QNX: A UNIX-like Multiuser Operating System

LAN and Multiuser PC Integration

November 1988 • $3.95 ($4.95 in Canada)

Micro Systems Journal

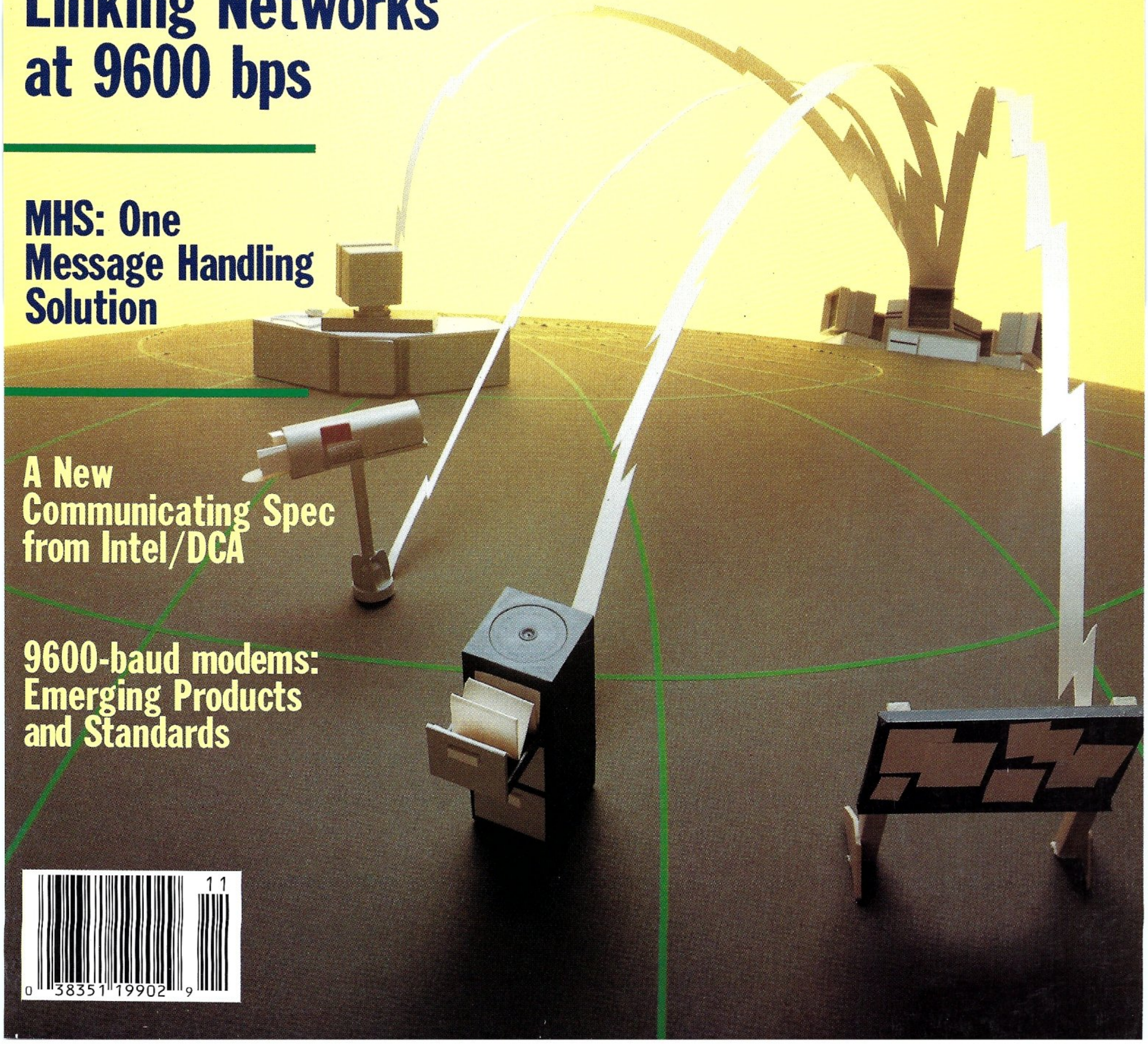
LAN Communications

Linking Networks at 9600 bps

MHS: One Message Handling Solution

A New Communicating Spec from Intel/DCA

9600-baud modems: Emerging Products and Standards



Meet Our High Powered Boards Of Directors

MICRONICS 386 MOTHERBOARDS,

acknowledged as the best in the business by experts such as PC Week, Byte, PC World and New York University, is proud to announce its third generation of products.

Micronics motherboards provide faster throughput, greater memory expandability and software compatibility. A tremendous performance advantage at a very affordable price.

Unlike the clones, Micronics motherboards run all major operating systems and offer true 32 bit memory expansion with 0 wait states.

Finally there is a company that fully utilizes the power of the 386.

The Chairman of the Board
is our newest and most powerful member. This 25 MHz board supports Weitek and 80387 math coprocessors. It has cache memory and the fastest 386 CPU available. This board has the power to analyze, communicate and compute data faster than any other.

Vice President of Engineering
with 20 MHz and cache memory, supports Weitek and 80387 math coprocessors. This board member is ideal for CAD/CAM, graphics applications and engineering simulations. It is also a superior data processing manager.

Our Founder and Chief Operating Officer
is our tried and true workhorse. It is the member most responsible for building our company and over the years has developed a loyal following. It is available in a 16 MHz model or 20 MHz models with or without cache memory.

Junior Executive
is our smallest and most compact unit. It fits into a small footprint or portable case and has all the power and performance of our full sized boards. It provides on-board support of the 80387 math coprocessor and due to its size, offers the user maximum flexibility.

For more information, pricing or your nearest Micronics dealer please call:
(800) 234-4386 • (408) 732-0940
Fax (408) 732-6048

MICRONICS
COMPUTERS INC.

935 Benecia Avenue, Sunnyvale, California 94086

CIRCLE 113 ON READER SERVICE CARD

Network Troubleshooting?

How's it going?

Not well, from the look of it. It's not one of your better days. The system's down. All eyes are on you.

They're acting like it's your fault. It's not. But it's your job. You can go down with the system, or you can turn to the best Ethernet network analyzers on the market.

Before your day gets any worse, call Excelan and find out about the LANalyzer EX 5000 Series. We'll give you the details, and, for a limited time, a free poster version of this ad.

EXCELAN 2180 Fortune Drive San Jose, California 95131 800-2-GET-LAN FAX 408-434-2310 Telex 176610
Excelan Europe, Ltd. Intec 2, Suite 17, Wade Road Basingstoke, Hampshire, RG24 0NE England, UK
Tel: (44) 256-842296 FAX (44) 256-842298

CIRCLE 122 ON READER SERVICE CARD

Finally, there's an SQL that gets back to BASIC. And COBOL. And C. And Pascal.

As a programmer, you've probably already faced it—the database dilemma. Do you use an SQL for easy database handling, or a true programming language for maximum power and flexibility?

Now you can do both with XQL,® the relational data management system from the developers of Btrieve.®

The Programmer's SQL. With XQL, you can access your data with the ease of Structured Query Language through simple subroutine calls from traditional programming languages. XQL supports standard SQL syntax, including subqueries, unions and security groups.

XQL Relational Primitive Operations. In addition, XQL lets you bypass the SQL level and perform highly efficient, relational primitive operations directly. You get all the functionality of a relational database model without the constraints of a 4th generation language.

Building on Btrieve. The heart of Novell's family of data management tools is Btrieve. By letting you access multiple records at a time, XQL adds a powerful dimension to Btrieve. XQL incorporates sophisticated data manipulation features which allow you to access data by field name, move forward or backwards through the database, compute fields from other fields or constants, and even work with composite records built from multiple, joined Btrieve files.

Like Btrieve, XQL offers features like multiuser support, fault tolerance, comprehensive documentation, and expert technical support. And you never pay royalties on your XQL applications.

Solve the database dilemma with XQL, the SQL that speaks your language. Only $795.* See your Authorized Novell Gold Reseller, or call us at (512) 346-8380.

For more information, call from your modem 1-800-444-4472 (8 bit, no parity, 1 stop bit) and enter the access code NVXQL13.

NOVELL

For software solutions, you should be seeing red.

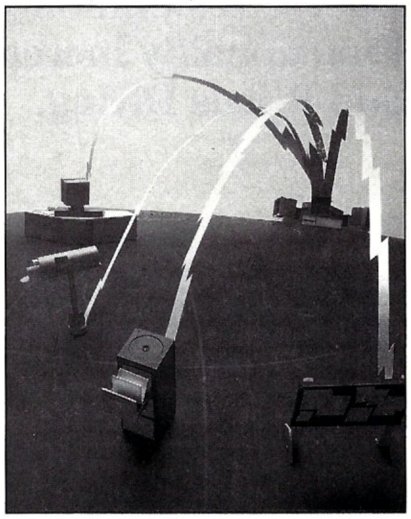
*Suggested retail price (US dollars) ©1988 Novell Inc., World Headquarters, 122 East 1700 South, Provo, Utah 84601 (801) 379-5900
Requires Btrieve 4.x and PC-DOS or MS-DOS 2.x, 3.x.

CIRCLE 65 ON READER SERVICE CARD.

LAN and Multiuser PC Integration

Micro/Systems JOURNAL

November 1988
Vol. 4, No. 11



About the cover: When Marshall McLuhen conceived of the global village more than 20 years ago he may not have been thinking specifically about computer technology, but high-speed data links have helped make the world a little smaller. As technology advances and the barriers of incompatibility continue to collapse, a workstation can conceivably be connected to any PC or network in the world. In this issue, *Micro/Systems* will examine methods to link LANs using 9600-baud modems, message handling, Intel's CAS specification, and related network communications technologies.

Micro/Systems Journal (ISSN 8750-9482) is published monthly by M & T Publishing, Inc., 501 Galveston Drive, Redwood City, CA 94063; (415) 366-3600. Second-class postage paid at Redwood City and at additional entry points. **POSTMASTER:** send address changes (form 3547) to *Micro/Systems Journal*, Box 3713, Escondido, CA 92025. **Change of Address:** Please send old label and new address to *Micro/Systems Journal*, Box 3713, Escondido, CA 92025. **Customer Service:** For subscription orders and changes of address, call toll-free: (800) 354-8400. For other subscription inquiries call (800) 321-333 (in California call (800) 331-4164); outside U.S. call (619) 485-9623. **Subscription Rates:** U.S., $29.97 for one year, $56.97 for two years. Foreign orders must be prepaid in U.S. funds with additional postage. Add $10 per year for surface mail to all countries or add $20 for airmail to Canada and Mexico; $28 for airlift/surface to other countries. Entire contents copyright ©1988 by M & T Publishing, Inc., unless otherwise noted on specific articles. All rights reserved.

High-Speed Modems — Data Rates Climbing, Functionality Increasing, and Prices Diving.

The high-speed modem marketplace is rapidly changing. Data rates are increasing rapidly and prices are plummeting dramatically. The number of features are also increasing, however. Many modems now perform diagnostics on themselves and often on the lines over which they communicate. Many programmed network functions and multiplexing are now incorporated in high-end, high-speed modems.

The low-end, high-speed modems are already selling for well under $1,000 (and in quantity for close to $500). When they first appeared, high-end modems commanded $5,000 to $6,000. Now they sell for less than $2,000. A price war is currently raging in the high-speed modem marketplace and additional price drops appear likely.

Although 2400 bps modems are still the hot sellers and sell at a fraction of the price of the 9,600 – 19,200 bps modems, it is apparent that the higher performance units pay for themselves quickly.

We reviewed many of these early "high-speed" modems in past issues (see *Micro/Systems Journal*, January/February 1987 and March/April 1987). In this issue, Charles Strom checks out three new units that have been introduced since our earlier reviews.

The big limiting factor in the growth of the high-speed modem marketplace is not price. Rather, it is a lack of standards. Although standards are being developed, the standards groups are moving slowly and most modem manufacturers cannot wait for the standards to be developed. The result is that we have each manufacturer pursuing its own idea of what the standard will be, with no compatibility between modems. Therefore, we now have a "Tower of Babel" where one manufacturer's modems cannot talk to another manufacturer's modems. Standards keep product prices down and make them more universally applicable. In his article, William Wong explores the different techniques currently in use and the standards being developed.

In the 19.2 kbps area, it may be several years before a standard will be established. And by that time, many of the nation's local and long-distance telephone networks will have been converted to digital transmission, and ISDN will be the way to go. Most experts agree that in five to six years, current modem technologies will be a thing of the past.

Most companies use high-speed modems to interconnect remote sites via dedicated leased lines. However, since the divestiture of the phone companies in 1984, leased-line prices have risen about 40 percent while dial-up service prices have dropped by about 40 percent. Because of low costs, many companies have switched from leased to dial-up service.

As dial-up service plays a more important role in corporate networks, users have sought more sophisticated network management tools to control dial-up modems and diagnose problems. Now that users have a choice of carriers, they need tools that allow them to gauge the quality of different services and isolate problems when they occur. A whole new breed of software tools is beginning to appear to fit this need. These tools provide diagnostic functions such as remote loop-back testing, and management tools to evaluate costs such as switched-traffic statistics and call-accounting data.

Sol Libes

Sol Libes
Editor

LAN and Multiuser PC Integration
Micro/Systems JOURNAL

EDITORIAL	
Founder and Editor	Sol Libes
Technical Editors	Stephen R. Davis
	Don Libes
Associate Editors	Lennie Libes
	Susan Libes
Contributing Editors	A.G.W. Cameron
	Bob Blacher
	Patrick Corrigan
	P.L. Olympia
Managing Editor	Thomas M. Woolf
PRODUCTION	
Art & Production Director	Larry L. Clay
Art Director	Joe Sikoryak
Typographers	Lorraine Buckland
	Margaret Anderson
CIRCULATION	
Director of Circulation	Maureen Kaminski
Asst. Circulation Mgr.	Andrea Weingart
Fulfillment Coord.	Francesca Martin
Newsstand Mgr.	Sarah Frisbie
ADMINISTRATION	
V.P. Finance & Operations	Kate Wheat
Business Mgr.	Betty Trickett
Accts. Payable Asst.	Luanne Rocklewitz
Accts. Receivable Asst.	Wendy Ho
ADVERTISING	
National Account Mgr.	Dwight Schwab
National Account Mgr.	Tami Brenton
Advertising Coord.	Shaun Hooper
Associate Publisher	Michael Wiener

M&T Publishing, Inc.

Chairman of the Board	Otmar Weber
Director	C.F. Von Quadt
President & Publisher	Laird Foshay
V.P. of Publishing	William P. Howard

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

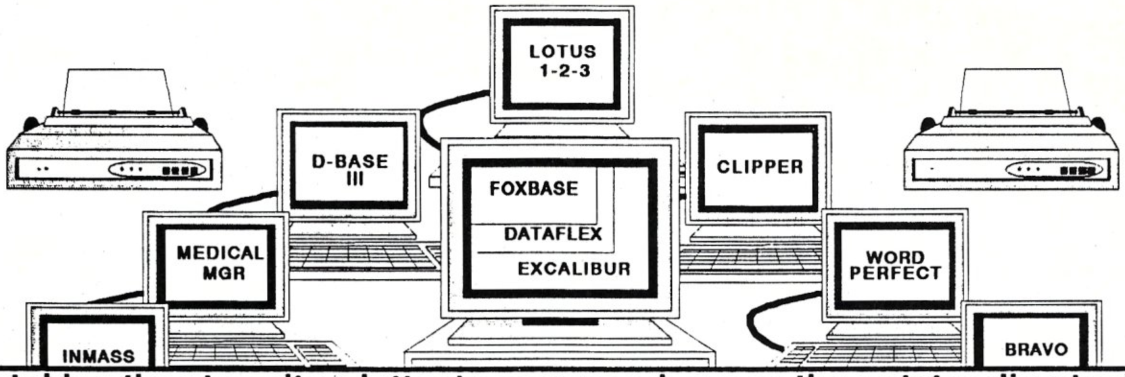
Article Submission: If you have a specific area of expertise or interest and would like to contribute, please write Micro/Systems Journal, P.O. Box 1192, Mountainside, NJ 07092; (201) 522-9347, or contact M&T Publishing, Inc., 501 Galveston Drive, Redwood City, CA 94063; (415) 366-3600. Please do not submit articles without first contacting the editors.

Correspondence: Please send letters to the editor to Micro/Systems Journal, 501 Galveston Drive, Redwood City, CA 94063. Other editorial correspondence may also be directed to P.O. Box 1192, Mountainside, NJ 07092. The editors may also be reached via MCI Mail (SLIBES or MSJ).

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

Back Issues: Available through M&T Books, 501 Galveston Drive, Redwood City, CA 94063 or by calling toll-free 800-533-4372 (in CA 800-356-2002). Price: 1 issue $5.00, 2-5 issues $4.50 each, 6 or more issues $4.00 each.

AS A VAR, YOU HAVE TO MAKE THE SYSTEM WORK.



"...I am taking time to write a letter to commend you on the outstanding technical support that your company has provided for Micromation, Inc.

On many occasions, I have called and have been put straight through or have been promptly called back. The tech support group has been very patient and helpful. When they didn't have an answer for one of my problems, they would create the problem there and work with me and my people until they had the answer. This kind of service I have received from every one of the tech support people in your company.."

Steve Durbin, Micromation, Inc.

The Buck Stops Here!

As a VAR or systems integrator, you know that the operating system software is only half of the package you need. The other half is the technical support team to work with you.

Concurrent Controls provides enhanced Concurrent Dos operating systems for one to nineteen users. Equally important, CCI provides responsive and professional support oriented just to VARs, system integrators and developers.

Support engineers who are trained in systems integration.

Support engineers who understand the inner workings of the operating system.

Support engineers who work daily with a wide range of vertical market packages and PC-DOS software on many different types of hardware.

Call us today for more details about our operating systems and comprehensive support.

CONCURRENT CONTROLS, INC.
925A Linden Avenue
South San Francisco, CA 94080
(415) 873-6240 FAX (415) 873-6091

CIRCLE 53 ON READER SERVICE CARD

News & Views

by Sol Libes

Random Rumors & Gossip

There are rumors that IBM will open up the OS/2 Extended Edition specifications, at least partially, to allow other communications hardware vendors to support EE's Communication Manager module. It is expected that IBM will supply enough information to enable device drivers to be written.

Intel has confirmed that it will introduce a 33-MHz version of its 80386, 32-bit microprocessor early next year.

Apple Computer is rumored to be readying a new version of its venerable old Apple II for introduction early next year. (And to think that Apple seriously considered stopping production of this cash cow when they were introducing the Mac . . . boy would they have lost a bundle.)

Engineering samples of 4-Mbit dynamic RAM chips have been shipped to key customers by **Siemens**, **Toshiba**, **Hitachi**, and **Texas Instruments**. Production is expected to start late next year. However, the 1-Mbit chips are expected to dominate the PC market for several years yet. **NEC** is also sampling the first 4-Mbit (512 × 8) EPROM.

High-Speed Modem Prices Down, Compatibility Up

Fastcomm Communications of Reston, Virginia, has introduced an $899, V.32-compatible, 9600-baud modem employing MNP Class 5 compression for an effective throughput of 17,000+ bps. Competing units with this capability typically sell for almost twice this price. This pricing is even less than most of the non-compatible, 9600-baud modems.

This reflects the competitive nature of the high-speed modem marketplace and improvements in manufacturing technology. It also indicates that standards compatibility is finally coming to the low end of the high-speed modem marketplace.

PS/2 Clones. . . Where are You?

Tandy began shipping the first PS/2 clone (using the MicroChannel Architecture) in July. As reported last month, Tandy sources indicated that sales of these units have been disappointing. **Dell Computer**, which had announced a 286-based Model 50/60 compatible a week before Tandy, has canceled plans to produce the announced MCA products. Instead, Dell has introduced several new systems using the old AT bus architecture.

IBM claims to have already made two million PS/2 machines. The majority, however, are Model 25 and 30 machines using the 8086 processor and the AT bus. The Models 50 and 60, 286-based systems using the MCA turned in poor performance figures compared to AT-compatible 286 clones and were recently upgraded. The Models 70 and 80 are 386-based machines. The Model 70 is a hot desktop unit that IBM did not start shipping until September. The Model 80 is a floor unit intended for server and multiuser applications.

The PS/2 machines have appealed almost exclusively to IBM's traditional customers. The rest of the market has moved even more aggressively to 286 and 386 AT clones, which provide a better price/performance factor. The AT/XT clone makers are enjoying their best year so far as IBM's market share keeps dropping. IBM's PS/2 technology licensing fees have dampened AT/XT clone maker interest in PS/2.

Virtually all major U.S. and Far East manufacturers have agreed to pay IBM royalties on PS/2 compatibles they make. A small number of clone makers are expected to show PS/2, MCA-based machines at the November Comdex show, and to begin shipments next year. A number of Far East clone makers are also expected to show PS/2 prototypes at the show. Although Dell Computer canceled its 286-based Model 50/60 compatible, it is rumored to be readying a 386SX-based MCA machine. And, chip vendors are introducing more PS/2 chip sets, indicating that they are gambling on growing interest by clone makers.

Incidentally, IBM has published a new PS/2 MCA technical reference manual (publication number 68x2330; $125). It covers all current models except the new 25-MHz Model 70 (a supplement is promised for this machine). To order call: 800-IBM-PCTB.

Intel Sampling 80486

Intel Corporation has reportedly shipped early samples of its new 80486, 32-bit microprocessor chip to **IBM** and **Compaq Computer Corporation**, its two largest customers (see "News and Views," *Micro/Systems*, August 1988).

Intel is boasting that the 486 "is a supercharger that will propel [PC] architecture into the workstation and minicomputer market." The chip will be "backwards compatible" with the 386, and Intel is promising that its processing speed will be four times faster than that of the 386. Although the initial version will be rated for a 25-MHz clock speed, subsequent releases are expected to operate at 33 MHz and 40 MHz.

Intel is expected to formally announce the device and provide technical data before year-end. Production is expected to begin late next year. The chip and support chips are expected to be considerably more expensive than early versions of the 386 components that operated at only 16 MHz. Thus we can expect 486-based systems to be in the $10,000 to $30,000 price bracket. The market for these systems will most likely be limited largely to LAN servers, multiuser systems, and high-performance workstations. The 486 is not expected to migrate down to PCs until 1991 at the earliest.

Feds Opt For UNIX

With a computer and communications budget of almost $10 billion for this year alone, the federal government has a lot of say in the future direction of computer systems. They are concerned with software portability and interoperability and have decided to standardize on the UNIX operating system. UNIX has the further asset of being largely an open system that is not processor dependent, and hence it encourages competition.

The government's National Bureau of Standards is nearing completion of work to make the POSIX UNIX application program interface standard a requirement for future government procurement bids. Apple Computer has already indicated that the next release of its A/UX implementation of UNIX for the Macintosh II will comply with the POSIX standard, and will be based on the Massachusetts Institute of Technology's X-Window System release 11, version 2.

Quotation of the Month

"In big corporations, OS/2 is going to be a minor portion of the market, maybe 25 to 40 percent, but in terms of the total PC-compatibles market, my feeling is it's going to be less than 5 percent.": Ed Juge, Director of Market Planning, Tandy Corporation, Fort Worth, Texas. □

Expand Your LAN

...With NPC's Family of Gateways

Network Products offers a family of network communications gateways, from high-speed direct connection to low-speed modem pooling. Expand your LAN to include asynchronous and packet-level hosts, Packet Data Networks, and remote asynchronous computers.

All of NPC's gateway products use NCSI, a versatile client interface program. Use the same popular terminal emulators with every member of NPC's gateway family, including:

ACS²
High-speed Asynchronous Gateway

XCS²
X.25 Packet Assembler/ Disassembler

NMP
Background Modem Server

NPC's gateways are available for NetWare/IPX or NetBIOS networks.

See us at Networld '88, Booth 880, where we will demonstrate our gateway family and the **T1B**, a T-1 Bridge for NetWare LANs.

NETWORK PRODUCTS CORPORATION

1111 South Arroyo Parkway, Suite 450, Pasadena, CA 91105 USA Telephone (818) 441-6504 Telex 910-240-7227

NetWare is a trademark of Novell, Inc.

CIRCLE 121 ON READER SERVICE CARD.

Make your programs millions of times smarter.

More and more, programmers and work-station builders are using DESQview 2.0 as a development tool. The reason is simple. They can create powerful, multitasking solutions today for the millions of DOS PCs in use today. Solutions comparable to those promised for tomorrow by OS/2.

Some of the applications under development right now using DESQview 2.0 API Tools: CAD, medical systems, insurance, 3270 mainframe communications, network management, real estate, typesetting, point of sale, education, commodity trading, stock trading and online voting.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get access to the powerful capabilities built into DESQview--multitasking, window-ing, intertask comunications, mailboxes, shared programs, memory management, mousing, data transfer, menu-building and context sensitive help.

Bells and Whistles

A program taking advantage of the DESQview 2.0 API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask commu-nications to rapidly exchange data between programs, share common code and data; or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

80386 Power

80386 programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview–side by side with other 80386 and DOS programs. The breakthroughs that make this possible: DOS Extenders from PharLap Software and AI Architects and DESQview support of these DOS extenders.

DESQview Developer Support

So if you are a developer, looking to create programs with mainframe capabilities, but wanting to sell into the existing base of millions of DOS PCs, we can be very helpful. We offer DESQview API Tools for assembler or C programmers: a debugger, a Reference Manual. And lots more tools on the way.

Plus, we offer DESQview API Developers Conferences. Come learn about the DESQview 2.0 API and 80386 DOS Extenders. Meet 80386 experts as well as those smart people who are creating DESQview 2.0 API workstation solutions.

For more information call or write us.

Bringing New Power to DOS.
DESQview 2.0 API Toolkit.

Quarterdeck

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405
(213) 392 9851

CIRCLE 86 ON READER SERVICE CARD.

You can deliver it before 199~~1~~2!

New Version 2.2
Compatible with DOS 4.0
Uses less memory, runs faster, and adds communications support

It's coming. An era of more powerful PCs. Easier to use PCs. With graphics and character-based programs working side by side. Talking to each other. Multitasking. Windowing. Menuing. Mousing. Letting your customers get their work done easier and faster.

Sell it all now.

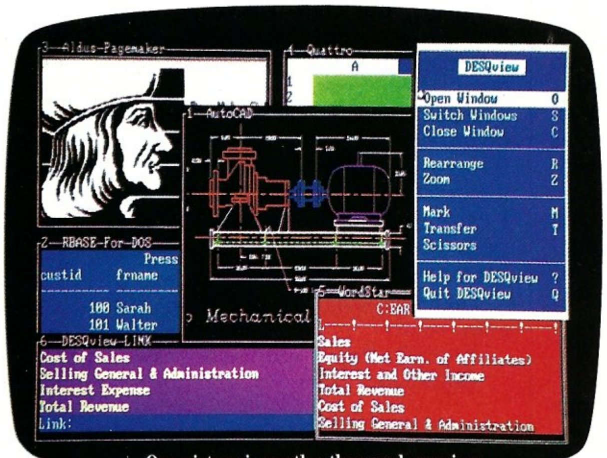
DESQview™ is the operating environment that gives DOS the capabilities of OS/2.™ And it lets your customers, with their trusty 8088, 8086, 80286, or 80386 PCs, leap to the productivity of the next generation. For not much money. And without throwing out their favorite software.

Add DESQview to your customers' PCs and it quickly finds their programs and lists them on menus. So they can just point to the program, using keyboard or mouse, to start it up. DESQview knows where that program lives. And what command loads it.

For those who have trouble remembering DOS commands, it adds menus to DOS. It even lets them sort files and mark specific files to be copied, backed-up, or deleted— all without having to leave the program they're in.

Best of all, DESQview accomplishes all this with a substantial speed advantage over any alternative environment.

For programmers, DESQview's API, with its strengths in inter-task communications and multi-tasking, brings a quick and easy way to adapt to the future. With the API's mailboxes and shared programs, programmers are able to design programs running on DOS with capabilities like those of OS/2.



```
                Press
custid      frname
------      ------
      100 Sarah
      101 Walter
```

```
Cost of Sales
Selling General & Administration
Interest Expense
Total Revenue
Link:
```

```
C:EAR
L----!-----!-----!-----!-----!-----!-----!
Sales
Equity (Net Earn. of Affiliates)
Interest and Other Income
Total Revenue
Cost of Sales
Selling General & Administration
```

One picture is worth a thousand promises.

Multitasking beyond 640K.

When your customers want to use several programs together, they don't have to leave their current program. Just open the next program. View programs in windows or full screen. Open more programs than they have memory for. And multitask them. In 640K. Or if they own a special EMS 4.0 or EEMS memory board, or a 386 PC, DESQview lets them break through the DOS 640K barrier and multitask. For instance, they can start 1-2-3 calculating and tell Paradox to print mailing labels while they're writing a report in Word Perfect, or laying out a newsletter in Ventura Publisher, or designing a building in AutoCAD.

DESQview even allows them to transfer text, numbers, and fields of information between programs.

Fulfill the 386 promise.

For 80836 PC users, DESQview becomes a 386 control program when used in conjunction with Quarterdeck's Expanded Memory Manager (QEMM)-386—giving faster multitasking as well as virtual windowing support.

And when you use DESQview on an IBM PS/2™ Model 50 or 60 with QEMM-50/60 and the IBM Memory Expansion Option, DESQview gives multitasking beyond 640K.

Experts are voting for DESQview. And over a million users, too.

If all of this sounds like promises you've been hearing for future systems, then you can understand why so many VARs and system integrators have chosen DESQview.

PC MAGAZINE EDITOR'S CHOICE NOV. 24, 1987

INFO WORLD 1986 AND 1987 PRODUCT OF THE YEAR

And why *PC Magazine* gave DESQview its Editor's Choice Award for "The Best Alternative to OS/2," why readers of *InfoWorld* twice voted DESQview "Product of the Year" why, by popular vote at Comdex Fall for two years in a row, DESQview was voted "Best PC Environment" in *PC Tech Journal's* Systems Builder Contest.

DESQview lets you sell it all now.

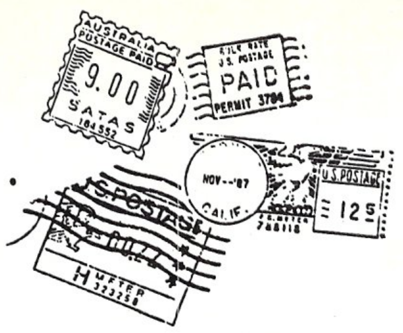
DESQVIEW SYSTEM REQUIREMENTS:

IBM Personal Computer and 100% compatibles (with 8086, 8088, 80286, or 80386 processors) with monochrome or color display; IBM Personal System/2• Memory: 640K recommended; for DESQview itself 0-145K• Expanded Memory (Optional): expanded memory boards compatible with the Intel AboveBoard; enhanced expanded memory boards compatible with the AST RAMpage; EMS 4.0 expanded memory boards•Disk: two diskette drives or one diskette drive and a hard disk•Graphics Card (Optional): Hercules, IBM Color/Graphics (CGA). IBM Enhanced Graphics (EGA), IBM Personal System/2 Advanced Graphics (VGA)• Mouse (Optional): Mouse Systems, Microsoft and compatibles• Modem for Auto-Dialer (Optional): Hayes or compatible• Operating System: PC-DOS 2.0-4.0; MS-DOS 2.0-4.0• Software: Most PC-DOS and MS-DOS application programs; programs specific to Microsoft Windows 1.03-2.1, GEM 1.1-3.0, IBM TopView 1.1• Media: DESQview 2.0 is available on either 5-1/4" or 3-1/2" floppy diskette.

Quarterdeck
150 Pico Blvd., Santa Monica, CA 90405
(213) 392-9851 Fax: (213) 399-3802

CIRCLE 101 ON READER SERVICE CARD.

There Is Mail . . .



In Praise of C

Dear Editor:

I have purchased several issues of *Micro/Systems* this year, and I have found "The C Forum" to be the most consistently interesting column in the magazine. In fact, I would be interested in acquiring columns from back issues.

I also have some observations on the July column ("Something For Everyone," *Micro/Systems*, July 1988). First, the suggestion by Roger Hayes to write conditional expressions with the constant first is a nice idea, but it doesn't completely solve the problem. You've still got to worry about possibly omitting an equal sign (=) in such expressions as

```
if (j==k)
```

I think a better solution is the one offered by compilers that have a compile-time option that causes warnings to be issued for assignment expressions in conditional statements. If you really want an assignment expression, the way to code around this is to write

```
if ((j=k) := 0)
```

rather than

```
if (j=k)
```

Functionally, these two statements are equivalent, but the extra verbosity provides some assurance that things will go the way you want them to.

Second, concerning the code submitted by Dave Luke that counts the number of bits in an *int*, I think that it would be better to initialize the unsigned *int* to 1 and to do left shifts rather than right shifts. This avoids the problem of trying to perform an unsigned divide, which on some machines takes more effort than a simple multiply.

As a further comment, would it be better to shift by more than one bit? As is, the code is quite general and will work for any size *int*, but, for example, is it proper to assume that *ints* will always be a multiple of two bits? Or four? I know of some machines that use 36-bit *ints*, in which case eight won't work. Of course, that probably counts as an over-optimization. After all, how many times in a program should you need to calculate the number of bits in an *int*?

Finally, the code listed as "Best One Liner" in the April issue ("The International Obfuscated C Code Contest," *Micro/Systems*, April 1988) appears to require a UNIX system. As I do not have access to one, I can't run the program. Even trying to compile it generated all kinds of errors. Since I can't find out for myself (as suggested by the author), would you tell me what it prints and why?

That's all for now. Keep up the good work.

Franklin Pierce Johnston
Columbus, Ohio

From the author: Thank you very much for your nice compliment about "The C Forum." I am putting them all together and will publish them as a book in the near future.

Your comments on the July column are apt, except I do not understand why you suggest a compiler might use division to implement a left shift.

As for the question regarding "The Best One Liner," I showed the output produced by the "obfuscated" code in the April "C Forum" column in the May "C Forum" column. I included a detailed explanation of that particular program. Rather than repeat it, let me point out that the critical difference in compiling programs under UNIX is that UNIX C compilers automatically define the symbol "unix" as 1. To get it to compile on your system, try adding the following line to the program:

```
#define unix 1
```

Don Libes

Fine Tuning 8086 Optimization

To the Editor,

I have been programming in 80x86 assembly language for several years and recently I have been optimizing assembly code. I therefore read with great interest Kevin Parker's article in the August 1988 issue, "Programming the 8086/8088 to Perform."

Although the article was very well written, I discovered several discrepancies and I also noticed that some key points were missed.

One recurring problem with the article is the way Mr. Parker counts clock cycles. In the text and accompanying examples, he lists the internal machine clock cycles for instruction clock cycles that the CPU uses to execute the instruction. The clock cycles required to fetch the instruction from memory are ignored. This oversight results in inaccurate comparisons.

Let's demonstrate this by examining Examples 1 and 2. We are told that the code in Example 1 executes in 35 clock cycles. What isn't mentioned is the additional 48 clock cycles that are required to fetch the 12 bytes from memory. The true clock cycle figure should be 83. In Example 2 we should add the 32 clock cycles required to fetch the 8 bytes. With these corrections, Example 1 seems to be 9 percent slower than Example 2. If we move away from the drawing board and put the two code fragments to the test, we find that this is close to correct. I timed 19,660,500 iterations of both examples on an 8-MHz XT clone. Example 1 took 4 minutes, 21 seconds; Example 2 took 4 minutes, 6 seconds. The result shows that Example 1 is 6 percent slower than Example 2.

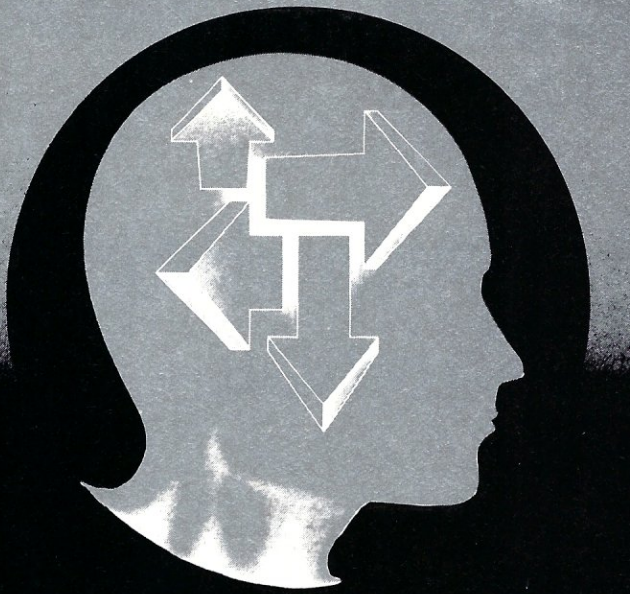
The point demonstrated in Examples 5 & 6 is correct, however the clock counts are wrong. Mr. Parker listed clock counts for MOVSB instructions, not MOVSW. Example 5 should read 4 bytes/104 clocks, and Example 6 should read 5 bytes/115 clocks. If we include the clock cycles for instruction fetches, Example 5 takes 120 clocks and Example 6 takes 135 clocks. On paper, this calculates to an 11 percent difference, the same as Mr. Parker's results. Once again, I timed the code fragments, this time with 8,388,480 iterations. Example 5 took 1 minute, 54 seconds, and Example 6 took 2 minutes, 4 seconds. The result is that Example 6 is 8 percent slower than Example 5.

I realize that it may seem that I am splitting hairs, but the point that I'm trying to make is that the instruction fetches must be considered in code timing. The best way to determine which way is fastest is to put both methods to a timing test. This will reveal timing errors that may have been missed on paper.

Mr. Parker demonstrates that word-

(. . . continued on page 68)

The Evolution Continues...



Line Up With 1dir+ 3.0

TOP OF THE LINE Bourbaki is proud to announce the next generation of DOS shells - **Wonder Plus 3.0.** Considered the most configurable shell available, version **3.0** breaks new ground with major enhancements to its already comprehensive hard disk management and custom menuing capabilities. To name a few - *Automated Command & Menu Generator, Dedicated Mouse support, Two Directory display, Enhanced Editor, Self-Running Tutorial . . .*

"What more could a user want from a shell"
Jack Stone - *DOS Faces, Personalities* - Federal Computer Week - 2/8/88

". . . nearly a 'must have' for experienced users"
The Editors - *Menu Maker Utilities* - InfoWorld 7/18/88

And they were talking about version **2.0**.

THE DIVIDING LINE **Wonder Plus** grows with YOU. Start off simple and add sophistication and complexity as you gain experience. If you're already "advanced," **Wonder Plus** will make you "expert."

Wonder Plus lives to be **Customized.** It is designed to be configured to suit YOUR needs and applications. Using things like the unique **Directory Personality** feature, you can set your system up to "**customize itself**" as you change from one application directory to another.

Our **Tech Support** is **"The Best in the Business."** When you buy **Wonder Plus**, you buy into Bourbaki's support. Our philosophy is, "You have the responsibility to ask for help, and we have the responsibility to provide it." Satisfied users are our greatest asset.

