95

JADE XPC IBM PC

- 640K of RAM
- 135 Watt Power Supply
- 8 Expansion Slots
- Deluxe Keyboard
- 90 Day Warranty



- 256K of RAM
- 63 Watt Power Supply
- 5 Expansion Slots
- IBM Keyboard
- 90 Day Warranty

256K of RAM, Two 360K Disk Drives & Disk Controller

$995

OPTION #1
256K of RAM
Two 360K Drives
Monochrome Graphics Card
Amdek 310A
IBM PC — $1995
JADE XPC — $1295

OPTION #3
640K of RAM
10 Megabyte of Hard Disk
One 360K Drive
130 Watts of Power
Parallel & Serial Ports
Color Card
Taxan RGB Monitor
IBM PC — $2995
JADE XPC — $1995

$1695

OPTION #2
256K of RAM
Two 360K Drives
Color Graphics Card
Taxan RGB Monitor
IBM PC — $2395
JADE XPC — $1395

JADE XPC PORTABLE

- 256K Expands to 512K
- 600 x 400 Hi-Res Video
- 110 Watt Power Supply
- Parallel and Serial Ports
- Four IBM Expansion Slots
- Deluxe Keyboard
- 90 Day Warranty

$1395 Includes Two 360K Disk Drives & Disk Controller

10 MEGABYTE PORTABLE
512K of RAM
10 MB Hard Disk
One 360K Drive
$1995

Place Orders Toll Free!

We accept cash, checks, credit cards or purchase orders from qualified firms & institutions. Minimum prepaid order $15.00. California residents add 6½% tax. Prices & availability subject to change without notice. Shipping & handling charges via UPS Ground 50¢/lb. UPS Air $1.00/lb. Minimum charge $3.00.

Continental U.S.A. (800) 421-5500
Inside California (800) 262-1710
Los Angeles Area (213) 973-7707

JADE Computer Products
4901 W. Rosecrans Ave. Box 5046 Hawthorne, CA 90251-5046

Subscription Problems?

We want to help!

*If you have a problem with your BYTE subscription, write us with the details. We'll do our best to set it right. But we **must** have the name, address, and **zip** of the subscription (new <u>and</u> old address, if it's a change of address). If the problem involves a payment, be sure to include copies of the credit card statement, or front and back of cancelled checks. Include a "business hours" phone number if possible.*

BYTE

Subscriber Service
P.O. Box 328
Hancock, NH 03449

McGraw-Hill

BYTE ADVERTISING SALES STAFF:

Dennis J. Riley, Advertising Sales Manager, 70 Main Street, Peterborough, NH 03458, tel. (603) 924-9281

NEW ENGLAND
ME, NH, VT, MA, RI,
EASTERN CANADA
Paul McPherson Jr. (617) 262-1160
McGraw-Hill Publications
575 Boylston Street
Boston, MA 02116

ATLANTIC
NY, NYC, CT, NJ (NORTH)
Leah Rabinowitz (212) 512-2096
McGraw-Hill Publications
1221 Avenue of the Americas—
39th Floor
New York, NY 10020

Dick McGurk (203) 968-7111
McGraw-Hill Publications
Building A—3rd Floor
777 Long Ridge Road
Stamford, CT 06902

EAST
PA (EAST), NJ (SOUTH),
MD, VA, W.VA, DE, D.C.
Daniel Ferro (215) 496-3833
McGraw-Hill Publications
Three Parkway
Philadelphia, PA 19102

SOUTHEAST
NC, SC, GA, FL, AL, TN
Maggie M. Dorvee (404) 252-0626
McGraw-Hill Publications
4170 Ashford-Dunwoody Road—
Suite 420
Atlanta, GA 30319

MIDWEST
IL, MO, KS, IA, ND, SD, MN, WI, NB, IN
Bob Denmead (312) 751-3740
McGraw-Hill Publications
Blair Building
645 North Michigan Ave.
Chicago, IL 60611

