THE JOURNAL FOR ADVANCED MICROCOMPUTING

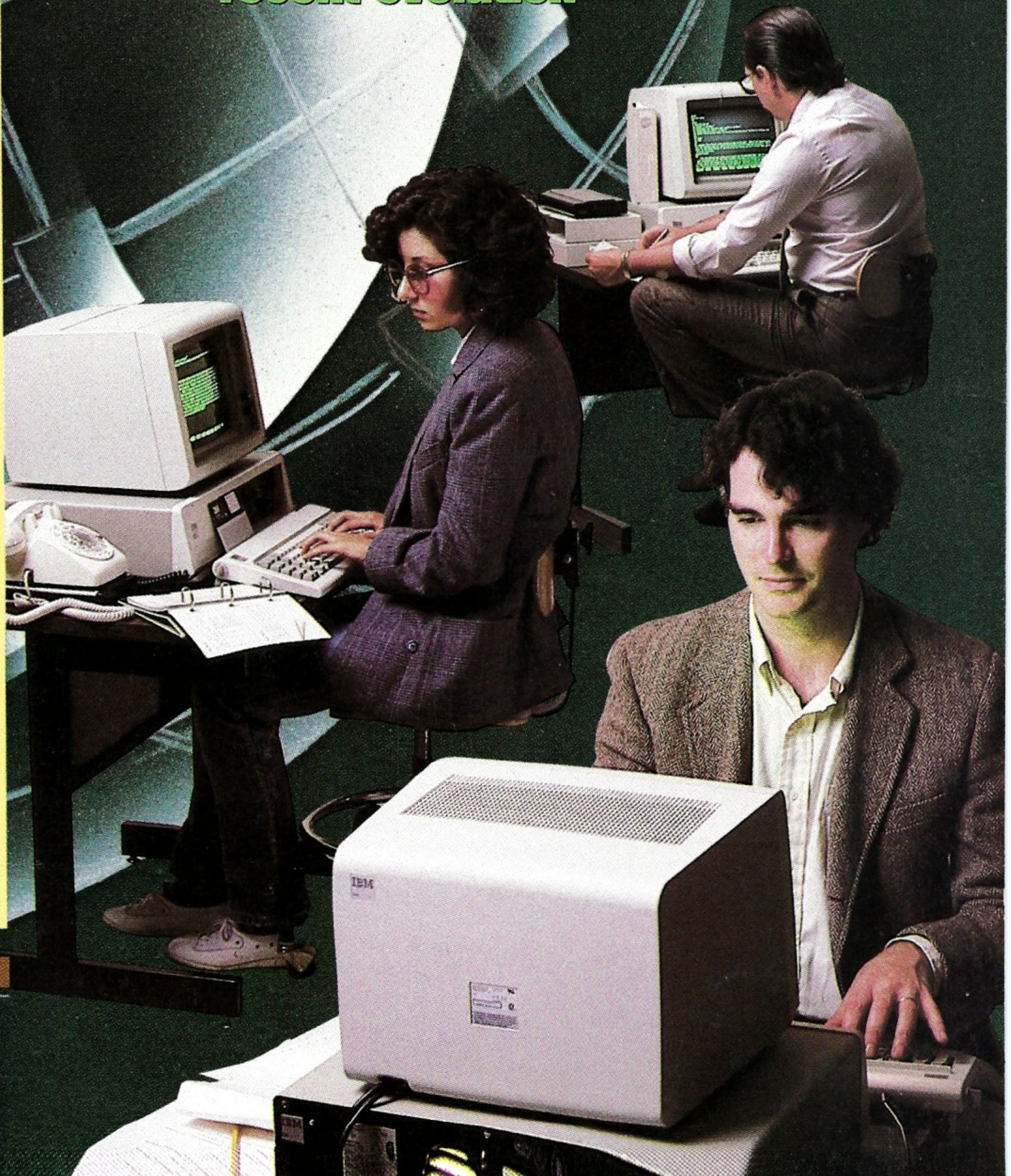
$2.95

Microsystems

Volume 5/Number 11

November, 1984

The expanding UNIverse—travel with us through its recent evolution



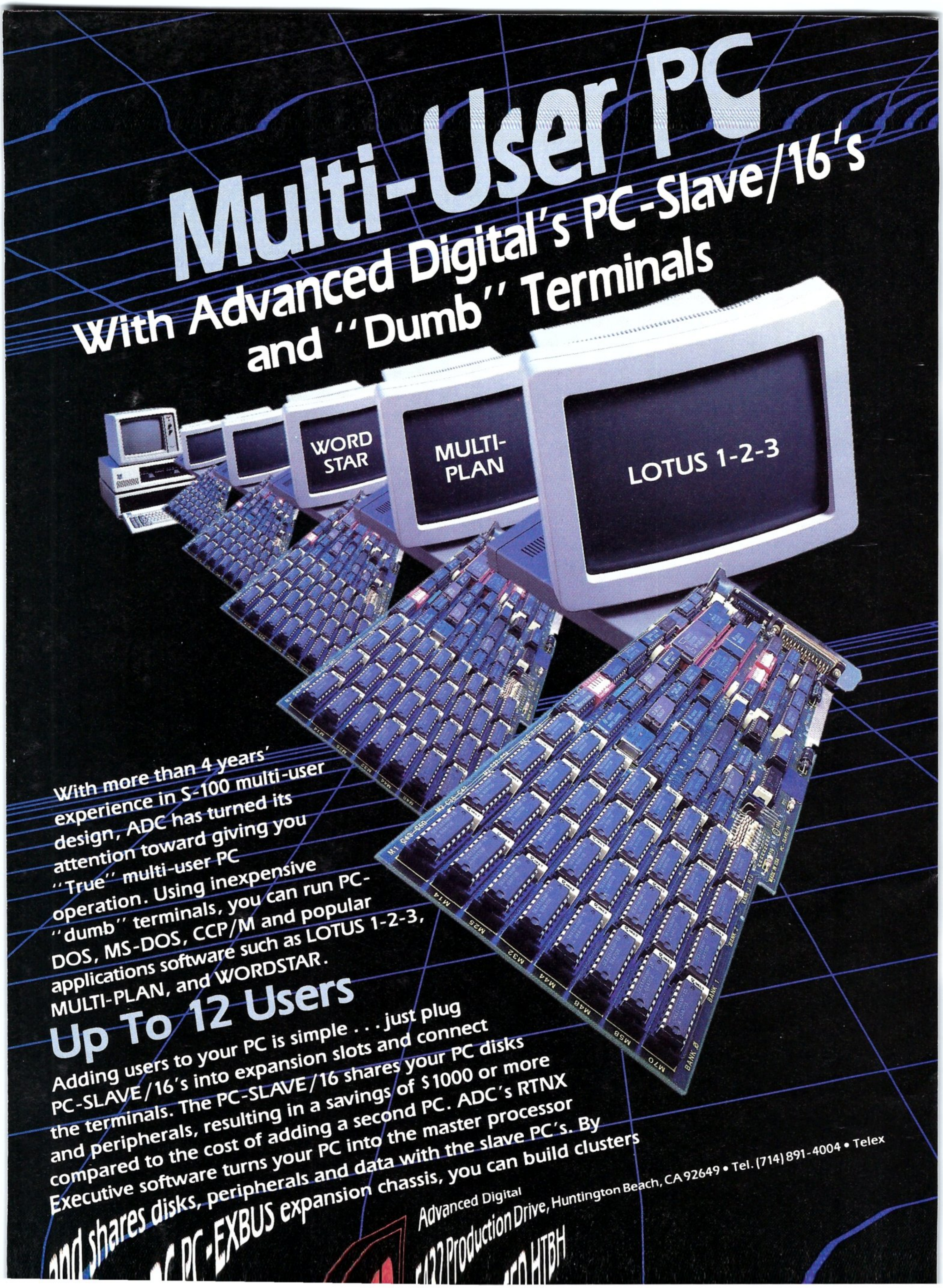
UNIX
Recent Evolution of UNIX
Talking with Andy Hall, Director System V Development
UNIX on a Codata 3300
How Portable is C?
The UNIX File: Shows, the Shell, and Shibboleths