THE BOTTOM LINE Don't settle for less than an integrated system that looks, feels, and does what you want. If you're the type who is always looking for better, faster, easier . . . ways to do things, you owe it to yourself to take a look at **Wonder Plus.** If you are currently using a "DOS shell," you'll see what the Best really means.

Wonder Plus runs on all IBM PC, AT, 3270/AT, PS/2, and compatible machines. It is also compatible with most networks.

Wonder Plus requires DOS 2.XX, or higher, 256K minimum available memory (can be run from EMS), and a hard disk.

Call or write for more information on **Wonder Plus,** the add-on **Muscle Language** module and the **S.F.E. Utilities. ALL 1dir & 1dir+** Users are eligible to **Upgrade** to **3.0.**

Take advantage of our special introductory price of $76.00 (+ $5.00 S & H) until 12/31/88.

To order, call 1-800-BUY-1DIR or (208) 342-5849 or write Bourbaki Inc., P.O. Box 2867, 615 W. Hays, Boise, Id 83701. *We offer a 30 Day Money Back Guarantee.*

WONDER PLUS

Bourbaki Inc.
rEvolutionary Software

CIRCLE 70 ON READER SERVICE CARD.

by Don Libes

Unraveling Threads

This column discusses *threads*, a word you may have seen in OS/2 advertisements and wondered about. Threads are not something new, but few people know about them. They are very elegant and represent a powerful concept that is available in any multiprocessing system. I will introduce them by telling you about a program I recently wrote.

This program would detect when my console was idle for several minutes. It would then paint images on the screen one at a time. At the rate of one per second, this was a very effective way of telling someone that my console was turned on without letting the screen burn in or using up very much CPU time.

My first cut at writing this program was something like the following:

```
int maxpictures = 0;

main()
{
   /* read in all the pictures */
   dir = opendir(PICTURE_DIR);
   while (read_next_picture()) ;

   /* now display them, one per */
   /*second until user presses */
   /*a key */
   while (TRUE) {
      if (key_pressed()) exit(0);
      draw_picture(pictures[
         random(maxpictures)]);
      sleep(1);
   }
}

int    /* returns FALSE if no more */
       /* pictures to read */
read_next_picture()
{
   picture = readdir(dir);
   if (NULL == picture)
      return(FALSE);
   pictures[maxpictures++] =
      read_picture(picture);
   return(TRUE);
}
```

Don Libes is a computer scientist working on artificial intelligence in robot control systems in the Washington, D.C., area. He is also the author of Life With UNIX, *published by Prentice-Hall.*

You can imagine what all of these routines did by reading their names.

This program worked nicely for a while. But one day it stopped working. At least, it *seemed* to have stopped working. When I ran it, it just sat there; the screen didn't change.

After a little investigation, I discovered that the program was actually still running correctly. It just wasn't making it to the display part of the program as quickly as it used to. It seems that someone on our system had received a supply of several hundred pictures from another site, and added them to our local picture directory. The directory had only had 20 pictures in it the week before, and the program took about five seconds to load them all in and begin displaying them. But now it took much longer — about five minutes to even begin displaying the first picture.

At first I was a little annoyed with the person who added all the pictures. But I realized that my anger was misdirected. After all, it was inevitable that more pictures would be added. Indeed, we ourselves were guilty of doing that, albeit at a much slower rate. However, the real culprit was the program itself. I resolved to rewrite the program so it would run more quickly.

Unfortunately, I found there was little I could do to speed up the picture-reading routine or the loop that read the pictures in. So, what I did instead was to rewrite the main routine as follows:

```
dir = opendir(PICTURE_DIR);
read_next_picture();
read_next_picture();
read_next_picture();

while (TRUE) {
   if (key_pressed()) exit(0);
   read_next_picture();
   draw_picture(pictures[
      random(maxpictures)]);
   sleep(1);
}
```

read_next_picture() would read in the next picture and update *maxpictures*. I called it several times before the loop, just so the program wouldn't always have to start with the same picture.

Notice that *maxpictures* was shared by the two routines. The routines were also sharing control, although as I've sketched it here, *read_next_picture()* was clearly subservient to *main()*. Ideally, I would have liked *read_next_picture()* to run at the same time as the drawing routine.

Both of my solutions shared control rather crudely. Because of that, the program was prohibited from reading faster than one picture per second, even if it were possible. And if it took longer than a second to read a picture, the screen updating would have slowed down correspondingly. Similarly, the keyboard was only being checked at the same interval.

We could rewrite this program in several ways. For example, we could have *main()* check after each picture read, whether a second has passed (or is likely to before the next picture is read), and if so, display a new picture.

And while a second doesn't seem that long a period to wait to get control back to the user, we would really like to check for keystrokes more frequently. In fact, the way the program is written, we are going to be waiting one second or longer, however much time it takes to read in a picture, which is actually close to one second on my system. So another possibility is to read in parts of a picture, intermittently checking for keystrokes.

This is a rather trivial application, and yet making it work intuitively is hard. The suggestions I have made all sound incredibly kludgy. The problem is that we only have one *thread of control*. In a single-threaded program like the two above, only one function can be in execution at a time.

Rewritten Using Threads

With a system like OS/2 which supports multiple threads, our original program becomes much simpler. First we create individual functions to perform the unrelated tasks.

```
read_all_pictures() •
{
   dir = opendir(PICTURE_DIR);
   while (read_next_picture()) ;
}

display_all_pictures()
{
   while (TRUE) {
      draw_picture(pictures[
         random(maxpictures)]);
      sleep(1);
   }
}

listen()
{
   getchar();
   exit(0); /* terminate process */
}
```

The only remaining problem is to get them all running at the same time. To do that, we write *main()* as follows:

```
main()
{
   createthread(
      read_all_pictures);
   createthread(
      display_all_pictures);
   createthread(listen);
}
```

createthread() takes a pointer to a function and calls it. Then, without waiting for it to return, *createthread()* itself returns! It's as if there was one program before the function call and two after. In reality, there are indeed two programs, but they share the same address space and symbol table. Thus, in the above example, we can share *maxpictures* between the two programs simply by using the same name.

When programs exist in the same address space like this, we call them *threads*, as in "threads of control." Here, the program has been separated into threads. The first one reads in the pictures. The second one displays them. The third one waits for a keystroke to terminate the program.

Notice how simple the subroutines are. The thread that reads in pictures is just a loop. It doesn't wait for anyone else. As soon as it finishes reading one picture, it can begin reading the next. The second thread is just as simple. It chooses a new picture, displays it, and goes to sleep for one second.

The third thread is amazingly trivial now. It just uses *getchar()* to wait for a character to be pressed. No interrupts, signal handlers, or I/O kludges need to be dealt with (which I passed over in my first solution). And it works much better than before. The program will terminate as soon as a key is pressed, rather than waiting up to a second. *exit()* is defined to terminate the process —this includes all the threads.

All programs start out single-threaded. The thread is whatever is defined by *main()*. In this case, *main()* created three new threads. For an instant, then, there were four threads running, but then *main()* terminated. The *main()* thread terminates just like any other thread — the thread goes away and leaves the remaining threads in a process in execution. In a similar way, the picture-reading thread will go away after it has read all the pictures.

In my second non-threaded attempt at the program, *read_next_picture()* continued to be called, even after all the pictures had been read. Clearly, the threaded solution is much cleaner, more efficient, and simpler to understand and code.

More Details About Threads

Don't confuse threads with processes. A process may have one thread or many. But each process has a different address space, while all the threads in a single process share the same address space. Some people refer to threads as *lightweight processes* or *tasks*.

Threads are not defined simply by subroutines. For example, it is possible to have two threads executing the same subroutine. What really defines a thread is a set of register values (including a PC). As threads are given a chance to run, the system saves the registers of one thread and restores the registers of another. However, the process's address map registers are not changed.

The register values of a thread include a stack pointer. Each thread has its own stack. When the system restores the register values from a new thread, the stack pointer is changed to point to the new thread's stack.

SERIOUS PROGRAMMERS LOOK FOR THE

PROGRAMMER'S POWER PACK

... the key to finding ther programs and utilities that they need to get ahead in their work.

PROGRAMMER'S POWER PACK is the one card deck that no serious programmer can live without...and it's free!

Look for the PROGRAMMER'S POWER PACK in your mailbox...it's coming soon!

If this makes you think of *setjmp( )* and *longjmp( )*, you have the right idea. *setjmp( )* and *longjmp( )* save and restore the registers in the same way. The only difference is that they have to be called explicitly by the user. When the operating system supports threads, it automatically saves and restores the registers as threads are given or relieved of control. Another difference is that *setjmp( )* and *longjmp( )* don't save all the registers, as I discussed in an earlier column ("C Forum," *Micro/Systems*, January 1988).

Threads In The Real World

Threads are not just a creation of OS/2. Many other systems support them, such as Mach, Xinu, Ada, and PolyForth. It is also possible to layer threads on top of any multiprocessing system, such as UNIX. Threads are conceptually handled strictly as multitasking between parts of one process, so there is no reason why the operating system has to know about it.

However, some implementations of threads have mixed the ideas of processes and threads a little, leading to some confusion. For example, OS/2 schedules threads, not processes. Thus, a process may get more of a slice of the processor by breaking the task into many threads.

It will take you a little while to become comfortable with multithreaded processes. In general, you should avoid them unless it is absolutely clear (as in the example above) that they are necessary. One of the drawbacks to threads is that they can be very difficult to write correctly and debug. Here's why.

Drawbacks Of Threads

Any programs that update shared data structures have to use some sort of locking in order to maintain integrity. For example, if two threads are incrementing the same variable, one thread could be temporarily suspended after loading the variable and before incrementing it. Meanwhile, the second thread could load and increment the variable. Then the first thread resumes adding one to the *old* value and saving it. The final value only reflects the operations performed by the first thread, and is thus referred to as the *lost update problem.*

Another problem is illustrated in my example. The careful reader will notice that it is possible for *display_all_pictures( )* to attempt to display a picture before any have been read in! Try correcting the program so this can't happen.

When processes manipulate shared resources, they go through the operating system. The operating system protects processes against lost updates. However, threads aren't protected against each other, and the thread programmer must take responsibility for this. Any system providing threads also provides a way of synchronizing thread control, such as monitors or semaphores. I don't have room to discuss that in this column, but there are many books on the subject of concurrent programming. A comprehensive text is *Concurrent Programming*, edited by Narain Gehani and Andrew McGettrick (Addison-Wesley, 1986).

Threads are not just a creation of OS/2. Many other systems support them, such as Mach, Xinu, Ada, and PolyForth.

More Thread Calls

All I've shown so far is how to create threads with *createthread( )*. I have actually simplified things a bit. For example, OS/2 calls this function *DosCreateThread( )* and requires a second argument that is a block of storage to use for stack space. The priority of the thread is the same as the priority of the calling thread.

Xinu, on the other hand, does things slightly differently. The Xinu *create( )* call creates a suspended thread. Then you call *resume( )* to start it running. Xinu also has slightly different parameters. For example, its second argument is the size of the stack rather than the address of the stack. Additional arguments include priority and optional arguments to be passed to the thread when it starts running.

I find Xinu threads to be slightly more elegant and orthogonal compared to OS/2 threads, although the differences really are quite minimal. The book *The Programmer's Essential OS/2 Handbook* by David Cortesi (M&T Books, 1988) does a nice job describing all the OS/2 thread calls along with other calls related to asynchronous processing.

There are only a couple of other thread functions. Threads can implicitly terminate by *return*, or they can be explicitly killed. A thread can also be suspended or resumed.

There also are functions to manipulate thread priorities. Typically, one function will get the priority of a thread, and another will set it. It is easy to think of a reason why you would want to prioritize threads. For example, a multithreaded editor might have a thread reading and echoing keystrokes, another maintaining a window of files in the current directory, and a third doing garbage collection for other threads. The keyboard and screen updating thread would have highest priority; the thread maintaining the list of files intermediate priority; and the garbage collection the lowest priority, effectively running only when nothing else in the program is running (or can run).

Threads do not exist in DOS, but then neither do processes so it is not surprising. I mentioned earlier that any multiprocessing system can support threads. UNIX, for example, has no thread calls built into the operating system, but it is easy to write your own.

I recently wanted to use threads on UNIX, so I ported Xinu to it. Xinu was about 900 lines of C code plus 6 lines of assembler. The assembler was necessary to save and restore the registers. No changes were necessary to the C compiler or libraries. If you are interested in this, you can read the paper "Multiple Programs in One UNIX Process," (*;login:*, July/August 1987). Xinu itself is described in *Operating System Design, The Xinu Approach* by Douglas Comer (Prentice-Hall, 1984). It is a wonderfully written book —a real gem in the field of operating systems.

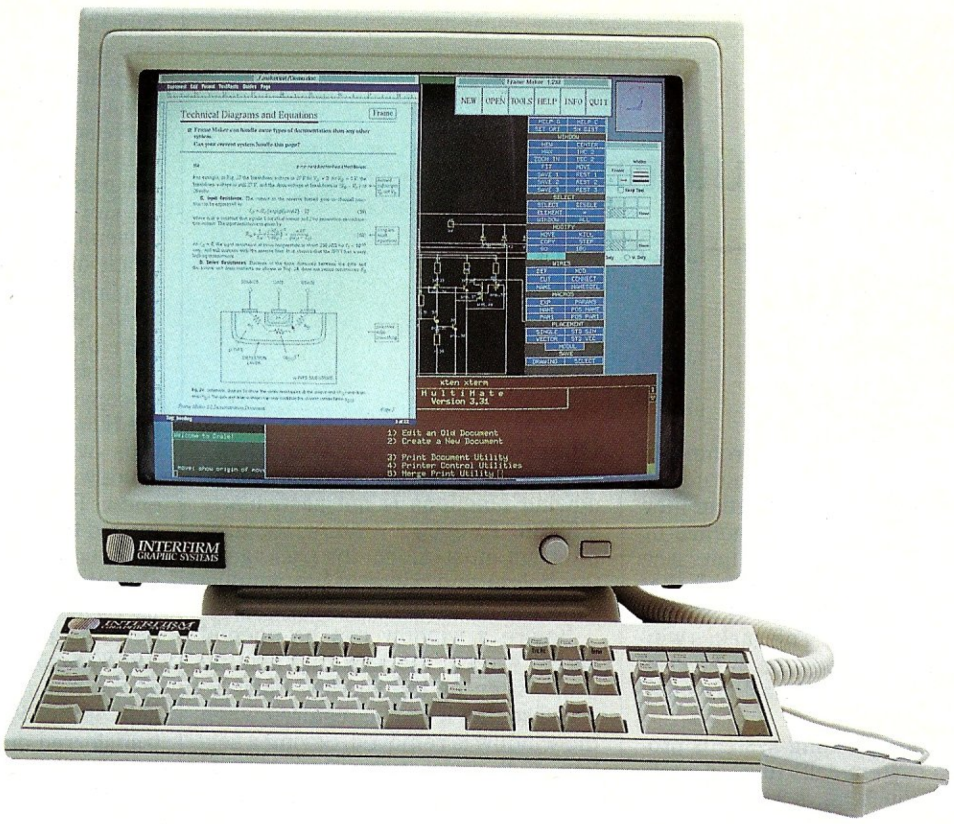
Conclusion

When used appropriately, threads can dramatically simplify programs that would otherwise be arcane or ugly. On the other hand, threads have drawbacks and pitfalls of their own, such as requiring the programmer to deal with synchronizing access to shared data structures.

I encourage you to become as familiar with threads as you are with processes. I guarantee you will find them handy. Threads are an important tool in the repertoire of an applications programmer on a multitasking system. □

Did you find this article particularly useful? Circle number 1 on the reader service card.

Get a window on the future . . .



```
1) Edit an Old Document
2) Create a New Document

3) Print Document Utility
4) Printer Control Utilities
5) Merge Print Utility
```

without giving up the present

At last, you can have the benefits of UNIX™ V.3 and high resolution graphics through X-Windows—without sacrificing the investment you've made in your DOS based 80386™ system.

The Easy Way to X-Windows

With the XSUBSYSTEM™ from Interfirm Graphic Systems, you'll have everything you need to convert your 80386-based machine into a powerful UNIX-based graphics workstation. Suddenly, you'll have all of the advantages of a multi-user, multi-tasking system—plus the incredible convenience of DOS programs runnng side-by-side with UNIX.

The XSUBSYSTEM Includes:

- 19″ High resolution color or monochrome display
- Graphics controller
- UNIX System V.3 from Interactive Systems Corp.
- The X Window System software
- Optional hard disk with all software pre-installed
- Optional TCP/IP Ethernet controller and software

The Future of 386 Based Graphics

The XSUBSYSTEM gives you the best of both worlds - you get to keep both your hardware and your DOS software while you upgrade to a UNIX V.3/X-Windows System.

For example, with Interfirm's XSUBSYSTEM, you can take data created in a DOS application and paste it into a document that you are creating with the Frame Maker™ desktop publishing system.

It's the most sensible way we know to give your 386 based system the power of a true graphics workstation - at a sliver of the price.

At Interfirm Graphic Systems, we are pioneering UNIX-based high-performance graphics, utilizing the X Window System and other emerging non-proprietary standards. When you look to the future of desktop computing, look to Interfirm.

INTERFIRM
GRAPHIC SYSTEMS
3940 Freedom Circle
Santa Clara, CA 95054
(408) 970-7070
800-338-4513

CIRCLE 109 ON READER SERVICE CARD

by Stephen Randy Davis

Creating Multitasking Windows —Part 1

Windowed user environments are becoming more and more popular, whether they are used for displaying graphics or for displaying text. Such an interface allows the user to view the task that is currently active as well as what is waiting in the queue. This makes windowed environments the preferred user interface for multitasking environments.

Windowed operating systems provide extra system calls to user applications for opening, scrolling, moving, and closing windows on the display. Programmers need only worry about the names and arguments of the windowing routines and not about how such magic is actually performed. Of course, DOS is not a windowed environment and therefore provides none of these display tools to the applications programmer. If a DOS application wants to open a window to display some piece of information, such as a menu or a help screen, the application must manipulate the window.

This does not mean, however, that the applications programmer must write the windowing routines. There are windowing packages available for Turbo Pascal and other languages, several of which we have examined in this column in the past. These packages provide the typical window manipulation routines that are otherwise missing under DOS. Of course, windowing only applies within the current application. Other programs that the current application might execute cannot be compelled to operate in the boundaries of a window without some help from the hardware. However, clever use of windows can still enhance the "look and feel" of an application.

Stephen Randy Davis is a technical editor for Micro/Systems Journal *and a programmer for a defense contractor in Greenville, Texas.*

Opening and Closing Windows

Implementing a set of simple windowing functions is not particularly difficult. To open a window, the user program must inform the package of the height and width of the window as well as its location on the screen.

Before a window can be opened on the screen, the area that is to be overwritten by the window must be saved so that it can be restored when the window is removed. Saving a section of the screen requires a good understanding of how screen memory is arranged. Visualize display memory as a large matrix. Each row of the video display contains 80 integers corresponding to the 80 columns across the screen. Integers are required because each location must contain both a character and a foreground/background color, each of which occupy a byte. A separate row appears for each of the 25 display lines. A window appears as a smaller matrix cut out of the middle of the screen matrix. Its configuration is the same except that, rather than remaining fixed, the number of rows and columns is specified by the applications program when the window is opened.

The memory used to store the section of display occupied by the window is allocated off the heap. It is convenient to store the address of this array along with the other window-descriptive data in a structure that we will call a *WindowNode*. The *WindowNode* contains all the information necessary to describe the window (except for the contents of the window itself, which are on the video display). The *WindowNode* can then be passed along to each windowing routine as a form of "calling card" to introduce the window.

Writing to a window is also quite simple. Generally the library provides *WindowWrite* and *WindowWriteLn* (or some other cleverly named functions) which write to the currently active window. Coordinates are taken from the upper left corner of the window. Character strings that threaten to run over the edge of the window and spill into the windows behind it must be truncated to fit. Scrolling the window is handled exactly like scrolling the entire screen —each line must be copied to the line above it and the last line must be cleared (see "Turbo Pascal Corner," *Micro/Systems Journal*, November/December 1986, for an example of direct screen manipulation). Application programs may *WindowWrite*, viewing their window as if it were the entire screen.

Closing a window is just a matter of copying the screen data saved in the *WindowNode* back onto the screen, thus overwriting whatever data the window contained and erasing its presence. The *WindowNode* is then returned to the heap. Another common operation, panning a window, is probably the most difficult operation to perform. This requires careful transfer of data both on the screen and in the *WindowNode* as background data at the boundary is copied into one side of the *WindowNode* while being transferred out the other side.

An example of such a simple set of windowing routines is shown in Listing 1. The functions *WindowOpen, WindowWrite, WindowWriteLn*, and *WindowClose* are all present. Each operates in the manner previously described, more or less. *WindowOpen* is passed the location and size of the window to be opened and returns the address of a *WindowNode*. This *WindowNode* must be passed to the other procedures that manipulate the window. Once the *WindowNode* has been closed, the pointer is no longer accessible.

Multitasking Tends to Shatter Windows

Notice that, in order to save room, the example program leaves out two features that are sometimes present in windowing packages. First, these routines only work in 80 column text mode. The older 40 column modes are generally not seen anymore, so this is not a limitation. However, the 43-line and 50-line modes of the EGA and VGA adapters are often supported, respectively. Second, this version of *WindowOpen* does not draw a box around the window. To do so, the special box characters in the upper 128 characters of the IBM display set are written at the boundary character positions.

From a multitasking standpoint, however, this set of windowing routines has a problem much larger than these minor annoyances —a problem that it shares with the vast majority of windowing packages available on the market: Only the "current window" is accessible. That is, if a window is opened by a task on top of another window, the background window cannot be accessed in any way until the top window is removed. As an experiment, modify the main program to scroll window 1 while window 2 is still open. (One way is to remove the call *WindowClose(W2)*.) Scrolling the background window causes the overlapping part of the foreground window to scroll as well. Data written to the background window appears right over the foreground window and the results are gibberish.

In a single-tasking environment, where windows are probably just drop-down menus, this limitation is understandable. In a multitasking situation, however, this is a serious problem. When a task opens a window, it does not know if it has opened its window over another window. More to the point, it does not know when another task opens a window over its own. A task should be able to continue accessing its window unimpeded, regardless of what other tasks do with their windows.

Clearly this approach is not acceptable for windows used in multitasking applications. In my next column, I will examine an approach I developed to handle this problem quite nicely. □

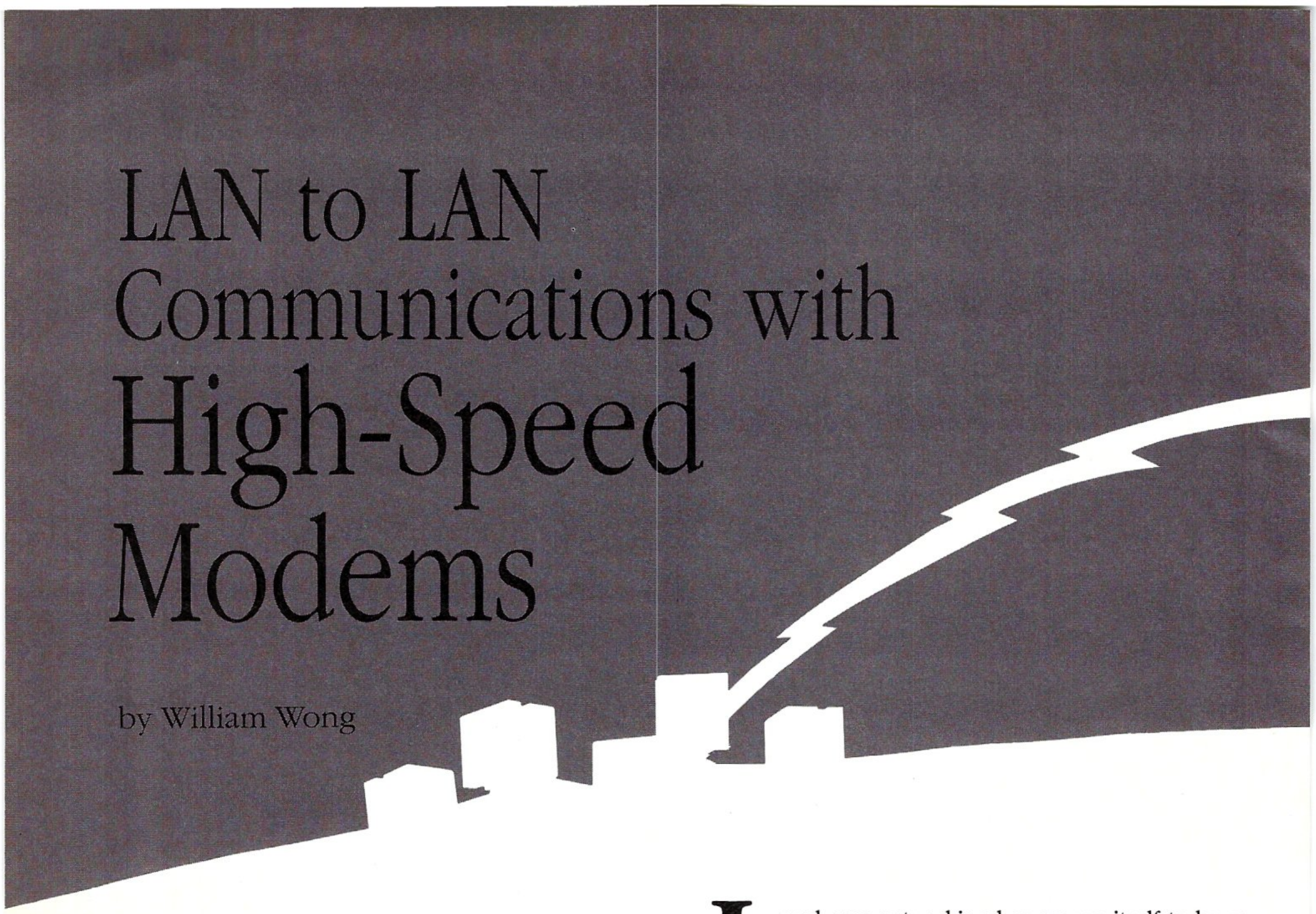
Did you find this article particularly useful? Circle number 2 on the reader service card.

Listing 1. A simple, single-tasking windowing package.

```
{Simple Windowing Package -
 Provide simple WindowOpen, WindowWrite, WindowWriteLn, and
 WindowClose procedures to allow user applications to readily
 open and manipulate single windows (text mode only).  Note in
 test code that while it is possible to open multiple windows,
 only topmost window may be scrolled or written to.
}

type
    Display = array [0..24] of array [0..79] of integer;
    DataBlock = array [0..1] of integer;
    Strng = string [80];
    WindowNodePtr = ^WindowNode;
    WindowNode = record
                   DeltaX, DeltaY     : integer;
                   WinX, WinY         : integer;
                   Color              : integer;
                   CurrentX, CurrentY : integer;
                   Data               : ^DataBlock
                 end;

const
    cga = $b800;                    {offset of CGA/EGA}
    mono= $b000;                    {          monochrome screen}
var
    Screen : Display absolute cga:0; {currently set for CGA/EGA}

Function WindowOpen (X, Y, Width, Height, Attr : integer) : WindowNodePtr;
var
    Pntr : WindowNodePtr;
    Size : integer;
    i, j : integer;
begin
    New (Pntr);
    with Pntr^ do
    begin
         {Save data into window}
         DeltaX    := X;
         DeltaY    := Y;
         WinX      := Width;
         WinY      := Height;
         Color     := Attr;
         CurrentX  := 0;     {set cursor to beginning of window}
         CurrentY  := 0;

         {Store off section of screen into windownode}
         Size := WinY * WinX * 2;
         GetMem (Data, Size);

         for i := 0 to WinY - 1 do
              for j := 0 to WinX - 1 do
              begin
                   Data^[i * WinX + j] := screen [DeltaY + i][DeltaX + j];
                   screen [DeltaY + i][DeltaX + j] := Color + Integer (' ')
              end
     end;
     WindowOpen := Pntr
end;

Procedure WindowClose (W : WindowNodePtr);
var
     i, j : integer;
begin
     with W^ do
     begin
         {put original screen back}
         for i := 0 to WinY - 1 do
             for j := 0 to WinX - 1 do
                  screen [DeltaY + i][DeltaX + j] := Data^[i * Winx + j];
         {now release memory to heap}
         FreeMem (Data, WinX * WinY * 2);
     end;
     Dispose (W)
end;

 Procedure WindowScroll (W : WindowNodePtr; Count : integer);
 var
     Index, XIndex, YIndex : integer;
 begin
     With W^ do
     begin
     CurrentX := 0;                  {carriage return}
     CurrentY := CurrentY + Count;   {line feed(s)}
     if CurrentY >= WinY then        {if beyond window's end...}
     begin                           {...scroll window's contents}
          Count := CurrentY - WinY + 1;
          CurrentY := WinY - 1;
          for index := 0 to (WinY-Count) - 1 do
          begin
               YIndex := Index + DeltaY;
               for XIndex := DeltaX to DeltaX + WinX - 1 do
                    screen [YIndex][XIndex]
                         := screen [YIndex + Count][XIndex]
          end;

          for index := 1 to Count do {blank bottom line(s)}
          begin
               YIndex := DeltaY + (WinY - Index);
               for XIndex := DeltaX to DeltaX + WinX - 1 do
                    screen [YIndex][XIndex] := Color + Integer(' ')
          end
     end
   end
end;

Procedure WindowWrite (W : WindowNodePtr; OutStrng : strng);
var
    i      : integer;
    Count  : byte absolute OutStrng;
    Offset : integer;
    Value  : integer;
begin
    with W^ do
         for i := 1 to count do
              if CurrentX < WinX then
              begin
                   value := Color + Integer(outstrng [i]);
                   screen [DeltaY + CurrentY][DeltaX + CurrentX] := value;
                   CurrentX := CurrentX + 1
              end
end;

Procedure WindowWriteLn (W : WindowNodePtr; Outstrng : strng);
begin
    WindowWrite (W, Outstrng);
    WindowScroll (W, 1);
end;

{give above routines a few trial calls}
Procedure ExerciseW (W : WindowNodePtr);
var
     i : integer;
begin
     for i := 1 to 100 do
     begin
          WindowWrite   (W, 'this is a silly string');
          WindowWriteLn (W, ' continuation');
          WindowWriteLn (W, 'another string')
     end
end;

var
     W1, W2 : WindowNodePtr;
begin
     W1 := WindowOpen (10, 10, 50, 10, $1300);
     ExerciseW (W1);
     W2 := WindowOpen (20,  5, 30, 17, $4200);
     ExerciseW (W2);
     WindowClose (W2);
     ExerciseW (W1);
     WindowClose (W1)
end.
```

LAN to LAN Communications with High-Speed Modems

by William Wong



High-speed modems can be a boon to support file transfers between LAN servers.

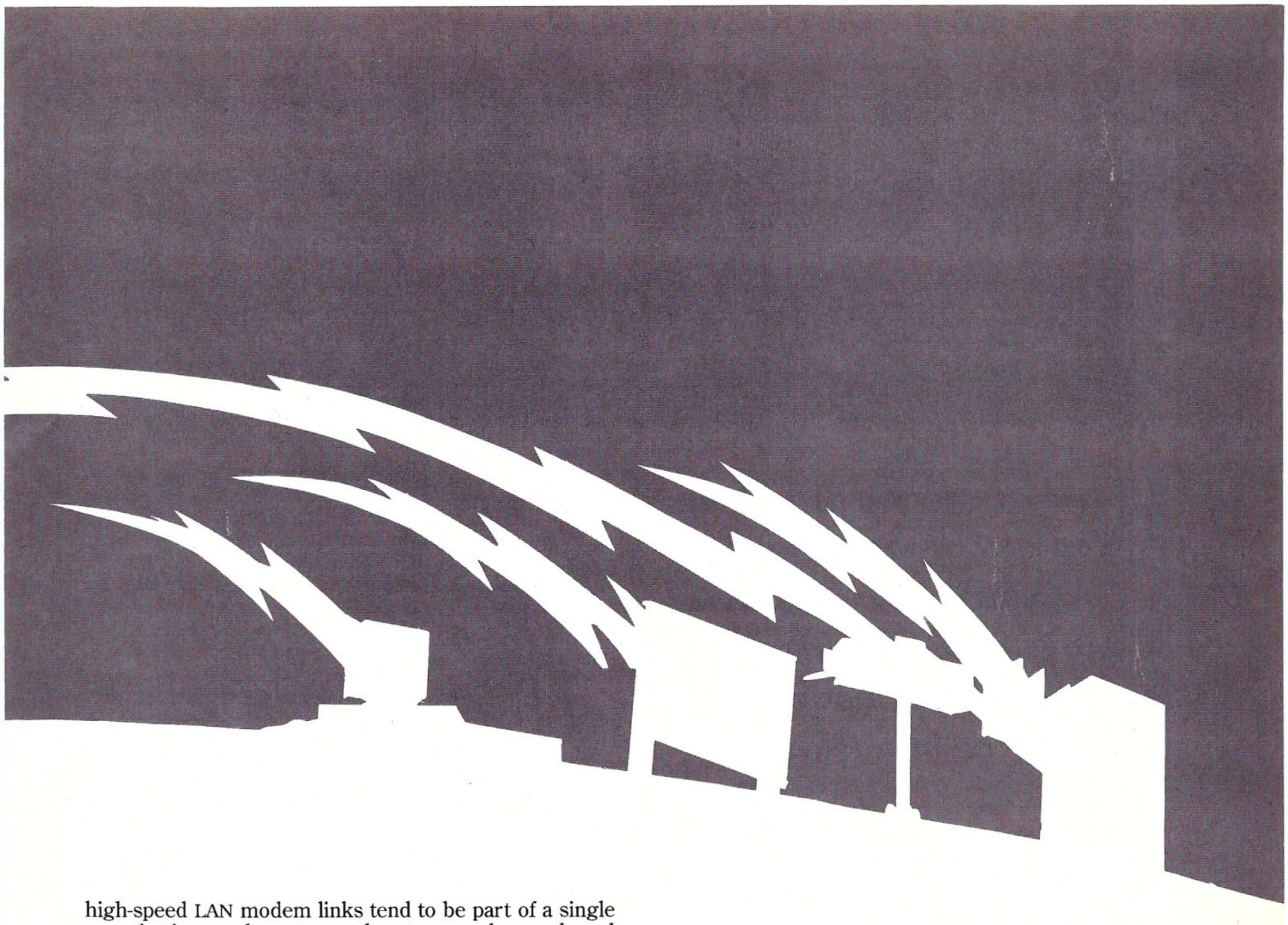
Local area networking has proven itself to be an effective approach to sharing data and computer resources. Now consider the possibilities offered by moving beyond the limits of the LAN and accessing remote resources, such as bulletin boards or other networks. This is where high-speed modem technology can prove to be a cost-effective means of expanding network connectivity.

In fact, a high-speed modem can be more cost effective in a LAN environment because it works as a shared device. It can significantly reduce telephone costs compared to a lower speed modem in many applications. Specifically, modems can be effectively used with LANs in three areas: server-to-server connections, remote workstation-to-server connections, and in modem pools. High-speed modems running at 9600 baud or faster can be used in all three areas with most PC LAN operating systems, including three of the most popular systems discussed here: 3Com 3Plus, Novell NetWare, and Banyan VINES.

Modem support can be made relatively transparent to users in most LAN operating systems if the LAN manager sets up both the LAN and the user modem support. Otherwise, modem operation can be a bit confusing and tedious for most users who tend not to know whether their modem is attached to COM1 or COM2, let alone what the modem switch settings are.

High-speed modems tend to have minimal compatibility in high-speed mode between different vendors. For example, a Telebit Trailblazer Plus will not work with a Hayes V9600 or a U.S. Robotics Courier HST. However,

William Wong is president of Logic Fusion, a systems development firm based in Morrisville, Pennsylvania.



high-speed LAN modem links tend to be part of a single organization, so the same modem type can be purchased for all links.

Compatibility between high-speed modems and LAN operating systems also tends to limit the number of choices. Server-to-server links, for example, tend to be restrictive and hard to customize, as are remote workstation-to-server links. Modem pools, however, tend to be more flexible because of the variety of low-speed modems that can be used in a pool and the ability to customize the modem configuration as needed.

Three Server-to-Server Connections

Server-to-server connections using modems fall into two specific categories: direct server-to-server links and indirect server-to-server links. The latter include links through X.25 networks, as well as vendor-specific products. Direct server-to-server links tend to be more common, and easier to install and maintain. 3Com, Novell, and Banyan all support direct server-to-server links using high-speed modems.

The 3Com Approach

3Com's 3+Route uses normal serial ports for modem support and operates from the server. The high-speed modems supported include Hayes, Microcom, Telebit, and DCA. The catch is that the server's other network interface adapters must be the more expensive, intelligent EtherLink Plus adapters. Leased-line support is also included.

3+Route is based on the Xerox Network Service (XNS) protocols, but it is not compatible with XNS. 3+Route does work with 3Com's bridge product, however, allowing multiple links between servers. 3+Route is also integrated with 3+Mail and NetBIOS support. Mail is automatically exchanged when a 3+Route link is made. Links can be set up for a fixed period of time or until the link is inactive for a set amount of time. This latter feature is very important for temporary links.

3+Route also supports 3+Remote, 3Com's remote workstation-to-server product. This allows a single link to be used for both server-to-server and remote workstation-to-server connections. This can be advantageous on PC servers that can support only COM1 and COM2. Fourteen ports can be supported on 3Com's 3S/400 server.

The NetWare Solution

Novell's Advanced NetWare supports server-to-server links with high-speed modems connected to either COM1 or COM2 for a limited number of connections, or using a Wide-area Network Interface Module (WNIM) which supports up to four serial links running at up to 19,200 baud. The WNIM is an intelligent communications processor with on-board, shared memory. Up to four links can be installed in a server for a maximum of 16 ports. The server can either be a normal NetWare file server or a communication server. The communication server can be a diskless workstation that loads the communication software from a file server, or it can contain its own disk, although this is only used for loading the communication software.

Intelligent communication boards and communication servers are both preferred for high-speed modem links on LANs because of the overhead imposed by a modem link. Other network interface adapters normally handle the exchange of data packets automatically, where serial links using COM1 or COM2 require the server's processor to get involved. This overhead can significantly reduce the performance of the network if the communications support is on a file server. Intelligent communication boards effectively link to any other network interface adapter. The advantage is that the boards are programmable and generally support other modem links, like remote workstations and modem pools.

NetWare modem links are integrated with the NetBIOS support and Novell's MHS mail support system. NetWare links can be setup at specific times for a specified duration, or until the link becomes inactive. Initially, high-speed modems were limited to full duplex V.32 modems, but now NetWare supports asymmetrical and "ping-pong" modems, like the Telebit Trailblazer and Hayes V Series.

Server-to-server links tend to be restrictive and hard to customize, as are remote workstation-to-servers. Modem pools, however, tend to be more flexible.