GREAT LAKES, OHIO REGION
MI, OH, PA (ALLEGHENY), KY,
ONTARIO, CANADA
Mike Kisseberth (313) 352-9760
McGraw-Hill Publications
4000 Town Center—Suite 770
Southfield, MI 48075

SOUTH PACIFIC
SOUTHERN CA, AZ, NM, LAS VEGAS
Jack Anderson (714) 557-6292
McGraw-Hill Publications
3001 Red Hill Ave.
Building #1—Suite 222
Costa Mesa, CA 92626

Karen Niles (213) 480-5243, 487-1160
McGraw-Hill Publications
3333 Wilshire Boulevard #407
Los Angeles, CA 90010

NORTH PACIFIC
HI, WA, OR, ID, MT, NORTHERN CA,
NV (except LAS VEGAS), W. CANADA
David Jern (415) 362-4600
McGraw-Hill Publications
425 Battery Street
San Francisco, CA 94111

Bill McAfee (415) 964-0624
McGraw-Hill Publications
1000 Elwell Court—Suite 225
Palo Alto, CA 94303

SOUTHWEST, ROCKY MOUNTAIN
UT, CO, WY, OK, TX, AR, MS, LA
(214) 458-2400
McGraw-Hill Publications
Prestonwood Tower—Suite 907
5151 Beltline
Dallas, TX 75240

WEST COAST SURPLUS AND RETAIL ACCOUNTS
Tom Harvey (805) 964-8577
3463 State Street—Suite 256
Santa Barbara, CA 93105

The Buyer's Mart
Karen Burgess (603) 924-9281
BYTE Publications
70 Main Street
Peterborough, NH 03458

Post Card Mailings National
Bradley Browne (603) 924-6166
BYTE Publications
70 Main Street
Peterborough, NH 03458

International Advertising Sales Representatives:

Mr. Hans Csokor
Publimedia
Reisnerstrasse 61
A-1037 Vienna, Austria
222 75 76 84

Mrs. Gurit Gepner
McGraw-Hill Publishing Co.
PO Box 2156
Bat Yam, 59121 Israel
3 866 561 321 39

Mr. Fritz Krusebecker
McGraw-Hill Publishing Co.
Liebigstrasse 19
D-6000 Frankfurt/Main 1
West Germany
69 72 01 81

Mrs. Maria Sarmiento
Pedro Teixeira 8, Off. 320
Iberia Mart 1
Madrid 4, Spain
1 45 52 891

Mr. Andrew Karnig
Andrew Karnig & Associates
Finnbodavagen
S-131 31 Nacka, Sweden
8-44 0005

Mr. Jean Christian Acis
McGraw-Hill Publishing Co.
17 rue Georges Bizet
F 75116 Paris
France
(1) 47-20-33-42

Mr. Arthur Scheffer
McGraw-Hill Publishing Co.
34 Dover St.
London W1X 3RA
England 01 493 1451

Mr. Savio Pesavento
McGraw-Hill Publishing Co.
Via Flavio Baracchini 1
20123 Milan, Italy
02 86 90 617

Seavex Ltd.
400 Orchard Road, #10-01
Singapore 0923
Republic of Singapore
Tel: 734-9790
Telex: RS35539 SEAVEX

Seavex Ltd.
503 Wilson House
19-27 Wyndham St.
Central, Hong Kong
Tel: 5-260149
Telex: 60904 SEVEX HX

Hiro Morita
McGraw-Hill Publishing Co.
Overseas Corp.
Room 1528
Kasumigaseki Bldg.
3-2-5 Kasumigaseki,
Chiyoda-Ku
Tokyo 100, Japan
3 581 9811