MS-DOS
Tutorial on PROMPT
MS-DOS Window: Bulletin Boards on the PC

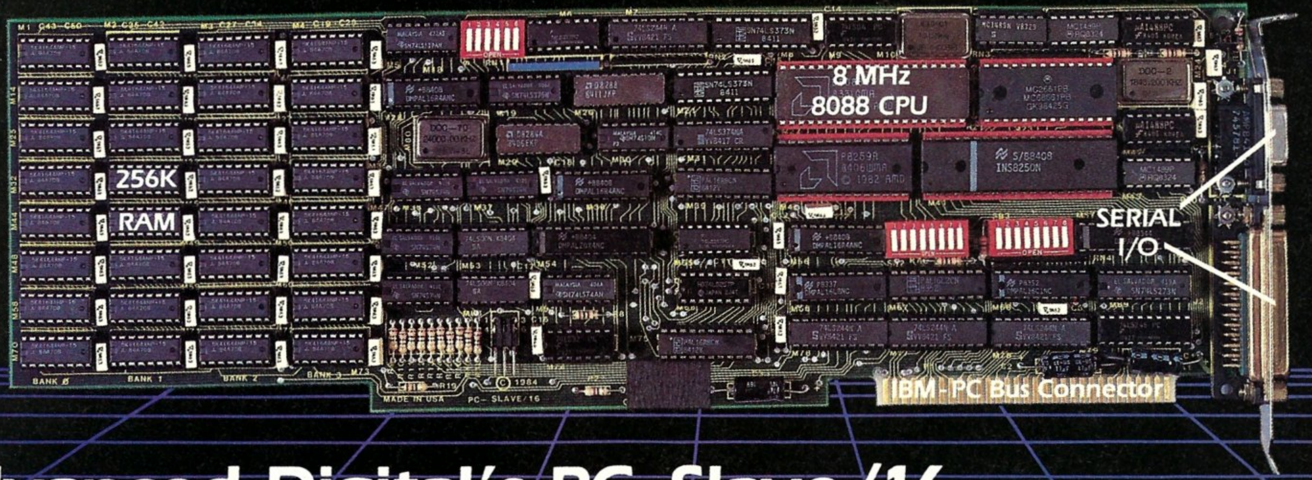
Graphics
Graphics Palette: HALO

Communications
X.25 Protocol

Product Reviews
Pfix86 Symbolic Debugger


Multi-User PC
With Advanced Digital's PC-Slave/16's and "Dumb" Terminals
WORD STAR
MULTI-PLAN
LOTUS 1-2-3
With more than 4 years' experience in S-100 multi-user design, ADC has turned its attention toward giving you "True" multi-user PC operation. Using inexpensive "dumb" terminals, you can run PC-DOS, MS-DOS, CCP/M and popular applications software such as LOTUS 1-2-3, MULTI-PLAN, and WORDSTAR.
Up To 12 Users
Adding users to your PC is simple . . . just plug PC-SLAVE/16's into expansion slots and connect the terminals. The PC-SLAVE/16 shares your PC disks and peripherals, resulting in a savings of $1000 or more compared to the cost of adding a second PC. ADC's RTNX Executive software turns your PC into the master processor and shares disks, peripherals and data with the slave PC's. By
PC-EXBUS expansion chassis, you can build clusters
Advanced Digital
Production Drive, Huntington Beach, CA 92649 • Tel. (714) 891-4004 • Telex

Powerful & Cost Effective



Advanced Digital's PC-Slave/16

Lets your PC Expand

The PC-SLAVE/16 creates a multi-processor PC cluster with shared data. Each user (up to 12) has his own "system" for local processing and shares the host PC's peripherals. Compare the costs of adding users to your PC using PC-SLAVE/16's with that of most networks. Then look at the performance . . . enjoy processing at almost twice the speed of the IBM-PC and ten times the speed of most Local Area Network schemes.

To expand your existing PC:	PC-SLAVE/16 and Terminal	Networking Multiple PC's	Savings
2-User System	$1650	$3600	$1450
4-User System	$4950	$9600	$4650

Runs Popular PC Software –

ADC's RTNX Executive software provides the PC-SLAVE/16 with a completely transparent interface to PC-DOS,* MS-DOS* or CCP/M* and emulates the PC's display on most popular terminals. You can run LOTUS 1-2-3, Multi-Plan and WordStar. Even the latest "Windowing" oftware will run with PC-SLAVE/16. To find out more about the PC-SLAVE/16, visit your ocal dealer or contact Advanced Digital Corporation.

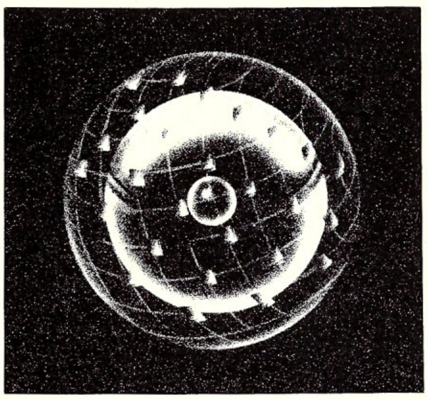
ADVANCED DIGITAL CORPORATION

Advanced Digital
5432 Production Drive, Huntington Beach, CA 92649 • Tel. (714) 891-4004
Telex 183210 ADVANCED HTBH CIRCLE 18 ON READER SERVICE CARD

Advanced Digital U.K. Limited
27 Princess St., Hanover Square • London WIR8NQ • United Kingdom
409-0011 409-3359 • Telex 265840 FINEST

Microsystems November

Volume 5/Number 11
November 1984



The expanding UNIverse —travel with us through its recent evolution

DEPARTMENTS

Editor's Page 6
News and Views 8
The S-100 Bus 12
The UNIX File 16
MS-DOS Window 24
Graphics Palette 30

Cover photo: Diana Church
Cover Art: Carol Gillot

The Worlds of UNIX 42
by Mark Rollins
A discussion of the development and future of UNIX introduces the articles that follow

The Evolution of UNIX from 1974 to the Present 44
by Ian Darwin and Geoff Collyer
UNIX in the last decade: the key ideas that made it evolve, and the individuals who contributed to its growth

A Conversation with Andrew D. Hall 58
Andy Hall, Director of the Bell Labs System V Development, talks about the future of UNIX

Codata 3300, Running Unisoft UNIX 68
by Bruce Hunter
Codata offers a reliable implementation of Unisoft UNIX at a moderate price

How Portable is C? 84
by Michael Tilson
Avoid some of the most common pointer and data typing problems that arise when programming in C

PFIX86 from Phoenix 92
by Michael Olfe
Phoenix Software has released a symbolic debugger for a standard PC/MS-DOS environment

The MS-DOS PROMPT Command has more Power than You Think 100
by Charles Petzold
PROMPT is more than a simple command—it will display the current directory path, change the color/graphics display mode, and redefine the keyboard

X.25 Communications Protocol 108
by Eric L. Beser
Part 3: Learn how to implement X.25 using the Intel 8274 and Western Digital WD2511 chips

Microsystems

STAFF

Sol Libes	founder & editorial director
Mark Rollins	editor
Chris Terry	technical editor
Susan Hurley	managing editor
Andrew Bender/Ian Darwin/David Gewirtz/Dave Hardy/Hank Kee/Steve Leibson/Don Libes/David McCune/Bruce Ratoff/Randy Reitz/Anthony Skjellum	contributing editors
Ann Ovodow	editorial coordinator
Tom Leander	editorial assistant
Will Kefauver	art director
Tracy Tyler/Tammi Colichio	art assistants
Jim Beloff	advertising director

ADVERTISING SALES OFFICES

New England, Midatlantic
Jeff Tompkins, *Microsystems*
Ziff-Davis Publishing Company
One Park Avenue
New York, NY 10016
(212) 503-5016

Advertising Coordinator
Michele Fischetti, *Microsystems*
Ziff-Davis Publishing Company
One Park Avenue
New York, NY 10016
(212) 503-5017

Southeast
Mark Browning, *Browning Publications*
P.O. Box 81306
Atlanta, GA 30366
(404) 455-3430

Midwest
William Biff Fairclough/
Jeff Edman, *The Pattis Group*
4761 W. Touhy Avenue
Lincolnwood, IL 60646
(312) 679-1100

Southern California, Southwest
Steve Taneman
Ziff-Davis Publishing
3460 Wilshire Blvd.
Los Angeles, CA 90010
(213) 387-2100

Northern California, Northwest
Jeff Cohen,
Ziff-Davis Publishing
11 Davis Dr.
Belmont, CA 94002
(415) 598-2290

Canada
Frank Lederer, *The Pattis Group*
501 Eglinton Ave., E. #202
Toronto, Ontario
M4P 1N4
(416) 482-6288

Direct Retail Sales
Lynn Kujawa,
Ziff-Davis Publishing
One Park Avenue
New York, NY 10016
(212) 725-7679

COMPUTER PUBLICATIONS DIVISION

General Manager	Kenneth H. Koppel
Senior Vice President	Eileen G. Markowitz
Vice President Editorial	Jonathan Lazarus
Vice President Licensing and Special Projects	Jerry Schneider
Vice President Creative Services	Herbert Stern
Circulation Director	Alicia Marie Ivans
Editorial Director	Ernest F. Baxter
Creative Director	Peter J. Blank
Marketing Manager	Ronni Sonnenberg
Business Manager	Gary A. Gustafson

ZIFF-DAVIS PUBLISHING

Richard P. Friese, *President* Albert S. Traina, *President, Consumer Magazine Division* Paul H. Chook, *Executive Vice President, Marketing and Circulation* Phillip T. Heffernan, *Senior Vice President* Sidney Holtz, *Senior Vice President* Edward D. Muhlfeld, *Senior Vice President* Philip Sine, *Senior Vice President* Baird Davis, *Vice President* George Morrissey, *Vice President* Rory Parisi, *Vice President* William L. Phillips, *Vice President* Selwyn Taubman, *Treasurer* Bertram A. Abrams, *Secretary*

MICROSYSTEMS (ISSN #0199-7955) is published monthly by Ziff-Davis Publishing Company, One Park Avenue, New York, N.Y. 10016.