Linking with VINES

Banyan VINES requires an intelligent communications adapter for both low- and high-speed modem support. These adapters must be placed into a Banyan or PC file server. An independent communication server, like that supported by Novell NetWare, is not supported. Two types of Persyst communications adapters and a Banyan communications adapter are supported. The Persyst boards support two or four high-speed channels. Banyan's Internetwork Communications Adapter (ICA) supports up to six channels with two operating up to 64 kilobits per second (Kbps) using DMA. This very high-speed mode is appropriate for synchronous links that may be provided by direct connections, high-speed synchronous modems, or attachments to T1 multiplexers. Up to four communication adapters of any type can be included in a file server. VINES also supports connections at a predetermined time and can break that connection when it becomes inactive. VINES StreetTalk naming system is automatically updated when two networks are linked together. They also retain the StreetTalk names after the connection is terminated. Mail is automatically exchanged when a link is made as well.

Initially, only Telebit and DCA 9600-baud modems were supported, and the Hayes V Series was added later. Non-intelligent high-speed modems are supported as well. Serial links can also be made using synchronous connections. X.25 support is also available. The 64 Kbps links on the ICA boards are more appropriate for dedicated long-distance connections.

VINES is limited in initiating modem links because it must be set up using the system console. One way around this is to use a remote system console connected to a modem pool port. On the other hand, a VINES 3.0 network can be connected to another VINES 3.0 system but, instead of sharing all resources, the networks only exchange a limited amount of information, such as mail. This can be important for networks where high security must be maintained, but many server-to-server links will be made with a variety of other systems.

The High-Speed Advantage

Server-to-server links can be very cost effective when they utilize a high-speed modem because 9600-baud connections tend to move large amounts of data between servers rapidly. The link also provides a complete set of services spanning two remote networks, which effectively creates one network. High-speed modems also tend to support data compression, which is important since most LAN operating systems do not perform data compression on information moving through the network. Using a modem supporting data compression to copy a text file across a modem link will provide significantly better performance.

High-speed asynchronous modems can also be cost effective in this type of application because they can be used to provide other services, like remote workstation support, or they can be part of a modem pool for dialing out to other services. Most LAN operating systems support the lower speed connections on a high-speed modem when it is used in a modem pool. In many instances, one or two high-speed modems may be all that is required for a modem pool.

Asynchronous high-speed modems are only one alternative to high-speed serial links. Synchronous modems may be more cost-effective in some cases, especially when data rates exceed the performance provided by asynchronous modems. Banyan's 64 Kbps channels are often used with leased lines or T1 multiplexers.

Another alternative to installing modems is not to use a serial connection but rather to tap into the normal network interface adapters, like Ethernet or Token Ring. 3Com's Bridge Communications Division makes bridge units that are transparent to the normal operation of a LAN and simply tap into the Ethernet cable. These are effectively high-speed repeaters.

If you need only minimal E-mail support and limited shared services, take a look at low-speed modems (e.g., 2400 baud). Move up to high-speed modems if more information is to be exchanged. The performance will not be up to what most LAN adapters support (1 to 10 megabits per second). Faster 64 Kbps links should be used for large amounts of information. If you are really serious, look into bridge hardware that taps directly into your network interface.

The advantage of using modems is that connections are made using a dial-up network, and one modem can be used to provide a link to a number of different networks. Also, you only pay for the time connected. The installation cost for a high-speed modem connection normally includes an additional charge for the support software (about $1,000), a pair of high-speed modems (about $1,000 each), and an intelligent communications adapter (about $1,000). For a network that can span a continent, $5,000 is a small amount to pay.

Remote Workstation-to-Server Connections

Remote workstations support is found in almost all high-end LAN operating systems. The remote workstation must have a disk to load the remote workstation software, as well as a serial port (COM1 or COM2) and a modem. Links are normally asynchronous, running as fast as 19,200 bits per second (bps). Connection speeds as low as 1200 bps are practical if network operations are limited to copying small files or exchanging electronic mail. At any speed, only data should be exchanged unless executable files are very small. In general, all support programs should already be available on the remote workstation.

Remote workstations have the advantage of providing complete network services to an off-site user, including access to file servers, print servers, communication servers, and any other type of service that may be available. Access to file and print services may be obvious but communication services may seem strange. One possible configuration would be to have a communication server connected to a mini or mainframe, where it is inappropriate for the user to dial in directly.

3Com 3+Remote, the server support, and 3+Remote PC, the remote workstation support, provide remote workstation access to a 3Com network. 3+Route can also be used at the server end. Most Hayes-compatible modems can be used, along with the high-speed modems supported with 3+Route.

Novell NetWare's remote workstation support is a standard part of Advanced NetWare. The server hardware support can be provided by COM1 or COM2, or a WNIM board.

Banyan VINES' asynchronous dial-in service provides remote workstation support. It is an added option and requires one of the intelligent network communications adapters discussed earlier.

There is not much to say about the differences between these product since they are functionally identical. All network services are available at a remote workstation, although they run more slowly than on a conventional network workstation.

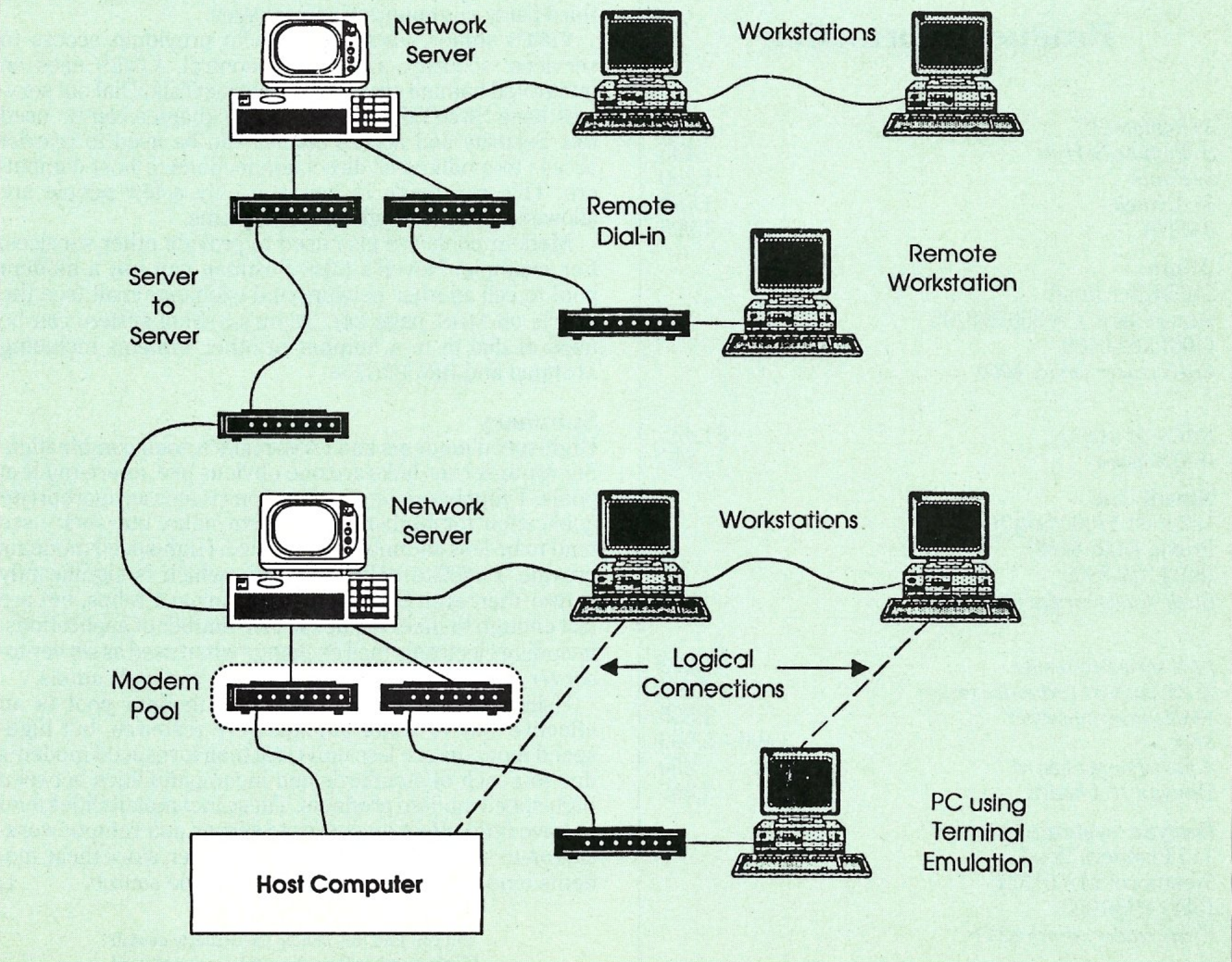
One alternative to using a remote workstation is to use a network workstation as a bulletin board system or to run a remote PC control program, like PC Anywhere from Dynamic Microprocessor Associates. Novell has a special version called NetWare Anywhere, which can also be used with the NetWare modem pool.

Remote PC control programs essentially allow the remote PC to control a network workstation. For most applications, the amount of information sent across the modem link will be less than that sent across the network. The PC control programs normally include file transfer support so information can be exchanged between the remote PC and the network file server.

Modem Pools

Sharing resources and exchanging information are the main reasons for having a network. Modem pools are a handy way to share a modem. Low-speed modems tend to be inexpensive enough that creating a network simply to pool modems is ridiculous. However, even sharing low-speed modems on an existing network can be very cost effective for a number of reasons. First, there is the savings in minimizing the number of modems. Second, it makes it easier to support both the users and the modems. Also, low-speed asynchronous modems are not the only type of modems that can be shared. High-speed asynchronous modems and synchronous modems, which are more

Figure 1. Two networks supporting remote dial-in, a server-to-server modem link, and a modem pool.



expensive than low-speed modems, can also be shared. Synchronous connections are normally made using IBM's SNA or, in a few cases, IBM's Bisynch support. SNA connections can be especially cost effective since all workstations attached to the network can be used to access a mini or mainframe. No special wiring or hardware needs to be added for this connection.

There are some disadvantages to using modem pools with asynchronous modems. One is the amount of time it takes to get a character from the keyboard to the remote computer and back. A network modem pool can add a significant amount of time to this delay because that character must travel through the network twice. However, this delay is only appropriate for full-duplex keyboard operation. Receiving a full screen of information happens almost as fast as on a PC with a modem attached to it since the only apparent delay is associated with the first character in a block. The other disadvantage is the amount of network overhead, the load it imposes on the network, and the impact a heavily loaded network will have on a running session. The overhead is required because the network must include a character in a packet that has the source and destination identifiers as well as command and error detection information. The overhead decreases as the amount of information sent at one time increases.

Synchronous or half-duplex connections have the same disadvantages, but to a much lesser degree since data tend to be sent in blocks. SNA terminal emulations are used in an environment where a screen is received, data is edited locally, and then information is sent back to the host computer as a block.

3Com's 3+Asynch provides asynchronous modem support while 3+SNA provides synchronous support. Terminal emulator programs are provided as part of the support. Modems can be accessed by name or by group.

Novell's NetWare Asynchronous Communication Server (NACS) provides support for an asynchronous modem pool. The NetWare Asynchronous Services Interface (NASI) program provides access to the modem pool. NASI has a large following in third-party communication programs. In fact, NACS comes with a minimal terminal emulation program that uses NASI. Most network users will want to use one of the third party products because they offer additional features, such as scripts and file transfer options.

NACS provide a number of options not found in other products, including automatic disconnect after a set period of inactivity. Also, NASI can support up to nine separate sessions at once, although most applications only support one. However, NACS does not provide any security or access-control options. Essentially, anyone can access a known port from any workstation as long as the port is not already in use.

Banyan VINES provides an asynchronous dial-out option, as well as SNA and Bisynch options. Terminal emulator programs are included. A program interface is available, but it does not have the following that NASI does. The asynchronous terminal emulator provides Kermit and Xmodem file transfer capability along with an XTalk-compatible script language. However, the program does not provide the same variety of features found in most third-party communication packages.

VINES shines when it comes to providing access to services, security, and access control. VINES uses an integrated naming system called StreetTalk. Dial-out services have StreeTalk names, and nicknames can be used too. Security and access control can be used to restrict access to modems or direct connections to host computers. This can come in handy if only a few people are allowed to use the high-speed modems.

Modem pools are also used to provide other services. For example, Novell's MHS Postman can use a modem pool to call another network and exchange mail (see the article on MHS, page 24). 3Com's 3+Mail system can be used to dial in to a number of other systems including MCIMail and IBM PROFS.

Summary

High-speed modems and LANs make a good combination. Server-to-server links are one obvious use, as are modem pools. Tying in remote workstations is also an appropriate application for high-speed modem links, but such uses tend to be less common due to price. High-speed modems provide a 9600-to-19,200 bps link, which is significantly slower than even StarLan, which runs at 1 mbps, but are fast enough to meet the needs of a number of applications, including electronic mail exchange when used as server-to-server and remote workstation-to-server applications.

Using high-speed modems in a modem pool is an effective way to share an expensive resource, but high-speed modems are less universal than low-speed modems due to a lack of standards and incompatibilities between high-speed modem products. These incompatibilities tend to have little effect on server-to-server and remote workstation-to-server configurations, however, since these modems tend to be purchased from a single source. □

Did you find this article particularly useful?
Circle number 3 on the reader service card.

Product Information

3+Remote PC	$295
3+Remote Server	495
3+Route	1,250
3+ASynch	1,795
3+SNA	4,995

3Com
3165 Kifer Road
Santa Clara, CA 95052-8145
(408) 562-6400
Circle reader service #250

NACS (4 users)	$1,495
WNIM board	895

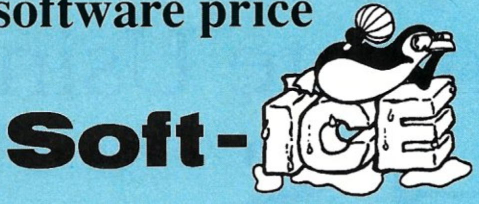
Novell, Inc.
122 East 17000 South
Provo, Utah 84601
(801) 379-5900
Circle reader service #251

LAN server-to-server	$1,000
X.25 support server-to-server	2,495
WAN server-to-server	1,595
SNA	3,490 – 5,990
4-port Persyst board	850
Banyan ICA board	1,495

Banyan Systems Inc.
115 Flanders Road
Westboro, MA 01581
(508) 898-1000
Circle reader service #252

SERIOUS DEBUGGING *at a* REASONABLE PRICE

All the speed and power of a hardware-assisted debugger at a software price



Hardware-level break points
REAL-TIME break points on memory locations, memory ranges, execution, I/O ports, hardware and software interrupts. More powerful break points than ANY software-only debugger on the market. Soft-ICE gives you the power of an in-circuit emulator on your desk.

Break out of hung programs
With a keystroke - no external switch necessary. Even with interrupts disabled.

Breaks the 640K barrier
Soft-ICE uses ZERO bytes of memory in the first 1MB of address space. This is especially useful for those subtle bugs that change when the starting address of your code changes. With Soft-ICE your code executes at the same address whether the debugger is loaded or not.

Works with your favorite debugger
Soft-ICE can be used as a stand-alone debugger or it can add its powerful break points to the software debugger you already use. You can continue to use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated.

Solve tough systems problems too
Soft-ICE is ideal for debugging TSRs, interrupt handlers, self booting programs, DOS loadable device drivers, non-DOS operating systems, and debugging within DOS & BIOS. Soft-ICE is also great for firmware development because Soft-ICE's break points work in ROM.

How Soft-ICE Works
Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses 80386 protected mode features, such as paging, I/O privilege level, and break point registers, to provide real-time hardware-level break points.

"Soft-ICE is a product any MS-DOS developer serious enough to own a 386 machine should have."
Dr. Dobb's Journal — May 1988

Both require 80386 AT compatible or IBM PS/2 Model 80. MagicCV requires at least 384K of extended memory. CodeView is a trademark of Microsoft Corporation.

RUN CODEVIEW IN ONLY 8K!

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 microprocessor to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView using less than 8K of conventional memory on your 80386 PC.

Don't let 640K be your limit!
If you are closing in on the 640K limit and would like the power of CodeView, MagicCV is for you.

Don't let the debugger hide the bug!
Even if you're not closing in on the 640K limit, running CodeView with MagicCV makes your debugging environment much closer to the end user's program environment. You can use CodeView to locate subtle bugs that only occur when there is plenty of free memory, or those difficult bugs that only occur when your program is running with a couple of TSRs loaded.

How MagicCV works
MagicCV uses the 80386 to create a separate virtual machine for CodeView. MagicCV uses between 4K & 8K of conventional memory as a bridge between the DOS environment and CodeView.

MagicCV is easy to use
If you are a CodeView user, you already know how to use MagicCV too. Just type MCV instead of CV; everything else is automatic.

Save $86

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

MagicCV	$199
Soft-ICE	$386

Buy Both and Save $86!

CALL TODAY
(603) 888 - 2386
***or FAX* (603) 888 - 2465**

30 day money-back guarantee
Visa, Master Card and AmEx accepted

NU-MEGA TECHNOLOGIES

P.O. BOX 7607 • NASHUA, NH 03060-7607

CIRCLE 103 ON READER SERVICE CARD.

Developing Applications That Communicate Using MHS

by Steve Hochschild and Carol Kime

The Message Handling Service utility allows network program developers to create applications that truly communicate.

Local Area Networks (LANs) enable applications to access a new level of integration. As LANs continue to grow, internetworks and bridges become an important link in business activities. One area rich with possibilities is the integration of widely dispersed networks that use asynchronous connections to access data and services.

With the exception of several specialized applications, few programs currently offer wide-area communications. Developers who choose to include wide area networking connectivity in their applications have been forced to deal with low-level programming issues, such as routing tables, modem control, and reliable file transfer. The typical design using libraries of communication source code, background tasks, and terminate-and-stay-resident (TSR) programs has significant disadvantages, including high memory consumption, limited portability, and rigid end-user interfaces. Developing applications that pass data over asynchronous telephone lines requires extensive effort and usually results in programs that are difficult to maintain.

Distributed processing and the growing number of desktop and departmental processors present challenging opportunities to developers. How do you use the aggregate processing capability of distributed computers? How do you make the location of computers, peripherals, applications, and data completely transparent to end users? And how do you integrate dissimilar systems and allow them to communicate with each other?

A Communications Engine For Wide Area Connectivity

After DOS was introduced in 1981, developers were able to build applications "on top of" standard file handling utilities. Similarly, the introduction of network operating systems offered developers standard platforms from which to share local resources among a variety of users and applications. These standard Application Programming Interfaces (APIs) free the developer from attending to the low-level functions of moving data and files around in local hardware environments.

One recent development in LAN-based technology is the concept of back-end engines that provide services to numerous, separate front-end applications. For example, there are back-end database engines that manage database access for all of the applications running on a network. There are also back-end communications engines that allow developers to create applications involving wide-area communications, cooperative processing, and application-to-application communications.

The Message Handling Service (MHS) utility developed by Action Technologies of Emeryville, California, is one such communications engine that is available for Novell's NetWare operating system and other DOS 3.X-compatible networks, as well as stand-alone computers. MHS trans-

Steve Hochschild is the MHS Product Manager and Carol Kime is a technical writer for the Novell Development Products Division in Austin, Texas.

ports messages across LANs, internetwork bridges, and asynchronous lines (see Figure 1). MHS provides a common software solution for any application that needs to cooperate, communicate, or coordinate with the same or different application. For maximum flexibility, MHS is typically installed on a dedicated PC for continuous monitoring of the messaging functions. However, MHS can be invoked only as it is needed, as would be done on a stand-alone PC.

MHS Components

The MHS engine consists of a number of components (see Figure 2). The Connectivity Manager queues waiting messages and routes each message using information contained in routing tables on the MHS host computer. If the recipient is located on a LAN, the message is placed in a directory available to the receiving application. Otherwise, the Connectivity Manager relies on the Transport Server to deliver the message. The Transport Server makes the physical connection and transmits the message. A Directory Manager component provides access to the routing tables to update the list of MHS usernames, workgroups, and hosts. And there are two utilities, SEAL and UNSEAL, that a developer can access from an application to send and receive messages.

By separating the application from the message transfer process, MHS adds modularity to the task, which provides a significant benefit to application developers. First, a programmer presents the information to MHS in a structured fashion. Then, after placing a few simple calls to the operating system, MHS handles the rest of the delivery.

MHS can efficiently and reliably transfer information across a network or between networks, taking care of

MHS Standard Message Format

The SEAL.EXE program takes a defined set of data, which is passed in an ASCII file called a Message Control Block (MCB), and produces a message in the Standard Message Format (SMF). Usually, this is the simplest and most reliable way to create MHS messages. But in those cases where the functionality of the MCB is too limiting, the application developer may choose to create the MHS message directly.

The SMF consists of a header, an optional message body, and an optional attachment file. While it is in transit, the SMF also has a 128-byte binary routing envelope, which is constructed from information stored in the header and modified by the Connectivity Manager component at each MHS host as the message moves through the network. The format of the header fields is defined, but its labels and meanings are not. Each application may create new header labels and use them to transport structural information as required. This extensibility permits an application to totally control the contents and structure of the messages it sends and receives. For example, an E-mail application may require additional header fields to keep track of the names of people that have been copied on a particular message since the 18-line MCB does not have the capability to track a list of copied recipients.

The beginning of the header is marked by the header signature, the control codes <^C><LF> (030Ah). A number of header fields follow the header signature. Header fields are five-part structures, with:

- An option specifier to indicate display and processing options for the field;
- A field keyword that identifies the header field type;
- A colon (:) that serves as a keyword delimiter;
- Keyword values that are the variables associated with the field; and
- Header field terminators, which are the control characters <LF> (0Ah), or <CR><LF> (0D0Ah).

The end of the header section is marked by the control characters <LF> (0Ah), or <CR><LF> (0D0Ah).

The minimum fields required to construct a valid SMF header are:

```
<^C><LF>          (Header signature)
Date
From
To
<LF>              (Header terminator)
```

The following fields are conditionally required:

Sender	When compatibility with MHS gateways is required.
Send-to	When compatibility with MHS gateways is required.
Message-id	When reliable tracking of messages is required.
MCB-options	When any delivery or receipt options are not set to their default value.
Error-report	When message is a status notification.
MCB-type	When message body is not plain ASCII text.
Date-delivered	When message is a delivery notification.
Date-received	When message is a receipt notification.
Message-type	When message body is not plain ASCII text.
Attachment-type	When attachment file is to be automatically imported.
Message-encoding	When message body is not encoded with the IBM-PC alphabetic character set.
Message-discard	When the receiving application is to be allowed to discard the message body.
In-reply-to	When the message is to be linked to a previous message.
Application-name	When message originates from an application that is not a user's preferred application, or when the originating user is a user of other applications.

The details of the SMF, along with implementation guidelines for gateway developers, are fully documented in the *MHS System Developers Guide.*

Using MHS in Manufacturing: A Case Study

One application in which MHS is currently being used is in manufacturing. Metatron Corporation, a company in Beaverton, Oregon, that develops shop floor control software, has an application that captures data from bar code readers. This data is then transferred to other manufacturing sites using MHS. Metatron's software package, MetaShop, exports data from its databases in ASCII format, sends this data via MHS to another site at a predetermined time, and then imports it back into the MetaShop application. This allows MetaShop users to share vendor order, timecard, production, and related information among all operation sites.

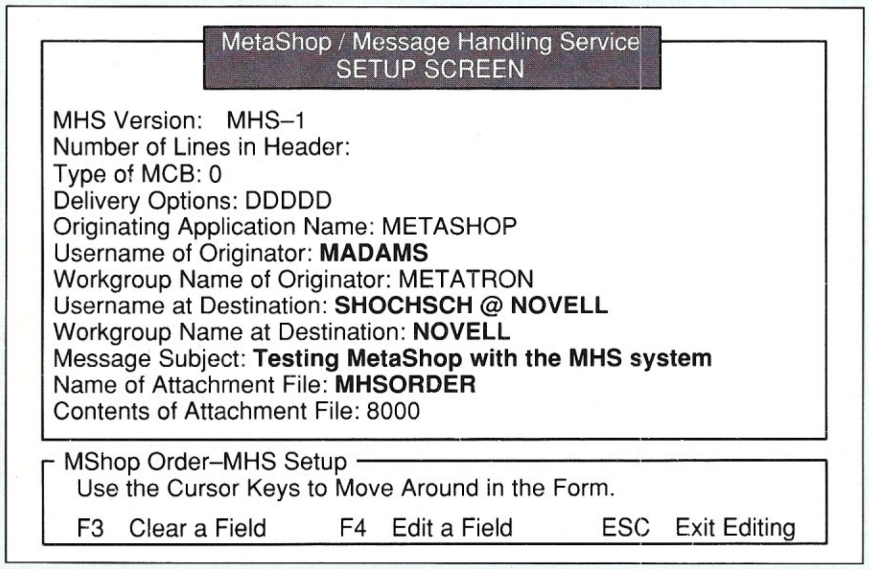
According to Mark Adams, a cofounder of Metatron, the integration of MHS with MetaShop was not difficult. Installing MHS was straightforward, and the Metatron engineers simply used the MCB API to SEAL and UNSEAL messages. The only coding necessary was to generate an application to create the routing header so the user could tell the system where they wanted the message sent (see Screen 1).

The SETUP SCREEN provides default responses for most of the entries; the user only needs to enter his or her name at "Username of Originator," and the "Username at Destination," and "Workgroup Name at Destination." As an option, the user can enter a message subject and specify the name of a file to send with the header, in this case, the file named "MHSORDER."

After the user presses Esc to exit from Screen 1, another menu gives the user the option of proceeding or aborting the operation. If the user chooses to proceed, the MetaShop application executes SEAL, which parses the header and provides an error-level code to MetaShop to indicate whether the header is correct. If there is a problem, MetaShop prompts the user to correct the appropriate header information.

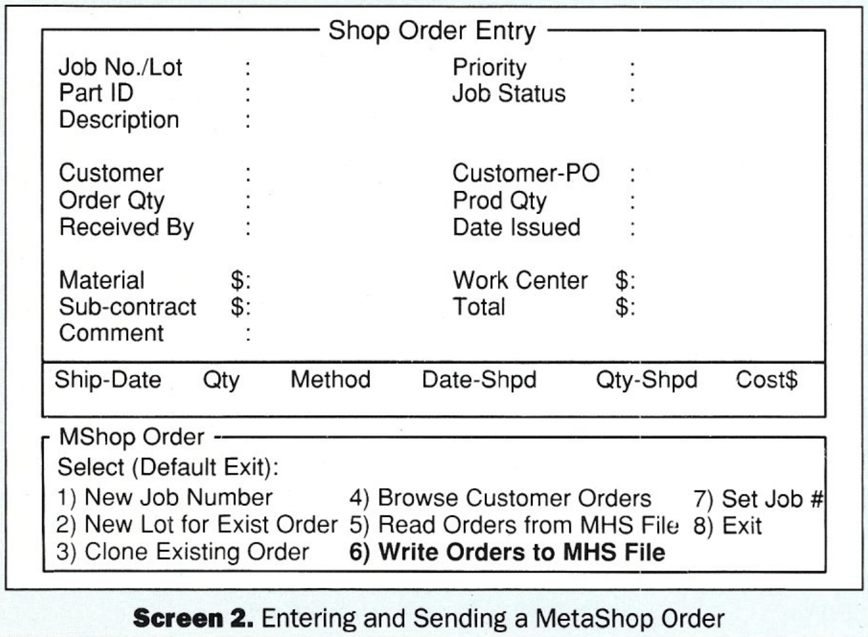
The MetaShop software already had export and import facilities for sharing data bi-directionally on the same LAN, but sharing data with other systems not directly connected to the LAN was improved using MHS. Using the MetaShop Export feature (see Option 6 at bottom of Screen 2), the user makes a menu selection to write the order to a file. MHS then sends the file to the address captured in Screen 1.

"One important area of expansion for the MetaShop/MHS integration is transferring order and status information between customers and vendors," Adams says. "As manufacturers' need for quick and accurate job status information from their vendors increases, more of this kind of information will be transferred via computer. At some point, automatic messages may be generated and sent at specific intervals to keep both customer and vendor aware of job status. The NetWare MHS utility can be a vital link in improving communications between suppliers, manufacturers, and customers."



```
MetaShop / Message Handling Service
SETUP SCREEN

MHS Version:  MHS–1
Number of Lines in Header:
Type of MCB: 0
Delivery Options: DDDDD
Originating Application Name: METASHOP
Username of Originator: MADAMS
Workgroup Name of Originator: METATRON
Username at Destination: SHOCHSCH @ NOVELL
Workgroup Name at Destination: NOVELL
Message Subject: Testing MetaShop with the MHS system
Name of Attachment File: MHSORDER
Contents of Attachment File: 8000

MShop Order–MHS Setup
  Use the Cursor Keys to Move Around in the Form.
  F3  Clear a Field        F4  Edit a Field        ESC  Exit Editing
```

Screen 1. Specifying the Message Address



```
                         Shop Order Entry
Job No./Lot      :                  Priority        :
Part ID          :                  Job Status      :
Description      :

Customer         :                  Customer-PO     :
Order Qty        :                  Prod Qty        :
Received By      :                  Date Issued     :

Material        $:                  Work Center    $:
Sub-contract    $:                  Total          $:
Comment          :

Ship-Date    Qty    Method    Date-Shpd    Qty-Shpd    Cost$

MShop Order
Select (Default Exit):
1) New Job Number            4) Browse Customer Orders     7) Set Job #
2) New Lot for Exist Order   5) Read Orders from MHS File  8) Exit
3) Clone Existing Order      6) Write Orders to MHS File
```

Screen 2. Entering and Sending a MetaShop Order low-level communication tasks such as modem control and transport protocols, determining the appropriate telephone calls to make, managing reliable file transfer, and reporting errors back to the sending application. Solving these problems at the operating system level creates a robust application platform and frees the applications developer to focus on the unique aspects of the application —the features and user interface.

MHS Transports Any Data

Although referred to as messages, the information packages that MHS can transmit include instructions to a receiving process, a database query, a work order, or an executable program. Up to 8,192 printable ASCII characters can be transferred as a message body, and you can optionally attach a variable length file that contains any type of machine readable data. This includes ASCII text, text generated by a word processor, spreadsheet data, database entries or entire databases, executable program code, digitized voice information, or pictorial data. MHS can transfer messages from person-to-person, person-to-process, process-to-person, or process-to-process.

Although the vast majority of electronic communications today is person-to-person, in the next decade more than 90 percent of electronic communication is predicted to be person-to-machine or machine-to-machine. Electronic mail will be a significant enabling technology to distribute applications, particularly between a company and its suppliers and customers. MHS is designed for distributed applications that must cooperate over a wide area network. For example, an application could use MHS to transmit daily transactions, monthly accounting reports, purchase orders, or shipping information to remote sites.

The MHS Technology

MHS uses store-and-forward technology. Because of this, the sending and receiving applications do not maintain an on-line connection as required by some other communication programs. With applications that use on-line connections, the receiving application has to be available to answer the incoming call or the transfer will fail. With store-and-forward technology, the application forwards the message to an MHS host where the message is stored until it is ready to be transported to the receiving application.

In addition, applications that share data through full-time connections are heavy users of operating system and network resources. MHS runs autonomously in an unattended PC and uses network resources only when it is collecting and delivering messages. The store-and-forward technology allows the application to operate with no performance degradation from interaction with off-LAN services and devices.

MHS messages are queued until a trigger is reached; the MHS user can specify the trigger to be a certain number of messages or a certain period of time. When the trigger is reached, MHS examines the message address, determines where the message needs to be transferred, makes the call, initiates the transfer, picks up any waiting messages, and hangs up. The two networks are on-line only for the relatively short period of time needed to transfer any waiting messages, thus saving the communications cost that a full-time connection would require.

MHS Addressing and Routing

Another important trait of MHS is flexible message addressing and routing. An address is the public identity of a user or group of users. A route is the particular wires or path that the network uses to deliver the packet to the address. Previously, application developers were forced to cata-

Figure 1. MHS is available for any DOS 3.x LAN or stand-alone PC

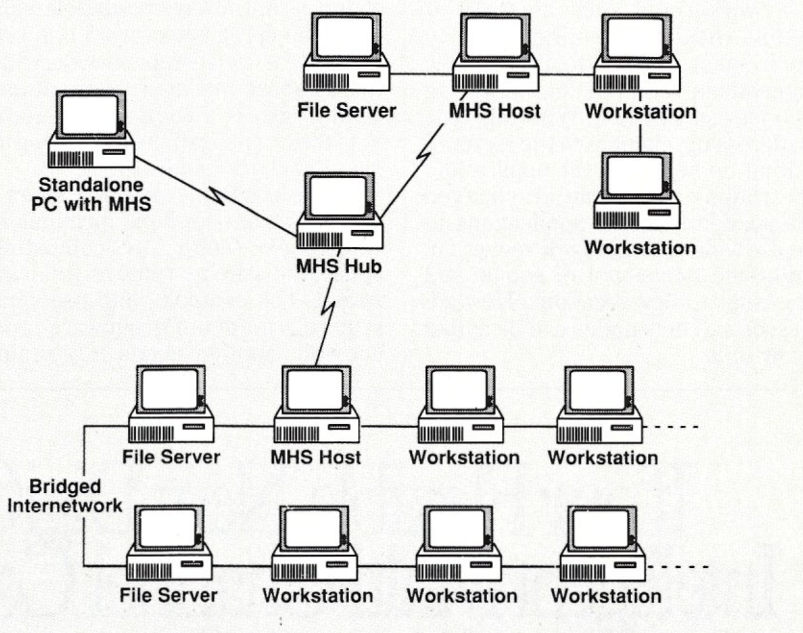
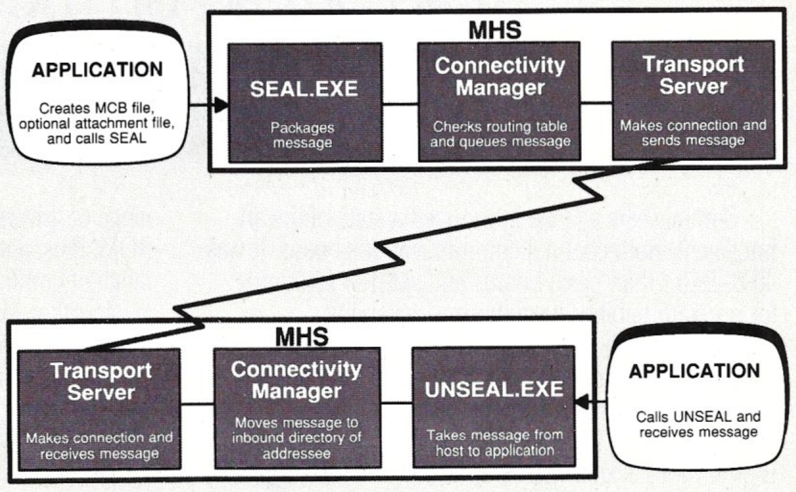


Figure 2. Applications communicate via MHS components



MHS
APPLICATION
Creates MCB file, optional attachment file, and calls SEAL
SEAL.EXE
Packages message
Connectivity Manager
Checks routing table and queues message
Transport Server
Makes connection and sends message
MHS
Transport Server
Makes connection and receives message
Connectivity Manager
Moves message to inbound directory of addressee
UNSEAL.EXE
Takes message from host to application
APPLICATION
Calls UNSEAL and receives message

log the address in the application. Any change in the location of either end meant updating the application, updating the communications software, or both.

MHS provides a simple addressing scheme that covers any type of network architecture and user (human or process). The rules for routing are structured so the system can handle unknown as well as known addresses. They also are flexible enough to permit routing optimization, depending on the most cost-effective network setup and the various levels of phone service.

MHS separates the address of the user from the routing required to direct a message to that user. An MHS message is addressed to a user at a workgroup, such as JDOE @ SALES. MHS uses routing tables to resolve that address to the specific connections required for delivering the message to JDOE. With this type of addressing, changes to the required routing do not affect the application. The routing tables can be changed independently of the applications to handle wide area network reconfiguration and movement of people and processes to new locations. New addresses and new nodes can be added at any time.

MHS Applications Communicate with One Another

MHS gives developers a standard set of messaging functions that make transparent communications between systems a matter of a simple function call. There are many advantages to developing and using applications that depend on a common software platform to perform a specific task. For the developer, the advantage is not having to deal with the development issues that surround that task. For users, it presents flexibility in choosing a front-end that is best suited to their needs.

In order to maintain compatibility, many organizations specify what applications employees can use. Standardization on a specific application is not always a suitable solution. The spreadsheet from one vendor may be used company-wide, while the database manager from another vendor also is a company standard. Yet, these applications may need to pass data back and forth.

These problems multiply when a network spans multiple locations or companies. Using MHS-compatible applications helps remove this constraint. For example, one user may require a premium groupware product while another needs only a minimal E-mail utility. The two users could each select the MHS-compatible application that best meets their individual needs, but these different applications could still communicate with each other.

Developing MHS-Compatible Applications

An MHS message has three parts:

1. A header;
2. Body text;
3. An associated parcel, which can be a file in any format.

MHS requires that the message be transported in a particular format, called the Standard Message Format (SMF). The SMF consists of a binary envelope and a header that contains information, such as who is sending the message, where the message is going, if there is an attachment file, and what delivery options to use. The SMF header can be extended to include special labels that are defined to meet specific needs. Applications use the SMF to send messages to multiple recipients and gain access to special message header and MHS services, such as information about copies sent to others, messages that have been resent, and response deadlines.

If our Flexible New DigiChannel/Xi Intelligent Multi-Channel Communications Board doesn't fit your application, read our ad on the right.

Several years ago, we introduced a state-of-the-art intelligent multi-channel communications board. It was 400% faster than basic boards and set new standards for programmability, flexibility and reliability.

Now, with the new **DigiCHANNEL/Xi**, we've advanced the state-of-the-art again. A 16 MHz 80186 co-processor makes it twice as fast. And plug-in I/O Mate modules make it even more flexible. So does its full range of drivers, and optional 512K of RAM and 128K of ROM. Plus, a synchronous channel opens a new range of communications options.

In other words, if your multi-user or multi-channel applications call for the last word in speed, configuration flexibility and programmability, call us about the new **DigiCHANNEL/Xi**.

If not, read our ad on the right.

DigiBoard
Plugging you into tomorrow

Call 1-800-344-4273. In Minnesota, (612) 922-8055.

CIRCLE 41 ON READER SERVICE CARD

An applications developer can directly create a packet in the SMF (see the accompanying box), or can use a simple Message Control Block (MCB) API.

The MCB Interface requires the developer to create an 18-line ASCII file and execute the MHS SEAL and UNSEAL utilities. SEAL converts the MCB to SMF for delivery by MHS; UNSEAL converts the SMF packet back to an MCB and places it in a specific directory available to the receiving application.

The MCB Interface

To send a message using the MCB interface, an application:

1. Defines the Message Control Block header (an 18-line ASCII file);
2. Creates the optional attachment file;
3. Executes SEAL.EXE.

In addition, SEAL generates return codes and files to which the application must be able to interpret and respond. The return codes are transmitted in the MCB file to report successful transfers or problems that must be resolved. Table 1 summarizes the MCB header.