IBM PC 256k, 2 floppies	**IBM PC 20 + 20** 256k, 1 or 2 floppies 20MB HD & 20MB Backup	**IBM XT 10 + 10** 256k, 1 or 2 floppies 10MB HD & 10MB Backup		**IBM AT 20 + 20** 512k, 1 or 2 floppies 20MB HD & 20MB Backup	**COMPAQ DESKPRO 10 MB** 640k, 1 or 2 floppies	**COMPAQ PORTABLE 286** Custom Configurations
IBM PC 10 MB 256k, 1 or 2 floppies	**IBM XT** 256k, 2 floppies	**IBM XT 20 + 20** 256k, 1 or 2 floppies 20MB HD & 20MB Backup		**IBM AT 30 + 60** 512k, 1 or 2 floppies 30MB HD & 60MB Backup	**COMPAQ DESKPRO 10 + 10** 10MB HD & 10MB Backup	**COMPAQ DESKPRO 286** Custom Configurations
IBM PC 20 MB 256k, 1 or 2 floppies	**IBM XT 10 MB** 256k, 1 or 2 floppies	**IBM AT 20 MB** 512k, 1 or 2 floppies		**IBM AT 40 + 60** 512k, 1 or 2 floppies 40MB HD & 60MB Backup	**COMPAQ DESKPRO 20 MB** Backup Available	**IBM XT COMPATIBLE** Custom Configurations
IBM PC 30 MB 256k, 1 or 2 floppies	**IBM XT 20 MB** 256k, 1 or 2 floppies	**IBM AT 30 MB** 512k, 1 or 2 floppies		**COMPAQ PORTABLE** 256k, 1 or 2 floppies	**COMPAQ DESKPRO 30 MB** Backup Available	**IBM AT COMPATIBLE** Custom Configurations
IBM PC 10 + 10 256k, 1 or 2 floppies 10MB HD & 10MB Backup	**IBM XT 30 MB** 256k, 1 or 2 floppies	**IBM AT 40 MB** 512k, 1 or 2 floppies	**FOR EXTRA SAVINGS** Call for the latest prices for your custom configuration. All systems are configured and tested at no extra cost and come only with Compumail's 90 day warranty.	**COMPAQ PORTABLE 10 MB** 256k, 1 or 2 floppies	**COMPAQ DESKPRO 40 MB** Backup Available	IBM COMPATIBLES ARE YOUR LOW COST ALLTERNATIVE