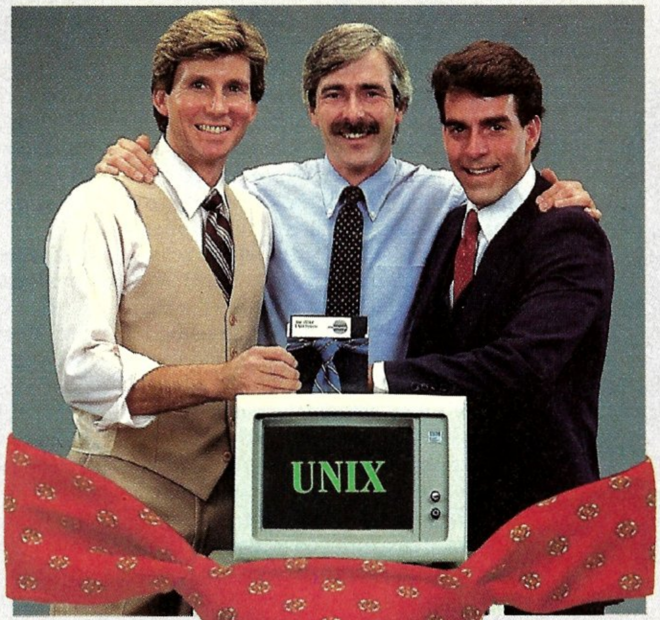
Second Class postage paid at New York, N.Y. 10016 and at additional mailing offices. POSTMASTER: Send address changes to MICROSYSTEMS, PO Box 2937, Boulder, CO 80322. Subscriptions are $26.97 for 12 issues. Canadian prices are $5.00 per year additional; other foreign $8.00 per year additional (U.S. currency only). For information or questions about subscriptions phone: (800) 631-8112.

Editorial correspondence is welcomed and should be sent to: MICROSYSTEMS, One Park Avenue, New York, NY 10016. Phone (212) 503-5555.

For information on commercial advertising, write to: MICROSYSTEMS Advertising Dept., One Park Avenue, New York, NY 10016 or call Jim Beloff at (212) 503-5015.

See us at COMDEX™/Fall '84
November 14-18, 1984
Riviera Hotel
Las Vegas, Nevada

WE DRESSED OUR UNIX* SOFTWARE FOR YOUR IBM PC/XT



Try it Risk FREE — the first AT&T-licensed UNIX implementation for the IBM PC/XT. Shipping since August 1983, Venix/86 is the popular choice among knowledgeable UNIX users and developers. Here's why!

Multi-User Capability...

Share the same PC, disk, and printer with up to three users! Simply plug in a CRT and run.

Multi-Tasking...

Edit a file, print a report, run a spelling check, format a diskette...all at the same time.

Berkeley Enhancements...

Including vi, termcap, more and the c shell.

Real-Time Extensions...

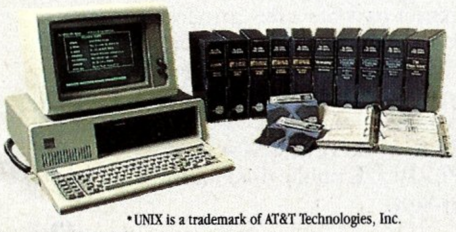
With semaphores, raw and asynchronous I/O, priority, shared data, I/O page addressing.

Quad-Screen Windowing...

Featuring four unique and powerful windows.

MS-DOS Partitioning...

Keep your DOS files and programs!



*UNIX is a trademark of AT&T Technologies, Inc.
Venix/86 implementation by VenturCom, Inc.

Lean and Clean...

192K RAM, 3.5 Mbytes on disk. Proven reliability.

Applications...

Networking, word processors, database managers, spreadsheets, menu interfaces.

One Source with Unisource...

Unisource is the leading publisher and developer of UNIX software for the IBM PC/XT and compatibles, DEC Professional 350, Rainbow, Micro-11, PDP-11, VAX series, and NCR computers. All our packages are fully documented and supported by our 800 user hotline. Call for a complete information kit or to arrange your 30-day Risk FREE Trial of Venix/86. Unisource Software Corp. Department 4130
71 Bent St., Cambridge, MA 02141.
Telex 92-1401/COMPUMART CAM

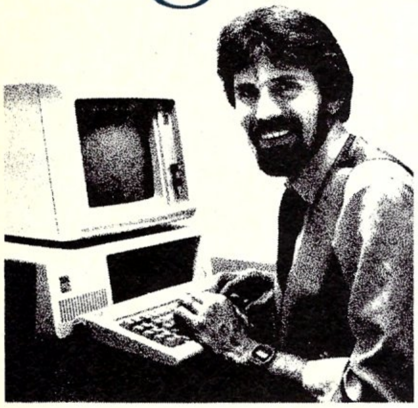
CALL 617-491-1264

UNISOURCE

Getting UNIX Software Down to Business

CIRCLE 82 ON READER SERVICE CARD

Editor's Page



Microsystems says goodbye

We have sad news.

This is the last issue of *Microsystems* that Ziff-Davis will be publishing. The reasons are purely financial, having nothing to do with the editorial content, which has been increasingly praised by our readers, advertisers, and the Ziff-Davis management. It is difficult to write about this while going through the emotional turmoil we on the staff are at the moment. All of us on the staff would like to thank you, the readers, for your support. We also wish to thank our authors for their outstanding articles, which have made *Microsystems* so highly respected in the industry.

Sol Libes' imagination and editorial flair set the viewpoint and laid the foundations. In my short tenure, we modified the viewpoint to maintain a state-of-the-art direction for system developers. This annoyed a handful of readers. Scores more, as well as Sol himself, supported and praised the change. The departure of *Microsystems* from the publishing scene is indeed a sad event.

The micro world has gone through immense changes since *Microsystems* first began publishing. For the skeptics, we still support the potential value of S-100 systems in today's world, if vendors provide the proper tools. We strongly support the moves toward standards that can satisfy the needs of both developers and end users. And we also support those in the industry who maintain a critical eye toward short-term profit motives that jeopardize the long-term benefit and health of the industry.

From Sol Libes

It saddens me deeply that this is the last issue of *Microsystems.* A great many loyal readers called when they heard the news, expressing their shock and disbelief. They found me in a thoroughly depressed state, and I appreciated their concerns. Many pleaded with me to resume publishing "my baby." Several offered to work on the venture. I seriously considered doing it again . . . but then I thought back to the days in 1980 when my wife and I worked 24 hours a day, seven days a week—and starting up a magazine today would be even more work, with greater risk.

The era of the early pioneers, when most of the work came from individuals working in garages and basements, has pretty much run its course. I look back on those early days with fondness. And I think I learned more from editing *Microsystems* than any reader did from reading it.

I am indebted to all the authors who wrote for the magazine, realizing that theirs was primarily a labor of love and discovery. To the people who worked on producing *Microsystems:* to Chris Terry, Mark Rollins, and the editorial and advertising staffs. To Will Kefauver, the art director who redesigned *Microsystems* into a magazine that was not only good to read, but good to look at. To our advertising and editorial production coordinators Peter Kasakove, Sue Conroy, and Lenny Fried. To all the subscribers who wrote and called to give their opinions, suggestions, praise and encouragement over the years.

My wife and I are continuing our teaching jobs (we are both Community College Professors in New Jersey), and I expect to write a few articles from time to time for various publications. I am continuing to remain active in the Amateur Computer Group of New Jersey (which I founded in 1975 and headed for five years), as well as the SIG/M and PC/BLUE public domain software user groups.

Microsystems *Editorial Staff. From left: Mark Rollins, Tom Leander, Ann Ovodow, Chris Terry, Susan Hurley, and Randolph Johnson.*

Microsystems *Advertising East Coast Sales Staff. From left: Michelle Fischetti, Jim Beloff, and Jeff Tompkins (not shown: Steve Taneman, Jeff Cohen, Jeff Edman, Biff Fairclough, Mark Browning, and Frank Lederer).*

by Mark Rollins

Gifford's Multiuser Concurrent DOS.™ The net that works!

Gifford has the network solution. It's simple, fast, secure, complete, and it works. Multiuser Concurrent DOS is based on Digital Research's Concurrent DOS, the only major microcomputer operating system specifically designed for networking.

Users can share disks and printers transparently, and can also take advantage of true multiuser features like file and record lockout. And Gifford has added a bundle of features that makes Multiuser Concurrent DOS easy to install and use. It lets you get right to work.

Our net is ARCNET.™

Multiuser Concurrent DOS utilizes Datapoint's ARCNET, the most popular network hardware in the industry. It's reliable, economical, and fast—so you can add users without overloading the network.

You can network up to 255 single and multiuser systems. You can connect single or multiuser Gifford or CompuPro® systems as well as IBM PC-XTs®. Dual processor Gifford and CompuPro systems can run thousands of 8 or 16 bit CP/M or MP/M applications. PC-XTs can run 16 bit CP/M and MP/M programs as well as most popular MS-DOS applications, such as Lotus 1-2-3.™

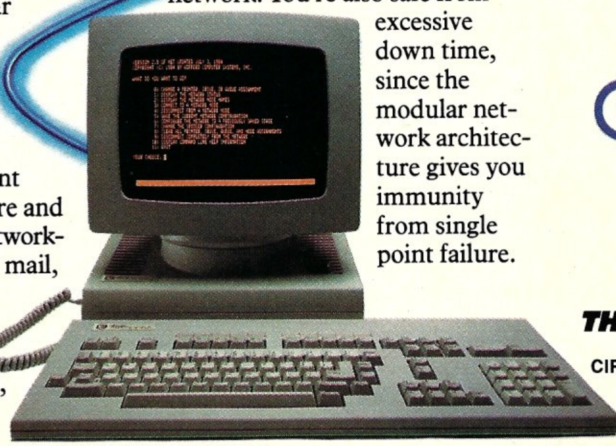
Gifford adds to your net worth.

Our enhancements of Concurrent DOS make it possible to get more and better work done in less time. Network-wide features include electronic mail, event calendar, inter-terminal communication, user time accounting and usage report generation, telecommunications, user expandable HELP facility, reminder messages, message of the day, automatic startup and shutdown procedures, and easily prepared files for initializing terminals, printers, and network nodes.

Gifford's Virtual Terminals™ increase productivity by offering full-screen concurrency; you can run up to four programs simultaneously from one physical terminal.