Information for some of the lines in the MCB header is variable, such as the name of the sender. Other information is produced by MHS, such as the date and time that the message was sealed. On line 4 of the MCB, the application developer controls how the message is delivered. With yes (Y) and no (N) responses in the first three columns, the application developer specifies whether MHS should notify the originator when the message is unsealed, notify the originator if the message cannot be delivered, and whether to return the contents of a message that is not deliverable. The other two delivery options that can be specified on line 4 let the

Table 1. Summary of the MCB header

Line	Description	Example
1	MHS version used by the submitting application (currently must be MHS-1)	MHS-1
2	Number of lines in header (currently must be 18)	18
3	Type of MCB: 0=message MCB; 1=status MCB (applications must enter 0)	0
4	Delivery options requested by originating application	NNYNP
5	Originating application name	X-TEST
6	Originator's username	CLEE.EMAIL
7	Originator's workgroup (or host) name	NOVELL
8	Recipient's username	JSMITH.EMAIL
9	Recipient's workgroup (or host) name	ATI
10	Message subject	SALES FIGURES
11	Name of attachment file	SALESDAT.DOC
12	Unique ID of sealed message (MHS supplies)	.
13	ID of referenced message	.
14	Date and time message was sealed (MHS supplies)	.
15	Date and time message was unsealed (MHS supplies)	.
16	Content and type of attachment file	1
17	Status and error report code (MHS supplies)	.
18	Blank line at end of header	.

If our Economical New DigiChannel/Xe Intelligent Multi-Channel Communications Board doesn't fit your application, read our ad on the left.

Because of our long experience, we provide engineering and support others can't. What's more, we listen. You've asked for a product that does its job — without all the "extra" features you don't need.

That's why we're introducing the **DigiCHANNEL/Xe**. Its 8 MHz 80186 co-processor makes it significantly faster than our former model — yet it costs less. You can still choose between 4 or 8 asynchronous channels, and 64K of SRAM is more than enough for such straight-forward multi-user and multi-channel applications as data collection and office automation. So if the new **DigiCHANNEL/Xe** sounds like a good fit, call us for more information.

If not, read our ad on the left.

DigiBoard
Plugging you into tomorrow

Call 1-800-344-4273. In Minnesota, (612) 922-8055.

See us at Unix Expo and COMDEX Fall.

CIRCLE 41 ON READER SERVICE CARD

originating application specify the grade of delivery (Normal, Urgent, Off-hours) and whether MHS should deliver the message to a recipient's preferred application if the recipient does not have access to the application specified in the address.

After the application creates the MCB, it issues a call to the MHS utility SEAL.EXE. The syntax for calling SEAL is:

```
SEAL -Ffilename | -Ddirectory_name
[-V]
```

You can specify that you want to seal a particular file, -F*filename*, or all files in a particular directory -D*directory_name*. The optional parameter, -V, causes SEAL to return status information.

SEAL constructs a routing header from the MCB information and stores the message file and the attachment file as a single MHS-formatted file that is ready to be transported. If the recipient of the message is a user of the local MHS host, SEAL immediately delivers the message. Otherwise, SEAL places the message in a subdirectory for the MHS Connectivity Manager to handle.

The receiving application periodically executes the UNSEAL.EXE utility to recover any waiting messages. For example, applications that are used continuously might check for messages every 10 minutes while applications that depend on receiving messages to operate might check for messages only once upon startup. When the application calls UNSEAL, UNSEAL looks in the incoming mesfsage subdirectory and retrieves any messages addressed to the user of that application. If UNSEAL finds a message, it converts the MHS-formatted message back into an MCB and attachment file and places them in a directory that is accessible to the application.

Conclusion

Application developers have an array of tools that provide interprocess communications (IPCs). NetBIOS, IPX/SPX, and Named Pipes are all real-time IPCs. MHS can be thought of as a "real-time enough" method of IPC. Some of the benefits of using MHS include: independence between communicating nodes due to the store-and-forward architecture; independence between the application and the actual message transfer due to flexible addressing and routing methods; straightforward implementation with a simple Message Control Block (MCB) API; unlimited functionality with the extensible Standard Message Format (SMF); any MHS-supporting front-end application can read MHS messages; delivery options and status codes that provide error control; and a common base for communications interconnectivity.

MHS is provided with every purchase of NetWare v2.1, and there are versions of MHS for any DOS 3.x LAN or stand-alone PC. MHS is a powerful enabling technology for software developers concerned with application-to-application messaging. □

Did you find this article particularly useful? Circle number 4 on the reader service card.

NetWare MHS is available free with NetWare 2.1 and above. For other NetWare versions and for stand-alone PCs, MHS is available from authorized Novell resellers and selected mail-order distributors for $100.00. Other network versions are available from Action Technologies, (415) 654-4444. The MHS Interface Guide documents the MCB API and is available from authorized Novell resellers and selected mail-order distributors for $145.00.

OMNIVIEW

ROCK SOLID MULTITASKING!

Packed with the power you need, OMNIVIEW takes you beyond the limits of DOS.

- Time-slices multiple jobs
- Takes full advantage of EMS 4.0
- Optional support for '386 Systems
- Manages megabytes of programs
- Controls resident utilities
- Lets you cut and paste between programs

OMNIVIEW unleashes your PC & converts it into an easy-to-use, multitasking workstation. Whether you have an 80386 or 8088 based computer, OMNIVIEW will squeeze out every ounce of power.

Solid & bulletproof, it's 47K assembly language engine gives you all the power without added overhead or complexity. Just ask the reviewers.*

"Sunny Hill provides true multitasking for most PC applications. It performs remarkably well, causing almost no noticeable reduction in the performance of each program . . . The user manual is well organzied and easy to read . . . It is an excellent product."

PC Tech Journal Sept '87

"It does an amazing job of handling not only multitasking, but in many cases, concurrent processing as well. I was amazed at it's success in handling the "impossible" combinations I threw at it."

"It is one of the best designed & implemented programs I've ever worked with. It is also one of the most bulletproof."

"What sets it apart from the pack is the fact that it runs in a network environment."

Data Based Advisor Aug '87

OMNIVIEW requires an IBM PC, XT, AT or Jr compatible, and PC or MS DOS 2.x to 3.x. To get your copy, call toll free:

(800) 367-0651

or send $79.95 + 3.00 S&H ($8.00 Intl.) to:

Sunny Hill Software

PO Box 55278, Seattle, WA 98155-5278

For more information call (206) 367-0650 *Omniview formerly Taskview

DoubleDOS trademark Softlogic Solutions, Concurrent PC-DOS reg. trademark Digital Research Corp., MS Windows trademark Microsoft Corp., DESQview trademark Quarterdeck Office Systems, Inc. TopView trademark IBM Corp.

CIRCLE 51 ON READER SERVICE CARD.

CAS: Communicating Applications Specification

This new specification from Intel and DSC makes it simple for developers to add transparent communications to applications.

Intel's Personal Computer Enhancement Operation (PCEO) based in Hillsboro, Oregon, and Digital Communications Associates, Inc., (DCA) of Alpharetta, Georgia, have pooled their resources to create a new communicating specification to simplify file transmissions between stand-alone PCs and PC workstations. DCA is also using the specification for mainframe communications.

Called the Communicating Applications Specification Version 1.0 (CAS), this new, open, high-level programming specification offers developers a common platform to automatically set up and route transmissions to electronic mail and facsimile machines. CAS provides a common platform for applications developers to add communications capabilities to spreadsheets, word processing, and other applications. Accompanying the introduction, Intel unveiled the first coprocessor to support the standard, the Connection CoProcessor Board (see the box).

According to Intel/DCA sources, CAS provides an interface for developers to communicate with other PC and non-PC environments. For example, E-mail developers can use CAS to provide a bridge between local area networks, or as a gateway to provide direct access to FAX machines for non-PC users. The idea is to provide a platform for developers to create applications that invoke communications transparently, without exiting the program.

CAS has been designed to support as many applications as possible in a hardware-independent fashion. CAS primitives have been kept simple and generic. Applications access CAS functions through interrupt 2FH (Multiplex Interrupt). As with DOS function calls, CAS functions are accessed by loading registers with defined values, issuing an interrupt 2FH, and then examining the returned values.

Included in the specification are core functions of communications scheduling and execution (see the functions described in Table 1). Basically, there are two software components in CAS: a Resident Scheduler and Transfer Agents.

The CAS Resident Schedule

With CAS, the developer initiates a communication request in the application by invoking a Resident Scheduler and indicating the target (i.e., distribution list), the transmittal document(s), and the desired time of communication. The Resident Scheduler can transparently access all available communications technologies (e.g., E-mail or FAX), taking information from the application and "lookup" information to determine the appropriate communication method and route.

The Resident Schedule carries a default multiplex num-

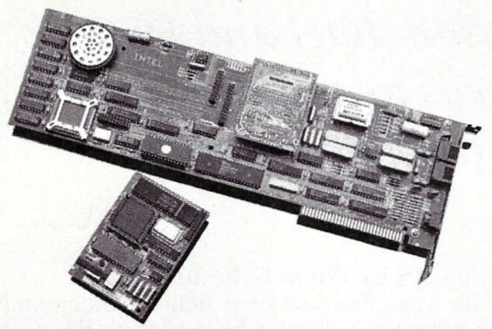
Function Name	Function Number
Get Installed State	00H
Submit a Task	01H
Abort the Current Event	02H
Reserved	03H – 04H
Find First Entry in Queue	05H
Find Next Entry in Queue	06H
Open a File	07H
Delete a File	08H
Delete All Files (in a queue)	09H
Get Event Date	0AH
Set Task Date	0BH
Get Event Time	0CH
Set Task Time	0DH
Get External Data Block	0EH
Get/Set Autoreceive State	0FH
Get Event Status	10H
Get Queue Status	11H
Get Hardware Status	12H
Run Diagnostics	13H
Move Received File	14H
Submit a Single File to Send	15H
Reserved	16H – 80H

Table 1. CAS Function Table

ber of CB. That is, the Resident Scheduler is invoked when the caller performs a multiplex interrupt with AH set to CB. To prevent conflicts with other multiplex services, this number is configurable.

There are three queues that the Resident Scheduler uses to handle Control Files: a Task Queue, a Receive Queue, and a Log Queue. Each of these queues contains information about events that is needed by the applications. An event can be classified as one of several hardware-controlled operations:

Send	The local computer transmits data to a remote computer.
Receive	The local computer receives data from a remote computer.
Polled Send	The local computer waits for a call and then automatically sends information to the computer that is calling.
Log	The local computer creates a detailed record of successful or unsuccessful transmissions.



The Intel Connection CoProcessor

The add-in Connection CoProcessor board for standard bus PCs is the first hardware product to support CAS. It allows users to send 9600 bps file transfers, E-mail, or Group III FAX transmissions, and to receive FAX and mail transmissions from network applications that support CAS. The board uses a 10-MHz, 80188 microprocessor and 286K of memory to download communications processing from the PC's CPU. This frees the microprocessor for safe background operations. The CoProcessor is currently compatible with DOS applications, and Intel plans to offer OS/2 compatibility in the near future.

According to Intel sources, the Connection CoProcessor runs four to eight times faster than traditional modems for file transfers, and it will support unattended operation for both transmission and reception. The user can compose and store up to 999 distribution lists in the electronic phone book (as outlined in the CAS standard). The board also keeps a complete communications log that records the status of each transmission.

The board carries a piggyback site for communication options. The first product in the family of options is the Connection CoProcessor 2400B Modem, a 2400 bps Hayes-compatible modem with an Intel 89024 chip set.

The Connection CoProcessor board has a suggested retail price of $995. the Connection CoProcessor 2400B has a retail price of $295. Both products are available from Intel PCEO.

Circle reader service #253

This kind of queue management is necessary to provide collision-free access to Control Files, and activity classification facilitates background operations (e.g., one application can examine the Log Queue while another is preparing a send activity).

When an application wants to perform a task, it must create a Task Control File that contains such information as the name of the file, the telephone number of the destination, and the time of the transmission. The application is then placed in the Task Queue and receives an "event handle" that keeps track of the task. Once the task is completed, the Resident Scheduler updates the Control File information and transfers it to the Log Queue, thus converting the file to a Log Control File. CAS functions can then be used to query the Log Queue and check the status of the task.

The Transfer Agent consists of computer code to connect, transmit, and disconnect from the recipient device, including transmission checking and a way to issue a status report. The Transfer Agent may execute on the host microprocessor or on intelligent communications hardware. Downloading these Transfer Agents from the microprocessor to intelligent, dedicated communications hardware allows communications to take place in the background, freeing the CPU for other processing tasks.

The Transmitting and Receiving Files

Once the Resident Scheduler is installed, a single file can be transmitted easily. First, CAS invokes function 00H to determine if the Resident Scheduler is installed. Once it has been determined that it is installed, the procedure is:

1. Create the data file to be transmitted.
2. Create the data structure for function 15H (Submit a Single File to Send).
3. Invoke function 15H.
4. Check for transmission errors.

To transmit multiple files, the procedure is to invoke CAS function 00H, then:

1. Create the data files to be transmitted.
2. Create a Task Control File.
3. Create one File Transfer Record (FTR) for each file to be transmitted, appending the FTRs to the Task Control File.
4. Invoke function 01H (Submit a Task).
5. Check for transmission errors.

Receiving files is accomplished in much the same way. First, invoke function 00H to determine if the Resident Scheduler is installed. Then:

1. Invoke 05H (Find First Entry in Queue) to check to see if receive events have occurred. If they have, then use the following steps:

    a. Invoke function 07H (Open a File) to open the Receive Control File. The tag field in the control file is then checked to see if the receive event is of interest to the application. If it is of interest, then move to step 2.
    b. Otherwise, invoke DOS function 3EH (Close File Handle) to close the Receive Control File and move to step 5.

2. For each File Transfer Record in the Receive Control File, check the "status of the file" field to determine

if the received files have been moved, renamed, or deleted. If they have not, then go to step 3. If no such files remain, the Receive Control File is closed and go to step 5.

3. Invoke function 14H (Move Received File) to move or rename the received file to the appropriate filename or directory, or invoke 08H (Delete a File).
4. Once all the received files have been dealt with, invoke DOS function 3EH (Close File Handle) to close the Receive Control File.
5. Invoke funtion 06H (Find Next Entry in the Queue) to check for additional receive events. If there are more receive events, repeat steps 1a – 4.

Industry Support for CAS

Intel is providing initial support of CAS at the PC level with its Connection CoProcessor board. DCA is working to connect the CoProcessor to its line of IRMA cards. DCA's 3270 emulation cards will be linked either through a direct cable attachment or through communications over the workstation bus.

The specification is best suited for delivery of simple files. For example, even though CAS can deliver text, spreadsheets, and other data, it can't differentiate the data so you cannot send a query as you can with an SQL server. Also notice that all this file manipulation makes CAS unsuitable for interactive use. While accessing events via files simplifies interaction between CAS and other software, it does make CAS unsuitable for interactive applications where large numbers of very small transactions are required.

CAS Version 1.0 is only applicable to modem and FAX technology, although DCA/Intel sources expect the specification to evolve to support other communications hardware as well. On the software side, Microsoft officials have indicated that they will add FAX output compatibility to Microsoft Works, and Lotus executives indicated CAS will be supported by Lotus 1-2-3. Borland is adding CAS FAX compatibility to its Sidekick Plus. LanFax/10 software from Alcom, Inc., is also compatible.

Other companies are planning to support CAS as well, including Novell. Symantec, the database software company, will support the Connection CoProcessor, providing transparent file transfers for users of Q&A 3.0. Ashton-Tate, Crosstalk Communications, WordTech Systems, Word Perfect, and other companies have added their support to the specification. □

Did you find this article particularly useful? Circle number 5 on the reader service card.

For a copy of the Communicating Applications Specification, contact either:

Intel Personal Computer Enhancment Operation (PCEO)
5200 N.E. Elam Young Parkway
Hillsboro, OR 97124-6497
(800) 538-3373

or

Digital Communications Associates, Inc.
1000 Alderman Drive
Alpharetta, GA 30201-4199
(800) 631-4171

THE DISKLESS WORK STATION

- A PC/XT Compatible Network Node
- Enclosure Dimensions are just 10" by 10" by 3-½"
- No Cooling Fans are Required, so the Unit is Perfectly Quiet

- Adds Security to the Company's Computer System
- Dealer & Distributor Inquiries are Welcome

PC and XT Are Trademarks of International Business Machines

MasterCard
Visa
Check
Money Order
C.O.D.

SLICER™

Slicer Computers Inc.
3450 Snelling Ave. So.
Minneapolis, MN 55406
612/724-2710
FAX 612/724-0878

CIRCLE 112 ON READER SERVICE CARD

Attention Graphics Programmers and System Developers

The TI 34010 is the best news to come along in the graphics processor arena in a number of years. PC Tech, Inc. has developed two TMS 34010 based PC compatible display adapters that allow the full power of the TMS 34010 to be used on PC/XT/AT compatible machines. **DGIS** is available for either board.

MONOCHROME

- 256k display memory
- 256k program memory
- Up to 48 mHz 6 MIPS operation
- Both ECL and TTL monitor drivers
- Multiple programmable dot clock frequencies
- Typical resolution 736 x 1024
- Capable of drawing 125 million pixels per second
- CGA / MDA / Hercules Emulation
- Suggested list price $595 ($1295 with full page black on white monitor.)

COLOR

- Up to 1 Mbyte display memory (8 bit / pixel)
- 1 - 4 Mbyte program memory
- 256 simultaneous colors from palette of 256,000
- Up to 48 mHz 6 MIPS operation
- Dual (piggyback) single slot board
- Multiple programmable dot clock frequencies
- Typical resolution 1024 x 768
- Custom dot clocks available
- Suggested list price $1295 (512K video memory, 1 Mbyte program memory)

TI 34010 Assembler, Debugger and "C" Compiler available!

Both boards support **DGIS** from Graphic Software Systems, Inc. Most major text and graphics applications are supported on both boards via **DGIS** and low level graphics interfaces . Custom implementations of either design and private labeling are also available.

Contact: **PC Tech, Inc.**
904 N. 6th Street
Lake City, MN 55041
612-345-4555
FAX: 612-345-5514

PC, XT and AT are trademarks of International Business Machines Corp.
DGIS is a trademark of Graphic Software Systems, Inc.
Prices subject to change.

CIRCLE 46 ON READER SERVICE CARD

High-Speed Modems: Techniques, Standards, and What to Look For

by William Wong

There is no high-speed modem standard, but there are specifics you can look for to maximize your chances for 9600-baud connectivity.

Three-hundred-baud and 1200-baud modems are old hat these days, and 2400-baud modems are as common as PCs. High-speed modems (9600 baud and up) are becoming more popular as prices are dropping and availability is improving. High-speed modems are becoming essential to send larger spreadsheets, database files, and bit-mapped images quickly across the country over phone lines.

However, unlike their lower speed predecessors, high-speed modems are starting to press the limits of voice dial-up lines. Standards in the high-speed modem arena are coming, but for now, plan to use the same modem at both ends of the telephone line because it is a "Tower of Babel" for fast-talking modems.

William Wong is president of Logic Fusion, Inc., a systems development firm located in Morrisville, Pennsylvania.

The transmission standards and modems examined in this article are based on the logical, full-duplex, asynchronous technology used by most PCs. This includes modems that may be internally based upon half-duplex modems with synchronous data transmission over the telephone network. Other types of modems include synchronous modems and logical half-duplex modems.

Existing and Proposed Standards

Modem standards have been developed by CCITT, ANSI, and EIA. These standards define various portions of a modem's operation, its connection to other modems via telephone lines, and various protocols. This article discusses some of these standards, starting with the computer-to-modem connection through the modem-to-modem and modem-to-computer connection at the other end.

The only common standard found on asynchronous modems is the EIA RS-232 standard for connecting Data Communication Equipment (DCEs) with Data Terminal Equipment (DTEs). The RS-232 standard defines all 25 signals (pins) of the subminiature D-type connector found on most modems, although most modems only implement a subset of these signals. The 9-pin, D-type connector used on the IBM AT serial adapter card is one subset that is usually sufficient, and has led to the sales of millions of 9-to-25 pin adapter modem cables.

Initially, modems were relatively dumb to the point of manual dialing. Autoanswer was supplied as an option. Now most modems have some form of intelligent control. Control of intelligent modems is dominated by the *de facto* Hayes AT command set. Unfortunately, there is no formal standard and most intelligent modems tend to implement a different subset and extend the command set in different directions to support special features.

The other standard common to all dial-up modems is the connection to the telephone network. Luckily this is another area that is well defined and well accepted, with a host of AT&T specifications and CCITT standards. These standards handle electrical connections as well as dialing and connection protocols.

Where things get tricky is in modem-to-modem communications. The most notable standards are the initial connection and modulation standards. These include Bell 103 and 212a, which correspond to CCITT V.21 and V.22. The Bell 103 modem is the venerable 300-baud modem, while the 212a is the common 1200-baud modem. Both are full-duplex modems. The standards specify the initial handshaking sequence, which includes a tone from the answering modem followed by a response tone from the calling modem. Different tones are used by an answering 212a modem so it can recognize a calling 103 modem, or vice versa. They also use different modulation techniques: Frequency Shift Keying (FSK) for the Bell 103 standard and Differential Phase Shift Keying (DPSK) for the Bell 212A standard. (See the box for a definition of these terms.)

Things get a bit more complicated as the transmission speed increases. Twenty-four-hundred-baud modems tend to use the CCITT V.22 bis standard. V.22 uses a Quadrature Amplitude Modulation technique. The V.32 standard specifies a full-duplex, 9600-baud connection, and each standard includes a way to recognize slower speed modems.

V.32 might have proven itself to be an effective standard for high-speed modems except for a number of problems. The major stumbling block is price; the V.32 standard requires very sophisticated circuitry to provide good echo cancellation. Also, some telephone networks will not support V.32 because of the way echo cancellation is accomplished. The transit time a signal takes between modems also affects echoes on the lines, which makes long-distance communication a problem. Making a good V.32 modem is expensive, so a number of manufacturers have addressed the high-speed modem market with different solutions, some based upon modification of existing standards.

The alternative to full-duplex communication is to use a half-duplex modem link and simulate full-duplex communication between DTEs using buffering and data flow control. Two half-duplex standards are V.29 and V.33, which run at 9,600 and 14,400 baud, respectively. The V.29 standard was originally designed for leased-line use and both standards are used on various facsimile machines. The problem with these standards is twofold. First, the line turnaround time for these standards tends to be slow, on the order of 250 milliseconds (ms). This was fine for the intended applications but it is too slow for most full-duplex applications. Second, a protocol is needed to handle the turnaround.

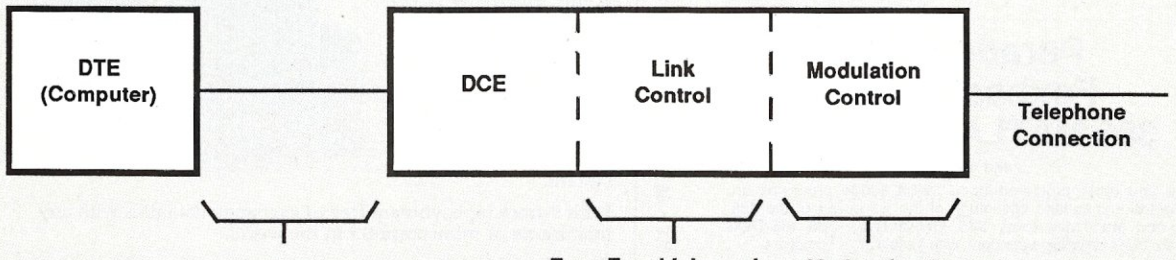
FAX machines normally transmit a large amount of information in one direction. Responses in the reverse direction occur infrequently, so the turnaround time of the modem is inconsequential to a FAX machine's operation. However, this type of delay is intolerable in interactive applications and in most file transfer applications.

However, the turnaround time is a limitation of the standard, not what can be implemented. As such, many manufacturers have implemented "fast turnaround," half-duplex modems based upon the V.29 standard. Examples are Microcom's MNP Class 6 and a half-duplex version of the V.32 standard used in Hayes' V series modems. These modems are often referred to as "ping-pong" modems. However, these modifications must be viewed as nonstandard at this time.

Full-Duplex Transmission

Another approach to logical, full-duplex modems is dynamic, asymmetrical modems that utilize a low-speed channel in one direction and a high-speed channel in the other. The direction of the channels can be changed, depending upon the amount of information being sent in a particular direction. V.34 is a tentative, CCITT-proposed, dial-up, asymmetrical modem standard that would use a 450-baud, low-speed channel and a high-speed channel running at either 14,400 or 28,000 baud. V.34 is under discussion and has not yet reached the formal draft stage.

Figure 1. Two modems with standards placed at appropriate points



Standards:

IEEE RS-232C
V.24

Error Free Links and Data Compression:

Standards and Proposals:
V.42 (Lap M and
, MNP level 2
, MNP level 3
, MNP level 4)

Nonstandard:
MNP level 1
MNP level 5
MNP level 6

De facto Standard Intelligent Modem Control:
Hayes AT Command Set

Modulation Control:

Standards and Proposals:
V.21,Bell 103
V.22, Bell 212A V.22 bis
V.32
V.34 (possible proposal)

Nonstandard Full Duplex Ping -Pong:
V.27 (Microcom)
V.29 (Microcom , MNP level 6)
V.32 (Hayes)

Nonstandard Full Duplex Asymmetrical:
PEP and PEP2 (Telebit)
USR HST (U.S. Robotics)

The connection and modulation techniques were all that were initially required for low-speed modems, but it was very possible for bad line conditions to cause errors in the data being sent over the telephone lines. Also, low-speed modems tended to match the asynchronous nature of the DTE while all high-speed modems utilize synchronous data transmission between modems and convert the data into asynchronous form for the DTE.

The next logical step in handling the more sophisticated high-speed modems is to make the link between modems error free. This is often done on lower speed modems by using communication programs for file transfers using protocols like Xmodem and Kermit.

Early file transfer protocols like Xmodem would send a block and wait for an acknowledgement from the receiver. The time taken to send a block is almost the same as the response time plus two turnaround delays with high-speed modems, so the throughput of a high-speed modem is effectively cut in half. Other synchronization delays can further degrade the operation of a file transfer using high-speed connections. Hence, these protocols are inappropriate for high-speed modems. Fortunately, this is one area where a standard is almost complete.

V.42 is a standard for error control and protocol detection that will provide error-free transmission over a standard, full-duplex connection. The full-duplex connections include CCITT modem standards such as V.21 (Bell 103), V.22 (Bell 212A), V.22 bis, and V.32. V.42 does not, however, have a specification that works on half-duplex connections.

Now comes the fun part. V.42 consists of two distinct protocols. A V.42-"compliant" modem will support the full standard, which includes both protocols. A "compliant" modem will talk to a V.42-"compatible" modem that implements only one of the protocols. There is no name for which "compatible" modem matches which protocol. The two protocols are referred to as LAP M and MNP. The latter is actually MNP Classes 2, 3, and 4. LAP stands for Link Access Protocol. The letters after LAP have no particular meaning, although you could read the M for modem. There are also LAP B and LAP D standards.

The only common standard found on asynchronous modems is the EIA RS-232 standard for connecting Data Communications Equipment (DCEs) with Data Terminal Equipment (DTEs)

There are a number of LAPs defined by CCITT based upon a synchronous protocol called HDLC (High-Level Data Link Control), which is effectively a superset of SDLC (IBM's synchronous protocol for SNA). LAP M is a modified form of LAP D. The LAP protocols are also used in ISDN and X.25 systems, which will be discussed later.

Why all this confusion with the standard V.42? A lot has to do with *NIH* (Not Invented Here), politics, technical

• 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 • 386 •

NOW!

C → 386
Pascal

Paradox 386
Foxbase+ 386
386-MATLAB,/Weitek

... and others ...

These and other protected-mode 32-bit 80386 programs are among the first to take advantage of the full power of the 386. They and practically every **386 protected-mode** MS-DOS program that's shipping were done with MetaWare's compilers.

It's no surprise. The recognized leader, MetaWare introduced the *first* **C** and **Pascal** compilers that generate protected-mode 386 code for running on any 386 MS-DOS machine (e.g., the Compaq 386 or the IBM PS/2-80) over a year ago. **High C** ™ and **Professional Pascal** ™ are well-established and proven.

Smart software developers aren't waiting! Industry leaders such as Borland (ANSA) and Fox use MetaWare's compilers to get dramatic increases in speed and functionality. Don't wait years for Microsoft's 386DOS—your competition will have a big jump on you!

Expand your application to the large 32-bit address space and the full 32-bit registers of the 80386. Go with the long-standing leader. Contact MetaWare for your 80386 software solution today!

(408)429-6382 telex 493-0879

Meta Ware™
INCORPORATED

903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060

The Clear Choice for Large Programming Projects —PC Tech Jrnl

© 1987 MetaWare. Trademarks: Paradox 386 Ansa. Foxbase+ 386 Fox Software. 386 MATLAB MathWorks. High C, Professional Pascal MetaWare. MetaWare

CIRCLE 56 ON READER SERVICE CARD.

9-Track Tape Subsystem

The Data Interchange Solution You've Been Waiting For!

Qualstar's ½" 9-track Ministreamers™ bring full ANSI data interchange capability to your PC or Macintosh™ computer system.

With 9-track tape, you are free to exchange data files with any mainframe or minicomputer in the world.

Our affordable Ministreamers come in both 7" and 10½" versions and use less desk space than an ordinary sheet of paper. They can provide 1600 thru 6250 BPI capability and may be used for disk backup as well as data interchange. Complete subsystem includes tape drive, coupler card, cables, and software.

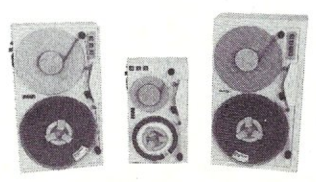
Qualstar has become the number one supplier of desk-top 9-track systems for a good reason; our tape drives have established an outstanding record for reliability and low cost of ownership.

Discover the many advantages 9-track tape has over other Micro/Mainframe links.

Call us today!

QUALSTAR®

9621 Irondale Avenue,
Chatsworth, CA 91311
Telephone: (818) 882-5822



Macintosh is a trademark of Apple Computer, Inc.

CIRCLE 67 ON READER SERVICE CARD.

superiority, and the installed base. LAPs have been a part of CCITT's standards for a long time and have been used in different areas so it seems logical to extend their use to high-speed modems. MNP, on the other hand, grew up with modems; it is a *de facto* standard whose specifications are set by Microcom, a modem manufacturer (see box for more details).

MNP comes in a number of classes that originally started out as hierarchical, but it has since split into a number of different areas. For example, MNP Class 6 deals with the "fast turnaround" V.29, half-duplex modem standard.

Data Compression

Finally, there is the issue of data compression. There are no real standards in this area. The exception is some of the higher MNP classes that can be viewed as a *de facto* standard in some circles. However, data compression techniques are found on almost all high-speed modems (with the exception of some V.32 products) since data compression can almost double a modem's average throughput.

Low-speed modems have the advantage of conforming to existing standards. High-speed modems have emerging standards, but few existing standards. The only industry-wide standard is V.32, which is found in higher priced modems. The remaining modems on the market employ a variety of proprietary techniques. The best way to see what the future has in store for 9600 baud is to look at some of the current high-speed modems.

Some High-speed Modems

The modems examined here are from: Anderson/Jacobson, Data Race, Hayes, Microcom, Telebit, and U.S. Robotics. They represent only some of the 9600-baud modems on the market and only some of the options available from the respective manufacturers. They were chosen for their unique characteristics.

Anderson-Jacobson

The Anderson/Jacobson AJ9631-SA is an intelligent, 9600-baud, V.32-compatible modem. It can operate in full-duplex mode as either an asynchronous or synchronous modem. Trellis-coded modulation (see box) is used for improved error performance, but that alone does not provide an error-free link. In fact, like most V.32 modems, the AJ9631-SA does not provide an error-free link using either a proprietary protocol or something like MNP. In the future, more V.32 modems will support the V.42 data-link standard.

The AJ9631-SA supports the main Hayes-AT command set for dialing. The alternative is to use the V.25 bis-compatible autodialer (V.25 is another CCITT standard for modem autodialers). Non-volatile memory is used to store optional settings, and a sophisticated control panel is included. A rack-mounted version, the AJ9631-SAR, is also available.

The AJ9631-SA supports 9600-baud transmission with a fallback to 4800. It does not support any of the lower speed standards, such as V.21, V.22, or V.22 bis. The one main feature of the AJ9631-SA is that it will talk to any other V.32-compatible modem at 9600 baud providing a full-duplex connection. As such, it has found a place in corporate America, which can afford the higher price of the V.32 modems.

Anderson-Jacobson's documentation is very complete, including a section on diagnostics. The AJ9631-SA is a big modem, almost six times the volume of a Hayes Smart-

TRAQQER™ ◆◆◆ Keeps Track...

- **Of C Functions**
- **Of C Structure and Union Definitions**
- **Of C Complex Data Objects**

And Lets You Find What You Need at the Touch of a Key

SYSTEM REQUIREMENTS: MS-DOS or PC-DOS, 2.0+; at least 256K; CRT with ANSI.SYS or monitor with ROM-BIOS support.

SUPPORTS: MDA, CGA, EGA, VGA, MCGA, HGA HARDWARE INDEPENDENT

$150.00

N.J. residents add 6% sales tax.

MS-DOS is a trademark of MICROSOFT CORP., PC-DOS is a trademark of IBM CORP

tropic software

168 WOODBRIDGE AVE., #2, HIGHLAND PARK, N.J. 08904 • (201) 572-0515

CIRCLE 115 ON READER SERVICE CARD.

The Race-VM and BMX-VM

modem, as are most V.32 modems because of the complexity of the echo cancellation circuitry. However, the larger size seems to be worth it since I found the AJ9631-SA worked flawlessly on good lines and had only the expected induced errors on noisy lines. Error-free transmission can be accomplished using a number of available communication packages that use their own data-link protocols. Some even include data compression.

Data Race

Data Race supplies a number of different high-speed modems. The two that I tested were the Race-VM I and Race-VM II. These modems are essentially the same when used for communication, but the Race-VM II adds a unidirectional printer port that uses the same telephone connection. This feature alone may be a good reason for purchasing a Race-VM II. The accompanying printer can be either a Centronics-compatible parallel printer or an RS-232 serial printer. Hardware- or software-flow control is used. The terminal port can support asynchronous connections from 300 through 19,200 baud, with full-duplex synchronous connections at 1200 or 2400 bits per second (bps) and half-duplex synchronous connections at 4800, 7200, and 9600 bps.

The Data Race modems are like many of the proprietary, high-speed modems in that they support a number of standard, low-speed protocols. The mix can be staggering but useful. Low-speed Bell 103, Bell 212A, and V.22 bis are supported as well as MNP levels 2, 3, and 4. The Race VM modems use V.29 for half-duplex synchronous, higher speed connections and a proprietary modification for "fast turnaround," full-duplex connections.

The Data Race modes must be applied when the op-

MNP: Microcom's Networking Protocol

MNP is a popular protocol that started as a way to provide an error-free link between communication programs. MNP moved into the modem itself as the protocols became more specialized. Microcom controls the MNP standard and allows others to use the standard for a lifetime license fee of $2,500. One disadvantage is that Microcom gets the initial edge on newly released MNP levels, such as MNP level 7, which has not been released to others as yet. However, there are advantages to making sure all the bugs are out and to see how an MNP level will be accepted. The MNP levels 1 through 6 that have been released and are:

Class	Description
1	Asynchronous Block Mode
2	Asynchronous Stream Mode
3	Synchronous Stream Mode
4	Advanced Synchronous Stream Mode
5	Data Compression
6	Fast Train (Turnaround) V.29 control

MNP level 1 is almost never found on modems. Both it and MNP level 2 can be easily implemented as part of a communication program, which means MNP can be used on any conventional modem that does not support MNP by using a communication program with MNP support. The asynchronous block mode is similar to Xmodem file transfers; it sends a block and waits for acknowledgement. Its throughput is actually slower than what a modem can provide because of MNP level 1 overhead. The asynchronous stream mode is more efficient, but still has a lower throughput than can be provided by the modem. MNP level 2 provides a full-duplex, error-free transmission link.

MNP level 3 moves into the modem and replaces the asynchronous DCE-to-DCE transmission with a synchronous protocol. Asynchronous data from the DTE is stripped of the start, stop, and parity bits, and the data is placed into an HDLC-like data stream. The synchronous data stream is actually more compact than the asynchronous data stream, even though there is additional control and error-detection overhead. This allows the actual throughput of a modem to exceed its bits-per-second rating, which is based upon asynchronous throughput. MNP level 3, and the higher level MNP protocols, must reside within an asynchronous modem since the protocol is synchronous.

MNP level 4 is an optimized version of MNP level 3. The data packet size has been reduced and adaptive packet sizing is added. Smaller packets are used if errors are frequently detected since a smaller packet has a lower probability of having an error. Larger packets are used if few errors are being detected. The packet size changes based upon line quality.

MNP level 5 provides data compression. It requires a lower MNP level to provide an error-free data link. MNP level 5 is one of the few public data-compression standards.

MNP level 6 differs from lower MNP levels in that it is used to control specific types of modems. It provides link speed negotiation from 300 to 9600 baud and supports "fast-train" V.27 and V.29 modems. MNP levels 4 and 5 are normally used with level 6 to provide an error-free data link.

MNP level 1 seems to be disappearing from normal use. In fact, the proposed V.42 standard includes only level 2, 3, and 4. Unfortunately, level 5 was left out and level 6 is inappropriate for the V.42 standard.

There are hundreds of thousands of MNP-compatible modems that will provide an error-free link between themselves. However, this statement can be a little misleading because most implementations currently support levels 1 through 4. The advantages found in levels 5 and 6 are found in newer modems, but they are currently few in number. The main advantage of MNP is that MNP modems talk together at some speed using an error-free link.

The Hayes V series 9600

tional printer is in use because the modems are effectively a small statistical multiplexer. The operation is totally transparent once the right cables are plugged in. Unfortunately, the parallel printer cable is not the same one used with the IBM PC, even though it does have a DB-25 connector on the modem for the printer.

Control of the modem is done using either a proprietary command set or a Hayes-compatible command set that has been extended to accommodate the various features found in the Race VM modems. Perhaps the feature most lacking in the Data Race product is the documentation, which is complete but lacks a good index. Well-organized documentation is necessary with a product that has so many options.

The Data Race modems use their own data link control protocol to talk between themselves, which means connection to non-Data Race modems using asynchronous DTEs is limited to lower speeds. However, this is true of almost all other non-V.32, high-speed modems.

Hayes

Hayes is probably the best known modem manufacturer to PC users, although it is actually one of the younger modem vendors. Even so, the Hayes AT command set for intelligent modems is a *de facto* standard. Hayes made its name by providing intelligent, low-speed modems to the PC marketplace, making the Hayes Smartmodem the yardstick by which other modems are measured. Hayes' new V series Smartmodems is a new yardstick. Unfortunately, unlike its prior modems, the V series is effectively a proprietary standard. There are currently three basic units in the V series: a 9600-baud unit, a 2400-baud unit, and an adapter that makes a conventional Hayes modem into a 2400-baud V series modem. The modem units come in internal and external configurations.