★ PRICE WAR ★ CALL US LAST WITH YOUR BEST QUOTES ★

PRINTERS

- **EPSON** Daisy Wheel DX-10/DX-15 ... CALL
  - FX-85+NLQ ... $339 FX-185+NLQ ... $459
  - Sht Fdr FX-85/FX-185 ... $249/$299
  - NLQ Upgrade Kit for FX-80/FX-100 ... CALL
  - LX-80/HI-80/JX-80 ... $219/$359/$469
  - Tractor LX-80/FX-80/HI-80 ... $50/$60/$70
  - LQ-1500 w/Parallel Module ... $889
  - LQ Tract/Sing/Dual Sht Fdr ... $70/$450/$750
  - LX-90 w/Par Interface & Tractor ... $279
- **OKIDATA** ... TOO LOW TO QUOTE
  - ... OKIMATE 20 w/IBM Plug & Print
  - 182-P/182-IBM ... 192-P/192-IBM
  - 193-P/193-IBM ... 84-P/84-IBM
  - 192 Tractor ... $70 2410 P IBM ... CALL
- **TOSHIBA** 1340P ... $499
- **TOSHIBA** 351P ... "REDUCED" ... $1049
  - 351 Tractor/Sht Fdr ... $190/$790
- **JUKI** 6100/6200/6300 ... $349/$549/$679
  - 6X00 Tractor/Sht Fdr ... $150/$250
  - 5510P (160cps) $369 5510 Color Kit ... $150
- **BROTHER** ... TOO LOW TO QUOTE
  - HR-10/HR-15XL ... CALL
  - HR-15 Trac/Kybrd/Sht Fdr ... CALL
  - HR-25P/HR-35P ... CALL
  - HR-25/35 Tractor/Sht Fdr ... CALL
  - 2024L LQ/Graphics-24 pin, 160/80 cps ... CALL
  - 2024L Sht Fdr-Narrow/Wide ... CALL
  - TWINRITER 5 ... World's First Dot Matrix & Daisy Wheel combined in ONE ... Best Price
  - TWINRITER 5 Trac/Sht Fdr ... CALL
- **C. ITOH** ProWriter Jr w/NLQ ... $219
  - 8510BPI $309 8510SEP+NLQ ... $379
  - 8510SCEP+NLQ $449 1550-EP ... $429
  - 1550SEP+NLQ $519 1550SCEP+NLQ ... $619
  - 24LQP $949 A10-30SP ... $439
  - Y10-20P ... $379 A10 Tractor ... $160
  - F10-40P ... $869 F10-55P ... $1069
  - F10 Tractor/Sht Fdr ... $190/$290
  - CX-4800P 4 Pen Plotter ... $429
- **QUME** LetterPro 20P (20 cps) ... $399
  - 20P Trac/Sht Fdr ... $150/$390
  - SPRINT 1140+/1155+ ... $1299/$1399
  - SPRINT Interface Module ... $80
  - SPRINT Trac/Sht Fdr ... $210/$690
- **PANASONIC** KX-P3151 LQ (22cps) ... $459
  - KX-P1090/91/92/93 ... $179/$249/$349/$449
- **STAR** Power Type/SB-10 ... $329/CALL
  - SG-10/SG-15 (120cps) ... $239/$389
  - SD-10/SD-15 (160cps) ... $349/$459
  - SR-10/SR-15 (200cps) ... $499/$599
- **NEC** PINWRITER P-2/P-3 ... $499/$699
  - Sht Fdr for P-2/P-3 ... $350/$420
  - 2050/3550/8850 ... $649/$999/$1399
  - SPINWRITER Trac/Sht Fdr ... $190/$790
  - ELF 360 Par (16cps) ... $419
- **DIABLO** Advantage D-25 ... $499
  - 630 API ... $1499 630 ECS-IBM ... $1599
  - Trac/Mech Single Bin Sht Fdr ... $230/$470
- **CITIZEN**: 18 months manufacturer warranty
  - MSP-10/15 (160/40cps) ... $299/$449
  - MSP-20/25 (200/50cps) ... $439/$589
  - Sht Fdr (Specify Model) ... $229
- **LEGEND** CP-VII ... $899 1200 ... $259
  - 880/1080/1380/1385 ... $199/$219/$259/$319
- **ANADEX** 9620B/9625B-PLUS ... $1199/$1299
  - 9725B COLOR ... $1399
  - WP6000/DP6500 ... $2299/$2569
- **HEWLETT PACKARD PRINTERS** ... CALL
- **SILVER REED** Daisywheel - Parallel Models
  - 400/500/550/770 ... $189/$269/$379/$669
- **TELEVIDEO** 750 Daisywheel-50 cps ... CALL
- **AB/ABC/ABCD** Switch Boxes ... CALL
- **KENSINGTON** Universal Printer Stand ... $20

FLOPPY/HARD DISKS

- **CDC/MITSUBISHI/PANASONIC/TANDON/TEAC**
  - Half Ht DSDD ... 'REDUCED' ... $90
  - Free Mounting Kit with 2 Drives
- **CDC/TANDON** Full Ht DSDD ... $99
- **HARD DISKS for IBM PC/XT**
  - 10 MB Int w/Controller ... 'REDUCED' ... $399
  - 10 MB Ext w/Controller ... 'REDUCED' ... $649
  - 20 MB Int w/Controller ... 'REDUCED' ... $529
  - 20 MB Ext w/Controller ... 'REDUCED' ... $799
  - 33 MB Int w/Controller ... 'REDUCED' ... $849
  - 33 MB Ext w/Controller ... 'REDUCED' ... $1049
  - 44 MB Int w/Controller ... 'REDUCED' ... $1049
- **MAYNARD** complete line
  - MAYNSTREAM portable tape backup 20MB ... $999
  - MAYNSTREAM portable tape backup 60MB ... $1299
- **HARD DISKS/BACKUP FOR IBM PC AT** ... CALL
- **GENOA GALAXAY TAPE BACKUPS**
  - 20 MB Int/Ext ... $749/$849
  - 20 MB Int/Ext ... $999/$1099
  - 32+60 MB Ext Disk/Backup ... $2399
- **TALL GRASS** NEW PC/T FORMAT
  - 25/35 MB w/60 MB Backup ... $2599/$3399
  - 50/80 MB w/60 MB Backup ... $4299/$5499
  - 60 MB Tape Backup only ... $1199
  - Controller ... $140 Cartridge (60 MB) ... $40
- **EVEREX** 10/20 MB Int w/Cont ... $579/CALL
  - EVEREX 60MB Int/Ext Tape Backup ... $899/$999
  - EVEREX Expansion Chasis Half Height ... $699
  - EVEREX Expansion Chasis Full Height ... $749
- **INTERDYNE TAPE BACKUP**
  - 10 MB Int/Ext ... 'REDUCED' ... $369/$549
  - 20 MB Int/Ext ... 'REDUCED' ... CALL
- **HARD DISKS/BACKUP FOR COMPAQ** ... CALL
- **Bernoulli** Box by Iomega 10+10 ... CALL
  - Auto Boot also Available
- **BERNOULLI** Box by Iomega 20+20 ... CALL
- **EMULEX** JAVELIN HD & BACKUP ... CALL
- **PWR SUPPLY** 130/150 Watts ... $89/$109