The safety net.

Multiple users can mean multiple security problems. Gifford's security enhancements include login account names and encrypted passwords to control access to the system. Users can be further restricted to specified terminals, user areas, programs, or nodes on the network. You're also safe from excessive down time, since the modular network architecture gives you immunity from single point failure.



Gifford nets a big one: Simplicity.

If you've gone through the ordeal of typing as many as seven commands just to get on and off a network, Gifford has your number.

A single, menu-driven network command handles all your network options. Everything you need is right in front of you. The net effect is simplicity—and sanity.

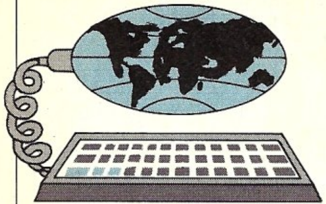
If you'd like to see how Gifford's Multiuser Concurrent DOS can solve your networking problems, or if you'd like to know about Gifford's selection of multiuser systems and software options, call (415) 895-0798.

Or write us at the address below. We'll send you a free networking brochure and give you the name of the nearest dealer.

Multiuser Concurrent DOS is a trademark of Gifford Computer Systems. Concurrent DOS is a trademark of Digital Research, Inc. ARCNET is a trademark of Datapoint Corporation. IBM PC-XT is a registered trademark of IBM Corporation. CompuPro is a registered trademark of CompuPro Corporation. Lotus 1-2-3 is a trademark of Lotus Development Corp. Virtual Terminals is a trademark of Gifford Computer Systems.

GIFFORD COMPUTER SYSTEMS
A subsidiary of Zitel Corporation
2446 Verna Court
San Leandro, CA 94577
(415) 895-0798 TELEX: 704521
Houston, TX (713) 680-1944
Los Angeles, CA (213) 477-3921
Amherst, NY (716) 833-4758

THE MULTIUSER COMPANY™
A division of Zitel Corporation
CIRCLE 181 ON READER SERVICE CARD



News & Views

Random rumors and gossip, plus a view of the industry's latest trends

by Sol Libes

There are reports that Sharp Electronics has been privately showing a prototype lap-sized system with UNIX in read-only memory . . . Seequa Computer Corp. reduced the list price of its IBM-PC-compatible transportable to $1,595, in a price war that is heating up. Considering that this machine, as well as virtually all the PC compatibles, is usually discounted from 10-20%, it means that the low end of the IBM-PC marketplace is fast approaching the $1,000 breakpoint . . . Commodore, which had demoed a low-cost UNIX-based system based on the Zilog Z8000, is now rumored to have dropped the project after the four leading engineers switched to Atari (now run by their old boss Jack Tramiel). Commodore filed a suit against the engineers for allegedly stealing design secrets. Atari is expected to introduce a 16-bit personal computer in January and a 32-bit machine next summer . . . There are rumors that IBM is seriously considering dropping Microsoft Basic for its PC and offering, instead, a version that is ANSI (American National Standards Institute) compatible.

Apple is rumored considering licensing the Macintosh technology to one or two manufacturers in an attempt to establish the Mac as an industry standard—à la the IBM-PC technology. Apple is also rumored to be hard at work on a color version of the Mac . . . The recently released PC-DOS 3.0 from IBM lacked multitasking capability and multiuser facilities. PC-DOS 4.0, an almost complete UNIX lookalike, is expected out sometime next summer. Rumored further down the line is an IBM-VM implementation for a desktop machine with PC-DOS and UNIX running as guest operating systems under VM. IBM already has CMS running on its XT-370, but without guest operating system capability . . . Texas Instruments is expected shortly to begin sampling its token ring Local Area Network chip set; this means that production will begin by mid '85, with systems being introduced by late '85 or early '86. Since this is most likely the chip set that IBM will use for their PC-LAN system, we can expect IBM to introduce their PC-LAN products within the same time frame . . . Digital Research (producers of CP/M) is rumored to be developing a graphics-based interface product for IBM's new 80286-based PC, code-named Crystal, under contract to IBM and working on development of a similar product for AT&T . . . IBM is rumored to be developing their own integrated software package for the PC based on its Query-By-Example relational database language used on its large mainframes. It is expected to include graphics, spreadsheet and electronic mail facilities and to require 512K of RAM. It is also expected to use a desktop metaphor similar to that used on the Apple Lisa . . . Data General is expected to enter the knee-top computer market with a machine similar to the Hewlett-Packard portable that has a 25-line x 80-column LCD display, 3.5″ disk drive, 128K RAM and IBM-PC compatibility . . . Digital Equipment Corp. is said to be close to announcing local area networking capability for their Professional 350 desktop unit running a UNIX-like operating system . . . Fortune System Inc., a maker of UNIX-base systems, is rumored negotiating to buy North Star Computer Inc.

IBM releases PC/AT

As predicted months ago in this column, IBM, in August, introduced a new, more powerful version of the PC. Called the PC/AT, it is based on the new Intel 80286 microprocessor (a true 16-bit device) and supports up to 3 MB of RAM (the PC supports 512K maximum), 1.6 MB floppies (PC supports only 360K units), a 20 MB hard disk (PC/XT has a 10 MB unit) and a new version 3.0 of PC-DOS. The basic model ($3995) comes with 256K of RAM and a single floppy; the expanded model ($5795) has 512K of RAM, floppy and hard disk drives, and parallel/serial I/O card. The PC/AT keyboard corrects the earlier PC keyboard problems.

IBM also announced it will introduce, in the first quarter of '85, a PC local area networking system to link together up to 72 PCs with a PC/AT functioning as a file server. A broadband coaxial cable LAN system, it transmits at 2 Mbits/sec and costs about $700 per station. IBM's LAN is thus slower and more expensive than some of the systems (e.g. Ethernet) that have been on the market for some time. Further, this is really an interim product, as IBM will introduce a token-ring LAN in another two years.

Also to be available in the first quarter of '85 will be PC/Xenix (a $1,000 software package), a three-user version of Microsoft's multiuser, multitasking operating system based on UNIX System III. Note that XENIX and several other UNIX versions for the PC/XT have been available, from other suppliers, for a long time. And many of these versions run faster, support more users, and cost less.

THE MI-286 DUAL CPU BOARD IS AT LEAST TWICE AS FAST AS COMPUPRO'S 8085/88... AND IT'S A DIRECT REPLACEMENT!

The 20-second revolution. It only takes about 20 seconds to bring your S-100 system up to its ultimate speed/power potential. Just pull out the old fashioned 8085/88 board and plug the MI-286 in its place. That's all there is to it. You're off and running with more power than ever before.

The 80286 and Z-80H. The MI-286 is the first dual-CPU board using the new, high speed Intel 80286 coupled with a Z-80H. It is designed for use with a variety of operating systems, including MP/M 8-16. It will support all your current 8086/88 and Z-80/8085 software. It can accommodate an optional 80287 math co-processor. In short, it gives you the best of both worlds.

Add more users. The MI-286. It improves throughput so dramatically you can add those extra users you've always wanted. It carries S-100 technology to its logical limit. And it only costs $1395!*

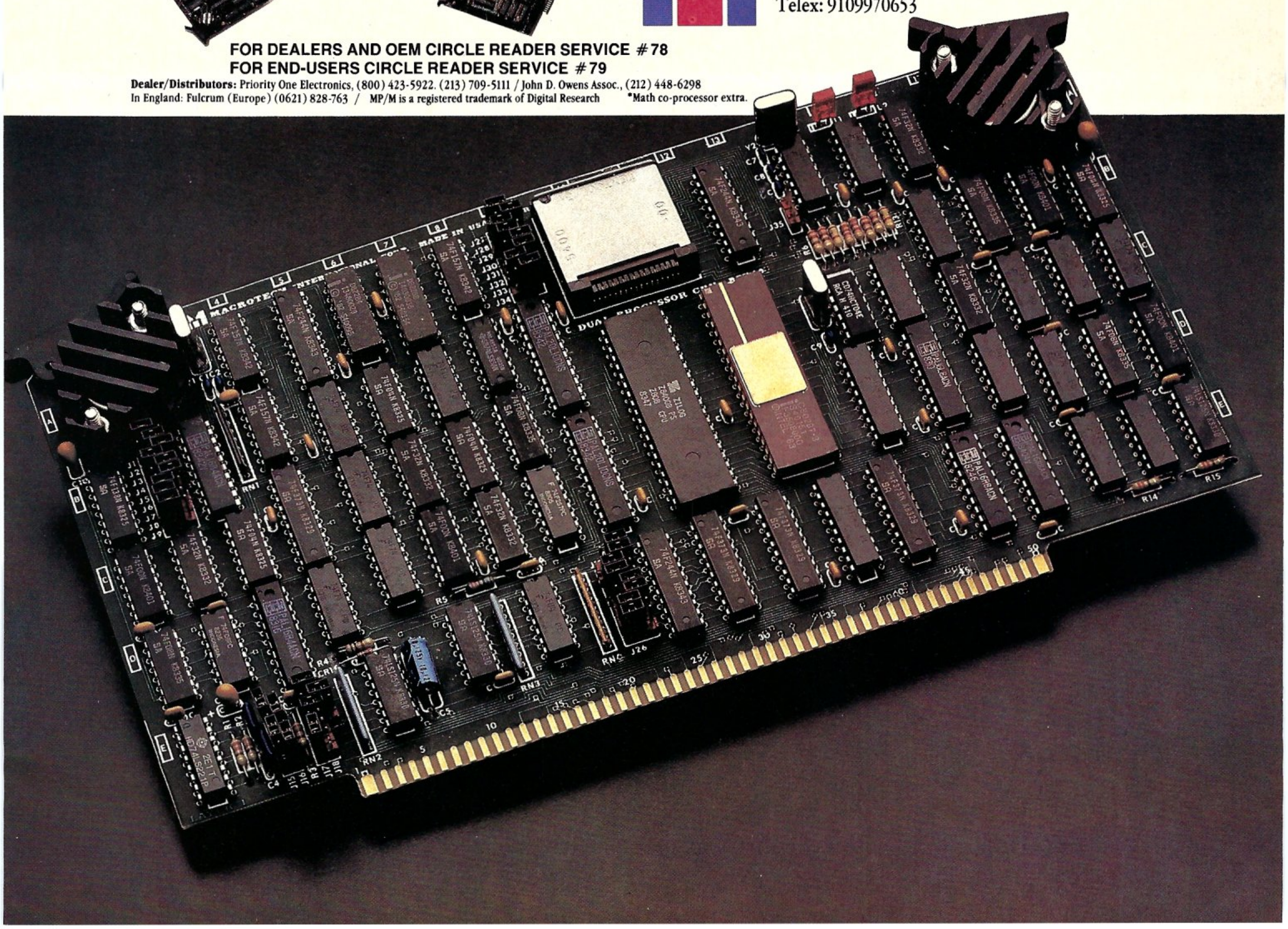
Upgrade your S-100. The MI-286 is only one of Macrotech's products designed to maximize the performance of your S-100 system. The MAX Dynamic Memory gives you up to 1 Mbyte of memory for your system memory and virtual disk applications. ADIT lets you control up to 16 different terminals, modems or printers from a single slot in your S-100 bus. And our static board is the S-100 world's first 1/2 Mbyte static memory. Call or write us today, and find out how easy it is to upgrade your S-100 system.