The external 9600-baud V series modem I tested uses a modified V.32, full-duplex standard, in a half-duplex, ping-pong mode. An error-free data link is provided between two V series modems using a modified LAP B protocol. This may be upgraded to a LAP M protocol found in the V.42 standard. The V.42 standard is supposed to run on existing full-duplex standards.

Hayes also uses a proprietary data-compression scheme that can double the effective throughput of the modem to 19,200 baud, assuming the proper data stream. The modem is obviously Hayes-compatible and supports the slower speed modem standards used by existing Hayes modems.

The new V series modems are slightly taller than their slower cousins and use a different power supply, but all other aspects of the products remain the same. The documentation is the best in the modem business. The thing that may keep the V series from becoming a standard is the fact that Hayes is keeping its technology proprietary. Currently, nothing talks to a Hayes V series at high speed except another Hayes modem.

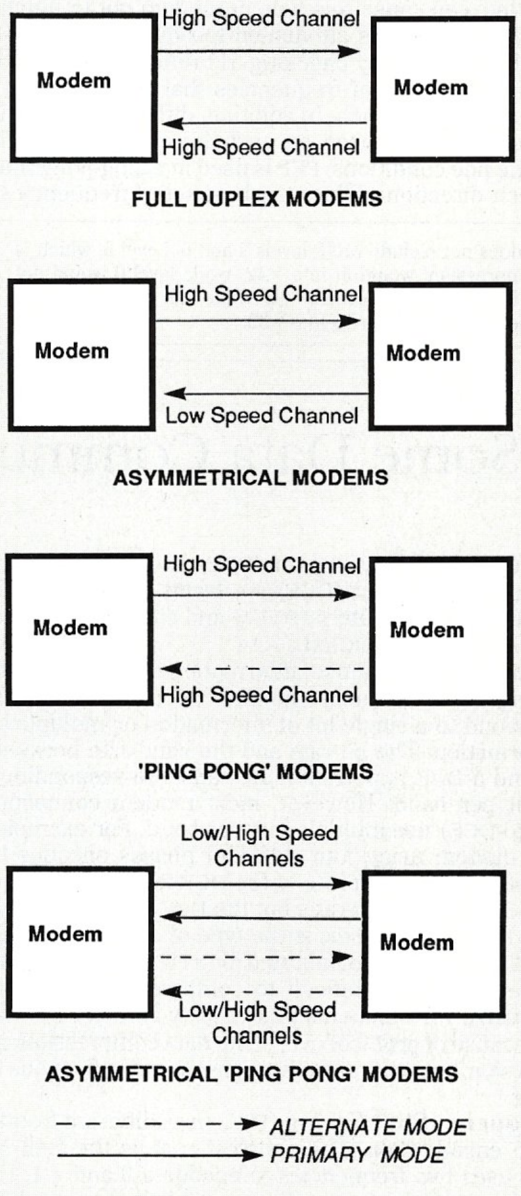
Microcom

Microcom is the originator, banner bearer, and developer of MNP. MNP has been incorporated into all of Microcom's modems, including the AX/9624c that I tested. The AX/9624c uses either a V.27 or V.29 connection, both of which are half-duplex connections with either the standard (slow) or fast train-link turnaround. It does not support MNP level 1, which is rarely used anyway, but it does support levels 2 through 6. This intelligent modem supports both the Hayes AT and Microcom's own SX command sets.

The AX/9624c also supports the lower-speed V.21, V.22, and V.22 bis modems, with or without MNP levels 2 through 4. The high-speed mode normally runs at 9600 bps, and MNP level 5 data compression can boost this up to 19,200 bps. Synchronous half-duplex connections to conventional V.27 and V.29 modems are also supported without MNP.

Microcom's documentation and support match Hayes', making it a good high-speed modem selection. The AX/9624c will talk to other MNP modems that incorporate

Figure 2. Data link (FSK, DPSK, QAM, PEP, ping-pong, and Courier HST)



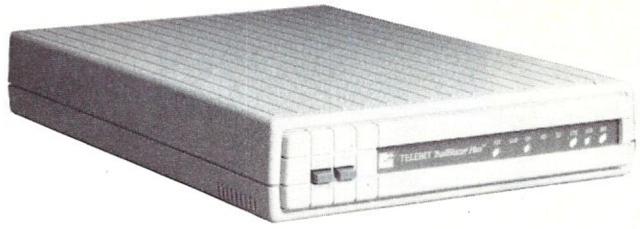
MNP levels 2 through 5 at speeds up to 2400 baud. It will talk at 9600 baud if the MNP modem also incorporates MNP level 6. For example, the U.S. Robotics Courier HST supports MNP levels 2 through 5 but uses a proprietary asymmetrical modulation technique for 9600-baud operation that is incompatible with MNP level 6, which is a modulation standard not a data-link standard.

Inclusion of MNP levels 2, 3, and 4 into V.42 has certain advantages but the lack of levels 5 and 6* may prevent it from being a CCITT standard. Even so, MNP has become a major force in the modem world, and its multivendor support may make other levels a *de facto* standard. In a world where compatibility is more important than being fastest, MNP and Microcom may make a major difference.

Telebit

Telebit is one of the first manufacturers to come out with a high-speed asymmetrical modem. The Trailblazer Plus, available in internal and external forms, utilizes a proprietary communication system called a Packetized Ensemble Protocol (PEP). Most conventional modems utilize a small number of tones that are shifted in phase or frequency to transfer information between modems. A bad telephone line can cause this type of modem not to function if one of the tones is attenuated too much. PEP tries to avoid this problem by checking 511 different frequencies and using a subset of frequencies that are found not to be attenuated too much. In addition, different modulation techniques can be used on each frequency, depending upon the line conditions. PEP is used in a ping-pong mode and each direction utilizes the best set of frequencies in the corresponding direction. The Trailblazer is actually an asymmetrical modem, so a low-speed link is always available in the opposite direction.

* V.42 does not include MNP levels 5 and 6. Level 5, which is data compression, would fit into V.42, while level 6 would not since it is a modulation control technique that corresponds to other modulation standards, like V.32.


Telebit Trailblazer Plus

The primary difference between the Trailblazer and other high-speed modems is that the Trailblazer can transmit data at 18,000 bps, at full tilt instead of the more conventional 9,600 bps modems. The Trailblazer also utilizes data compression, bringing throughput to 19,200 bps. Trailblazer's data compression scheme only needs to reduce the data stream size by a small percentage to reach this rate instead of cutting it in half.

Like many other high-speed modems, the Trailblazer uses an adaptive modulation scheme that slows down transmission when a telephone line degrades, and speeds up when the line improves. The difference occurs because of the way PEP works with a large number of frequencies, allowing the Trailblazer to make the incremental differences smaller than some other modems, which cut their transmission speeds by a third or a half.

The sophistication of the Trailblazer matches that of a high-performance PostScript printer. It contains both a

Some Data Communication Terminology

Data Terminal Equipment (DTE) and **Data Communication Equipment** (DCE) are terms used to describe terminals and computers (DTEs) and communication devices like modems (DCEs).

A **baud** is the amount of information that is transferred in a single time unit on a communication line. A baud can correspond to a single bit of information or multiple bits of information. The bit rate and the baud rate between a DTE and a DCE is normally the same, corresponding to one bit per baud. However, most modem connections (DCE-to-DCE) use multiple bits per baud. For example, a DPSK modem using four different phases encodes two bits per baud. The bit rate of the DCE-to-DTE is the same as the DCE-to-DCE bit rate, but the DCE-to-DCE baud rate is half that amount. This is the type of encoding used by a Bell 212A modem. Baud rates for a DCE-to-DCE connection become more difficult to compare and are of less importance with modems that employ an error-free data link because of protocol overhead, data compression, and conversion between asynchronous and synchronous encoding.

Frequency Shift Keying (FSK) uses different frequencies to encode data. The simplest case is the Bell 103, which uses two frequencies to encode a 0 and a 1. Full-duplex operation is normally accomplished using different frequencies for each direction. FSK usually encodes a single bit per baud.

Differential Phase Shift Keying (DPSK) uses a single frequency to transmit information in one direction. The phase of the signal is tracked by the receiving modem, and the transmitting modem changes the phase of the signal by some pre-defined amount. For example, a phase change or 0, 90, 180, and 270 degrees could be used to encode two bits of information in one phase shift. The number of different valid phase changes depends upon how the modem is built. More valid phase changes decrease the difference between changes and increases the complexity of the modem because it must be more accurate. There is also a limit to the accuracy of the phase change that a telephone line will replicate.

Quadrature Amplitude Modulation (QAM) uses multiple frequencies simultaneously with changing amplitudes to encode information. This technique also encodes multiple bits per baud like DPSK.

Trellis coding is an error-detection correction encoding scheme used with modems that encode multiple bits per baud. It is more common in high-speed modems that may encode as many as eight bits per baud. It is analogous, although not identical, to error-correcting codes found on computer memory systems.

CHARTER SUBSCRIPTION INVITATION

Here At Last! The Authoritative Source of Database Application and Product Information

Inside DBMS Magazine

DBMS Magazine offers in-depth articles examining the capabilities of programmable databases as tools for solving real business problems. Authoritative features show how a specific product is used to solve a specific problem. Sample some of the feature articles to be presented:

- 50 Nasty dBase Bugs, and How to Squash Them
- Has Ashton-Tate Finally Designed a Decent, Multiuser Database?
- Why 386 Databases Can be Either the Fastest, or a Waste of Money
- Is FoxBase+ dFastest Around?
- Why Microsoft and Ashton-Tate teamed up on SQL: Server – What this Ambitious Product Will and Will Not do for You
- R:Base Puts OS/2 to Work
- Understanding Paradox's PAL
- Can Oracle Squeeze Into a PC?

...with regular columns written by experts, providing tips on developing and modifying databases, such as:

- Putting dBase IV's Arrays to Work
- Cross tabulations in R:Base
- Sharing Files Between 1-2-3 and dBase
- Using Clipper's Valid Statement
- Tips on Creating Reports in Paradox

Who Should Read DBMS Magazine?

Professional developers, MIS/DP departments, consultants, VARs and serious users will find DBMS Magazine to be the source of important information for serious users of microcomputer database software.

Charter Offer and Benefits with No-Risk Guarantee

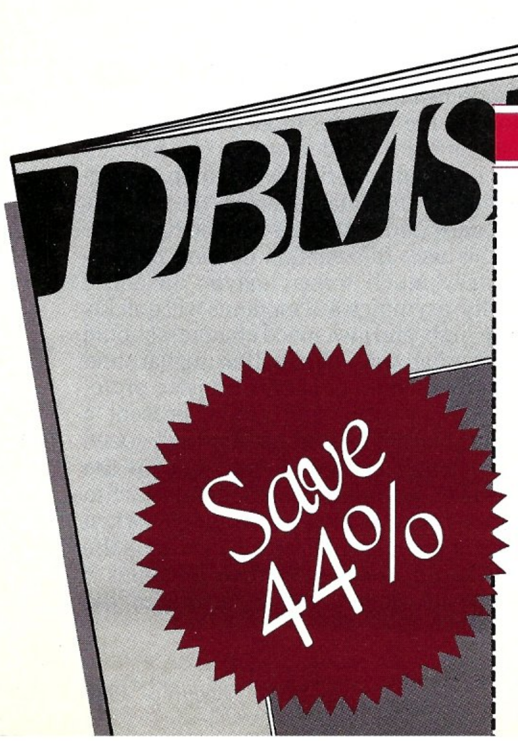
For a limited time DBMS Magazine will offer Charter Subscriptions with substantial cash savings.

You will also be entitled to continued savings — on all renewals and on any gift subscriptions you may order.

If at any time you decide DBMS is not for you, simply cancel. We will send a full refund for all unmailed issues.

Order Today!
Send your order form to:

DBMS Magazine
P.O.Box 3770
Escondido, CA
92025-9638



CHARTER SUBSCRIPTION ORDER FORM

☐ **Yes,** please send my first issue and enter my Charter Subscription to DBMS Magazine. My full year's subscription (11 additional issues) is only $19.97. I save 44% off the newsstand rate.

SEND NO MONEY!
We will bill you later, after you receive your first issue. GUARANTEE! You may cancel at any time for any reason and receive a full refund on all unmailed issues.

MAIL TO: **DBMS Magazine, P.O.Box 3770, Escondido, CA 92025-9638**

NAME ______

COMPANY ______

ADDRESS ______

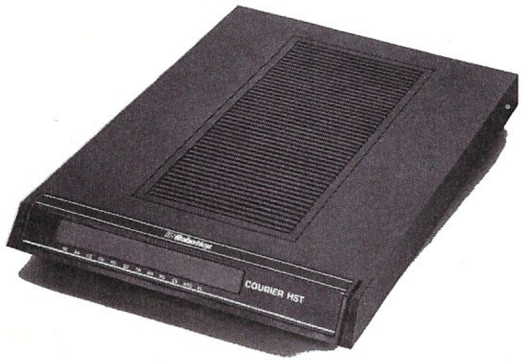
CITY/STATE/ZIP ______

Allow 6-8 weeks for delivery of first issue. Offer good in the U.S. only. Newsstand rate (1 year): $35.40

20M9

high-speed digital signal processor and a Motorola 68000. All this horsepower is needed to keep up with the changing environment and the large amount of data moving through the modem.

The Trailblazer is very versatile. It's synchronous mode supports SDLC instead of being just a synchronous modem. MNP levels 1 through 3 are supported when the modem is operating on the slower but standard modulation modes at 300, 1200, and 2400 baud. The Trailblazer can also be configured to optimize its use with other protocols, such as Xmodem, Ymodem, Kermit, and UUCP. Although this may sound strange, it becomes quite important when using high-speed modems in new applications. For example, Xmodem will tend to slow down the effective throughput of a high-speed modem because it waits for an acknowledgement of each block. The acknowledgement time is a small percentage when using a low-speed modem but it can actually exceed the time needed to send a block of data using a high-speed modem. The Trailblazer simulates the connection by accepting data from one end using the appropriate protocol. It then sends the data to the other modem using its own protocol, and the other end converts the data back to the selected protocol using buffers inside the modem itself. Although the selection is not totally transparent, it can make a major difference for long-distance connections with bulletin board and UNIX systems.



U.S. Robotics Courier HST

Although the Trailblazer uses a different asymmetrical mode of operation than the U.S. Robotics Courier HST, the two are currently combined in developing the V.34 proposal for asymmetrical high-speed modems. However, the current Trailblazer implementation does not match the proposal. Even so, the Trailblazer is a flexible, high-speed modem that can meet or beat any other on the market.

U.S. Robotics

U.S. Robotics is another well-known modem manufacturer that also has a line of low-speed, Hayes-compatible, MNP modems. The Courier HST is its high-speed modem, which incorporates MNP levels 1 through 5 for low-speed connections and a proprietary protocol called USR HST for high-speed connections. Low-speed connections include the usual 300-, 1200-, and 2400-baud standards. In high-speed mode, the Courier HST is similar to the Telebit Trailblazer because it uses an asymmetrical data link. The slow-speed channel is 450 bps and the high-speed channel is 9600 bps and uses Trellis-coded modulation. However, the high-speed channel does not use as large a number of different frequencies as the Telebit PEP scheme. Also, the high-speed channel only runs at 9600 bps and not the 18,000/19,200 bps speed found on the Trailblazer.

The Courier HST is a well-made product. Its documentation was the best of the group; well designed, indexed, and full of useful and understandable examples and appendices. The modem also seems to be more readily available from dealers while many other high-speed modems have more restricted distribution channels.

Overall, these modems represent the main technologies currently employed in high-speed modems. The majority are not full-duplex modems internally, and all are synchronous modems internally. With the exception of some V.32 modems, all high-speed modems provide an error-free data link, and most provide data compression as an option.

MNP levels 1 through 5 tend to be found in a high-speed modem, but only when the modem is utilizing low-speed modulation standards. MNP level 6 is probably the closest thing to a *de facto* high-speed modem standard enabling different vendor's modems talk to one another at high speeds.

However, multiple vendor support of a particular modem technology is not restricted to publicly available standards like MNP. Telebit Trailblazer modems are available from a number of other sources that have licensed Telebit's technology. Even so, most high-speed modems only talk among themselves when it comes to high-speed connections.

High-Speed Modem Selection Considerations

Choosing a high-speed modem can be a difficult task since the major use can greatly affect its performance. The ideal choice is a V.32 modem with an error-free data link protocol like MNP level 4, which also supports the lower speed modem standards. However, this type of unit tends to be very expensive. It is also restricted to 9,600 bps without the addition of data compression.

The alternative is to buy a high-speed, ping-pong or asymmetrical modem. Ping-pong modems tend to be implemented using half-duplex standards, such as V.27 and V.29 or some other modulation technique like Hayes' modified V.32 half-duplex data link. The turnaround times can vary quite a bit but tend to be significantly lower than the 253 ms found in the standards. Some are as fast as 27 ms, while others may be as slow as 85 ms. This time can affect full-duplex, interactive applications since a keyboard character must go from the keyboard to the PC and back to the remote screen. This is essentially twice the turnaround time plus any internal buffering delays and any delays in the PC.

Asymmetrical modems fair a bit better if the full-duplex data transfer is close to the transfer rates of the modem. Interactive applications are a good example of where an asymmetrical modem usually works better than a ping-pong modem. Normally, information from the user is entered at the keyboard and a 450 bps channel is more than adequate for this use. Information coming back to the screen is normally much larger, maybe an entire screen. In this case, the asymmetrical modems will quickly detect which way the high- and low-speed channels should be setup and remain that way, eliminating turnaround delays. Likewise, many unidirectional file-transfer protocols, either block/ACK or streaming mode, present a similar load, with the file being sent on the high-speed channel and flow control messages being sent on the low-speed link. However, asymmetrical modems start to have the same characteristics as ping-pong modems when the amount of information in both directions exceeds the capacity of the low-speed channel. It then becomes a matter of which type of modem can turn around the high-speed channel the fastest.

In general, asymmetrical modems work better for interactive applications. Both types of modems are good candidates for file transfers or LAN-to-LAN links. In fact, it may be more important to consider the communication programs being used, since their operation may limit the effective throughput of the data link. For example, using the Xmodem file-transfer protocol on any high-speed modem will generally cut throughput to half or less the optimum transfer rate of the modem. More advanced protocols, like those found in HyperAccess from Hilgreave, Inc., Hayes' Smartcom III, Blast, and other communication programs, are needed to effectively utilize high-speed modems. This is in part due to line turnaround times as well as protocol overhead and the speed of the PCs. It does not make a lot of sense to use a high-speed modem on a dual floppy disk PC.

The importance of V.42, LAP B, and LAP M are somewhat minimal at this point since most high-speed modems are based on proprietary technology, usually in more than one area. The purpose of a standard is to make one implementation work with another, and high-speed modems currently do not yet fit as interchangeable parts. The standard may start to have an impact in the early 1990s.

Hayes has indicated that it will be putting an X.25 PAD (Packet Assembler Disassembler) into its modems. This is an interesting option for some corporate users, but it will probably be of no interest to the majority of high-speed modem users. Essentially, this type of modem moves the PAD support for a port on an X.25 network into the modem. It can improve efficiency and possibly add power at the modem end if properly implemented. However, you need an X.25 network to use it. It is especially common in Europe.

Finally, there is the issue of MNP. It is an effective standard for providing an error-free data link at low speeds, and is an existing *de facto* standard adopted by a few vendors. However, those vendors that have implemented MNP levels 2, 3, 4, and 5 have not necessarily moved onto levels 6 and 7. Instead, they have used proprietary technology to provide high-speed data links, using proprietary protocols and data compression schemes. So MNP provides a good deal more flexibility at low speeds and may provide more connection alternatives at high speeds. At this point however, it is uncertain whether the higher MNP levels will become the dominant force in the high-speed modem world, especially since they are not part of the proposed V.42 standard.

Summary

High-speed modems are finally here at reasonable prices and good reliability. They do not need conditioned or leased lines, and will work with practically any communication software package although some software uses the high-speed modems more efficiently than others.

The main thing to remember is to buy high-speed modems by the pair because, for the most part, no two vendor's modems will talk to one another at high speeds. There are exceptions to this rule and MNP level 6 (a modem modulation control technique) may become more of an influence in this area depending upon its acceptance by modem vendors other than its creator, Microcom.

Luckily, buying a high-speed modem does not leave you out in the cold. Most continue to support multiple low-speed protocols and even error-free data links used on the slower speed modems. The AT command set dominates the modem marketplace, which makes operation with most PC communication packages possible.

In addition to buying a pair of modems, or at least the same kind as the modem at the other end of the line, you will want to keep in mind the major application for which the modem will be used. Line turnaround time can become a factor in interactive use, and some file-transfer protocols work better with asymmetrical modems than with ping-pong modems.

At this point there is no one "safe" high-speed modem to buy if you want to buy only one. However, any of the high-speed modems on the market should be considered if you can specify all the modems that will be used in a company or for a particular application. □

Did you find this article paricularly useful?
Circle number 6 on the reader service card.

Product Information

J9631-SA $3,095

Anderson/Jacobson
521 Charcot Avenue
San Jose, CA 95131
408-435-8520
Sources at Hayes indicate that the J9631-SA will be replaced by the AJ-9651, a 9600-baud unit with the same functions, a more compact design, and a lower price.
Circle reader service #256

Race-VM I $1,195
Race-VM II 1,395

Data Race, Inc.
12758 Cimarron Path
San Antonio, TX 78249
512-692-3909
Circle reader service #257

V Series 9600 $1,299

Hayes
705 Westech Drive
Norcross, GA 30092
(404) 449-8791
Circle reader service #258

AX/9624c $1,299

Microcom, Inc.
1400A Providence Highway
Norwood, MA 02062
617-762-9310
Circle reader service #259

Trailblazer Plus $1,345
Internal version 1,195

Telebit Corporation
1345 Shorebird Way
Mountain View, CA 94043-1329
1-800-TELEBIT
Circle reader service #260

Courier HST $995

U.S. Robotics, Inc.
8100 North McCormick Blvd.
Skokie, IL 60076
(312) 982-5010
Circle reader service #261

Testing Three High-Speed Modems

Profiles of the Hayes V Series 9600, the Telebit Trailblazer Plus, and the NEC DSP9630 Modems

by Charles H. Strom

Editor's Note: *This review of high-speed modems is part of an ongoing look at modem technology. It covers new units introduced since our previous reviews. In the January/February 1987 issue,* Micro/Systems Journal *published a review of the Electronic Vault UPTA96, Racal Vadic 9600VP, and Trailblazer RAI12E-T1 modems. In the March/April 1987 issue,* Micro/Systems Journal *published reviews of the Codex 2260 and USR Courier HST modems.*

In the early days of microcomputing, I made extensive use of my state-of-the-art PMMI modem. At the time, this S-100 board was on the cutting edge of technology. It used FSK (frequency shift keying) modulation, the standard 300 bps Bell 103 protocol (where bps stands for bits per second, not to be confused with baud, which is the number of analog state changes per second). However, careful circuit design and high-quality components allowed routine operation at 450, 600, and, under ideal conditions, even 710 baud. Since the late '70s, 1200 bps has become the norm, and today 2400 bps operation is routine as well. However, given the explosive growth of the telecommunications industry, it is safe to say, "the faster the better."

Charles H. Strom is a chemist who has been working with computers for more than 10 years. He is a sysop on General Electric's GEnie national time-sharing service.

In recent months, we have witnessed a proliferation of high-speed modems designed for asynchronous operation at sustained data rates of 9600 bps and higher. At the same time, the continuing personal computer revolution has imposed additional demands. The microcomputer user expects a relatively low price as well as ease of use. My partner, Dave Kozinn, and I have been working with three state-of-the-art, high-speed modems for several months. We have examined these devices from the standpoint of real-life experience rather than evaluating them in a laboratory setting. With 2400 bps and slower modems, it is a simple matter to call a variety of local and long-distance systems, national services, and so on. Such is not the case with the high-speed units because they are still relatively uncommon. We had to be satisfied with making several interstate calls from New York to New Jersey or Pennsylvania. Thus, our experiences cannot be considered exhaustive, but I feel that we put these 9600-baud units through their paces.

The Tower of Babel

The personal computer scene today is a world of emerging standards. We have IBM-compatible computing with its implied hardware and software platform, and Macintosh computing, which is certainly a standard by default since there is only one supplier. In the data communications arena, there are standards as well. Bell 103 is the old 300 bps standby, Bell 212 is synonymous with 1200 bps asynchronous operation, and so on. Sadly, such is not the case with the current crop of 9600 bps modems. A standard 9600 bps protocol does indeed exist, the CCITT-defined V.32. This is a full-duplex method that until recently has been relatively expensive to implement because of the sophisticated filter components necessary for full-duplex operation.

A number of workarounds have emerged. They can be considered pseudo-V.32, or asymmetric modems. Indeed, the Hayes V series Smartmodem 9600 claims to be a V.32 half duplex modem in its specifications. This is a contradiction in terms, but that is beside the point. We will examine the Hayes unit and another pseudo-V.32 modem, the Telebit Trailblazer Plus. Unfortunately, these modems are not compatible with one another. As we delve into the technical features of these modems, the reasons for their incompatibility will become obvious. Their designs are based on differing philosophies. It is a terrible failing that these modems are incompatible. Possibly, each firm thinks that its own design will succeed in the market based on its merits. After further examination, we will try to hazard some opinions about this viewpoint. In any event, the present situation can be compared to a mechanized Tower of Babel, where many different modems cannot talk to each other. Lastly, we will look at the new

NEC DSP9630, a true V.32 unit that does offer a measure of compatibility in that any true V.32 modem can communicate with it.

The Hayes V Series 9600

Hayes is one of the first manufacturers of modems geared toward the microcomputer market. The company's technical innovations and the high quality of its equipment have established Hayes modems as the *de facto* standard in this market segment. Hayes is clearly interested in extending its dominance to the commercial and mainframe marketplaces through its 9600 bps modem. It appears that the company wishes to establish its protocol as *the* standard protocol.

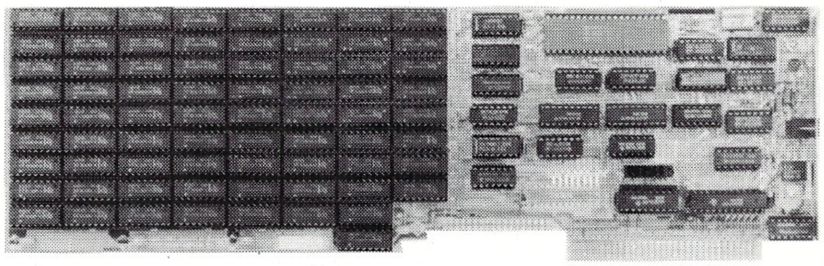
The 9600 bps unit is identical in form to other Hayes modems. The package is surprisingly compact and consists of two full-sized circuit boards sandwiched together in the familiar brushed aluminum case. Without any cooling slots, the modem runs warm, but the design appears to take heat dissipation into account since operation was unaffected by overheating. The front panel LED indicators are also identical to the Hayes 1200- and 2400-baud modems, except that the HS light denotes 9600 bps operation.

Although the Hayes can operate in synchronous modes, we did not test this facility since it is not of great interest to personal computer users. The modem operates well at speeds of 300, 1200, and 2400 bits per second and does not appear to compromise anything in terms of low-speed performance. It was a simple matter to disconnect my Hayes 2400 Smartmodem and replace it transparently with the new unit without any modifications in my normal operating procedures. All speeds, including 9600 bps, are accessed in the standard Hayes manner of sending commands to the modem at the desired rate. When the special, error-control mode is selected, it is possible (and advisable) to set the communications link between the DCE (the computer) and the local modem to a rate higher than the actual modem-to-modem speed. Adaptive compression is enabled in this mode. The modem examines the data stream and encodes the data based on the frequency of a character's occurrence, while the remote modem decompresses the data in a similar manner. This reportedly results in an effective data transfer rate which, depending on the nature of the data transmitted, could approach 19,200 bps. Needless to say, in addition to the hardware capability of sending data from the computer at 19,200 bps, the computer also must be capable of feeding the data at this rate in a sustained manner. A 4.77-MHz PC can't do this. We used 16-MHz, 80386-based units for this review, and although this is more processing horsepower than necessary, it allowed us to concentrate on the modems themselves rather than worrying about what was driving them.

Error-Control Mode

Although the Smartmodem 9600 is capable of operating in high-speed mode in a dumb, asynchronous manner, such a mode is of limited utility. Hayes mentions that such a mode can allow communication with a 100-percent compatible modem, but none exist yet, so this point is moot. The normal procedure for high-speed operation would be to go into error-control mode to call another Hayes modem. Parenthetically, Hayes does manufacture a V-Series Smartmodem 2400 as well as an accessory that can be retrofitted with its non-V series units to add error-control functionality. Upon remote carrier detect, the Hayes performs a feature negotiation query with the remote unit. Assuming the remote is a V series modem, error-control mode is established and reported. Failing this, a normal, high-speed link is established (or the connection is terminated, depending on a user-set S register). Assuming error-control mode is established, several protocols are called into play to increase transfer rates as well as data integrity.

GET A FLAME



The Blue Flame II is the latest in our line of very high-performance disk emulators for PC's, XT's, AT's, '386's, and all clones. It's extremely fast: 800Kbytes per second transfer rate, ten times faster than hard disks. Even faster than IBM's VDISK program! And big: Up to 8 megabytes per board, 32 megabytes per logical drive. Much bigger than extended or expanded memory. It doesn't waste any of your computer's memory address space for storage. And the Blue FLame II is reliable: With no moving parts, it can be accessed continuously for years with no failures. Don't try this at home with your hard disk!

Not just another RAMdisk, the Blue Flame II has an external AC-powered battery-backup option: Data isn't lost when the computer is turned off. And "Reset" isn't a dirty word anymore. Even during a blackout, the battery maintains data for 10 hours.

The Blue Flame II is available fully-populated, with 8 megabytes, for $2495. 4 megabytes for $1395. 2 megabytes for $795. Battery Backup option costs $130. Call us for information on our SemiDisk products for S-100, and Epson QX-10/QX-16.

If you want greater software speed, improved data security, increased hardware reliability, get a Flame. If you need the hottest disk performance possible, get a Flame. A Blue Flame II SemiDisk.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, OR 97075
(503) 626-3104

CIRCLE 47 ON READER SERVICE CARD.

The Hayes V series modems use a subset of the X.25 standard of the CCITT. This protocol was conceived for use between synchronous terminals and public data networks and is widely used today. X.25 is a multilayered protocol. The packet layer provides multiple communications circuits as required on a typical network. The next lower layer, the link layer, deals with a single communication path. The link layer is called LAP B, for "Link Access Procedure – Balanced." The link layer is the only layer implemented on the V Series, but Hayes hints that the packet layer will be added in future products, along with multiple connection capability. Furthermore, Hayes has recently proposed adding what it terms AFT, or Asynchronous Framing Technique, to the CCITT. AFT is an extension of LAP B that will replace the lowest level framing and error-detection aspects of the protocol and extend them to asynchronous links.

As previously discussed, the other major proprietary feature of the V series modem is adaptive compression. The modems constantly monitor the type of data being transmitted and revise their compression tables so as to maximize transfer efficiency. This allows an effective throughput approaching 19,200 bps.

Test Results

We had an opportunity to test a pair of Hayes modems on several occasions to make transmissions between New York City and northern New Jersey using AT&T long-distance lines. We verified that a vanilla IBM PC- or XT-compatible running at 4.77 MHz is not fast enough to feed the modem characters properly and thereby maximize throughput, and we underscore the value of 8 MHz-or-better 80286 or 80386 machines for such endeavors.

Our communications program of choice for these tests was Omen Technology's Pro-Yam. Although complex and not very user-friendly (a command-line mode and no menus), Pro-Yam is one of the most powerful and versatile programs available for DOS.

With both computers connected to the modems at 9600 bps, we averaged better than 1700 characters per second using Zmodem*, a streaming protocol that is optimized for error-free links such as X.PC, MNP, or, in this case, a proprietary hardware link. These transfers were of ASCII files. Interestingly, throughput decreased dramatically using Zmodem on Hayes' units to transfer binary files. The only logical explanation is some sort of negative interaction with the Hayes proprietary compression algorithms. Ymodem protocol, which is basically a superset of Xmodem using a 16-bit CRC calculation and a 1024-byte block size, came in at under 900 characters per second (irrespective of the type of data in the file). It is not surprising that it was slightly less efficient because of the handshake every 1024 characters. Simple Xmodem with CRC was much less efficient, averaging about 350 characters per second. Once again, such a result is expected. Strangely, when we tried to link each computer to its modem at 19,200 bps, we ran into lots of retries with the Zmodem protocol. We were able to lower the sending end to 9600 while maintaining the receiving end at 19.2 Kbps, but this did not seem to have a dramatic effect on the throughput.

X.25 was conceived for use between synchronous terminals and public data networks . . .

The Telebit Trailblazer Plus

Telebit is a relatively small player in the modem world. It introduced the Trailblazer in the summer of 1985, and has been continually refining it. The model tested, called the Trailblazer Plus, is about twice the size of the Hayes. Like the Hayes, the modem uses half-duplex communication and simulates full-duplex operation by a rapid turnaround. This ping-ponging is very effective in giving the illusion of a full-duplex circuit, but, of course, it is not as efficient as a genuine V.32 link. To be fair, the modem uses some intelligence in determining the ratio of time spent in each direction, depending on the demands of the session. In a typical file transfer, which is an application that lends itself well to the use of high-speed modems, most of the data flows in one direction, so this scheme is quite successful.

The Trailblazer has a novel approach for maximizing data throughput. A standard asynchronous full-duplex modem divides the available frequency spectrum of the voice line into two bands, one for each direction. Different combinations of phase shifts and amplitude variations allow up to four bits per baud in the V.22 bis scheme as implemented in full-duplex 2400 bps modems. However, the Trailblazer divides the audio frequency range into 512 separate carriers, evaluates the signal quality at each frequency, and typically labels about 400 carriers as usable. The modem then assigns 2, 4, or 6 bits per baud to each carrier, once again based on this signal quality evaluation.

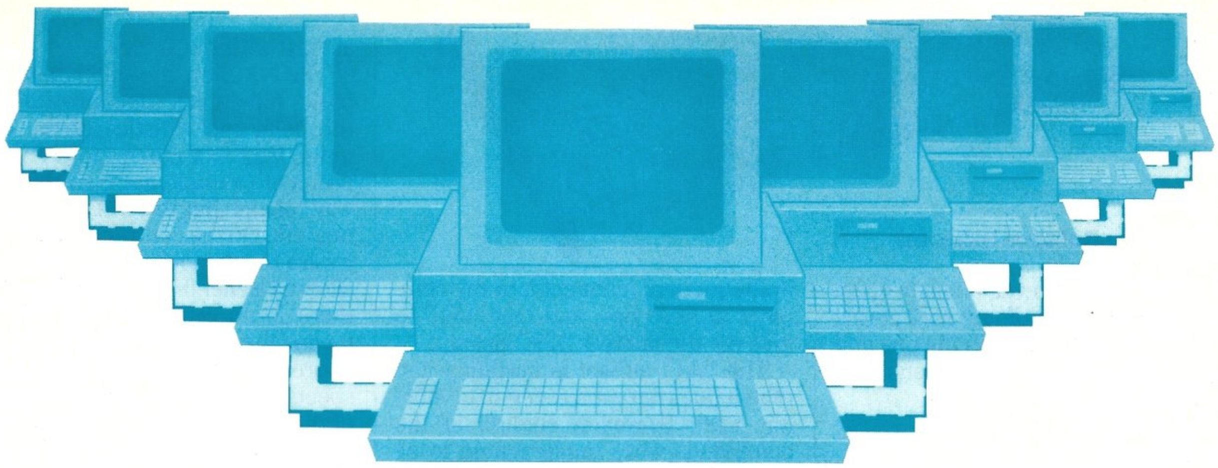
This elegant approach is augmented by another novel approach introduced in the Plus series called protocol "spoofing." Simply put, two Trailblazer modems are connected in error-free mode using their own proprietary protocol. The computer on either end is fooled, however, into using one of the standard protocols such as UUCP, Xmodem, or Kermit. Thus the computer is communicating with the local modem using such a protocol with virtually instantaneous "ACK-ing" as blocks are transmitted. The local computer has no way of knowing that these ACKs are coming from the local modem rather than across the communications line. Indeed, it is quite possible to have a computer at one end employing Xmodem software protocol while the distant computer is employing Kermit protocol.

Protocol spoofing allows easy use of existing communications software, and minimizes turnaround delays since the actual detection and correction are handled transparently by the proprietary modem-to-modem link. An added plus is the possibility of using the Trailblazer as a sort of protocol converter, since the spoofing need not be identical on either end. The simplicity and effectiveness of this technique are tributes to the designers.

With respect to the balance of features and operating techniques, the Trailblazer is quite similar to the Hayes. It, too, offers lower speed operation that is compatible with the accepted standards, local terminal-to-modem speed that is higher than modem-to-modem speed (to maximize throughput when using the built-in data compression), feature negotiation and retraining, and so on. The Telebit does offer some interesting S registers that allow analysis of line conditions in either direction, either averaged or instantaneous. In addition, MNP protocol is available in non-PEP (slow) mode.

Telebit does license its PEP technology, and at least one other manufacturer, Ven-Tel, offers compatible

* For an in-depth discussion of Zmodem, refer to "In the Public Domain," *Micro/Systems*, July 1988, pp. 64 – 66.



Tomorrow's LAN Solutions – Today.

EtherLAN Plus™/The Affordable Ethernet Solution

In today's LAN world, every new LAN manufacturer seems to have a new idea of what a network should be. Standards for the industry are few and far between, but are there if you want to pay for them. Wouldn't it be nice if for once you could buy a LAN system, at a price you can afford, and be fully compatible with the ethernet standard? Well now you can!

Turn Key Ethernet

EtherLAN is here to give you everything you need to set up an ethernet LAN system. You'll receive the EtherLAN network adapter, 25 feet of thinwire ethernet coax cable, T connector and easy to understand Installation and User's Guides, all for one low price. In addition to this, you'll also receive complete software needed to get your new EtherLAN system up and running quickly. You'll get NETBIOS software and also a complete network operating system to make using your new EtherLAN system even easier.

SimpleWARE™ Makes EtherLAN Unbeatable

The operating system you'll get works as a shell above DOS, so all your normal DOS programs will run on EtherLAN as well as locally. Even better yet, you can use standard DOS commands to control your network, so you are not stuck spending hours to learn a new set of commands. The operating system functions transparently, so you won't even know you're on a network.

Features You Can Rely On.

- Fully ethernet and cheapernet compatible (IEEE 802.3).
- Fast — a full 10,000,000 bit per second.

EtherLAN 6 Node Comparison

500 Kbytes	Read and Write	Retail
EtherLAN	11 and 10 seconds	$4199
3Com®	14 and 12 seconds	$5670*
Novell SFT®	10 and 9 seconds	$8265*

*Figures from PC Magazine.