MULTI-FUNCTION CARDS

- **AST** SIXPAK 64K ... $219 384K ... $245
  - Rampage! ... Advantage 0-K ... $379
- **ORCHID** Blossom 0-K ... $169 384K ... $199
  - Conquest for IBM PC/XT (up to 2MB) ... CALL
  - Cram-RAM for IBM XT short slot (up to 2MB) ... CALL
  - Eccell for IBM AT (up to 4MB) ... CALL
- **TECMAR** Captain 0-K ... $169 384K ... $199
  - MAESTRO AT w/Treasure Chest ... Best Price
- **INTEL** Above Board (up to 4MB) ... CALL
- **PARADISE** 5-Pack 0-K ... $159 384K ... $189
- **BT6** Plus w/64k ... $175 384k ... $199
- **MF-100** 0-K ... $179 384k ... $209
  - Par, Ser & Game Ports, Clock/Cal
- **QUADBOARD** 0-K ... $189 384K ... $219
  - Silver Quadboard: 0-K to 640k ... CALL
  - Gold Quadboard: Multifunction & Color ... CALL
  - Quadmeg-AT (up to 4 MB) ... CALL
  - Quadport-AT/Expansion Kit ... CALL
- **INSTALL RAM** & test brd for PC/AT ... $20/$40

MONITORS

- **PGS** MAX-12/HX-12/HX-12E ... $169/$419/$499
  - HX-9/HX-9E/SR-12 ... $419/$499/$549
  - SR-12 w/Scan Doubler ... $699
  - SR-12 w/Sigma 400 ... $999
- **TAXAN** COMPOSIT 115 Green/116 Amber ... $129
  - MONO 121 Gr/122 Amb ... $139/$149
  - COLOR 620 4 text modes 640X262 Res ... $359
  - COLOR 630 4 text modes 640X400 Res ... $399
  - COLOR 640 Top of the line monitor ... $489
  - COLOR 640 w/TAXAN 555 RGB Card ... $749
  - COLOR 640 w/Persyst BoB Brd ... $799
  - COLOR 640 w/Sigma 400 ... $939
  - COLOR 640 w/Sigma 400 & PC Mouse ... $1039
- **AMDEK** 300G/300A/310A(M) ... $139/$149/$159
  - COLOR 300/500/600 ... $239/$349/$419
  - 710 ... $489 w/Tecmar Graphics Master ... $899
- **IBM** Enhanced Color Monitor ... CALL
- **ROLAND** MB-142 14"/Color Monitors ... Best Price
- **TECMAR** Color ... $529 w/Graphics Master ... $939
- **TILT/SWIVEL** Monitor Pedestal ... $30