Macrotech International Corp.
9551 Irondale Avenue
Chatsworth, CA 91311
Phone: (818) 700-1501
Telex: 9109970653

See us at
Comdex, Las Vegas
Booth #316

FOR DEALERS AND OEM CIRCLE READER SERVICE #78
FOR END-USERS CIRCLE READER SERVICE #79

Dealer/Distributors: Priority One Electronics, (800) 423-5922. (213) 709-5111 / John D. Owens Assoc., (212) 448-6298
In England: Fulcrum (Europe) (0621) 828-763 / MP/M is a registered trademark of Digital Research *Math co-processor extra.



News & Views

Continued from page 8

Overall, the PC/AT appears to be a "ho-hum" upgrade of the PC/XT, leaving opportunities for PC-compatible makers to bring out AT compatibles with better performance. For example, consider that the 80286 can directly address up to 16 MB of memory; that most small multiuser systems have more than three users and want tape backup for the hard disk, and that 20 MB of disk space is really not adequate for a three-user XENIX system, etc. However, with all the IBM power behind it, it should do well.

Random news

Yates Ventures of Palo Alto, CA, a leading UNIX market research firm, has been bought out by another market research firm, International Data Corp., of Framingham MA . . . The International Personal Robot Congress and Exposition, to be held in the U.S. in 1986, is planning to have a robot Ping-Pong contest. Robot builders wishing more information on the contest should write to *Dr. John Billingsly, Department of Electrical and Electronic Engineering, Portsmouth Polytechnic, Anglese Road, Portsmouth, England.*

32-bit micro status report

Motorola has begun shipping samples of its 68020 32-bit microprocessors to selected customers and expects shortly to have samples generally available. . . production quantities are not expected until mid '85.

National, who renamed their 32-bit micro the "32000 family" (which is also the name used by AT&T for their 32-bit micros) is reported to be already shipping samples of their 32032 microprocessor to selected customers and may actually beat Motorola into production.

Zilog has begun to release detailed information on their Z80,000 32-bit chip, promising samples late this year. The device will include on-chip virtual memory mangement, cache memory, six-stage pipelining, burst memory transfer and multiprocessing support. Via 32-bit addressing it can access 4 gigabytes in each of four address spaces. The Z80,000 32-bit chip will use a 25 MHz clock, and Zilog claims 3.7 MIPS performance.

Intel has begun supplying detailed information on its 80386 32-bit chip to selected customers and expects to ship samples to those customers by mid '85; production is not expected until 1986. In the meantime Egil Juliesen, of Future Computing, recently made this interesting remark about Intel's new 80286 super 16-bit microprocessor (used in the new IBM PC/AT): "For us, 80286 really means 2-86, which really means that by February 1986 it will be in volume production."

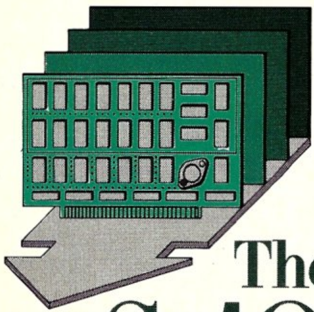
Japan penetrating U.S. PC market

According to Future Computing, a respected market research organization in Richardson TX, the Japanese now have 5% of the U.S. personal computer systems business. However, if one also includes the sales of peripherals (e.g. printers, monitors) sold under their own label, then they now have 22% of the business. Further, if one looks inside the machines at the chips, disk drives, etc., then Japan has 31% of the value of the U.S. personal computer market.

Quotation of the month

"AT&T has never represented the state of the art in UNIX."—*Mark Ursino, Operating System Product Manager, Microsoft*

Sol Libes, Box 1192, Mountainside, NJ 07092



The S-100 Bus

Tips on buying an S-100; safe power switching; slow EPROMs

Continuing with last month's discussion for new S-100 users, I have compiled (with the help of all the new users who have written to me in the last year) a short list of things to consider when purchasing a used (or maybe even a new) S-100 machine.

1. Be sure that the computer is functioning properly when you buy it. You would be surprised how many people don't check out a machine before they take it home! If it doesn't work before you buy it, it won't work after you buy it—unless you are a wizard. Since most S-100 machines run CP/M or a derivative or similar operating system, diagnostic programs for memory and I/O testing are readily available in the public domain libraries from SIG/M and CPMUG. It is usually a good idea to test at least the system's memory, I/O, and mass storage devices. If you find any problems in these three areas, then you should reconsider buying the system unless you are skilled at troubleshooting and repairing S-100 machines.

Problems in the mass storage (usually floppy and hard disks) are often very expensive to remedy. You can count on spending at least $100 to repair a bad floppy drive, and $500 or more to repair a bad hard disk. Most floppy drives can be repaired in the field or at a local repair shop. So can most of the older cartridge drives, although they are usually *very* expensive to repair. Winchester-style hard disk drives are usually cheaper to repair than the cartridge type, but can be repaired only at the factory.

2. Be sure that the machine you are buying will do what you want it to do. Check that the frame has enough slots for your future as well as your current needs. Front panels look nice, but rarely work satisfatorily with IEEE-696 boards, and usually require replacement or modification. Check that the boards you intend to add to the machine will work—if possible, plug them in *before* you buy the machine.

3. Check that any peripherals you already have (especially terminals and printers) will be able to work with the new machine. Many of the older S-100 machines are set up for serial list devices and have no provision for parallel printers (which are today the most popular type).

A classic problem that occurs when installing any machine that requires an external terminal is incompatibility of the console I/O devices. The problem appears to be simple, but can assume nightmarish proportions. One reader bought a machine that booted from floppy disk and assumed a console speed of 19,200 bps; his terminal, however, operated at a maximum data rate of 9,600 bps.

When he bought the machine it seemed a simple task to change the console I/O rate, but when he got the machine home he discovered that the console speed control routine was contained in EPROM; to change the console speed to suit his slower terminal would have required burning a new EPROM. Finally, he gave up and just bought a new terminal capable of supporting the higher speed.

4. Beware of undocumented hardware. Much of the older hardware available for S-100 machines was manufactured by companies that are no longer in business. At the very least, you should get schematics for every board and for the machine itself. Many of these (especially for MITS, Processor Technology, and TDL/Xitan boards) have been collected and republished in book form. However, some boards, such as plug-in modems, EPROM programmers, and terminal emulators, use parts that are no longer available.

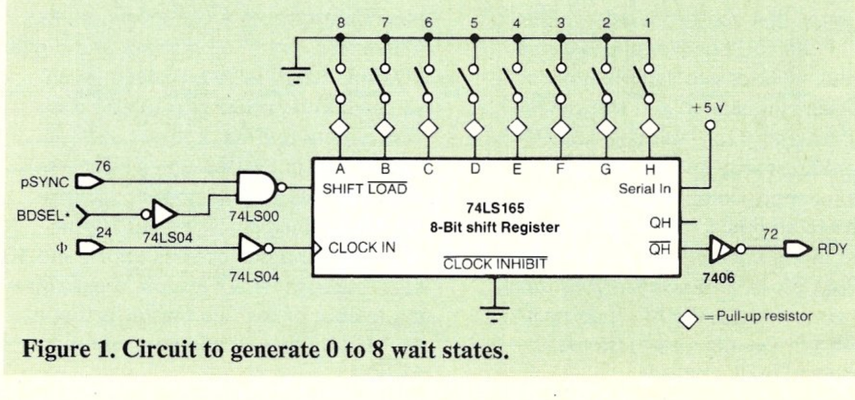


Figure 1. Circuit to generate 0 to 8 wait states.

by Dave Hardy

Thus, if they fail, they become useless relics. In addition, you may have to modify them in order to make them work in an IEEE-696 environment; without adequate documentation, this becomes an extremely difficult task, and you would probably be better off replacing the old board with a new, IEEE-696-compatible one.