- NO dedicated servers are required.
- Transparent operation —operates as a shell above DOS.
- Easy to use DOS type commands or pop up menus.
- Share disks, subdirectories, printers and plotters.
- DOS file and record locking.
- Up to 5 printers per server.
- Time and Date sharing.
- Electronic Message System.
- Low memory requirements.
- NETBIOS compatible.
- Everything you need is included at one low price.

The best news of all is the price.

At only $699.95 per node, you get all hardware, software, cabling and manuals needed.

Introductory Offer

Four Node Kit for only $2499.00. Save over $300.00.

With features like these, and an affordable price per node, how can you lose?

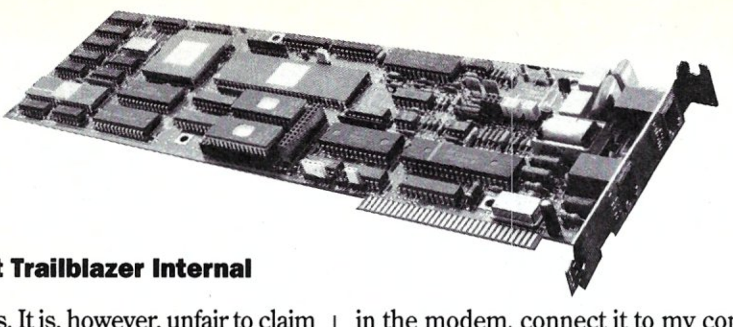
Call Toll Free
1-800-262-8010
1-714-529-8850 (in CA)

VISA VISA, M/C, COD, or Prepaid orders accepted. MasterCard

Simple Net Systems, Inc.
545 W. Lambert Rd., Suite A
Brea, CA 92621
FAX: (714) 529-2413

Requires IBM PC/XT/AT or compatible, running DOS 3.10 or higher. EtherLAN is a registered trademark of Simple Net Systems, Inc. Other brand and product names are trademarks of their respective holders.

CIRCLE 123 ON READER SERVICE CARD



Telebit Trailblazer Internal

modems. It is, however, unfair to claim PEP as a standard of any sort at this point. This leaves us with the one and only high-speed, asynchronous protocol extant, namely V.32.

Test Results

We ran file transfer tests on the Trailblazer much like those on the Hayes, this time between New York City and Philadelphia. In the cases where protocol spoofing was not used, we obtained results similar to the Hayes, namely transfers on the order of 900 characters per second. The use of Xmodem protocol spoofing raised the throughput dramatically to better than 1300 characters per second. When we tried Xmodem spoofing and hardware compression on an ASCII file (which, as mentioned above, will generally benefit more than a binary file), the transfer rate increased to better than 1600 characters per second.

The NEC DSP9630

This modem represents a radical departure in concept from the previous two products discussed here. The DSP9630 is a two-wire, full-duplex, V.32 modem. It is physically imposing compared to the others, measuring about 15 × 12 × 3-inches and weighing almost 14 pounds! Although the price is on the same order of magnitude, it is clear that this modem is directed primarily toward the mainframe market. It has a bewildering selection of buttons and indicator lamps as well as an alphanumeric display. The manual is written in terse, extreme "modem-ese." This modem is certainly not meant for the neophyte.

The actual operation is rather straightforward, certainly moreso than either the Hayes or the Telebit. This is in part because the DSP9630 is a simple 9600 bps modem (with fall-back to 4800 bps). It does not offer 103, 212, or V.22 operation, so there is less to fiddle with. While the plethora of switches and function keys allow all sorts of options —including leased-line operation with automatic fall-back to dial back-up, passworded mode change from a remote modem, stored telephone numbers, and a variety of test modes —the NEC modem offers a subset of the Hayes "AT" command set. I was able to plug in the modem, connect it to my computer at 9600 bps, and type "AT" to give me the familiar "OK" prompt.

The "AT" subset is certainly just that; it does not even recognize "ATZ" for reset. However, enough of the commands were implemented to allow us to use Pro-Yam to communicate with no trouble. There were, however, a few kinks. My partner, David Kozinn, had all kinds of trouble getting his 9630 to work. Every time he plugged his data line connector into the modem, it went off-hook into an endless mode of retraining, disconnecting, going off-hook, and so on. We feared the modem was defective, but as a last resort, David tried his voice line and it worked fine. Strangely, he uses the data line for his normal communications and used it for the other high-speed modem tests we performed as well without problems.

Test Results

Once we passed the initial hurdle, operation was straightforward. We connected at 9600 bps with no trouble and never saw a single stray character, either through AT&T or MCI lines. Zmodem protocol gave an average of between 900 and 950 characters per second, while Xmodem-1K offered about 850 characters per second and Xmodem less than 600 characters per second. Unfortunately, even V.32 modems are still pretty rare, so we were not able to call other locations to check for compatibility with other brands. There is no reason to think that the 9630 won't communicate with a non-NEC V.32 modem.

Conclusion

All three modems were used on a daily basis for several months. After the complexities of initial setup (which were tougher with the Hayes and the Telebit than with the NEC unit), all proved reliable and delivered good performance. The Trailblazer has the most impressive design concept with the combination of a microprocessor-based unit, allowing a simple ROM change to alter basic characteristics, the protocol spoofing, and the PEP scheme.

It is clear from our tests, though, that the Hayes unit held its own as far as transfer efficiency is concerned, at least with ASCII files.

The NEC, on the other hand, is a basic, full-duplex modem without all the bells and whistles of the other units. V.32 is a much more complex protocol to implement, so this in no way means that the NEC is a simpler modem. On the contrary, its electronics are much more complex, and they allow a cleaner firmware design. True, there is no error-free link, but this means the software must handle this chore as is done with today's lower speed non-MNP modems. The ability of the 9630 to communicate with any V.32 modem is a very significant plus in my opinion.

If an application calls for the purchase of multiple modems, then of course any proprietary scheme will do. At present, many UUCP nodes are using Telebit Trailblazers since the firm was far-sighted enough to offer special price incentives to these operators. Similarly, Hayes is attempting to become a *de facto* standard by allying itself with Compuserve, which has announced its intention to use the V-Series 9600 bps modems. However, if I were a gambler, I would place my money on the future of established CCITT standards such as V.32 rather than going off on a tangent. I predict that in two or three years, the asymmetric modems will be a thing of the past, but V.32 will be very much with us. In the meantime, the Tower of Babel remains. □

Did you find this article particularly useful? Circle number 7 on the reader service card.

Product Information

V Series 9600	$1,299

Hayes Microcomputer Products, Inc.
P. O. Box 105203
Atlanta, GA 30348
(404) 449-8791
Circle reader service #262

Trailblazer Plus	$1,345
Internal Version	1,195

Telebit Corporation
1345 Shorebird Way
Mountain View, CA 94043
(800) TELEBIT
Circle reader service #263

NEC DSP9630	price unavailable

NEC America
Technical Support
1255 Michael Drive
Wood Dale, IL 60191
(800) 826-2255
Circle reader service #264

C CODE FOR THE PC

source code, of course

	MS-DOS File Compatibility Package (create, read, & write MS-DOS file systems on non-MS-DOS computers)	$500
	Bluestreak Plus Communications (two ports, programmer's interface, terminal emulation)	$400
	CQL Query System (SQL retrievals plus windows)	$325
	GraphiC 4.1 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$250
	Greenleaf Data Windows (windows, menus, data entry, interactive form design)	$220
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF)	$175
	TurboTEX (TRIP certified; HP, PS, dot drivers; CM fonts; LaTEX)	$170
	Essential resident C (TSRify C programs, DOS shared libraries)	$165
	Greenleaf Functions (296 useful C functions, all DOS services)	$160
	Essential C Utility Library (400 useful C functions)	$160
	Essential Communications Library (C functions for RS-232-based communication systems)	$160
	WKS Library Version 2.0 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	ME Version 2.0 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$130
	TurboGeometry (library of routines for computational geometry)	$125
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$115
	Minix Operating System (U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP implementation for PCs)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
NEW!	Tele Operating System (TeleKernel, TeleWindows, TeleFile, & TeleBTree by Ken Berry)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
	Wendin Operating System Construction Kit or PCNX, PCVMS O/S Shells	$80
	C Windows Toolkit (pop-up, pull-down, spreadsheet, CGA/EGA/Hercules)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
NEW!	Polyglot Lisp-to-C Translator (includes Lisp interpreter, Prolog, and simple calculus prover)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	lp (flexible printer driver; most popular printers supported)	$65
	Quincy (interactive C interpreter)	$60
	PTree (parse tree management)	$60
NEW!	Icon-Tools (full-featured icon display and editing system)	$60
NEW!	Polyglot TSR Package (includes reminder, bookmark, virus catcher, cache manager, & speech generator)	$50
	MicroFirm Toolkit (28 Unixesque utilities for MS-DOS)	$50
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$50
	HELP! (pop-up help system builder)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; uses Galacticomm modem card)	$50
	Make (macros, all languages, built-in rules)	$50
	Vector-to-Raster Conversion (stroke letters & Tektronix 4010 codes to bitmaps)	$50
	Coder's Prolog (inference engine for use with C programs)	$45
	Virtual Memory System (least recently used swapping)	$40
	C-Notes (pop-up help for C programmers ... add your own notes)	$40
NEW!	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Biggerstaff's System Tools (multi-tasking window manager kit)	$40
	PC-XINU (Comer's XINU operating system for PC)	$35
	CLIPS (rule-based expert system generator, Version 4.2)	$35
	Tiny Curses (Berkeley curses package)	$35
NEW!	Polyglot RAM Disk (change disk size on the fly; includes utilities)	$30
	SP (spelling checker with dictionary and maintenance tools)	$30
	Clisp (Lisp interpreter with extensive internals documentation)	$30
	Translate Rules to C (YACC-like function generator for rule-based systems)	$30
	6-Pack of Editors (six public domain editors for use, study & hacking)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
NEW!	Pascal Compiler & Interpreter (P-codes, standard Pascal)	$25
	ICON (string and list processing language, Version 7)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX)	$25
	LEX (lexical analyzer generator; an oldie but a goodie)	$25
	Bison & PREP (YACC workalike parser generator & attribute grammar preprocessor)	$25
	AutoTrace (program tracer and memory trasher catcher)	$25
NEW!	Data Handling Utilities in C (data entry, validation & display; specify Turbo C or Microsoft)	$25
	Arrays for C (macro package to ease handling of arrays)	$25
NEW!	ANSI Forms (forms manager based on ANSI codes)	$20
	C Compiler Torture Test (checks a C compiler against K & R)	$20
	Benchmark Package (C compiler, PC hardware, and Unix system)	$20
	TN3270 (remote login to IBM VM/CMS as a 3270 terminal on a 3274 controller)	$20
	A68 (68000 cross-assembler)	$20
	List-Pac (C functions for lists, stacks, and queues)	$20
	XLT Macro Processor (general purpose text translator)	$20
NEW!	KeySwap (swaps CapsLock and CRTL key functions on XTs and ATs)	$20
NEW!	OOPS (collection of handy C++ classes by Keith Gorlen of NIH)	$20
	C/reativity (Eliza-based notetaker)	$15

Data

DNA Sequences (GenBank 52.0 including fast similarity search program)	$150
Protein Sequences (5,415 sequences, 1,302,966 residuals, with similarity search program)	$60
Dictionary Words (234,932 words in alphabetical order, no definitions)	$60
U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
USNO Floppy Almanac (high-precision moon, sun, planet & star positions)	$20
NBS Hershey Fonts (1,377 stroke characters in 14 fonts)	$15
U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

acw!info@uunet.uu.net

CIRCLE 62 ON READER SERVICE CARD

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping on prepaid orders   For delivery in Texas add 7%   MasterCard/VISA

PC-Based Communication Using Interrupts

by Bill E. Swafford

A hardware interrupt program like this one can be useful in optimizing data communications.

When a hardware interrupt occurs in an 8088-based microcomputer system, the flags, CS, and PC registers are pushed onto the stack. Then the CS and PC registers are loaded with the values contained in the interrupt vector associated with the hardware element that caused the interrupt. This means that the next instruction to be executed will be the first instruction of the hardware element's interrupt handler. The interrupt handler saves all the registers that it will modify, performs its function, restores all saved registers, and issues an IRET instruction. The IRET instruction pops the stack three times and places the values in the PC, CS, and flags registers, in that order. The program that was interrupted thus continues to run with no ill effect from the interruption.

This interrupt potential can be exploited to optimize data communication by defining an interrupt handler as a producer process that buffers incoming data and runs asynchronously with a consumer process that deals with the data. To appreciate the validity of this assertion, we must consider how polling and interrupt-driven systems differ. In a polling system, a check-if-ready test is made in a loop until a character is available from the input port. When it is available, the character is read from the input port and processed. If there is extensive processing to be performed, data may be lost before the next check-if-ready test is made. In an interrupt-driven system, the interrupt handler is the producer process that takes characters from the input port and places them in a buffer so they can be consumed by the consumer process as it performs its various functions, such as editing for special characters, display, and so on. Since the interrupt handler has so little to do, it can do it very quickly and hence can buffer incoming data at a higher rate of speed than can be done in a polling environment.

It doesn't take much thought to understand that this producer/consumer relationship will work only as long as there is room in the buffer to hold incoming data. If the consumer process processes the incoming data much more slowly than it is produced, then a buffer of infinite length would be required. However, there is a simple fact that saves the day here. Most communication occurs in burst mode, in which data is sent in blocks of finite length. Also, the sender, having sent one block, will not send another until an "okay" on the previous block is sent by the receiver. Clearly, this allows an error checking scheme to be implemented between the sender and the receiver. Very often, instead of or in conjunction with this chain of events, before sending each character, the sender checks for a special value (XOFF) transmitted by the receiver that signals that the sender is not to send another character until a second special value (XON) is sent by the receiver.

In order to use this producer/consumer model as the basis for a communications system, there must be a hardware communications element capable of generating the appropriate interrupts. Such a device is the INS8250 asynchronous communications which is used to support serial communications in the IBM PC. All IBM PCs and true compatibles provide full interrupt support for two serial ports, COM1 and COM2, using the 8250 chip. There is a slight complication. In the design of the IBM PC, there is an Intel 8259 programmable interrupt controller between the 8250 and the CPU. Interrupts generated by hardware are given to the 8259, which then decides whether, and in what order, to pass them on to the CPU.

Bill E. Swafford is an associate professor with the Department of Applied Computer Science, Illinois State University, in Normal, Illinois.

The decision to pass an interrupt on to the CPU is based on the previous programming of the 8259, while the order in which interrupts are presented to the CPU is determined by the priority of the hardware element generating the interrupt. In the IBM PC, the priority for the hardware interrupts, high to low, is: interval timer, keyboard, COM2, COM1, hard disk, floppy disk, and printer.

The software presented here is a complete environment that uses interrupt-driven serial communications for input with either COM1 or COM2. The only output in the system is the use of XON/XOFF logic to stop the sender when the interrupt buffer is within 256 bytes of being full, and to restart the sender when the interrupt buffer is only half full. These are special cases and do not explain the check-if-ready logic that was just recommended for output. In the hope that it can form the basis for a communication system for interested programmers, C source code, which indicates how this logic is implemented, is included in Listing 3. All C code presented here has been tested using Version 2.1 of the Datalight C Compiler, and all assembler code has been assembled using Version 4.0 of the Microsoft Macro Assembler. In the discussion that follows, hardware details are relevant only to the IBM PC and true compatibles. Complete details concerning interrupts and the 8088 can be found in Osborne (1978) and Rector and Alexy (1980).

The 8259

The 8259 programmable interrupt controller must be initialized before the 8250 is initialized. This is because the 8259 is a leading edge-triggered device, and thus recognizes an interrupt only by its leading edge.

The 8259 is attached to the I/O bus of the 8088 via two ports, the control port at 020 hex and the mask port at 021 hex. The 8-bit value in the mask port determines which interrupts, if raised, will be passed on to the CPU. The two communication interrupts are governed by the two bits exactly in the middle of the 8-bit field; the leftmost for COM1 and the other for COM2. A value of 0 in a bit means that the interrupt associated with that bit is passed through while a value of 1 means that the interrupt is suppressed. To set the mask so that a particular interrupt is passed through to the CPU without changing previous settings made by DOS or other software, we first read the mask port and AND its value with a byte that has all 1s except in the position of the interrupt we wish to activate. We then write this new mask to the mask register at port 021 as shown in Listing 2.

The control register at port 020 must also be used by this software. After an interrupt has been serviced, an end-of-interrupt signal must be sent to the 8259 via its control register. If this is not done, no further interrupts will be passed on to the CPU. Thus, it is the responsibility of the interrupt handler to write this signal, 20 hex, to the control register of the 8259. Complete details on the 8259 can be found in Osborne (1978).

The 8250

The 8250 is attached to the I/O bus of the 8088 at base port address 3F8 for COM1 and 2F8 for COM2. The

Listing 1. The main PC communication interrupt module.

```
#include <stdio.h>
#include <dos.h>

union REGS inregs,   /* needed by kbhit() */
           outregs;  /*  (REGS defined in dos.h) */

char *intbuff;       /* address of buffer queue */
int front, rear;     /* front and rear of queue */
int qlength;         /* size of buffer */
int overs;           /* buffer overflow count */
int bcount;          /* buffer character count*/
int port;            /* COM1 at 3F8 & COM2 at 2F8 */
int whichcom;        /* COMX is COM1=12 or COM2=11 */
char lowbyte;        /* baud rate divisor low byte */
char highbyte;       /* baud rate divisor high byte */

main()
/*
  This code is included to provide documentation of
system initialization and proper use of the assembler
routines.
*/
{
  int i;
  char buffer[4096];
  char c;
  extern int queout();
  extern int empty();
  extern int intson();
  extern int intsoff();

  intbuff = &buffer[0];      /* initialize externals */
  qlength = sizeof( buffer );/*  before assembler */
  front = qlength;           /*  routines are called */
  rear = front;
  bcount = 0;
  overs = 0;
  port = 0x03f8;            /* 0x02f8 if COM2 */
  whichcom = 12;            /* 11 if COM2 */

  setbaud();                /* ask user for baud rate */
  intson();                 /* initialize the system */
  while( !kbhit() ) {       /* loop until key pressed */
    if( !empty() ) {          /* when data available */
      c = queout();             /* get next byte */
      putchar(c);               /* display it */
    }
  }
  getch();                  /* clear the keyboard */
  intsoff();                /* interrupts off */
}

int kbhit()
/*
   kbhit returns 1 if a key is pressed, 0 otherwise
*/
{
  int ret;

  inregs.h.ah = 1;
  ret = int86(22,&inregs,&outregs); /* Datalight C */
  ret = ( (ret & 0x0040) == 0 ) ? 1 : 0;
  return (ret);
}

setbaud()
/*
  Baud rate divisors are selected by the user.  The
  actual baud rate divisors are in the table labeled
  divisors in the assembler module interrpt.asm.
*/
{
  int  i, in = 0;
  extern char divisors[];

  while( (in < 'A') || (in > 'P') ) {
    printf("\n\nSet baud rate to\n\n") ;
    printf("    A.    50 \n");
    printf("    B.    75 \n");
    printf("    C.   110 \n");
    printf("    D.   135 \n");
    printf("    E.   150 \n");
    printf("    F.   300 \n");
    printf("    G.   600 \n");
    printf("    H.  1200 \n");
    printf("    I.  1800 \n");
    printf("    J.  2000 \n");
    printf("    K.  2400 \n");
    printf("    L.  3600 \n");
    printf("    M.  4800 \n");
    printf("    N.  7200 \n");
    printf("    O.  9600 \n");
    printf("    P. 19200 \n");
    printf("\nEnter baud rate choice here --> ") ;

    in = getche();
    in = toupper(in);
    putchar('\n');
  }
  i = (in - 'A') * 2;
  highbyte = divisors[i];
  lowbyte = divisors[i+1];
}
```

Listing 2. The assembler source code.

```
;  This library contains the four routines queout,
;  empty, intson, intsoff, as well as the interrupt
;  handler insertit and the baud rate divisor table,
;  divisors.  The routines have been assembled and
;  tested with the Microsoft MASM assembler, version
;  4.0.  This code conforms to the assembler interface
;  for the Datalight C compiler, and has been tested
;  within a C environment using version 2.10 of the
;  compiler.  Rights to use this code in way desired or
;  to reproduce it are granted, provided that the
;  author's name remains on all coppies.
;
;                Bill E Swafford
;                Applied Computer Science Department
;                Illinois State University
;
dgroup   group   data
data     segment word public 'data'
         assume  ds:dgroup
;
;  Global variables defined in the  main C module
;
         extrn  intbuff:word
         extrn  front:word
         extrn  rear:word
         extrn  qlength:word
         extrn  overs:word
         extrn  bcount:word
         extrn  port:word
         extrn  whichcom:word
         extrn  lowbyte:byte
         extrn  highbyte:byte
;
;  Baud rate divisor table.  The first byte of each
;  pair represents the high byte of the divisor.
;
         public divisors
divisors db   9,0                  ; 50  baud
         db   6,0                  ; 75
         db   4,23                 ; 110
         db   3,89                 ; 134.5
         db   3,0                  ; 150
         db   1,128                ; 300
         db   0,192                ; 600
         db   0,96                 ; 1200
         db   0,64                 ; 1800
         db   0,58                 ; 2000
         db   0,48                 ; 2400
         db   0,32                 ; 3600
         db   0,24                 ; 4800
         db   0,16                 ; 7200
         db   0,12                 ; 9600
         db   0,6                  ; 19200
data     ends
;
pgroup   group   prog
prog     segment byte public 'PROG'
         assume  cs:pgroup
;
;  queout()
;  usage:  c = queout();
;          c is of type char or int
;  queout removes a byte from the circular queue and
;  sends XON when necessary.
;
         public queout
queout   proc near
;
         cli                 ; interrupts off
         mov  bx,front   ; get get front pointer
         cmp  bx,qlength ; time to wrap?
         je   reset
         inc  bx         ; if not, just increment
         inc  front      ; ... in both places
         jmp  getdata
reset:   mov  bx,0       ; wrap back to start
         mov  front,bx
getdata: mov  ah,0       ; zero upper byte
         mov  cx,intbuff
         add  bx,cx      ; point to front of queue
         mov  al,[bx]    ; get current data byte
         dec  bcount     ; correct byte counter
         mov  bl,byte ptr cs:xonoff ; XOFF flag
         cmp  bl,0       ; have we sent XOFF ?
         je   notyet
         mov  bx,qlength ; should we send XON?
         mov  dx,bx
         ror  dx,1       ; divide qlength by 2
         sub  bx,dx      ; check for < half full
         cmp  bx,bcount
         jb   notyet
         mov  bl,0
         mov  cs:byte ptr xonoff,bl ;turn off flag
         push ax         ; save accumulator
         mov  al,17      ; XON
         mov  dx,port    ; name of port
         out  dx,al
         pop  ax         ; get accumulator back
notyet:  sti             ; interrupts on
         ret
;
;  empty()
;  usage:  empty()  or  !empty();
;
;  empty returns 1 if the queue is empty, 0 otherwise
;
         public empty
empty:   cli             ; interrupts off
         mov  ax,front   ; get front pointer
         cmp  ax,rear    ; is the queue empty?
         je   itis
         mov  ax,0       ; queue is not empty
         jmp  return
itis:    mov  ax,1       ; queue is empty
return:  sti             ; interrupts on
         ret
;
;  intsoff()
;  usage: intsoff();
;         no return value
;  intsoff disables communication interrupts passed
;  through the 8259 and disables the generation of
;  interrupts by the 8250.  It also restores the
;  interrupt vector to its state before this system
;  was started.
;
         public  intsoff
intsoff: cli             ; interrupts off
         mov  dx,21h
         in   al,dx      ; 8259 mask
         and  al,0E7h    ; disable COMM interrupts
         out  dx,al
;
         mov  al,00h
         mov  dx,port    ; disable 8250 interrupts
         inc  dx         ; interrupt enable reg
         out  dx,al
;
         mov  dx,word ptr cs:compc ; PC of int handler
         push ds         ; save data segment
         mov  ax,whichcom ; request set int vector
         mov  ah,25h     ; set for COMX interrupt
         mov  ds,word ptr cs:comcs ; CS of int handler
         int  21h        ; get DOS to set vector
         sti             ; interrupts on
         pop  ds         ; restore proper data segment
         ret
;
;  intson()
;  usage: intson();
;         no return value
;  intson preforms most of the initialization for this
;  system.  First the appropriate communication
;  interrupt vector is saved and then set to point to
;  the interrupt handler, insertit.  The data segment
;  register is saved in the code segment so that it
;  can be obtained by the interrupt handler.  Then the
;  8259 is programed to pass the chosen communication
;  interrupt through to the CPU.  Finally, the 8250 is
;  programmed for the chosen baud rate, 8 data bits and
;  1 stop bit, to assert the data-terminal-ready,
;  request-to-send, and OUT2 values, and to generate an
;  interrupt when input data is available.
;
         public  intson
intson:
;     first get old interrupt vector
;
         push es         ; Datalight requires we save es
         mov  ax,whichcom ; set get vector function
         mov  ah,35h     ; ...for COMX interrupt
         int  21h        ; ... DOS returns it in ES:BX
         mov  word ptr cs:comcs,es ; ...set CS
         mov  word ptr cs:compc,bx ; ...set PC
;
;     now set interrupt vector to point at insertit
;
         mov  dx,offset cs:insertit ; PC of int handler
         push ds
         mov  ax,whichcom ; COM1 or COM2?
         mov  ah,25h     ; request set int vector
         push cs         ; requires code segment value
         pop  ds         ; ... in data segment register
         int  21h        ; get DOS to set vector
         pop  ds
         mov  word ptr cs:dssave,ds ; save our DS
         sti             ; interrupts on
;
;     initialize 8259 PIC chip
;
         mov  dx,21h     ; point to 8259 mask register
         in   al,dx      ; get 8259 mask
```

. . . listing continues

discussion that follows refers only to COM1, but the details are the same for COM2 as well, provided that all port values are translated to 2Fx. The accompanying software is set up for COM1, but can easily be converted to work with COM2 by initializing the global variables, *port* and *whichcom*, to the appropriate values.

Before the 8250 can be used it must be initialized. The baud rate, the number of stop bits and data bits, and what modem signals to assert must be set. This process entails writing certain values (bytes) to specific control registers on the 8250. These control registers can be accessed as ports on the I/O bus of the PC. The specific control registers of interest here are the data (both input and output), the line control, the least significant and most significant baud rate divisor, the modem control, and the interrupt enable registers. These registers are addressed by 3F8, 3FB, 3F8, 3F9, 3FC, and 3F9, respectively. Duplicates in this list are explained by the fact that values written to the line control register can alter the function of certain other control registers.

To initialize the 8250, we first write 80 hex to the line control register, and then write the low and high bytes to the baud rate divisor registers to set the desired baud rate. After the baud rate is set, we write 07 to the line control register. Because the high order bit is 0, 3F8 becomes the address of the data port and 3F9 becomes the address of the interrupt enable register. The digit 7 in 07 programs the 8250 to generate or expect 8 data bits and 1 stop bit for each character. Writing 0B hex to the modem control register asserts the clear-to-send, data-terminal-ready, and OUT2 signals. The first two values may or may not be needed, depending on your modem settings and cable. However, the OUT2 signal must be set in order for an interrupt to occur [see IBM Corporation (1983)]. We then enable the generation of an interrupt when an incoming byte is completely assembled, by writing 01 to the interrupt enable register. Details regarding programming the 8250 are fully explored in Kane (1978) and IBM Corporation (1983).

Software Overview

The main module (Listing 1) consists of definitions, the main driver, and two subroutines, *setbaud* and *kbhit*. *setbaud* is a simple program that queries the console to determine the baud rate desired by the user so that the global variables *lowbyte* and *highbyte* can be set to the proper values. Together these values represent a two-byte baud rate divisor. *kbhit* uses a BIOS interrupt to determine if a key has been hit by the user.

The main driver initializes the global variables that determine the communication port to be used, defaults the interrupt buffer to the empty state, and calls *setbaud* to determine the baud rate. Now that all global variables have been set, the assembler initialization routine, *intson*, is called to prepare the interrupt environment for the interrupt handler, *insertit*, and initialize the hardware elements. At this point, we loop until a key is pressed, obtaining a character from the buffer, when available, and sending it to the console display screen. When a key is pressed, the loop is terminated and the assembler clean-up routine, *intsoff*, is called.

The assembler source (Listing 2) contains the subroutines *queout*, *empty*, *intson*, and *intsoff*, as well as the interrupt handler *insertit*. *intson* saves the old interrupt vectors, sets new ones to point at the interrupt handler *insertit*, and initializes the 8259 and the 8250. *intsoff* restores the old interrupt vectors and reprograms the 8250 and the 8259 to terminate the generation of communication interrupts. The routine *empty* performs the very important function of synchronizing the consumer proc-

Tools for C

C-terp

"... C-terp is a nicely balanced, integrated-development environment"
-- Robert Ward, Debugging C

"... easy to use, powerful, and a time saver"
-- Larry Jordan, The C Journal

"C-terp is a high-power interpretive environment ... an excellent product."
-- Computer Language

"C-terp comes closest to this ideal [C development environment]"
-- John Unger, BYTE

For MS-DOS, specify compiler
$298.00 Microsoft, Turbo, Lattice, Aztec, C86
$350.00 MS-DOS Combo (C-terp and PC-lint)
$398.00 Xenix 286 / 386

PC-lint

"PC-lint has everything going for it: flexibility, speed, good documentation, and a reasonable price" -- Stephen D. Cooper, San Francisco PC Users Group

"... a remarkable well thought-out product which will check for just about every conceivable coding error" -- Andrew Binstock, The C Gazette

"Gimpel has provided ways to make Lint shut up about all those errors you either know or don't care about" -- Don Malpass, IEEE Software

Only **$139** first copy, **$100** each additional
Specify MS-DOS or OS/2

Generic Lint

Lint in shrouded source form for use on your mainframe or mini, including OS-9, VAX/VMS, Unix, IBM VM/CMS - MVS, etc. Requires only K&R C to compile. Pricing starts at $798. Call for details.

(215)584-4261

CALL TODAY

30 Day Money-back Guarantee

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426

PA residents add 6% sales tax. Outside USA add $20.

C-terp, PC-lint, Generic Lint are trademarks of Gimpel Software.

CALL ADVERTISER DIRECTLY

ess with the interrupt handler by telling it when data is available for consumption. The routine *queout* assumes that data is in the queue, so *empty* should be called first to determine if it is safe to call *queout*. The interrupt handler *insertit* and the two routines *empty* and *queout*, are the only modules that reference the buffer. This buffer is managed as a circular queue as described in Augenstein and Tenenbaum (1979). □

Did you find this article particularly interesting? Circle number 8 on the reader service card.

References

Augenstein, M., and A. Tenenbaum. *Data Structures and PL/1 Programming*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

IBM Corporation, *Technical Reference for the IBM Personal Computer*. 1983.

Kane, J., with A. Osborne. *Some Real Support Devices, An Introduction to Microcomputers*. Vol. 2 – 3. Berkeley, CA: Osborne & Associates, 1978.

Rector, R., and G. Alexy. *The 8086 Book*. Berkeley, CA: Osborne/McGraw-Hill, 1980.

```
         mov  bx,12
         cmp  bx,whichcom ; COM1 or COM2 ?
         jne  itsbaker
         and  al,0efh   ; enable IRQ4 for COM1
         jmp  onward
itsbaker: and al,0f7h   ; enable IRQ3 for COM2
onward:  out  dx,al     ; reset 8259 mask
;
;       initialize 8250 chip
;
         mov  al,80h    ; set divisor latch access
         mov  dx,port   ; point to serial port
         add  dx,3      ; increment to line control reg
         out  dx,al
;
         mov  al,lowbyte ;low byte for divisor latch
         mov  dx,port   ; divisor latch least sig bit
         out  dx,al     ; set low value
;
         inc  dx        ; divisor latch most sig bit
         mov  al,highbyte ; ... rate to requested speed
         out  dx,al     ; set high value
;
         mov  al,07h    ; set 8 bits & 1 stop bit
         mov  dx,port   ; set address
         add  dx,3      ; ... of line control register
         out  dx,al
;
         mov  al,0bh    ; set DTR & RTS & OUT2
         mov  dx,port   ; set address
         add  dx,4      ; ... of modem control register
         out  dx,al
;
;       enable 8250 interrupt
;
         mov  al,01h    ; 8250 receive data available
         mov  dx,port   ; set address
         inc  dx        ; ... of interrupt enable reg
         out  dx,al
;
         pop  es        ; restore es for Datalight
         ret
;
;  This data area is actually in the code segment
;  because the interrupt handler must be able to load
;  the correct value for the data segment before it
;  attempts to address variables.  Only the data
;  segment need be here.
;
dssave:  nop            ; save area for DS reg
         nop
xonoff:  nop
comcs:   nop            ; save area for COMX CS
         nop
compc:   nop            ; save area for COMX PS
         nop
;
;
;  insertit (interrupt handler)
;  usage: insertit is NEVER called.
;
;  insertit reads the input port and places the data in
;  the next spot in the circular buffer addressed by
;  intbuff and rear.  If the buffer is full, insertit
;  disregards the data and increments overs.  insertit
;  checks to see if the buffer is within 256 bytes of
;  being full and sends an XOFF if so.
;
         public  insertit
insertit:
         sti            ; allow other interrupts
         push ax        ; save all the registers used
         push bx
         push cx
         push dx
         push ds
;
         mov  ds,word ptr cs:dssave ; restore DS reg
         mov  dx,port   ; set addr of trans & recv reg
         in   al,dx     ; get data byte
         mov  bx,rear   ; get rear pointer
         cmp  bx,qlength ; should we reset?
         jne  itsok
         mov  bx,0      ; back to start of buffer
         mov  rear,bx   ; save it for real
         jmp  noinc
itsok:   inc  bx        ; new rear pointer
noinc:   cmp  bx,front  ; is the queue full?
         je   bovers
         mov  rear,bx   ; save hew rear pointer
         mov  cx,intbuff ; point to buffer
         add  bx,cx     ; bump to current byte
         mov  [bx],al   ; place data in buffer
         inc  bcount    ; increment byte count
         mov  bx,qlength ; should we send XOFF?
         sub  bx,256    ; subtract cow catcher
         cmp  bx,bcount
         ja   retint
         push ax        ; save accumulator
         mov  al,19     ; XOFF
         mov  dx,port   ; name of port
         out  dx,al
         mov  al,1      ; set XOFF flag
         mov  cs:byte ptr xonoff,al ; and save it
         pop  ax        ; get accumulator back
         jmp  retint
bovers:  inc  overs     ; count as buffer overflow
retint:
         cli            ; interrupts off
         mov  al,20h    ; point to control reg of 8259
         out  20h,al    ; tell 8259 we are done
         pop  ds        ; restore saved registers
         pop  dx
         pop  cx
         pop  bx
         pop  ax
         iret           ; return from interrupt
;
queout   endp
prog     ends
         end
```

Listing 3. The C source code indicating the logic behind interrupt-driven serial communications for input with either COM1 or COM2.

```
#define xmit_buf_empty   ( inp(port+5) & 0x20 )
#define send_data(value) ( outp(port,value) )

int port = 0x3F8;                  /* COM1 */

send(c)
char c;
/*
  usage: send(c);
         c is type char

  send is a very simple routine that illustrates how to
  use the "check if ready" logic to send data through
  the 8250 after it has been initialized.  The #define
  statements for xmit_buf_empty and send_data use
  Datalight C builtin functions inp and outp to access
  the 8250.
*/
{
  while( !xmit_buf_empty )      ;  /* wait till it is */
  send_data(c);                    /* send next char */
}
```

Simply the BEST C and Pascal on AT, 386, Sun, Apollo, RT, VAX, 370

"The most rock-solid C compiler in the industry. Superb technical support and portability. Superior code generated."
Gordon Eubanks, Symantec — Q&A (386).

"It simply works, with no trouble, no chasing strange bugs, and excellent warning and error messages ... a professional product."
Robert Lerche, Bay Partners.

"For large-scale software development, the highest quality C compiler available on the market today. Pragmas are great. Quality of support is exceptional." **Randy Neilsen, Ansa—Paradox (DOS,OS/2).**

"15% smaller and 15% faster than Lattice C."
Robert Wenig, Autodesk.

"Our software is running anywhere from 30 to 50% faster than when compiled under Lattice." **David Marcus, Micronetics.**

"We switched from Lattice due to a 10% reduction in code size. The compiler is very stable." **Lee Lorenzen, Ventura Software — Ventura Publisher, marketted by Xerox Corp.**

"Best quality emitted code by any compiler I've encountered. Often amazing." **Bill Ferguson, Fox Software — FoxBase (386).**

"Messages sometimes pointed out type mismatches, incorrect-length argument lists, and uninitialized variables that had been undetected for years [in 4.x bsd]." **Larry Breed, IBM ACIS [RT PC].**

"Diagnostics turned up bugs missed by other compilers. Rapid bug fixes by technical support, someone who knew what he was talking about. 80386 code is well optimized."
Tim Addison, Logistics Data Systems.

"386 protected mode support is fantastic, especially the access to large amounts of memory. It's mainframe compute power on a PC." **Dan Eggleston, Viewlogic.**

"The preprocessor supplied with Professional Pascal is quite useful. The code quality and control over segmentation and memory models are superior to MS Pascal." **Bob Wallace, QuickSoft.**

Check Out These Reviews

- **High C ™:**

Computer Language	February 1986, '87	
Dr. Dobb's Journal	August 1986	
PC Magazine	Jan. 27, 1987	(80386 version)
Dr. Dobb's Journal	July 1987	(80386 version)
BYTE Magazine	November 1987	(80386 version)

- **Professional Pascal ™:**

PC Magazine	Dec. 29, 1985	
Computer Language	May 1986	
PC Tech Journal	July 1986	
Journal of Pascal, Ada, & Modula-2	Nov.-Dec. 1986	
BYTE Magazine	Dec. '86, June '87	(80386 version)

Why MetaWare compilers

- They are specifically designed for serious software developers.
- They are reliable and robust: they don't break at every turn.
- Their generated code is the best, or near best, on each architecture.
- Their superior diagnostic messages help you produce better products more quickly.
- Your source can be ported with ease to the most popular systems.
- You can link mixed-language modules from our compilers, others
- You can benefit from high-level, personal technical support.
- You can take advantage of the latest ANSI C extensions, and/or extensive Pascal extensions. **High C** has been tracking the ANSI Standard for two years; **Professional Pascal** will soon have a VS Pascal compatibility switch and several Apollo Pascal ext'ns.
- You can take advantage of the latest 387 and Weitek 1167 support — we have the only compilers with Weitek real mode support.