DISPLAY CARDS

- **HERCULES** MonoGraphics ... $279 Color ... $149
- **HERCULES COMPATIBLE** MonoGraphics ... $199
- **EVEREX** The Edge ... Best Price Ever
  - Graphics Edge ... Even Better Price
- **AST** Monograph Plus ... $339/$369
  - Preview for Mono Graphics ... $279
  - Graphpack w/64k ... $549
- **PERSYST** Short Color/Mini Mono ... $159/CALL
  - Color Combo: Multifunction & Color ... from $299
  - Mono Combo: Multifunction & Mono ... from $299
- **PERSYST** BoB Board ... $339
- **PERSYST** w/TAXAN 440 ... $799
- **IBM** EGA (Enhanced Graphics Adapter) ... CALL
  - Gold Quadboard: Multifunction & Color ... CALL
- **STB** Chauffer ... "REDUCED" ... $299
- **PARADISE** Modular Brd ... from $259
- **ORCHID** Turbo Graphics Controller ... CALL
- **TECMAR** Graphics Master ... $429
  - G Master w/TECMAR Color Monitor ... $939
  - G Master w/AMDEK 710 Color Monitor ... $899
- **GENOA** Spectrum ... Best Price
- **MYLEX** Chairman ... $399
- **SIGMA** COLOR 400 (specify Monitor) ... $479
  - COLOR 400 w/PGS SR-12 ... $999
  - COLOR 400 w/TAXAN 440 ... $939
  - 3 Button PC Mouse w/PC Paint ... Add $100

CHIPS ETC.

HIGHEST QUALITY!

Major Brands

- **64K RAM** Set ... $7 10+ Sets ... $6 50+ ... $5
- **128K RAM** Set ... $35 10+ Sets ... $33 50+ ... $30
- **256K RAM** Set ... $30 10+ Sets ... $28 50+ ... $25
- **8087** Maths Co-Processor ... Best Price
- **80287-3** for IBM AT ... 'REDUCED' ... $239
- **ORCHID** PCturbo w/128k ... $549 640k ... $699
- **QUADRAM** QUADsprint ... 'REDUCED' ... $499
- **MICROWAY'S Fastbreak/# Smasher** ... CALL

MODEMS

- **EasyData** 12B 300/1200 Bd Int w/PCTalk III ... $199
- **SmarTEAM** 212A 1200 Bd Ext w/o sft ... $209
- **HAYES** 1200 B w/SmartCom II ... $319
- **HAYES** 1200 Ext. w/o Software ... $349
  - Smartcom II ... $89 NEW 2400 Bd Ext ... $589
- **HAYES** Transet 1000 ... "REDUCED" ... $299
- **POPCOM** C-150 Int w/PFS:Access ... $289
- **POPCOM** X-150 Ext w/PFS:Access ... $299
- **AST** Reach! 1200 Bd short card w/Crosstalk ... $359
- **PROMETHEUS** PROMODEM 1200 Ext ... $269
  - Options Processor ... $79 w/Modem ... $65
  - Alphanum Display ... $79 w/Modem ... $65
  - PROMODEM 1200 Ext + Both Options ... $399
  - PROMODEM 1200 B Int w/MITE software ... $229
- **QUADMODEM** II shortcard w/Crosstalk XVI ... CALL
  - 2400 Bd Upgrade Kit for Quadmodem II ... CALL
- **Ven-Tel** Half Card w/Crosstalk XVI ... $349
  - 1200 Plus Ext w/o software ... $349
- **BIZCOMP** IntelliModem EXT ... $319
  - ST/XL/XT ... $289/$319/$359
- **WATSON** Integrated Voice Data Modem ... CALL
- **EVEREX** 1200B Internal w/software ... $249
- **U.S. ROBOTICS** Complete Line ... CALL

SPECIAL ITEMS

- **OBERON OMNI READER**
  - A must for any one using word processing. Works like a scanner, transferring text from paper to your computer. Reads common type faces. Will save hours of laborious typing. It is a great technological breakthrough at a very reasonable price. Comes complete with interface and software for your computer ... $599
- **IBM** TERMINAL 3161-1 ... CALL
- **IBM** TERMINAL 3161-11 ... CALL
- **GENERAL** Parametric's VideoShow ... CALL
- **CHORUS** PHOTOBASE/PC-EYE ... CALL