5. If you plan to upgrade the machine in the future, make sure that it will be able to support the new hardware. If the machine has an IEEE-696-compatible motherboard, then you should have little trouble upgrading to 16- or 32-bit operation (assuming that all its boards are also IEEE-696 compatible). If the motherboard does not conform to IEEE-696 in all respects, then you may have to invest a lot of money before you can upgrade the system.

You should also check the machine's power supply ratings to make sure that it will be able to handle more boards and additional drives. Many of the newer S-100 machines are designed for a specific application and may not be able to provide enough power for items that the manufacturer does not include. Don't assume, because the machine has 10 slots, that you can fill all of them with any kind of board; old static RAM boards are power hogs, and a 16K board may well draw twice as much current as a modern 64K board.

Slow EPROMS again

A problem that appears to be plaguing many owners who are updating to high-speed processors is how to make their slow EPROMs work at 6, 8, or even 10 MHz. Although somewhat faster versions of most of the popular EPROMs are available, 8 MHz or 10 MHz EPROMs are not likely to be available for quite some time. The solution, as I have several times previously pointed out in this column, is to use a wait-state generator.

Most EPROM boards have built-in generators for one or two wait states, but some have no wait-state capability at all. At clock speeds of 6 MHz and above, even two wait states are not enough for many of the available EPROMS.

The circuit shown in Figure 1 (from *Interfacing to S-100/IEEE-696 Microcomputers*, by Sol Libes and Mark Garetz, Osborne/McGraw-Hill, 1980) is a simple wait-state generator that can add up to eight wait states to any S-100 RAM or EPROM board. Being basically *very* lazy, I have used this circuit dozens of times to avoid having to replace existing boards when "souping up" an S-100 computer.

The signal BDSEL* is taken from the board's existing I/O decoder and is used, along with the system clock and pSYNC signals to force the RDY line low for from 0 to 8 bus cycles, by clocking the parallel word loaded from the switches onto the RDY line.

Safe power line switching

Many readers have asked for a simple circuit that can be used to allow their computer to switch an AC power line via a parallel I/O line. There are many ways to build a switching device of this sort, but the simplest is to use a commercially available solid-state relay. There are several advantages to this approach. It is inexpensive (about $10), easy to install (four wires), provides a high degree of protection for your computer, and can handle relatively large AC loads (20A or more, depending on the relay you choose).

Typically, the solid-state relay has four terminals: the input terminals (usually marked '+' and '-') are connected to your computer's parallel output line and signal ground, respectively. The load is generally no more than 15 milliamperes— well within the drive capability of a standard 74xx chip. The AC switch terminals (generally marked '-') are connected in series with the AC load in exactly the same manner that a light switch is wired in a table lamp.

A word of caution

Since these relays use optical coupling between the switching circuit and the load circuit, there is no direct electrical path between your sensitive TTL computer ICs and the potentially *very* harmful 120VAC power line. A word of warning, however. Although the optocoupler in the relay completely isolates your computer from the AC power line, the 120VAC present on the AC terminals of the relay is **lethal.**

Mount the relay *outside* the computer case. Make sure that all 120VAC connections are adequately insulated with friction tape or a non-conducting cover. Check the manufacturer's specifications for maximum load, and make sure that the controlled AC device load does not exceed the relay ratings. You may also have to provide a heat sink for the relay, depending on which one you choose.

If you neglect these precautions you may kill more than your computer!

Two relays I have used for medium-power applications (less than 10A) are the Monsanto MSR100 and the Crydom (International Rectifier) D1202-S. Each of these devices allows a 3-to-32VDC input to switch up to 1.2kW of AC power.

Dave Hardy, 736 Notre Dame, Grosse Pointe, MI 48203.

Available for the IBM PC, AT, XT, jr., and true compatibles*

CLEAR THE DESK, SIDEKICK'S HERE!

The Super Organizer

ALWAYS JUST A KEYSTROKE AWAY . . .
No matter what software you're running!
WHETHER YOU'RE RUNNING LOTUS, WORDSTAR, dBASE OR WHATEVER . . .
JUST A KEYSTROKE AND A SIDEKICK WINDOW OPENS . . .

- A CALCULATOR
- A NOTEPAD
- AN APPOINTMENT CALENDAR
- AN AUTO DIALER
- A PHONE DIRECTORY
- AN ASCII TABLE

"SIDEKICK IS A $50 SOLUTION TO A $5,000 PROBLEM."
Garry Ray, **PC WEEK**, *July 24, 1984*

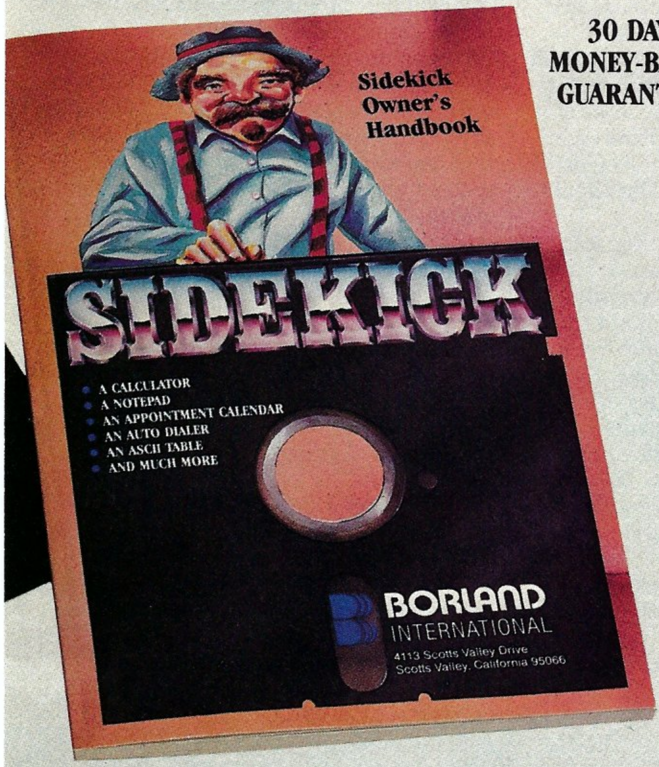
INTRODUCTORY OFFER

$49.95

Copy Protected Version

Here's Sidekick in action. That's Lotus 1-2-3 running underneath. In the Sidekick Notepad you can see data that's been imported from the Lotus screen. On the upper right, that's the Sidekick Calculator.

30 DAY MONEY-BACK GUARANTEE



NOW . . . Whether you're working in dBase, Lotus, Wordstar or whatever . . . you can unleash the full power of your computer . . . and make a lot of extra space on your desk at the same time.

Whenever you're using your computer . . . from start to finish of your session . . . Sidekick™ will be there . . . ready to serve. And it's as lightning-fast and compact as only Borland knows how to make it.

There's a notepad that has a full-screen editor that can time and date stamp your notes, and then save them to disk. You can even pull information into the notepad directly from the screen of your "underlying" software.

Suppose you're working in Lotus and the phone suddenly rings. Give your Sidekick a call and it pops right up over Lotus with the notepad you need. Or an appointment calendar . . . one you can never misplace.

What if you need to do a quick calculation? A keystroke instantly brings up the calculator. And the results of your calculations can even be transferred to your "underlying" software.

Need to make a phone call? Up pops your personal phone directory. Type in the name you want . . . and Sidekick jumps right to the phone number. Another keystroke, and the phone is automatically dialed for you.†

There's lots more, too. You can move the Sidekick windows anywhere on the screen you like. And you can have as many on screen at a time as you need. There's even an on-line help window for each of Sidekick's features.

We designed it because we needed it. If you've ever been writing a report and needed to do a quick calculation, or jot down a note, then you need Sidekick, too.

†Only with Hayes Smartmodem and compatibles.

GOT *YOUR* SIDEKICK YET?

YOU CAN ORDER YOUR COPY OF SIDEKICK™ TODAY!

For VISA and MasterCard orders call Toll Free **1-800-255-8008** in California **1-800-742-1133**
(lines open 24 hours, 7 days a week) Dealer Distributor Inquiries Welcome 408-438-8400

SIDEKICK™ $49.95 ☐
Non-copy protected Version: $79.95 ☐
(Plus $5.00 shipping and handling.)

Check ☐ Money Order ☐
VISA ☐ MasterCard ☐
Card # ______________

Expiration Date ______________

**PC jr. runs non-copy protected version only.*

Please be sure your computer is an IBM PC, AT, XT, jr., or true compatible!

NAME ______________

ADDRESS ______________

CITY/STATE/ZIP ______________

TELEPHONE ______________

California residents add 6% sales tax. Outside U.S.A. add $15.00. (If outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars.) Sorry, no C.O.D. or Purchase Orders.

SG9

THE SOFTWARE CHAMPIONS

BORLAND INTERNATIONAL

Borland International
4113 Scotts Valley Drive
Scotts Valley, California 95066
TELEX: 172373

The UNIX File

Shows, shell efficiency, and shibboleths

by Ian F. Darwin

This issue of The UNIX File runs over several small but useful topics. I look at upcoming conferences, offer hints on **make** and on shell efficiency, and attack a common myth about UNIX.

Long-range planning

Get out your new 1985 calendar and mark these dates—the really big shows of the UNIX world in 1985. To start the year off right, Dallas, Texas will be the site of a USENIX and */usr/group* "disjoint" conference. Both organisations are holding their conferences at the same time in the same city, but under different roofs. Details such as cross-registration are not yet (end of August 1984) announced. This should be a very large event.