Power Tools for Power Users

Ashton-Tate: dBase III Plus, MultiMate; Autodesk: AUTOCAD, AUTOSKETCH (8087, '387, Weitek); Boeing Computer Services (Sun); CASE Technology (Sun); CAD/CAM giant Daisy ('86, '386, VAX); Deloitte Haskins & Sells; Digital Research: FlexOS; GE; IBM: 4.3/RT, 4680 OS; Lifetree Software (Pascal): Volkswriter Deluxe, GEM-Write; Lugaru: Epsilon; NYU: Ada-Ed cmplr; Semantec: Q&A; Sky Computers; ... (Product names are trademarks of the companies indicated.)

Industrial-Strength

MetaWare C and Pascal compilers are designed for professional software developers. These tools are loaded with options to control them for special purposes. You can adjust the space-time trade-off in code quality. You can adjust external naming conventions to agree with linkers and operating systems. You can specify segment names for segmented architectures, and to help place code or data in particular places for embedded applications. You can select from five memory models for the 8086 family. And on and on.

A Partial List of Optimizations

Common subexpression and dead-code elimination, retention and reuse of register contents, jump-instruction size minimization, tail merging (cross jumping), constant folding, short-circuit evaluation of Boolean expressions, strength reductions, fast procedure calls, automatic mapping of variables to registers (where advantageous), ...

"Platform" — Code Quality

Sun,Apollo,SGI	— 18%, 3%, 26% > resident compiler (Dhrystone).
PC: DOS, OS/2	— 3-10% > Microsoft C; 30% > MS Pascal, LatticeC.
386 32-bit DOS	— no competitors, since November, 1986.
286, 386 UNIX	— 66% better than pcc (Dhrystone, 386).
VAX VMS	— ≈ DEC's excellent C and Pascal; better features.
VAX Ultrix	— 19% > pcc (Dhrystone); much > Berkeley Pascal.
RT PC/4.3bsd	— 89% > the original port of pcc (Dhrystone).
370 CMS,UNIX	— much better than any C, and VS Pascal.
AMD 29000	— >40,000 Dhrystones! Available in Q2, cross.

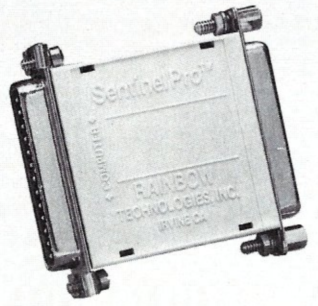
(408) 429-6382, telex 493-0879. ***Since 1979.***

MetaWare™
INCORPORATED
903 Pacific Avenue, Suite 201 • Santa Cruz, CA 95060-4429

The Clear Choice for Large Programming Projects — PC Tech J.

CIRCLE 55 ON READER SERVICE CARD

Hard Locks for Soft Parts.



At Rainbow Technologies, we think protecting software developers' investments is very serious business. That's why we designed the first fully effective security solution for software running on PCs and other computers.

Our family of virtually impenetrable Software Sentinel hardware keys provides the highest level of software protection the developer can get. While remaining invisible to the end user.

Take a look.

Key Sentinel Family Features.

Prohibits unauthorized use of software ▫ No need for copy protection ▫ Unlimited backup copies ▫ Virtually unbreakable ▫ Pocketsize key ▫ Transparent operation ▫ Transportable

Software Sentinel.

- Runs under DOS and Xenix, on IBM PC/XT/AT and compatibles
- Algorithm technique (Never a fixed response)
- Serial or parallel port version
- Minimal implementation effort
- Higher level language interfaces included
- 100 times faster than fixed-response devices (1ms)

Software Sentinel-C.

- For developers who want to customize or protect multiple packages with one device
- 126 bytes of non-volatile memory that is programmed before shipment of software
- We supply a unique programming adapter for programming the unit
- Higher level language interfaces included
- Runs under DOS on PC/XT/AT and compatibles
- Parallel port version only

Software Sentinel-W.

- Designed for workstations, supermicros and minicomputers
- Serial port only (modem-type)
- Algorithm technique
- We provide detailed interface specifications: Developer creates a port driver
- Interface requirements: 25 pin DB25P or DB25S; RS232/RS422/RS423
- Only signals used: DTR & RTS from computer; signal ground; DSR or optional DCD from Software Sentinel-W or external device. TXD, RXD, CTS, RI passed through.

Call For Software Sentinel Evaluation Kit Pricing.

RAINBOW TECHNOLOGIES
18011-A MITCHELL SOUTH IRVINE, CA 92714 USA
(714) 261-0228 TELEX: 386078 FAX: (714) 261-0260

CIRCLE 95 ON READER SERVICE CARD.

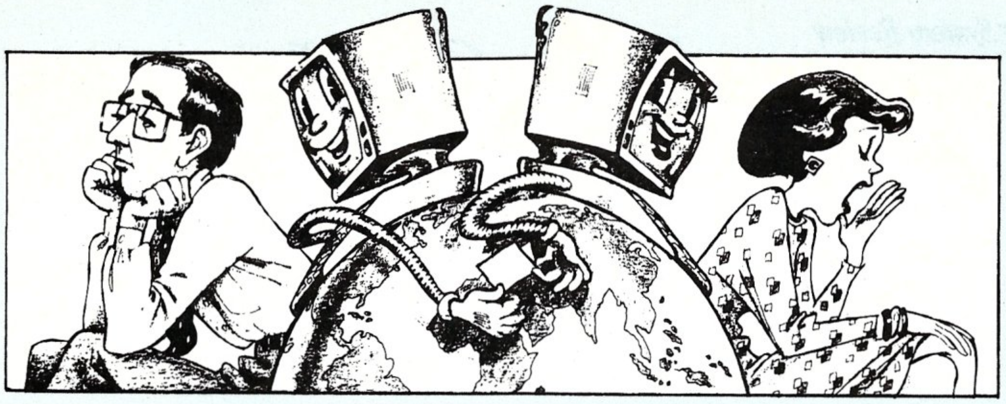
ADVERTISER INDEX

RSN	*Advertiser*	*Page*
62	Austin Code Works	49
70	Bourbaki, Inc.	11
53	Concurrent Controls, Inc.	5
	DBMS	41
41	Digiboard	28 – 29
99	Emerald Microware	65
122	Excelan Inc.	1
	Gimpel Software	53
109	Interfirm Graphics Systems	15
	Invisible Software	63
	M&T Books	67
	M&T Books	59
97	Max Software Consultants	69
55	MetaWare Incorporated	55
56	MetaWare Incorporated	36
113	Micronics Computers, Inc.	C-2
	Micro/Systems Reprints	69
	Nanosoft Associates	64
121	Network Products Corporation	7
65	Novell Development Division	2
103	Nu-Mega Technologies	23
46	PC Tech	72
43	Pure Data	69
72	Qualstar Corporation	36
86	Quarterdeck Office Systems	8 – 9
95	Rainbow Technologies, Inc.	56
94	Santa Cruz Operation	C-4
47	SemiDisk Systems, Inc.	45
123	Simple NET Systems, Inc.	47
112	Slicer Computers Inc.	33
81	Stargate Technologies, Inc.	13
51	Sunny Hill Software	30
80	Teletek Enterprises, Inc.	C-3
115	Tropic Software	37

ADVERTISING REPRESENTATIVES

National Account Manager	Dwight Schwab (415) 366-3600
National Account Manager	Tami Brenton (415) 866-1957
Advertising Coordinator	Shaun Hooper (415) 366-3600

Associate Publisher
Michael Wiener



Ten Reasons Not to Use PeaceNet

1. I don't like working with others
PeaceNet is a computer network and communication system for people who believe that global planning and cooperation are necessary to reverse a trillion-dollar-per-year arms race; it is linking users throughout the United States and in over 70 other countries.

2. I've got all the information I'll ever need
PeaceNet is for those who appreciate that information is always growing and changing; its bulletin boards, conferences, and databases provide information about everything from Central America to Star Wars.

3. I love playing phone tag
PeaceNet's electronic mail system renders those endless conversations with secretaries and answering machines obsolete.

4. I don't know how to use my computer
PeaceNet helps novices with simple, entertaining manuals and round-the-clock staff for answering their questions.

5. I enjoy copying, labeling, and stamping letters
PeaceNet enables you to send messages to hundreds of other users with one simple command.

6. I've got plenty of money to waste on postage and phone bills
PeaceNet is for people who want to save money; it lets you send documents across the world faster than Federal Express™ for pennies per page.

7. I don't mind getting action alerts a week late
PeaceNet does mind and can help your organization send out time-urgent alerts instantly.

8. I don't have the right kind of computer equipment
PeaceNet is available to anyone with a computer terminal and a modem.

9. An effective peace movement isn't worth 50 cents a day
PeaceNet users disagree.

10. It's all hopeless, anyway
Then why read this magazine when Modern Wrestling would suffice?

Use PeaceNet's electronic mail, conferences, bulletin boards, databases, and Telex services to gather, organize and share valuable up-to-date and comprehensive information about:

Peace and anti-nuclear efforts, Central American issues, environmental protection, anti-apartheid efforts, human rights, Native American issues, and much more.

PeaceNet is the largest and most rapidly growing on-line community of progressives anywhere in the world. And by using any home computer and a modem it is only a local phone call away. Call or write today for more information.

PeaceNet:

3228 Sacramento Street
San Francisco, CA 94115 (415) 923-0900

The QNX Multiuser/Multitasking Operating System

Looking for a UNIX-like operating system? Quantum Software's system may be just what the integrator ordered.

by Jon Johnston

QNX (*cue-nix*) from Quantum Software is a multiuser, multitasking operating system with excellent networking capabilities.

QNX was first introduced in 1982 and is based on research done at the University of Waterloo, Ontario, by Dan Dodge and Gordon Bell. Although cosmetically similar to UNIX, QNX's design is based on a message passing architecture that gives it speed and versatility. This structure allows both real-time executive speed and resource transparency in a network.

QNX is a modular operating system. It is structured as a group of cooperating tasks, with the kernel at the lowest level, "administrators" (explained below) at the next level, then user applications. This structure provides an incredibly flexible system that is easily customized. Customization can be accomplished simply by writing another administrator task that runs at the same level with other administrators, or by "mounting" device drivers. Tampering with the kernel may result in unforeseen complications and is unnecessary, as is the addition of layers between an application and the operating system.

The message passing architecture of QNX is also unique. Unlike other operating systems that provide a separate set of system calls to allow communications between tasks, all QNX system calls (including standard calls such as *fopen* and *fclose*) are inherently capable of message passing. This is possible beacuse of the ability of the kernel to provide task synchronization and because of the creation of virtual circuits.

Jon Johnston is a freelance writer and a service technician for an IBM dealership in Bloomington, Minnesota.

Two versions of the operating system are currently available: a real-mode version for use on PCs, and a protected-mode version which addresses up to 16 megabytes of memory for use on ATs and 80386-based micros. Both versions boot and operate in a network environment in only 256K. The real-mode version can handle up to 64 tasks, while the protected-mode supports up to 150 tasks. Two parallel ports are supported by both versions, with up to 32 serial ports supported in real mode and 58 in protected mode. A version specifically designed for the 80386 is currently under development, but no release date has been specified.

The Structure of QNX

The kernel occupies approximately 10K of memory and is responsible for the synchronization of task switching and message passing. It is a preemptive, prioritized task scheduler. A task switch can be accomplished in 0.31 milliseconds (ms) on an 8-MHz AT, and in 0.14 ms on a 16-MHz 80386.

Four main administrators reside in the level above the kernel: *Task*, *Fsys*, *Dev*, and *Idle*; along with three other "optional" administrators: *Net*, *Queue*, and *Timer*.

Task is responsible for the creation and termination of tasks, memory allocation, and task name registration. It is the highest priority task in the system.

Fsys is the file system manager. The QNX file system keeps files as contiguous as possible, with each file consisting of anywhere from 512 byte blocks to one terabyte. When *Fsys* is no longer able to keep the file contiguous, an "extent" is created that tells the file system the location of the next contiguous file segment. Extents symbolize a type of linked-list of the file structure. Therefore, unlike MS-DOS, in which a head seek will consist of moving back and forth between reading the .FAT table and reading the file, the QNX file system does a continuous read on the file, thus improving hard-disk performance. QNX features tree-structured directories, as well as file permissions similar to UNIX. File security is provided by assigning a number to each user, thus associating each user as a member within a group. Numbers for both groups and members range from 0 to 255, with member 255 of each group acting as group leader. All members of the group 255 are super users. Attributes associated with the user's number may be used to limit access to the following: the owner only, users within a group, users not in a certain group, or all users.

Dev is the device administrator and is concerned with devices responsible for character input/output, such as the console and parallel and serial ports.

Idle eats up idle system time by executing a null task. It is run at the lowest priority.

Net is available in the networked version of QNX. On boot up, it checks to see whether or not there is a network card in the computer, and if not, deallocates itself. *Net* occupies approximately 20K and performs message passing between nodes. The network cards are proprietary to Quantum, and each card is equipped with a boot PROM that allows machines to boot across the network. The cards are of the ARCNET type, requiring RG62U coaxial cable between nodes.

Queue is optional and is used to perform queued message passing. Since the primitives SEND and RECEIVE are unbuffered and normally block a sending task until a reply has been received from the destination task, *Queue* is provided to permit the sending task to resume operation. The queued messages may be buffered across the network, since *Queue* buffers the entire message.

Timer, also optionally loaded, adds timing capabilities to the system. The source code to *Timer* is available as an example for writing an administrator, and *Timer* itself may be extended.

Two other optional tasks are *Locker* and *SpoolDev*. *Locker* supports record locking and concurrent file sharing and is compatible with UNIX System V. *SpoolDev* creates a virtual device that spools information to a disk, then spools it to a physical device in a first-come, first-serve order.

Tasks

Tasks in QNX are run on a prioritized, time-sliced basis. Time-slicing can be set by the utility, TIME_SLICE. Priorities range from 0 to 15, with 0 reserved for the kernel and 1 – 4 reserved for the QNX and user-written administrators. By using the utility PRI, the user may set the priority of a normal application task between 4 and 15.

Three methods are provided for intertask communication: message passing, ports, and exceptions.

Messages consist of up to 65,535 bytes of data in a pre-defined structure that is agreed upon by both the sender and the recipient of the messages, usually an application communicating with its tasks. The kernel is not responsible for the content of the messages, but with message synchronization.

Message passing is accomplished using a pair of primitives, SEND and RECEIVE. Task A sends a message to Task B, then Task A blocks, waiting for a reply. Task B receives the message and replies to Task A, which then is unblocked and continues processing. Another situation might arise when Task A does a RECEIVE, then blocks, the difference being that Task A has requested a RECEIVE before a SEND is sent by another task.

For message passing between

nodes, virtual circuits are created. The use of virtual circuits enables applications to share resources, such as printers, modems, or files, and to execute tasks remotely, regardless of where those resources may be physically located. Nodes may be added or removed without changing the applications that may be running on the network.

Virtual circuits exist as virtual tasks on each communicating node. A virtual circuit provides a link between the two nodes, specifying the size of the largest message that the communicating tasks will send, and occupying only an entry in the task table entry of each node.

An example of resource sharing through the creation of virtual circuits is Qterm, a communications application. Using modem pool management, which involves telling Qterm where various modems of differing baud rates are located throughout the network, a user may access a remote modem. A file, */qterm/modems*, contains information that modems are located on nodes 2 and 3, with baud rates of 2400 baud. By invoking

```
qterm m=? baud=2400
```

from node 1, Qterm automatically searches the modem pool for a free 2400-baud modem and uses it as though it were located on the local node.

QNX uses ports to provide communications between interrupt handlers and tasks. It is also a very easy and speedy method to determine the task ID (TID) of another task, and therefore enable communications with that task. By "attaching" a task to one of the 40 ports provided (16 of which are reserved), other tasks may "detach" to a preset port (preset by an application) to obtain the TID of the task to which they must send communications. Since tasks aren't blocked by "attaching" and "detaching" as they are by message passing, ports are rapidly available for use when tasks are located on the same node.

"Exceptions" are abnormal events that are created when a task dies unexpectedly, similar to a "signal" in UNIX. There are 32 exceptions in QNX, with the first 16 reserved by the operating system for such occurrences as modem hangup, communication errors, task termination, memory violations, divide by zero, or a missing shared library.

Installation and Use

The standard QNX package is composed of three disks: Boot, Utilities, and the C compiler. The Boot disk contains both versions of the operating system along with the utilities necessary for installation and the Install program.

Install automates the installation process and explains each step to the user. It first mounts the hard drive, thereby making it accessible. For this process, the user must choose a disk driver from a menu selection. Drivers are available for PCs, ATs, PS/2s (ST506 and ESDI), and the Hewlett Packard Vectra. Next, other parameters are verified, including number of tracks, heads, and sectors per track. The second step is to create a QNX partition using *fdisk*. QNX's *fdisk* is compatible with DOS's *fdisk*. (I also had no problems with an SCO Xenix partition.)

The use and appearance of QNX are quite UNIX-like, as the shell and utilities resemble those found in UNIX systems.

The installation procedure continues by remounting the hard disk to use the QNX partition, formatting the QNX partition with a default interleave of 6:1, initializing the file system, and marking known bad tracks and blocks. The boot disk and all remaining disks are copied to the hard disk (directories are created as needed), and the operating systems image file is set to boot. The last step is to create a system initialization file, SYS.INIT. Caches are mounted for file system performance and the size may be set by the user. Up to eight virtual consoles may be created on the computer. Terminals may have only one.

Once Install is completed, QNX is bootable and usable, but several other steps must be completed before QNX is network-ready.

The first of these steps involves creating a user file, PASS. Pass contains the following user information: Userid (Username), Password, User Number (in form of Group.Member, e.g., 1.255), Login Directory (Home Directory), and Login Command.

The login command may be used to pass control to a batch file or to call a custom application, disallowing shell access when the application terminates. The PASS file must be created before additional nodes are added. After each user's information is entered in PASS, a directory is created for each user. In order for the user to be able to access this directory, the attributes must be changed; the user must be given ownership. This is done with the *Chattr* utility.

Once the network card is physically installed in the computer, the PROM must be initialized. As the computer boots, the network card is found by the BIOS and displays "Node nn" (nn being the default number, typically 1 before installation). The user then hits the Esc key twice and is presented with the PROM initialization menu. At this point, several choices must be made:

1. *Boot from the Network?* Nodes may be booted locally from their hard drive or across the network. Booting from across the network involves downloading the operating system image file.
2. *Local Node ID.* The Local Node ID must be unique. No switch settings are required.
3. *Boot Node ID.* The node from which the operating system image file is downloaded must be specified.
4. *Alternate Node ID.* An alternate boot node must be indicated.
5. *Boot Filename.* The version of the operating system that the booting node will be downloading must be specified. Depending on the type of machine, switching between running in real or protected mode may be changed here.
6. *Hardware Interrupt Level.* The interrupt at which the network card is set must agree with the hardware jumper setting.
7. *Exit the Menu and Boot.* Selecting this option will boot the computer, if it is the server.

Additionally, several steps specifying remote node access must be completed at the server before remote nodes can boot properly. These involve allowing network access and may be included in SYS.INIT, once again, for simplicity.

1. *search 3 +remote.* This sets the search order in order to allow remote access to the server's hard drive.
2. *nacc 3 +read +write.* This sets

network access for non-super users. Network access for each resource (serial ports, CPU, parallel ports) may be allowed on each node. CPU access may be granted in order to run remote tasks.

System initialization files must be created for each node. These are files like SYS.INIT, but in the form of SYS.INIT.NN (nn being the Node ID number). If the proper SYS.INIT.NN isn't found by the remote node, it will default to SYS.INIT.

3. *netboot &*. This activates the netboot task, allowing the server to accept boot requests from the remote nodes.

At this point, the QNX network installation is complete. Installation at the server and overall setup are rather simple, with no complications such as serialization, copy protection, or key cards. Addition or removal of nodes is also easy, involving only the creation of a system initialization file and the initialization of the network card in each node.

The use and appearance of QNX are quite UNIX-like, as the shell and utilities resemble those found in UNIX systems. Commands are typically entered from the command line, with parameters for different functions added as desired. One small time-saving detail of the system is its ability to display all possible command parameters by entering that command and following it with a question mark.

In comparison with SCO Xenix, QNX requires much less disk space, as well as shelf space. Instead of the eight manual set from SCO weighing about 40 pounds, the QNX manuals will fit into a three-manual set about the size of three DOS manuals. All of the QNX software installed on the hard disk requires less than three megabytes compared with approximately 20 megabytes or more for Xenix.

The QNX manuals are not for the unambitious. They are decently organized, and they contain a large amount of information. They start with an explanation of installation, then move on to an introduction to the operating system and the way that QNX handles devices, files, consoles, terminals, and the shell.

The utilities section offers the user a brief explanation of each utility at the beginning, along with an asterisk to mark which utilities will be important to a novice user. More than 100 utilities are included for configuration, disk, file, and directory manipulation, information handling, networking, and printing.

Support Software

Quantum offers a large number of products for use with QNX. The development system includes a C compiler, assembler, debugger, and utilities such as *Make*. Compilation is done in such a way that any source compiled under the protected mode will run under real mode without modification. The library is included with the compiler and has a multitude of functions for string manipulation, I/O, file access, terminal control, memory management, graphics, timing, math, interprocess communication, task control, record locking, and a host of others. Quantum offers a BASIC compiler, and a FORTRAN compiler is available from Southdale. Computer Innovations is porting its optimizing C compiler, with a debugger and support for the six memory models to QNX as well.

A BIOS emulator called QDOS is available, along with DFS, the DOS File System, which allows QNX to read and write DOS files. QDOS comes in two versions: a real-mode version and QDOSP for protected-mode operation which reserves the entire 640K for DOS use. QDOS and QDOSP allow DOS to run as a single task in QNX. QDOS is not compatible with certain manufacturers' versions of DOS since it must patch DOS in order to run properly. It does run many prominent DOS applications, such as Wordstar Professional, Lotus 1-2-3, and dBASE III PLUS. QDOS is in the process of being replaced by RunDOS, a full DOS emulator.

Other available software includes Qterm, an excellent communications and terminal emulation package complete with the QCL (Quantum Communications Language) interpreter; C-Tree, an ISAM file manager; Mail, for network mail; AP, an appointment scheduler; and Doc, a text processor. For mainframe communications, Corman Technologies provides X.25 software complete with X.25 card, and MicroNode, a 3274 Cluster Controller Emulator, is available from Xicom.

Software available from third-party vendors include Zim, a powerful 4GL database from Zanthe Information; WPro, a word processor from Klondike Software; and Profit Plan, a Lotus-like spreadsheet from Elsid Software Systems. On-Line Data Corporation offers OnCmd, a dBASE III PLUS clone that comes complete with language compatibility. A plethora of vertical applications for industry and office automation are also available, including inventory control, accounting, and point-of-sale systems for stores.

This year, Quantum sponsored QNX '88, a seminar for QNX developers. Over 200 people were in attendance, and Quantum announced the release of three new products and a new version of Ctree, the QNX file manager. The new version of Ctree will allow greater distributed server-type applications through the file manager. Also announced are: QRCS, a revision control system for source code control and maintenance; Ditto, a remote execution program that will allow an operator to display remote screens on the local display, or run a system remotely while re-routing output and input; and RunDOS, a DOS emulator different from QDOS, which will run a DOS application in real mode (with 635K available for the DOS application) and allow it to read and write QNX files and devices, thereby giving the DOS application many of the capabilities of QNX.

Conclusion

Because of its speed and flexibility, QNX can be used to replace more expensive minicomputer systems. When the QNX system becomes over burdened, another microcomputer may be added to boost performance and add resources without having to alter the existing system or applications.

Unlike other operating systems of its type, QNX lacks a hungry appetite for memory and hard-disk space while providing an environment inherently suited to networking. It is, therefore, a viable alternative to UNIX, Novell, and OS/2. □

Did you find this article particularly useful? Circle number 9 on the reader service card.

Product Information

QNX Operating System (1-user)	$450
including C compiler	650
QNX Operating System (4-user)	900
C compiler alone	300
BASIC Compiler	300
QDOS	125
Qterm	150
Ctree	295
Mail	150
AP	125
Doc	250

Quantum Software Systems Ltd.
Kanata South Business Park
175 Terrence Matthews Crescent
Kanatam Ontario K2M 1W8
(613) 591-0931
Circle reader service #265

by A. G. W. Cameron

Number Crunching in a 386/Weitek Environment

I spend a good deal of the time running a three-dimensional, smooth-particle hydrodynamics (SPH) code on a variety of personalized computers, both workstations and souped-up PCs. Generally speaking, there is little advantage in trying to do these calculations on a supercomputer because the central part of the calculations is not vectorized and is unlikely to be, so we would be using a supercomputer as a scalar machine. This is not cost-effective unless you can get the time free. (Actually we are using quite a bit of time on Crays at Los Alamos Observatory in collaboration with colleagues there.) The problem, then, is to deploy your resources in such a way that you get the largest number of floating point operations per dollar.

Cost-Effective Computation

My initial solution to this problem was to get a Sun 3/260 workstation with a Floating Point Accelerator (see "The Scientific Computer User," *Micro/Systems Journal*, May/June and September/October 1987). This ran roughly 10 percent as fast as the Los Alamos Cray X-MP (although a souped-up X-MP with hardware indirect addressing and hardware gather-scatter would gain a factor of two for each of these features). The factor is a rough one because parts of the calculation can be vectorized, and hence this ratio varies somewhat with the number of particles used in the calculation. Later, the Sun 3 was upgraded to the Sun 4/260 (see "The Scientific Computer User," *Micro/Systems*, July 1988), but we were very disappointed because our program only ran 20 percent faster. I believe the problem is that the Sun 4 uses a 128-kilobyte cache to achieve high speed, but this is relatively ineffective when many large arrays need to be manipulated.

When you are interested in number crunching, there is a further contradiction in the use of a workstation such as the 4/260. Sun expects that such a machine will generally be used as a server, running the disk accesses for a number of other machines on an Ethernet network. Such activity detracts from its use for number-crunching. I am currently using the 4/260 to support a Sun 3/110 and I soon expect to support two Sun 386i/250 workstations as well, although each of these will have a 327-MB disk of its own so that the additional load will not be too great.

Meanwhile, I have also pursued the use of PCs as dedicated number crunchers. The first experiment in that direction was the use of a Definicon 780+/4 board with 4 MB running in an XT clone (see "The Scientific Computer User," *Micro/Systems*, January 1988). This has been very successful, but the board is only one-third as fast as was our Sun 3/260, and about four-thirds as fast as a VAX 11/780. It typically takes about one month of continuous operation to calculate one of our hydrodynamics cases on the Definicon, and my feeling is that this is about the lower limit of acceptable speed. It is no longer the most cost-effective way to go in today's market.

There was a period in late 1987 when the Compaq 386/20 with the Weitek floating point option running at 20 MHz (instead of the 16 MHz used in both versions of my Sun workstation) appeared to be most cost effective. So it became logical to buy one.

The Compaq 386/20

I ordered the Compaq 386/20 through Businessland, which gave me a reasonably good discount. It took about three months to get delivery, and even then the delivery was not complete. The 135-MB tape backup unit came a few weeks later. But there I sat with a 386 machine and only 1 MB of RAM on the base memory board that sits in the special 32-bit Compaq proprietary memory expansion slot. I had ordered two 4-MB expansion memory modules. Then, for quite a long time, I would phone Businessland about once a week and ask, "Where is my memory?" Each time I would be told, "Compaq had promised there would be some memory in our last shipment, but there wasn't any. They say there will be some on the next delivery."

I have run into a surprising number of other people caught in the same bind. Some, but not all, of them have been attendees at the PC Technical Group meetings of the Boston Computer Society. I get the impression that Compaq has been using all of its available DRAM chips to push new machines out the door, ignoring the fact that a large number of these will be nearly total cripples; people can be strung along with unfulfilled promises for quite a long time.

Eventually it was clear that if I wanted to get any advantage out of the 386 protected mode, I would have to make other arrangements. I have now installed an Intel Above Board with 2 MB of RAM, and it is possible to make some progress. This memory has to be used over the standard AT 16-bit bus rather than the proprietary 32-bit bus, and therefore the performance I obtain is substandard compared to that promised by Compaq. This should be kept in mind in connection with performance figures that I quote in this article. But at least I can do some operations in 386 protected mode.

The Compaq 386/20 was one of the first machines to offer a Weitek socket. This socket is designed to accept a 387 chip, but it has extra rows of holes all the way around for the installation of the Weitek 1167 chip set. The Weitek 1167 chip set actually contains the Weitek 1164 and 1165 floating point chips that were used in the Floating Point Accelerator in my Sun 3/260 and on the CPU board of my Sun 4/260. There is also the 1163 chip that acts as an interface to the 386. Compaq offers these on an 1167 insert board that also takes a 387 chip for compatibility with the ordinary Intel floating point instruction sets. A cable connects to the Weitek socket. But I chose to go with the MicroWay 1167 daughterboard, which has all these same chips but inserts directly into the Weitek socket

A.G.W. Cameron is Professor of Astronomy at the Harvard-Smithsonian Center for Astrophysics.

on the motherboard. This socket lies underneath the space that may be covered by insert boards in several slots on the motherboard, and this may cause some problems because it prevents the use of any boards with low skirts. I managed to put most of the short boards in these slots, but I had to do some juggling of slots so that the Intel Above Board could fit between the chips on the MicroWay daughterboard.

386 Protected Mode

There is a loud and steady drum beat in the microcomputer community these days in favor of OS/2 and applications that run under it. This leaves me completely cold. I have not the slightest interest in running slowly in 286 protected mode. 386 protected mode allows you to run faster by a factor of two, and the preceding discussion will have indicated how I feel about factors of two in computation speed. Nor am I interested in just 386/387 computation in 386 protected mode. You gain another factor of two to three by using Weitek chips rather than the 387. If it were not for these factors, the Compaq would not have been interesting to me.

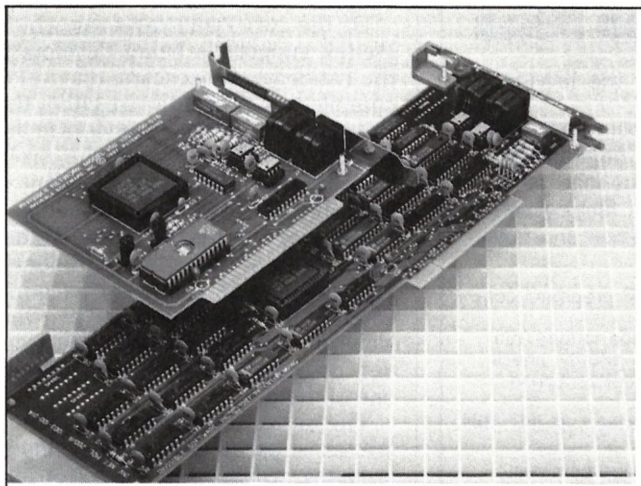
386 protected mode allows you to run faster by a factor of two.

In the inexcusable absence of a proper 386 operating system, it is necessary to use a DOS extender control program to execute programs that have been prepared to use the protected mode. There are several choices here, just as there are a variety of different FORTRAN compilers that use one or more of these DOS extenders. Four of these FORTRAN compilers were reviewed in these pages by Dan Feenberg (see "Protected Mode FORTRAN Compilers," *Micro/Systems*, June 1988). However, only two of these compilers support the Weitek chips at the time of this writing, although the other two are actively developing such support. The two compilers that do support the Weitek chips are the NDP FORTRAN-386 from MicroWay and the SVS FORTRAN-386 from Science Applications International Corporation (SAIC). Feenberg described the general features of these compilers, so I shall concentrate on the Weitek aspects.

NDP FORTRAN makes extensive use of the Phar Lap tools, including the assembler and linker as well as the DOS extender. SVS FORTRAN uses the X-AM DOS extender from IGC (Intelligent Graphics Corporation), but requires the Phar Lap linker. In both cases, the vendors supply batch files that render the DOS extenders all but invisible to the user. Thus I do not have much to say about them except that they work very well.

X-AM from IGC is really invisible unless you look very hard. The batch file imports the X-AM control program into the current directory and renames it FILENAME.COM, where FILENAME is the name of the file that you have compiled. Your compiled and linked file is called FILENAME.REX. Hence, FILENAME.COM loads into memory and takes over control, loading FILENAME.REX in its turn and then sitting and waiting to handle calls to DOS services, which run in 386 real mode. If you wish, you can combine these files into a standalone .EXE program, and anyone wishing to distribute a program would choose to do that. But for the ordinary working scientist or engineer, that final step is unnecessary and usually a waste of time.

NDP FORTRAN does not use a slick trick like that. Instead, after compiling and linking your program, you



"Good performance under load, elegant design, low price, simple cabling requirements, and broad compatibility."
—PC Week

"The Invisible Network is a darn good deal."
—PC Magazine

"A joy to use. I highly recommend it."
—Data Based Advisor

Best value in a complete LAN

New! Invisible Network.

Best value. New Invisible Network™ from Invisible Software is a *real* LAN. It transfers data at 1.8Mbps. In one high performance package, it combines power, reliability, ease of installation, and low cost. We have included all the hardware and software. And backed it with a lifetime limited warranty plus a 30-day money-back guarantee.
Broad compatibility. Invisible Network includes the NET/30™ Network Operating System at no additional expense. Invisible Network is also fully compatible with the IBM NETBIOS standard, the IBM-PC LAN Program™ and Novell Advanced Netware™ So you can run all the most popular network applications.
Installs simply. You easily install the LAN yourself by inserting an interface card (with built-in modular jack) into each network computer. It's that simple.

	Model 100	**Model 200**	**Model 200/A**
Compatibility	IBM PC/XT/AT	IBM PC/XT/AT	IBM PS/2 Micro Channel
Speed	0.7 Mbps	1.8 Mbps	1.8 Mbps
Price	$199.00	$299.00	$399.00

For more information call **(415) 221-0916** or write.

Invisible Software, Inc.
481 47th Ave.
San Francisco, CA 94121
(415) 221-0916

CALL ADVERTISER DIRECTLY

Introducing

NANODISK

"Disk Cache for the IBM PC"

Make your floppy drive and hard disk run close to RAM disk speeds. Dramatic speed improvement for most programs. Supports cache of any size in main or expanded memory.

Requires IBM PC/XT/AT or true clone.

only **$29.95**

MultiDos Plus

"multitasking for the IBM-PC."

Ideal for developing applications in process control, data acquisition, communications, and other areas. Check these features which make **MultiDos Plus** an unbeatable value.

- Run up to 32 DOS programs concurrently.
- Operator commands to load/run programs, change priority, check program status, abort/suspend/resume programs.
- Programmatic interface via INT 15H for the following.
  - * Intertask message communication.
  - * Task control by means of semaphores.
  - * 256 priority levels.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.
  - * and more!

Requires IBM PC/XT/AT or true clone, and enough memory to hold **MultiDos Plus** (48 KB) and all your application programs.

$24.95 *or* **$99.95**
With source code
(Written in Lattice C and Microsoft Assembler.)

Outside USA add $5.00 shipping and handling.
Visa and Mastercard orders only call toll-free: 1-800-872-4566, ext. 350., or send check or money order (Drawn on U.S. Bank Only) to:

NANOSOFT

13 Westfield Rd, Natick, MA 01760
MA orders add 5% sales tax.

CALL ADVERTISER DIRECTLY

run it with a statement like *RUN386 FILENAME* (the compiled and linked program is named FILENAME.EXP). The Phar Lap RUN386 control program operates similarly to the X-AM control program.

I do have a gripe about the X-AM procedure. The X-AM DOS extender requires that the size of the runtime stack be set with a program called XAMPARM. If you guess too small a size, the program quits with a generally meaningless dump of register contents and a meaningful error "0E." You have no choice except to run XAMPARM over and over again, increasing the stack size each time until the program is satisfied (or until you run out of memory). There should be a much more graceful way to handle this.

Of the two FORTRANs for which Weitek support is being developed, the Language Processors, Inc., (LPI) compiler can presently use either the Phar Lap or IGC DOS extenders in the 386/387 version. One of the problems that is holding up the Weitek version is that LPI cannot produce a single program that supports Weitek under both of these DOS extenders. LPI will have to make up its mind whether it wants to produce two versions, one to support each DOS extender, or whether it will support only one of them.

The other FORTRAN for which Weitek support is being developed is Lahey, which uses the A.I. Architects 386 DOS extender. I reported on an early version of this DOS extender in *Micro/Systems*, February 1988. Lahey promises to have its version of Weitek support available by the time this appears in print.

The FORTRAN Compilers

Table 1 shows some benchmarks for both the NDP and SVS FORTRANs, each compiling either for the 387 or for the Weitek 1167. These five floating-point programs were employed by Avram Tetewsky and Dan Feenberg ("FORTRAN Compilers for the PC," *Micro/Systems Journal*, September/October 1987) to compare the performances of a wide variety of computers. Table 1 includes various measurements that I have made, together with a few of their results for comparison. An appended /16 indicates use with the Above Board Extended memory in the Compaq.

S-whet and D-whet are the well-known whetstone benchmarks, in single and double precision, respectively. Here high numbers (in kilowhetstones per second) are better. REMEZ is an IEEE digital signal-processing program that is floating point intensive. SVD is a single value decomposition program that operates on 50 × 50 matrices. Roots finds complex roots of real polynomials. In these three cases, the numbers are the execution times in seconds, and small is beautiful.

In these benchmarks, the NDP FORTRAN generally did significantly better than the SVS FORTRAN. On Roots, the discrepancy was quite large because the SVS (Silicon Valley Systems) compiler is notoriously slow on complex arithmetic. The NDP manual contains a long section on benchmarking that has obviously been prepared by Steve Fried at MicroWay, who has published a number of similarly written articles on floating-point processors and computational efficiency. (I feel I know his writing style.) It is interesting to discover that current efforts to improve NDP FORTRAN have centered on speeding up the whetstone benchmark. This is a legitimate way to improve performance, but the resulting benchmark speeds may not carry over to superior performance on real-world problems.

In any case, even though it is handi-

Table 1. Some comparison floating point benchmarks

Computer	S-kwhet kwhet/s	D-kwhet kwhet/s	REMEZ sec	SVD sec	Roots sec
VAX 11/780 FPA	1191	734	3.00	n/a	n/a
Definicon 780+/4	1157	1051	2.41	15.22	4.45
VAX 11/785 FPA	1800	1157	1.30	7.90	0.399
Compaq 386/20/387/SVS/16	1889	1627	2.19	9.20	1.32
Compaq 386/20/387/SVS/32	2014	1750	2.09	9.20	1.32
Compaq 386/20/387/NDP/16	1830	1649	1.82	10.80	1.21
Compaq 386/20/387/NDP/32	1893	1715	1.76	10.50	1.21
Sun 3/260/FPA	3561	2373	0.65	2.21	0.68
Compaq 386/20/1167/SVS/16	4115	2546	1.27	4.72	0.72
Compaq 386/20/1167/SVS/32	4109	2543	1.15	4.66	0.77
Compaq 386/20/1167/NDP/16	4305	2459	1.04	4.29	0.35
Compaq 386/20/1167/NDP/32	4330	2662	0.94	4.23	0.33
Sun 4/260	5219	3281	0.48	1.36	0.45
VAX 8650 FPA	6100	3895	0.53	2.20	0.130
IBM 3081D	5850	5680	0.30	n/a	n/a

capped by the Intel Above Board, it is clear that the Compaq 386/20 delivers a performance intermediate between the Sun 3/260 and the Sun 4/260. This fully justifies its use for intense number-crunching activities. Once I had the Above Board installed, I first brought up the SVS FORTRAN, since I was familiar with it from using it on my Definicon board. I trimmed the size of a very large array in the SPH program, modified the system-specific calls to timing routines, and had SPH up and running in half an hour. The SVS compiling and linking

I am looking forward to when I can use the DOS extender in a multitasking environment.

is extremely fast, and the compiler produces nicely compact code. The execution speed is about 75 percent of that for the same SPH case running on the Sun 4/260. Since then the Compaq has been running SPH around the clock except for interruptions for needed chores. I am looking forward to the day when additional memory (when or if ever delivered by Compaq), together with a promised revision of DESQview, will allow me to use the DOS extender in a multitasking environment, so that these interruptions will be less disruptive.