MISC. ADD ONS

- **CABLE** Parallel, 6ft ... $20 Serial, 6 ft ... $25
  - Keyboard Extension, 6 ft ... $10
- **FLOPPY** Drive Controller ... $109
- **FLOPPY** Controller w/P,S,G & Clock/Cal ... $199
- **FLOPPY** & HD Controller ... "REDUCED" ... $299
- **HAYES** Transet 1000 ... REDUCED ... $299
- **MOUSE SYSTEMS** PC Mouse w/PC Paint ... $129
- **MICROSOFT** MOUSE w/PC Paintbrush
  - Bus/Serial ... $129/$139
- **KEYTRONIC** KB 5151 ... $165 AT Upgrade Kit ... $15
  - KB 5153T w/Touch Pad ... CALL
- **AB/ABC/ABCD** Switch Boxes ... CALL
- **COMPUTER ACCESSORIES** P 1-2-3 ... $299
  - POWER DIRECTOR P2(5) ... $89 P12(6) ... $129
- **KENSINGTON** Master Piece (5 outlets) ... $89
- **KENSINGTON** Universal Printer Stand ... $20
- **KENSINGTON** Master Piece Plus ... CALL
- **PC** Keyboard Storage Drawer ... $99
- **STANDBY PWR SUPPLY** w/surge protection
  - 200 Watts ... $269 300 Watts ... CALL
  - 800 Watts (Sine Wave) ... $679
- **TILT/SWIVEL** Monitor Pedestal ... $30
- **PWR SUPPLY** 130/150 Watts ... $89/$129
- **EVEREX** Expansion Chasis Half Height ... $699
- **EVEREX** Expansion Chasis Full Height ... $749

CALL FOR OUR WEEKLY UNADVERTISED SPECIALS

AMERICAN EXPRESS VISA

COMPANY POLICY: Min. order $100. Prices & availability subject to change. We ship UPS only. Shipping/handling charges vary. Prices reflect cash discount on prepaid orders. Add 1% for COD orders, 3% for MC/VISA & 5% for AMEX. COD requires cashiers check. All merchandise sold is new and all sales are final. Refused shipments subject to 20% charge (Min $50). Products shipped in factory cartons come with manufacturer warranty. For others, including IBM systems, call tech support for return auth. # for warranty repairs. Non-defective items returned as defective subject to 10% service charge (Min. $50). Not responsible for hardware or software compatibility of any product. No open acct PO's or foreign orders. No showroom, demonstrations or walk-in sales. Personal or company checks take 3 weeks to clear. FOR ADVANCE PAYMENTS or PICKUP. PLEASE CALL FIRST FOR WORKORDER #.

COMPUMAIL 805-987-7015

406-C CONSTITUTION AVE., CAMARILLO, CA 93010

WHEN ORDERING PLEASE REFER TO AD # S805

BYTE's Ongoing Monitor Box

BOMB CARD EXPLANATION

Just as with your regular issues of BYTE, the editors would like to survey how effectively each article in our 1985 BYTE special issue serves our readers. Please fill out the card opposite the readers' index, mail it in, and look for the tabulation of your votes in the February BYTE.

First-place winner will receive a bonus of $100, second-place author will receive $50, and the article that scores third will be mentioned in these pages. Articles written by anyone under the aegis of BYTE automatically forfeit the bonus to the next-placed article.