Ironically, the fact that the techies are in town will draw the marketing types to the UniForum show, and the fact that the marketing types are "at a distance" will draw the techies to the USENIX conference. The dates for this are January 22-25, 1985. UniForum, which bills itself as "the premier UNIX marketing event," will be the opening conference at the brand-new Dallas Infomart. USENIX, the largest UNIX technical conference, will be at the Fairmont Hotel in Dallas.

According to Steve Glaser, who is General Chair of the USENIX committee for the summer 1985 conference, *summer's* USENIX get-together is scheduled for Portland, Oregon, June 12-14, 1985, with a day of tutorials and other meetings on the 11th. USENIX and the Software Tools User Group are organising this conference. Steve Bourne—best known as the author of the Bourne shell, now working at Silicon Graphics—is the chairman of the technical program, so that is in good hands. There will be no major */usr /group* participation, although they will probably have a booth in the Exhibitors' area along with vendors and others displaying products and training.

Steve Glaser said in a recent (end of July, 1984) USENET posting: "We're still trying to settle on most of the details. We wanted to wait to see how some of the new stuff that they tried in Salt Lake City worked out. If you were there, did you turn in the survey form from your registration packet?" Steve tells me that the program has settled down a little since that posting; expect the call for papers and more details on the Summer 1985 USENIX to appear around December, 1984.

For conference registration or vendor space rental information, contact *USENIX* at *Box 7, El Cerrito CA 94530*; phone *415-528-UNIX (528-8649)* and/or */usr/group, 4655 Old Ironsides, Santa Clara CA 95050*; phone *408-986-8840.*

More on *make*

Here's another hint on the **make** utility. You often want to maintain the originals of 'shell files' in a source directory far away from the executable directory (directory of executable program, not directory which is 'executable' in terms of the UNIX permission scheme) in which the "production" versions live. It's common to call these files "commandname.sh" (for example, the xyz command is stored as xyz.sh). Here is a makefile which copies xyz.sh and its subordinates into the executable directory bin/xyz. It will also copy one or more of several members of a library into the production copy of this library without renaming them. Both examples use the magic **make** variable '$?', which refers to just the name or names of the dependencies that are out of date. Both rules use the UNIX **touch** program to create an (empty) file in the current directory solely to keep track of file modification vs. installation dates. A look at the example (Listing 1) should clarify this.

For example, if xyz.sh has been modified (i.e., its modification time is later than that of the file **install**), but xyzpart2.sh has not, then the command **make install** will result in xyz.sh being copied to the bin directory as xyz; xyzpart2.sh will not be copied.

The 'for' loop is continued over several lines in the makefile. Alternately, it could be entered as one long line. The semicolons are delimiters to the shell, the backslashes to **make** itself. If you type the whole for loop on one line, keep the semicolons but omit the backslashes. If you tried to put the for loop on a line by itself, without the backslash, then **make** would feed that line off to a shell, which would reject it as an incomplete command. Then it would feed the 'do' command to a shell, which would reject it as incomplete, and so on. Always continue long shell constructs carefully in a makefile. Note also that where you want the shell to get a '$' character, you must enter two of these into the makefile, since the sign of the dollar is a delimiter to both **make** and the shell.

The **make** program has a number of '$' variables ($@, $?, $*). This is just one example of what you can do with them.

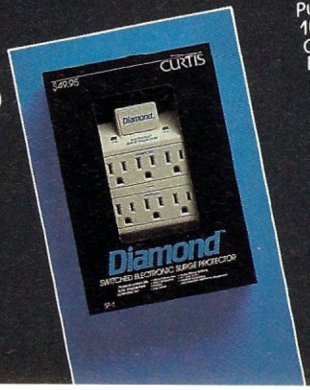
ELECTRONIC SURGE WITH THAT JEWELERS

Curtis electronic surge protectors cost less and offer a lifetime performance warranty while protecting your valuable computer hardware and software from dangerous, potentially damaging powerline overvoltage transients – "surges, spikes, and glitches" – that can cause instant failure or drastically reduce system life. With our specially designed EMI-RFI filters, your system is protected from troublesome office equipment, general hash created by small motors, fluorescent lamps, and the like.

DIAMOND SP-1

Switched; adapts duplex outlet; six outlets. • 15 Amps, 125 Volts, 1875 Watts • Operating Temperature: −40°C to +85°C • Peak Pulse Power Dissipation @ 25°C, 585,000W for 100 Sec. • Maximum Clamping Voltage @ Test Current 50A (8/20 sec.) 340V Clamping Occurs Line to Line (normal mode) and Line to Ground (common mode) • Clamping Response Time: 1 Nanosecond (1×10/−9 sec.) • Dielectric Test: 3000VAC 60 Sec. • Forward Surge Rating @ 25°C, 1440 Amps for 1/120 Sec.

$49.95



RUBY SPF-2

Switched; EMI – RFI filters; plugs into duplex outlet; 6 ft. power cord with built in circuit breaker/switch; six outlets. • 15 Amps Total, 125 Volts, 1875 Watts • Operating Temperature: −40°C to +85°C • Peak Pulse Power Dissipation @ 25°C, 585,000W for 100 Sec. • Maximum Clamping Voltage @ Test Current 50A (8/20 sec.) 340V Clamping Occurs Line to Line (normal mode) and Line to Ground (common mode) • Clamping Response Time: 1 Nanosecond (1×10/−9) • Dielectric Test: 3000VAC 60 Sec. • Forward Surge Rating @ 25°C, 1440 Amps for 1/120 Sec. • Mode Noise Protection: Normal and Common Mode • Noise Rejection Frequency: 150 KHZ - 30MHZ • Attenuation: 5DB - 58DB

$89.95

UL/CSA pending

Curtis Manufacturing Co. Inc.
305 Union Street
Peterborough, N.H. 03458

Manufacturing Company, Inc.

CURTIS

PROTECTORS TOUCH

Each of these precious jewels, will easily blend with the colors of any modern office or home surroundings. For safe, trouble-free performance, give your computer that jewelers touch.

Available at all ComputerLands, Sears Business Systems Centers, Entre Computer Centers, Businessland, Inacomp, Micro Age, Eczel, or your local microcomputer dealer. Over 1000 locations nationally! In Canada: Micro-Computer Products, P.O. Box 235, Ajax, ONT, Canada L1S 3C3, (416) 427-6612.

For the Curtis dealer nearest you call (603) 924-7803

CIRCLE 28 ON READER SERVICE CARD

EMERALD SP-2

Switched; plugs into duplex outlet; 6 ft. power cord with built in circuit breaker/switch; six outlets. • 15 Amps, 125 Volts, 1875 Watts • Operating Temperature: $-40°C$ to $+85°C$ • Peak Pulse Power Dissipation @ 25°C, 585,000W for 100 Sec. • Maximum Clamping Voltage @ Test Current 50A (8/20 sec.) 340V Clamping Occurs Line to Line (normal mode) and Line to Ground (common mode) • Clamping Response Time: 1 Nanosecond ($1 \times 10/-9$ sec.) • Dielectric Test: 3000VAC 60 Sec. • Forward Surge Rating @ 25°C, 1440 Amps For 1/120 Sec.

$59.95

SAPPHIRE SPF-1

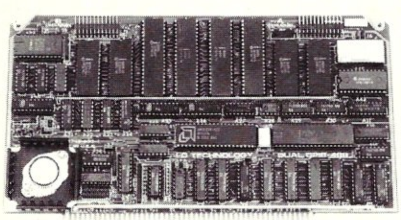
Switched; EMI-RFI filtered; adapts duplex outlet; three outlets. • 15 Amps Total, 125 Volts, 1875 Watts • Operating Temperature: $-40°C$ to $+85°C$ • Peak Pulse Power Dissipation @ 25°C, 585,000W to 100 Sec. • Maximum Clamping Voltage @ Test Current 50A (8/20 Sec.) 340V Clamping Occurs Line to Line (normal mode) and Line to Ground (common mode) • Clamping Response Time: 1 Nanosecond ($1 \times 10/-9$) • Dielectric Test: 3000VAC 60 Sec. • Forward Surge Rating @ 25°C, 1440 Amps for 1/120 Sec. • Mode Noise Protection: Normal and Common Mode - Noise Rejection Frequency: 150 KHZ – 30MHZ • Attenuation: 5DB – 58DB

$79.95

INPUT/OUTPUT TECHNOLOGY, INC.