Things were not so easy with the NDP FORTRAN. Dan Feenberg noted that the NDP FORTRAN (a version of the well-known Green Hills FORTRAN) has much longer compilation times than SVS, and the code is not as compact. This caused problems in getting SPH to run in my available 3 MB. It took me a good part of three days to port the program. I had to cut down array sizes even further. NDP FORTRAN allows you to choose one of four buffer sizes, ranging from 8096 bytes to over a MB, to handle I/O. I was unhappy about this piece of clumsiness, but I had to work to get everything to run with the 8096-byte buffer in the interest of keeping the runtime code size small. That meant that I had to arrange all input tables to be in ASCII and I had to take all output in ASCII. This lengthened the input time substantially but affected only the first time step in SPH. Eventually I succeeded in getting SPH to run. The results of the benchmarks shown in Table 1 led me to hope that I would get improved performance from SPH using NDP FORTRAN.

The results were a great surprise. After the first time step, SVS FORTRAN ran SPH with further time steps of 3.2 minutes. NDP FORTRAN ran SPH with time steps of 5.5 minutes! This comparison involved the same case. I am at a great loss to understand the difference in these two performances. Clearly MicroWay has a lot more development to do to increase the efficiency of its NDP FORTRAN. Meanwhile, the Sun 4/260 ran the same benchmark in 2.4 minutes. I would expect the SVS FORTRAN's performance to be more comparable to the Sun 4/260 if given decent memory expansion modules in the Compaq.

Addendum:
Finally, at the end of August, I have received a 4 MB (third-party) memory module for the Compaq, which operates on the 32-bit proprietary bus. The benchmark results obtained using this memory have been added to Table 1 with /32 appended to the appropriate rows of the table. The improvements in performance range from none at all (where the routines were presumably already executing within the lowest 1 MB) to about 10 percent. One iteration of the SPH program became 2.95 minutes with SVS and 4.84 minutes with NVP, improvements of or 8 and 13 percent, respectively. Thus, with SVS, the Compaq 386/20 runs SPH at 81 percent of the speed of the Sun 4/260. □

Did you find this article particularly useful? Circle number 11 on the reader service card.

Product Information

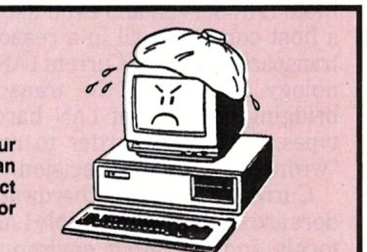
NDP FORTRAN 386	$595
for *Phar Lap tools* add	495

MicroWay
Cordage Park
Plymouth, MA 02360
(617) 746-7341
Circle reader service #266

SVS FORTRAN 386	$695
for *Phar Lap linker* add	495

Science Applications Int'l Corp.
5150 El Camino Real
Los Altos, CA 94022
(415) 960-5931
Circle reader service #267

Disk formats can be a REAL headache!



How many times have you needed to use a file, but your disk was for the wrong computer? Micro Solutions can take the pain out of data transfers by giving you direct access to your Macintosh, IBM PS/2, AT, Apple II, or CP/M diskettes on your PC or AT.

UniForm-PC .... **$ 64.95**
Read, write, copy, and format diskettes for over 275 CP/M and MS-DOS computers on your PC or AT, including 3½", 96 TPI, high density, and 8" formats. Use your DOS programs and commands on your CP/M diskettes.

CompatiCard I .... **$ 169.95**
THE universal four drive floppy controller board for the PC or AT. Runs standard 48 TPI, 96 TPI, & high density 5¼" drives, 720K & 1.44 MB 3½" drives, and 8" floppy drives. Use up to four floppy controllers on the same system. Comes with utility disk.
CompatiCard I w/ UniForm-PC program .... $ 225.00

MatchPoint-PC .... **$ 169.95**
Use this board and software package with your existing floppy controller to handle NorthStar hard sector and Apple II format diskettes. **Free UniForm-PC included.**

MatchMaker .... **$ 139.95**
Plug in your Macintosh drive to transfer files between Mac and MS-DOS disks.
MatchMaker w/ external Mac drive .... $ 325.00

Wouldn't it be nice if your PC or AT could speak CP/M?

UniDOS Z80 Coprocessor Board .... **$ 169.95**
Run CP/M programs in your PC or AT with this 8 Mhz. Z80H half-card. UniDOS stays memory resident until a CP/M program is encountered, then transfers execution to the Z80. Terminal emulation built in for Kaypro, Xerox, VT100, and ten others. All standard CP/M system calls are supported. **Bonus: UniDOS and UniForm-PC included.**

30 day money back guarantee on all products. Please include $5.00 for shipping, $7.50 for COD. Call or write for full catalog.

EMERALD MICROWARE

Emerald Microware
P.O. Box 1726
Beaverton, OR 97075
(503) 641-0347

CIRCLE 99 ON READER SERVICE CARD.

by Patrick H. Corrigan

Selecting LAN Hardware — A Primer

LAN technology is rapidly changing, both in terms of hardware and software. We are seeing the emergence of faster, more capable, more "intelligent" LAN hardware, as well as increasingly sophisticated LAN operating systems. In addition, interconnectivity of dissimilar systems is becoming more practical, as well as interoperability of those systems. With today's technology, a PC user can access local files, files from a central LAN server, files from a remote LAN server, and even files from a host computer, all in a reasonably transparent fashion. Current LAN technology also allows the transparent bridging of different LAN hardware types, making it harder to make a "wrong" LAN buying decision.

Currently, many LAN hardware vendors are supporting multiple LAN protocols and operating environments. Many vendors also are providing interfaces, bridges, and gateways that allow multiple equipment types to be interconnected. Similarly, LAN operating system vendors are porting their software to multiple hardware types. For example, many LAN hardware vendors are currently supporting Novell's NetWare, the IBM PC Network Program and NetBIOS, and TCP/IP (Transmission Control Protocol/ Internet Protocol, which is the Department of Defense standard used widely in defense and aerospace, engineering, and educational environments). On the software side, Novell's NetWare, Banyan's VINES, and CBIS's Network OS currently support internetworking of multiple LAN hardware types, and provide for gateways to mainframe and minicomputers, remote computers and LANs, and wide area networks (WANs). Even cable type and cable system topology are less of an issue. Many of the leading LANs can now be cabled with different cable types and with different topologies.

For integrators of PC LANs, all of this means that the issues concerning selection of LAN hardware are not as clear cut as they once were. The issues that we must address in our equipment selection process include: LAN operating systems supported, CPU interfaces supported, performance, transmission method, cable access scheme, layout or topology, and supported media.

Patrick Corrigan is a partner in The Corrigan Group — Information Services, an independent consulting firm based in Corte Madera, California, that specializes in local area networks and office automation.

Operating Systems Supported

For most PC LAN systems, the most important component will be the network operating system. Therefore, the selected hardware must support the chosen OS. Many LAN hardware vendors now support multiple network operating systems for PCs, including Novell NetWare, Banyan VINES, and the IBM PC LAN Program, giving the LAN integrator the flexibility to change operating systems later. Some vendors also support software that allows interconnection of non-PC type equipment, such as DEC VAXes, Apple Macintoshes, Sun workstations, and others, into a PC network.

Supported CPUs

If you are planning to connect to non-PC type equipment, it is important that LAN interfaces (and operating software) are available. For example, Ethernet interfaces are available for almost every major micro- and minicomputer system, and for many mainframes.

Proteon provides a wide variety of interfaces for its Pronet 10 Token Ring network. TCP/IP software is available for many Ethernet interfaces and for Pronet 10. TCP/IP allows dissimilar systems to transfer files and emulate each other's terminals, among other things.

Transmission Method

Choosing between baseband or broadband transmission methods is a major consideration. Baseband delivers a single signal at a time on a cable system, while broadband delivers multiple signals on the same cable system at different frequencies, as with cable television. For most applications, baseband systems provide a cost-effective data dissemination solution. In certain environments, however, the multiple signal carrying capability of broadband LANs (providing, in effect, multiple LANs on the same cable set) justifies the greater complexity and higher costs incurred for a broadband system.

Cable Access Schemes

In the average office environment, the method used for controlling access to the LAN cable is not usually a major consideration. In specific environments, however, cable access could be critical. In manufacturing control systems, for example, the "deterministic" access scheme provided by token-passing networks (802.5 Token Rings, Pronet, ARCNET, and others) is usually preferable to the "probablistic" access scheme of contention networks (Ethernet, G-Net, and others). Token-passing access guarantees delivery of data packets, while contention schemes only offer a high probability of delivery.

Network Throughput and Performance

LAN performance is a combination of many factors. The factor most often quoted is the raw data transfer rate. This is the speed at which data can travel across the LAN. However, factors such as cable access method and network interface efficiency greatly affect actual throughput. Remember that no one factor determines overall performance. The following things all contribute to LAN hardware performance (or lack of it):

- *Cable access scheme:* Generally speaking, contention or CSMA (Carrier Sense Multiple Access) is more efficient on lightly loaded networks. Token passing, on the other hand, is usually more efficient on busy

LANs.

- *Raw data transfer rate:* This is the most quoted figure related to LAN performance. Other factors are often equally important.
- *Network interface efficiency:* Various factors affect the efficiency of the network interface. A board with an on-board processor and on-board RAM can off-load communications tasks from the computer's CPU. The methods used for interface-to-host and interface-to-LAN data transfer can affect performance.

Cabling decisions should entail a great amount of thought and preplanning. The actual design of a building may make one topology easier to install . . .

Topology or Cable Layout

Cabling is often a key issue in LAN selection. Although cable itself is relatively inexpensive, installing cable in a building can be very expensive. For this reason, cabling decisions should entail a great amount of thought and preplanning. The actual design of a building may make one cable topology easier to install than another.

Supported Media

The media or types of cable supported are often more important than the cable layout itself. We have recently seen movement toward using unshielded, twisted-pair phone wire for cabling short distances and fiber optics for cabling longer distances. In addition, some LANs support the use of infrared light beams, microwaves, radio waves, and even AC power lines. The ability of a LAN to support multiple types of cable provides for flexibility when installing and expanding.

These are some, but not all, of the factors to consider when selecting LAN hardware. In future columns we will explore specific aspects of these options in greater detail. □

Did you find this article particularly useful? Circle number 10 on the reader service card.

There Is Mail . . .

(continued from page 10)

sized moves are more efficient than byte-sized moves (Example 4). I would like to suggest that the code fragment in Example 4:

```
        SHR CX,1
        REP MOVSW
        JNC NO_ADD
        MOVSB
    NO_ADD:
```

be changed to:

```
        SHR CX,1
        REP MOVSW
        RCL CX,1
        REP MOVSB
    NO_ADD:
```

If this portion of code is in a loop, considerable time can be saved because JNC NO_ADD results in 24 clocks if the jump is taken, and the second method always takes 14 clocks. Also, the second method doesn't suffer the penalty of dumping the prefetch queue. The second method is NOT an advantage if the variable BYTE_COUNT is even most of the time.

This brings up a point that should have been mentioned in the article: Example 4 should only be used if BYTE_COUNT is a variable. If BYTE_COUNT is a constant, (defined at assembly time) then let the assembler do the work. For example:

```
        BYTE_COUNT EQU 13
        ...
        MOV CX,BYTE_COUNT/2
        MOV DI,OFFSET DESTINATION
        MOV SI,OFFSET SOURCE
        REP MOVSW
        IF BYTE_COUNT MOD 2
            MOVSB
        ENDIF
NO_ADD:
```

Another important point that was missed concerns word alignment. Mr. Parker mentioned the importance of having memory operands on even boundaries. Of equal importance is having loop targets on even boundaries. The same penalty incurred by fetching a non-aligned piece of data is incurred by jumping to a non-aligned address. A good place to demonstrate this is in Example 12:

```
        MOV SI,OFFSET TBL_ST
        MOV DX,1000/4
        EVEN
NORM:
        SHL WORD PTR [SI],CL
        etc.
```

If an 8086 is executing this code, 1000 clock cycles could be saved by insuring that the target address is on an even boundary.

By the way, the instruction MUL AX,DX is not valid (Example 6). It should read MUL DX;. AX is always the destination in MUL.

The leading zero counter problem is something that I haven't thought of. After examining Mr. Parker's solutions, I decided that, if I needed to tackle a similar problem, Example 16 is too complicated and Example 17 is too absurd. Therefore, I came up with a simpler version of Example 16. The result is in EXAM16.ASM. This routine will execute consistently at 46 Parker clocks for results >= 8, and 34 Parker clocks for results < 8. I will let someone else calculate the real clock cycles.

(By the way, the instruction ADD SI,AX in Example 17 should be MOV SI,AX.)

As I mentioned earlier, I found this article interesting; it provoked a lot of thought. Please continue to print assembly language articles.

Adrian B. Crum
La Habra, Calif.

From the author: Evidently you have misunderstood the section that discussed pre-fetch queue operation. As explained in the article, under "ideal" conditions, the Bus Interface Unit (BIU) fetches instructions as fast as the Execution Unit (EU) can execute them. Thus, instruction fetches occur in parallel with instruction execution, so the time required to fetch instructions (4 clock cycles-per-byte for the 8088, 4 clock cycles-per-word for the 8086) can be ignored.

Is the "ideal execution time" assumption of the examples all that bad? As the 8086 User's Manual points out, actual instruction execution time is normally within 5 – 10 percent of the calculated execution time (Table 1 of the article).

However, adding 4 clock cycles per intruction byte (or word for the 8086) to account for instruction fetches is clearly wrong. After all, what would be the point of having a pre-fetch queue if you can't remove instruction fetch time from the performance equation?! I calculated and then measured the execution time of your test code for Examples 1 and 2. Calculations were made for execution time only (as in the article) and for execution time plus 4 clocks per instruction-word (your method for 8086). The results are summarized in Table A. From Table A, we see that the 8086 measured execution time is within 4 percent of the calculated ideal time for Example 1 and agrees exactly for Example 2. Your method is off by 30 – 45 percent.

Why are my 8086 results so far off from your measurements? Looking at the hardware, your XT clone probably uses an 8088 CPU with dynamic RAM that may or may not require wait states. The different CPU (mine is an 8086) alone would account for about 20 seconds of the difference because of the 16-bit memory-operand instructions. (An additional 4 clocks is required to execute each of the PUSH and POP instructions in your test code on an 8088.) The refresh cycles required by dynamic RAM can reduce performance by 5 – 10 percent and wait states required by slow memory (dynamic or static) will reduce performance even further. However, the impact of these factors doesn't seem to explain the large differences between our results.

The difference could be explained, however, if you ran your test code from inside a debug program such as DEBUG, SYMDEB, or CODEVIEW. In this situation, timing measurements may yield erroneous results because the debug software may execute several instructions for each user-program instruction (to provide the instruction trace capability). Another source of error can be the presence of interrupts.

Table A. Measured versus calculated performance of test code for article Examples 1 & 2

8086 RESULTS	Example 1	Example 2
Ideal execution time (calc)	2 min 54 sec	3 min 16 sec
Execution time + 4 clks/inst. word (calc)	3 min 13 sec	4 min 15 sec
Measured execution time	2 min 59 sec	3 min 16 sec
8088 RESULTS	Example 1	Example 2
Ideal execution time (calc)	3 min 16 sec	3 min 36 sec
Execution time + 4 clks/inst. byte (calc)	5 min 32 sec	5 min 34 sec
Measured execution time (Mr. Crum's result)	4 min 21 sec	4 min 6 sec

* Measurements done on an 8-MHz 8086 with 16-bit wide static RAM

The real-time clock and the printer handler are two that exist in most PC clones. The number of interrupts, their periodicity, and the length of their service routines affect the "apparent" execution time of the code under test. The only way to avoid these measurement errors is to disable all interrupts for the period of the test.

You conclude from your results that Example 1 is actually slower than Example 2. The results in Table A show that this assertion is wrong. But, it is also wrong to calculate the "relative" performance between Examples 1 and 2 using only the results in Table A because about half of the measured time is spent executing "overhead" code (LOOP, PUSH, and POP instructions). To compare the performance of the two examples, you should measure the execution time of the "overhead" code (test code without macros EXAM1 and EXAM2) and then subtract that result from the test code measurements. After performing this test, performing the calculation, and dividing by the number of iterations, I found that Example 1 executes in 37 clock cycles and Example 2 in 44 clocks. These results agree closely with the numbers shown in the article; the ideal time calculation estimates that Example 1 is 20 percent faster than Example 2 while the measurement shows a 16 percent speed increase.

Your letter does bring up some good points concerning Example 4. The leading zero counter is another good example of trading off code speed for size; it fits nicely between the 2nd and 3rd entries of Table 3 in the article.

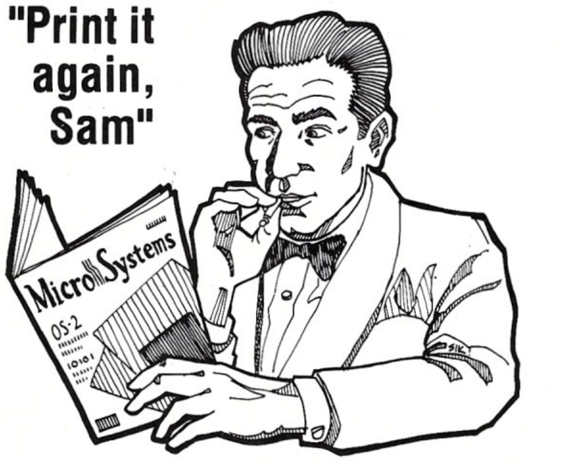
The point you make about Examples 5 and 6, however, is not really correct. The discussion in the article referred to the 8086, so Examples 5 and 6 show the results for MOVSW instructions executing on an 8086, not an 8088. For an 8088, 8 more clocks per MOVSW are required, which is the same performance penalty that is paid for unaligned word moves on the 8086. Sorry for the confusion this may have caused.

Your point about aligning jump targets for 8086 machines can improve execution time in some instances, but your time estimate is a bit optimistic. My test measured a time difference of 6.25 usec between code fragment (CF) 5 (aligned jump target) and CF 6 (un-aligned jump target). This represents a savings of 1 clock cycle per loop iteration. After trying a few different combinations of instructions that execute with various degrees of efficiency (execution time/# of instruction bytes), I have concluded that the amount of improvement provided by instruction alignment is no more than 2 clocks per loop iteration. So, for 8086 machines, 16-bit memory operand alignment will "always" affect performance by 4 clock cycles per transfer while jump target alignment will improve performance by 1 – 2 clocks per iteration, depending on the nature of the code at the beginning of the loop. Don't forget, however, that the big performance improvement in program loops is to reduce the number of iterations by duplicating the code inside the loop.

In summary, I have found that the standard execution timetable offered in Table 1 of the article provides reasonable accurate speed estimates, at least accurate enough to select between two pieces of code or algorithms. Under normal circumstances (not an excessive number of program jumps), the tables support speed estimates that are usually within 5 percent of measured execution time. As pointed out in the article, you should "measure" the execution time if more accuracy is needed. And when measuring execution time, minimize the amount and account for the execution time of the testing overhead code; use an oscilloscope or logic analyzer if possible to provide the best precision.

Kevin Parker

"Print it again, Sam"



Spread the good news about your product with a Micro/Systems Reprint

You can have your article or review professionally reprinted on 60 lb. paper stock, in 4-color, 2-color or black & white. Call today for more information.

Shaun Hooper
(415)366-3600

* Minimum quantity 100 reprints

dp_MAX

dBase III Tools in Turbo Pascal 4.0

Complete Support for dBase III files.
DBF, NDX, DBT file & record Access.
Fully Compatible dBase III B+Tree ISAM.
Library of 100+ functions in TP4 Unit.
Allows 250+ files & indexes open at once.
LRU file caching, round robin file manager.

$$\text{dp\_MAX} = \int_{-\infty}^{+\infty} \frac{\text{dBASE III +}}{\text{Turbo Pascal 4.0}}$$

Max Software Consultants, Inc.
4101 Greenmount Avenue
Baltimore, MD 21218
(301)-323-5996

$149

CIRCLE 97 ON READER SERVICE CARD.

New Products

New Software Products

VenturCom Adds Two Products for PC-Based UNIX

VenturCom, Inc., has released VENIX/286 System V Release 2.4, a standard UNIX System V Release 2 that runs on the IBM PC/AT, PS/2, and compatibles. This real-time implementation of UNIX for the 80286 and 80386 complies with the AT&T System V Interface Definition (SVID). Real-time extensions guarantee immediate system response, including asynchronous I/O and direct I/O which can increase I/O throughput by a factor of 10. It also has real-time capabilities for preemptive priority scheduling, direct hardware access, and high-resolution alarms. VENIX 2.4 is available as a Runtime System or a Development System. The Runtime System provides basic operating system services with real-time capabilities and supports standard AT peripheral boards: display adapters (MDA, CGA, EGA, VGA), smart RS-232 boards, $3^1/_2$-inch floppies, Wangtek streaming tape, and more. The Development System offers a software development environment with two C compilers, the AT&T Intel 80286 Software Generation System and the VENIX proprietary Intel 8086 Software Generation System. It also includes a FORTRAN compiler and many familiar Berkeley and AT&T utilities. The VENIX 2.4 Runtime System is priced at $400 for single quantities. The complete Development System, including a Developer's Programming Guide and Reference Manual, is available for $800 for two users, $1,000 for multiple users. End-user and administrator documentation are available separately.

Also new from VenturCom is RTX/386, a "turbocharger" for 386 workstations running UNIX or 386/ix. VenturCom sources claim that RTX/386 can improve disk access speed by 2 to 10 times for online transaction processing, as well as group applications such as database management systems, calendars, and mail. It is said to be useful to improve response times for X-Windows, interactive graphics, and communications servers. RTX/386 extends real-time functionality, including asynchronous I/O, preemptive priority scheduling, direct hardware access, and physical memory access.

For more information, contact **VenturCom**, Inc., 215 First Street, Cambridge, MA 02142; (617) 661-1230.

Circle reader service #15

New Office Package Supports a Host of Users

BOS National (Business Operating Software) has released BOS 2000, a fully integrated, multiuser office software package that runs on a wide variety of hardware from the PC to the VAX. Designed to meet the needs of small to medium-sized businesses, the package includes BOS Writer (word processor), BOS Speller (spell checker), BOS Finder (database), and BOS Planner (financial planner/spreadsheet). All BOS products operate under the BOS operating system and are available in single-user, multiuser, and LAN versions. BOS software can also be run under DOS, Microsoft Windows, and DEC VMS.

The BOS 2000 Office Software Package is priced at $1,350 for the single-user version, $2,700 for up to 12 users, and $5,400 for up to 40 users. Larger systems are quoted individually. For more information, contact **BOS National, Inc.**, 2607 Walnut Hill Lane, Dallas, TX 75229; (214) 956-7722.

Circle reader service #16

A Network Calendar Resolves Schedule Conflicts

Network Scheduler is the latest group connectivity software to be released by Sumware Inc. This time-management software provides a shared calendar for all network appointments and provides a calendar for individual network users. Up to 16 network users can be scheduled for a meeting through Network Scheduler, including meeting confirmation via electronic mail. An appointment book is part of the system that shows scheduling by the date or month. This software program has 27 levels of security, and provides a database for names, addresses, phone numbers, and notes, and a word-processing function for memos and letters. It is compatible with IBM PC hardware and will operate on most Novell, IBM, 3Com, StarLAN, and Ungermann Bass networks.

Network Scheduler sells for $695 in the LAN configuration ($120 for the single-user version). For more information, contact **Sumware Inc.**, 23121 Verdugo Drive, Suite 101, Laguna Hills, CA 92653; (714) 855-3062.

Circle reader service #18

NetWare Now Supports DOS 4.0

Novell has announced that NetWare will support IBM's DOS 4.0. The NetWare Workstation Shell Kit for DOS 4.0 is now available and includes an unlimited user license. The new shell can be purchased for $50, or downloaded from Novell's NetWire on-line support network.

For more information, contact **Novell, Inc.**, 122 East 1700 North, Provo, UT 84601; (801) 379-5900.

Circle reader service #19

New Hardware Products

Micom Adds Multiplexer for T1 Feeder Networks

Micom Systems, Inc., has introduced the Micom Box Type 6 statistical multiplexer for use in point-to-point or T1/2.048 Mbps networks. The Micom Box Type 6 features dual microprocessors and direct memory access (DMA) and multiplexes asynchronous and/or synchronous data simultaneously over a single leased line operating at up to 19.2 kbps or a digital wideband link operating at up to 72 kbps. The unit has flexible clock speeds from 1200 bps to 72 kbps with up to 32 async/sync user channels, and requires only a single RS-232 or V.35 port. Each channel may be individually configured for data rate, flow control, and protocol, among other features. The MB6 can be expanded with slide-in, eight-channel expansion cards.

Prices for the Micom Box Type 6 start at $4,320 for the eight-channel unit. For more information, contact **Micom Systems, Inc.**, 4100 Los Angeles Avenue, Simi Valley, CA 93062; (800) 642-6687.

Circle reader service #20

Lowell Introduces UPS Series

The Emerson AP101 series is a new uninterruptible power source from Lowell Corporation. This UPS provides 120 VAC or 240 VAC, 60 Hz power and has a single rotary switch and LEDs to guide the user through start-up. It comes standard with internal static and maintenance bypasses, remote emergency off, and a 10-minute battery. The AP101 comes in 3-, 5-, and 10-KVA versions and has options for an RS-232 port, remote alarm relays, extended battery time, and up to six receptacle panels.

List prices for the new Emerson UPS range from $9,000 to $17,000. For more information, contact **Lowell Corporation**, 97 Temple Street, Worcester, MA 01613; (508) 756-5103.
Circle reader service #21

SCSI Hard Disk Kit Offered for NetWare 2.1

Rancho Technology has a new hard-disk installation kit to optimize NetWare 2.1. The NetWare 2.1 kit, RT10AT-NS, includes 16-bit SCSI host adapter, software, and cables. The RT10XT-NS includes an 8-bit SCSI host adapter, software, and cables. The RT100XT-NS is similar to the RT10XT-NS, but with the addition of a built-in floppy disk controller on the host adapter board. A separate SCSI NetWare 2.1 driver is also available.

The RT10AT-NS sells for $245, the RT10XT-NS sells for $199, and the RT100XT-NS sells for $215. The separate driver is available for $45. For more information, contact **Rancho Technology**, 8632 Archibald Avenue, Rancho Cucamonga, CA 91730; (714) 987-3966.
Circle reader service #22

Smart Switches Handle FAX and Modem Traffic

Three new FAX/modem switches have been introduced by High-Tech Resources, Inc., to solve barge-in problems and monitoring problems associated with toggle switch systems. The switch family is compatible with all FAX machines, dial modems, and key telephone systems. The Model V/F/M FAX/modem switch allows a FAX or modem to share a voice line and features 24-hour auto-answer with preset rings. It installs ahead of the Key Service Unit (KSU), bypassing the KSU to provide dedicated line status. The Model F/M/A is designed for single-line or multiline installations to monitor incoming FAX tones and/or may be switched to the FAX or modem either locally or remotely using the Star telephone key. The F/M/A FAX switch can also be used as an answering machine, instructing callers to "Press Star for the FAX," "Leave a message at the tone," or "Leave a message and then press Star to connect with the FAX." The model M/F FAX/modem switch is designed to handle two devices on a dedicated phone line by deciphering the incoming tones and connects the appropriate receiving device. Company sources indicate that the M/F switch works particularly well with FAX boards, Hayes modems, and dial-up printers.

The FAX/modem switches range in price from $100 to $350. For more information, contact **High-Tech Resources**, 4225 West Glendale, #102, Phoenix, AZ 85051; (800) 422-2832 or (602) 931-0793.
Circle reader service #23

16-port Serial Card Offers Total UNIX Compatibility

Dastra America has released the MCIO/16, a 16-channel, intelligent serial card for use with IBM PCs, XTs, and ATs. The card features default firmware that is specially designed for SCO Xenix, UNIX, and other multiuser operating systems. MCIO/16 supports the full range of UNIX System V line disciplines and a variety of standard, asynchronous line protocols (all of which are compatible with RS-232C). It supports all common line speeds from 50 to 34,400 bps. MCIO/16 users can implement such features as line editing and character set translations, and build an intelligent, integrated subsystem. Two models of connector boxes are used, both of which use eight D89P subminiature male connectors compatible with PC/AT cable.

The MCIO/16 is available in two configurations, an 8-port version and a 16-port version. It is priced at $1,095 and $1,495, respectively. For more information, contact **Dastra Americam Inc.**, 976 North Lemon Street, Orange, CA 92667; (714) 633-2240.
Circle reader service #24

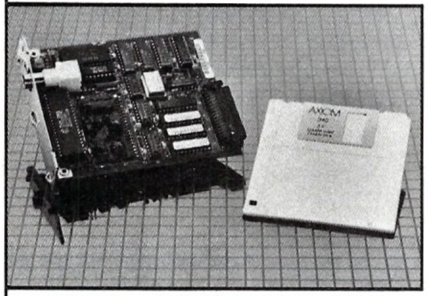
Upgrade Boards Offer 386 Processing Power

The STD-386 is a 386-based accelerator board from Seattle Telecom & Data, Inc., that offers full compatibility with IBM's OS/2, DOS and UNIX 386. STD-386 will fit virtually any card on the expansion bus. It is also compatible with a variety of network operating systems, including NetWare, VINES, and 3Com. The accelerator board operates at three speeds for software compatibility. The STD-386 uses static column, high-speed 32-bit memory, the same memory as that of the Compaq 386, with the addition of two-way interleaving which has been optimized for 32-bit processors. It features its own 80386 microprocessor and a choice of 80387 math coprocessor or Weitek 1167 Floating Point daughterboard (with 4 mflops floating point).

Also new from STD is the STD-386XT, a "plug-and-play" 80386 motherboard designed to fit the IBM XT and AT. The replacement motherboard is available in either 16- or 20-MHz versions, is configured with eight expansion slots, and has a choice of 1, 2, 4, 8, or 16 MB of memory. It comes delivered with an 8-MHz 80287 math coprocessor socket on board and offers an optional synchronous (16- or 20-MHz) 387 daughterboard.

The STD-386 sells for $1,995. The 16-MHz 386 processor motherboard with 1 MB of memory is $1,750; $2,550 for the 20-MHz version. For more information, contact **Seattle Telecom & Data**, 12277 134th Court NE, Suite 205, Redmond, WA 98052; (206) 820-1873.
Circle reader service #25 □

GET EASY ARCnet® ACCESSIBILITY FOR YOUR TOSHIBA PORTABLES



Now you can easily link your laptops from Toshiba to the power and flexibility of ARCnet®. Pure Data is shipping the new PDT508™ fully compatible ARCnet interface card for Toshiba T1100 PLUS, T1200, T3100 and T5100 portable computers.

The PDT508 installs easily in the Toshiba expansion slot and comes complete with configuration and diagnostics software and there are no DIP switches to fumble with. This Pure Data network innovation also includes on-screen system activity and diagnostic capability.

All Pure Data products are covered with our exclusive five year limited warranty from date of purchase.

For more information about the PDT508 call today. Dealer inquiries are welcome.

Pure Data
COMMUNICATING QUALITY

TORONTO
(416) 731-6444

1740 South I-35, Suite 140
Carrollton, Texas 75006
(214) 242-2040
FAX (214) 242-9487

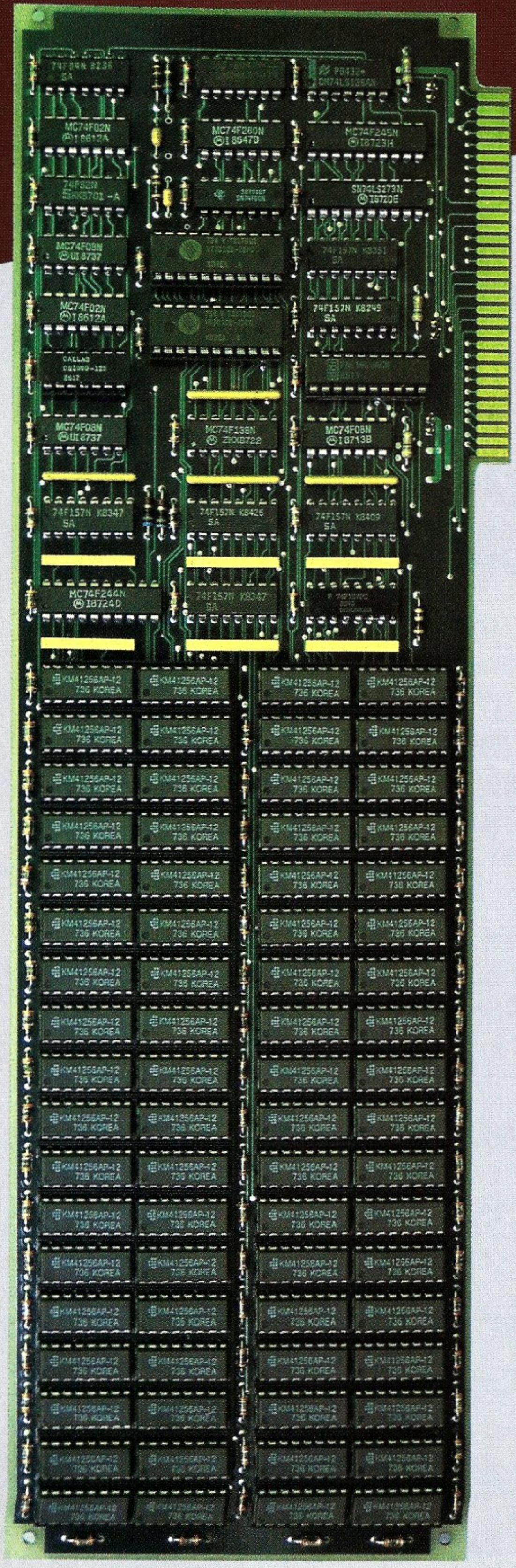
LONDON, U.K.
(0628) 770011

ARCnet is a registered trademark of Datapoint Corp.
PDT508 is a trademark of Pure Data Ltd.

CIRCLE 43 ON READER SERVICE CARD

X-BANDIT

Designed for the EMS 4.0 Standard



DESIGN PHILOSOPHY

• The Teletek X-Bandit was specifically designed to utilize the advanced features of the Lotus/Intel/Microsoft EMS 4.0 Specification. It is available in both 8 and 16 bit versions for use in the IBM XT, AT, and compatibles.

MEMORY

• Segmented Memory Mapping allows the user to fill out unused memory segments between 640K and 1 Megabyte.

• Split Memory Addressing allows the user to fill out conventional memory to 640K.

• Extended Memory Addressing is available for the PC/AT version.

• 2 MB capacity in a single slot. Up to 8 MB per system.

• Parity checking.

SOFTWARE

• Easy menu-driven auto configuration software.

• Device driver includes print spooler and RAM drive.

• Supports multitasking with the appropriate shell-resident software package.

SPEED

• 6/8/10/12 MHz speed with 0 wait states. 16 MHz speed with 1 wait state.

WARRANTY

• One year parts and labor.

TELETEK

4600 Pell Drive
Sacramento, CA 95838
(916) 920-4600
Telex #499-1834

CIRCLE 80 ON READER SERVICE CARD

DOS system running Lotus 1-2-3

SCO XENIX system running SCO Professional

THIS IS AN IBM PS/2 MODEL 80 RUNNING DOS

Under DOS, this PS/2™ is a powerful 80386-based single-tasking, single-user computer that can run thousands of DOS applications. In 16-bit, 8086 mode.

One at a time.

When OS/2™ software becomes available, the PS/2 can become a multitasking, single-user computer running in 16-bit, 286 mode that can also single-task those DOS applications under OS/2.

One at a time.

With DOS or OS/2, the PS/2 will support one user.

	1 user (DOS)	1 user (OS/2)
Cost per system**:	$12,389	$12,594
Cost per user:	$12,389	$12,594

THIS IS AN IBM PS/2 MODEL 80 RUNNING SCO XENIX

Under SCO XENIX,® this PS/2 becomes a powerful 80386-based multitasking, multiuser computer that can run thousands of XENIX applications. In full-tilt, 32-bit, 386 mode.

Many at a time.

And using SCO VP/ix,™* the PS/2 can multitask DOS applications under SCO XENIX.

Many at a time.

With SCO XENIX, the PS/2 will support one user.
Or 9 users. Or even 33 users.

And it can do all that today because you can get SCO XENIX for the PS/2—*now!*

	1 user	9 users	33 users
Cost per system**:	$14,559	$19,726	$40,402
Cost per user:	$14,559	$2,192	$1,224

SCO XENIX System V and the SCO XENIX family of software solutions are available for all industry-standard 8086-, 80286-, and 80386-based computers, and the IBM® Personal System/2™ Models 50, 60, and 80.

SCO™
THE SANTA CRUZ OPERATION

(800) 626-UNIX (626-8649)
(408) 425-7222
FAX: (408) 458-4227
TWX: 910-598-4510 SCO SACZ
uucp: ...decvax!microsoft!sco!info!

*SCO VP/ix available as separate product.

Cost comparisons are based on most recently published U.S. domestic suggested list prices. **Cost model: Base machine: IBM PS/2 Model 80, 70Mb disk, 1Mb RAM, IBM 8512 color monitor, 1Mb additional IBM RAM, IBM ProPrinter XL,® **1-user DOS system:** Base machine, plus DOS 3.3, WordPerfect 4.2, Lotus® 1-2-3,® dBASE III PLUS.® **1-user OS/2 system:** 1-user DOS system; substitute OS/2 for DOS. **1-user SCO XENIX system:** Base machine, plus SCO XENIX 386 for PS/2, SCO VP/ix, SCO Lyrix® (word processing), SCO FoxBASE+™ (dBASE III PLUS workalike), SCO Professional™ (1-2-3 workalike). **9-user SCO XENIX system:** 1-user SCO XENIX system, plus intelligent 8-user multiport card, 8 IBM 3151 ASCII terminals. **33-user SCO XENIX system:** 9-user SCO XENIX System, plus 3 more intelligent 8-user multiport cards, 24 more IBM 3151 ASCII terminals, 4 Mb additional RAM, additional 70Mb disk.

CIRCLE 94 ON READER SERVICE CARD