ARTICLE#	PAGE	ARTICLE	AUTHOR(S)
1	11	An Annotated Bibliography of Recent Books	Crabb
2	39	Public-Domain Utilities	Edwards
3	59	ROM BIOS Extensions for the PC AT	Norton
4	66	The 8087/80287 Performance Curve	Fried
5	92	Moving from the 8088 to the 80286	Claff
6	105	Writing Desk Accessories	Wadlow
7	127	A MIDI Recorder	Swearingen
8	143	Circuit Design with Lotus 1-2-3	Haynes
9	159	Adding a Hard Disk to Your PC AT	Shiell, Markoff
10	167	Fixed Disks and the PC AT	Shiell, Markoff
11	177	A Survey of Debuggers	Drake, McCaffrey, Sadowsky
12	187	IBM Compatibility Issues	Dahmke
13	195	Benchmarking the Clones	Edwards, Hartwig
14	203	Four Hard Disks for Under $1000	Grehan
15	209	Programming the Enhanced Graphics Adapter	Wilton
16	223	IBM PC Interrupt Service Routines	Dunphy
17	231	Pick, Coherent, and THEOS	Rochkind
18	243	One Million Primes Through the Sieve	Peng
19	247	TopView	Byers
20	253	When Your PC Doesn't Work	Williams
21	259	IBM PC Family BIOS Comparison	Shiell, Markoff
22	269	Device-Independent Graphics	Clarkson
23	283	IBM PC Disk Performance and the Interleave Factor	Kolod

TIPS

SUBSCRIBERS ONLY!*

Use BYTE's Telephone Inquiry Processing System

Using TIPS can bring product information as much as 10 days earlier.

SEND FOR YOUR SUBSCRIBER I.D. CARD

1) If you are a new subscriber or have lost your I.D. card, circle #1 on the Reader Service Card; **attach mailer label.** We will immediately send your personal TIPS subscriber card.

GET PREPARED

2) Write your Subscriber Number, as printed on your Subscriber I.D. Card, in boxes in Step 5 below. (Do not add 0's to fill in blank boxes)

3) Write numbers for information desired in boxes in Step 7b below. (Do not add 0's to fill in blank boxes.)

CALL TIPS

4) Now, on a Touch-Tone telephone dial: **(413) 442-2668** and wait for voice commands.

ENTER YOUR SUBSCRIBER AND ISSUE NUMBERS

5) When TIPS says: "Enter Subscriber Number"
(Enter by pushing the numbers and symbols [# or * enclosed in the boxes] on telephone pad ignoring blank boxes)
Enter [ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ] [#] [#]

6) When TIPS says "Enter magazine code & issue code"
Enter [1] [#] [1] [3] [5] [#] [#]

ENTER YOUR INQUIRIES

7a) When TIPS says "Enter (next) Inquiry Number"
Enter one inquiry selection from below (ignore blank boxes)

b) Repeat 7a as needed (maximum 17 inquiry numbers)

1. [ ] [ ] [ ] [#] [#]
2. [ ] [ ] [ ] [#] [#]
3. [ ] [ ] [ ] [#] [#]
4. [ ] [ ] [ ] [#] [#]
5. [ ] [ ] [ ] [#] [#]
6. [ ] [ ] [ ] [#] [#]
7. [ ] [ ] [ ] [#] [#]
8. [ ] [ ] [ ] [#] [#]
9. [ ] [ ] [ ] [#] [#]
10. [ ] [ ] [ ] [#] [#]
11. [ ] [ ] [ ] [#] [#]
12. [ ] [ ] [ ] [#] [#]
13. [ ] [ ] [ ] [#] [#]
14. [ ] [ ] [ ] [#] [#]
15. [ ] [ ] [ ] [#] [#]
16. [ ] [ ] [ ] [#] [#]
17. [ ] [ ] [ ] [#] [#]

END SESSION

8) End session by entering [*] [*] [9] [1] [#] [#]

9) Hang up after hearing final message
If you are a subscriber and need assistance, call (603) 924-9281.

Available 24 Hours, 7 Days a Week

If you are not a subscriber fill out the subscription card found in this issue or, call BYTE Circulation 800-258-5485.

*Domestic and Canadian Subscribers Only!

R·E·A·D·E·R S·E·R·V·I·C·E

* *Correspond directly with company.*

TO GET FURTHER information on the products advertised in BYTE, either pick up your touch-tone telephone and use TIPS (if you are a subscriber), or fill out the reader service card. Either way full instructions are provided following this reader service index which is provided as an additional service by the publisher, who assumes no liability for errors or omissions.


NUMBER
SMASHER™


Maxell Corp. of America, 60 Oxford Drive, Moonachie, NJ 07074
maxell