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

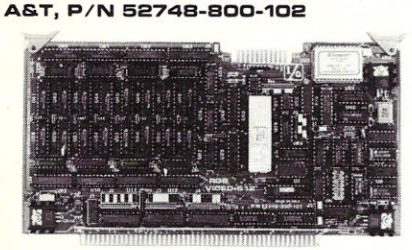
Uncompromising Additions to your S-100/IEEE-696 BUS



DUAL GPIB-488 INTERFACE BOARD

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

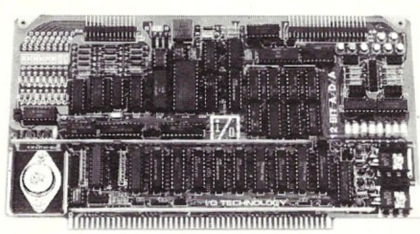
A&T, P/N 52748-800-102



RGB COLOR GRAPHICS BOARD

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

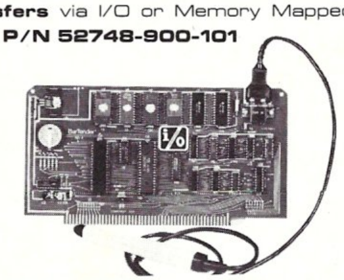
A&T, P/N 52748-300-101



12-BIT A-D-A CONVERTER BOARD

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

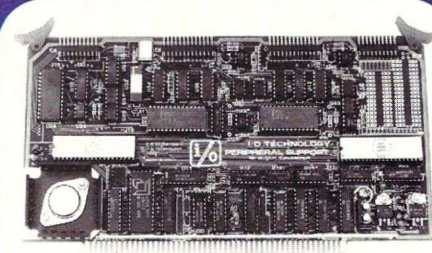
A&T, P/N 52748-900-101



BAR CODE PROCESSOR BOARD

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

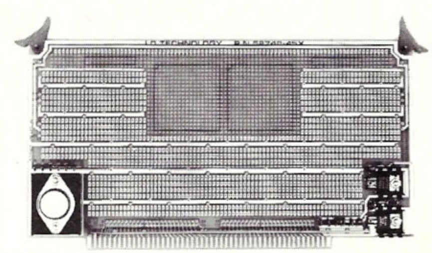
A&T,52748-500-101 Without Wand
A&T,52748-500-201 With Wand



PERIPHERAL SUPPORT BOARD

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

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



MULTI-PURPOSE PROTOTYPING KIT

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

P/N 52748-450

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

CIRCLE 10 ON READER SERVICE CARD

SPECIFICATIONS SUBJECT TO CHANGE WITHOUT NOTICE

UNIX File

Continued from page 16

Efficiency tip—shell "exec"

Here's an easy-to-remedy minor inefficiency. Fixing it can reduce (by one) the number of processes active in your system, which in turn may reduce the amount of swapping going on. The inefficiency is caused by the way shell files operate.

The shell, however, is not part of the operating system 'kernel,' so it must create new programs just like any other program does. UNIX programs that want to create new lifeforms must use the **fork()** system call, which creates an *exact* copy of the program that issued it. This newly created "child" process in most cases will immediately suicide by **exec()**-ing another program. The fork( ) system call creates a second process identical with the issuer; the exec( ) system call causes the issuing process to start running a different program.

Now it often happens that the parent process waits around for the child process to live out its lifespan and die of illness or old age (be terminated by an error condition or signal, or terminate itself when done). For example, every time you type a command to an interactive shell, the shell forks, and the parent shell sleeps while the child process **exec** execs the program you asked for, which eventually either exits or bombs out. When this happens, the parent is notified by the responsible authorities (in this case the UNIX kernel) and goes about its business without remorse. Usually it will read the next command from the terminal, and repeat this whole process.

When the shell's input is from a shell file (or command file), the situation is similar. It reads a command, forks, then the child shell execs the requested program and the parent shell waits for completion. Many shell files, however, have only a few commands in them. If the last command is a long-running one (such as an interactive edit), then the parent shell (the one reading commands from the shell file) will needlessly wait around for the edit to finish, taking up swap space and other system resources.

Of course, there's an easy way to turn this to your advantage, now that you know how it works. The shell provides a built-in **exec** command, which causes it to execute the named program *without* first doing a fork( ). The result is that the shell reading the shell file self-destructs (harmlessly), and is replaced by the called program (in this example, the editor). Result? "One less shell to wait for, one less process slot; One less group of files stay open . . ." Apologies to pop singer Marilyn McCoo, but you get the idea. A bit of efficiency gained.

Here's an example (Listing 2) using a simple shell script, **phoneup**, which updates my private phonebook file. The phonebook file lives in my home directory, and I don't like to have to type the full pathname (or even $HOME. . .) when I'm doing something three levels of directory away and I found a new or changed phone number.

So, use of the shell **exec** can save having extra shells around. Remember that you can only **exec** the last command in a shell file, since after the exec the shell that is reading the shell file is gone! Try it and see. Use the **exec** judiciously, and you should find your system doing a little less swapping and a little more processing.

Listing 1

```
BIN=/u/ian/bin
LIB=/usr/lib

install: xyz.sh xyzpart2.sh xyzpart3.sh
        for i in $?; \
        do\
                F=`basename $$i .sh`; \
                cp $$i $(BIN)/$$F; \
        done
touch install

lib:    xyz.jcl abc.jcl
        cp $? $(LIB)
touch lib
```

Listing 2

```
% cat bin/phoneup
#! /bin/sh

# update one's private phonebook

${EDITOR-/bin/ed} $PHONEBOOK
% phoneup                      # try running it
*!ps                           # look at processes
  PID TTY TIME COMMAND
   79 02  0:08 sh              # my login shell
  121 02  0:01 sh              # shell waiting for 'ed'
  122 02  0:01 ed              # the editor
  123 02  0:01 sh              # shell waiting for 'ps'
  124 02  0:02 ps              # 'ps' finds itself
!
*q
% ed bin/phoneup
 75
$s/^/exec /p
exec ${EDITOR-/bin/ed} $PHONEBOOK
*w
 80
*q
% phoneup                      # try it again
*!ps
   79 02  0:08 sh              # login shell still around
  126 02  0:01 ed              # look ma, one less shell
  127 02  0:01 sh              # shell waiting for 'ps'
  128 02  0:02 ps
!
*q
%
```

UNIX File

Continued from page 21

Can you use 'exec' in the '!ps' escape to save yet another shell? It's left as an exercise to the reader—try it and see. If you can, is it worth the extra typing? Hint: you can't use the time(1) command to find out. This is because it would get run by the shell that you are eliminating, so it doesn't 'see' the delay —if there is one—that the shell created by the '!' escape induces by waiting for the 'time ps' to finish.

Old shibboleths die hard

Scratch a dozen UNIX critics and you'll find ten or eleven false myths or misconceptions about UNIX—what it is, what it does, and so on. Here's a look at one of these (the misconceptions, not the critics).

The UNIX system was designed by programmers. Most early UNIX systems were located inside Bell Labs; most of the next batch of (mostly Sixth Edition) UNIX sites were at universities. Both these groups of installations had UNIX gurus around to fix things up when problems developed. One result was that it was considered acceptable for filesystems to get corrupted now and again by system crashes, since there was always expertise on hand to fix things up. So far, so good.

The myth is that UNIX is still at this stage. The reality is that this problem had been fixed by 1979. That's five years ago.

The **fsck** program was a standard part of Seventh Edition (V7) UNIX. Written by T.J. Kowalski based upon earlier **fcheck and related programs, fsck** has been adopted by every serious version of UNIX since V7, including 4.1BSD, 4.2BSD, System III, System V and V8 (an internal research version not available outside Bell Labs). This program is normally run for you automatically at startup time. If you have a small UNIX system, you probably use **fsck** and don't even know it. **fsck** produces those '*** Phase 1 . . .' messages and the DUP/BAD, Remove/Reconnect messages (which are suppressed in some systems) and the one-line summary about each filesystem's size, number of files and free space. The job of **fsck** is to seek out and eradicate any inconsistencies in the file system. Normally such errors will not occur in a filesystem, unless UNIX itself was halted in the process of updating the disk system, i.e., if the system crashed when files were being written to disk.

fsck has turned out to be so reliable that most people now take UNIX filesystem correctness for granted. Only rarely are files lost, unless the hardware is bad. The UNIX Administrator's manual uses the example of a disk which writes when it should be reading.

And in fact a malfunctioning disk is the only cause of lost files on my system, and that was while editing this month's column! A dozen power failures over the last half year or so have not resulted in any lost files. At the University, we occasionally lose files after a serious crash on timesharing systems with a dozen or dozens of people on.

Files can also be lost if changes to the kernel or a new device driver are being tested. And as everyone knows who's used computers for any length of time, you should always have a backup anyway, for the occasional time when files do get lost. User errors are far more common as a cause of lost files than filesystem damage.

In addition, the UNIX protection mechanisms prevent amok programs from scribbling on directories, unlike some widely used single-user systems and their multiuser adaptations, in which an errant BIOS call can eat a directory. Look for operating systems that have many 'directory fix' or 'directory dump' or 'crashed disk recover' programs available both commercially and in the public domain. UNIX does include an **fsdb** program, but I've never learned it since I've never needed it.

Scratch a UNIX critic and you'll find a UNIX myth.

The conclusion? UNIX has a filesystem reliability about as good as most other operating systems, and superior to several common ones. The 'UNIX file system always falls apart' myth belongs in the Version 6 days of a decade ago; let's leave it there and get on with UNIX today.

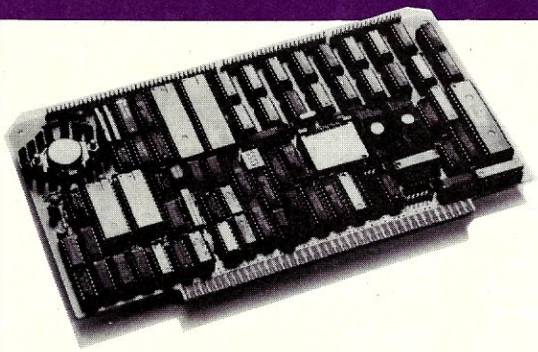
Please feel free to write in with questions or comments. Addresses for regular mail and electronic mail are given in the box. I'm always glad to hear from readers with comments either on the column itself or on their reactions to particular UNIX systems or products.

Ian Darwin, Box 603, Station F, Toronto, ON Canada M4Y 2L8.

uucp *network address:* **ihnp4!darwin!ian.**

LOMAS DATA PRODUCTS PRESENTS:

THUNDER 186



THUNDER 186 offers the following standard features:

- 8 Mhz 80186 processor
- Floppy disk controller (controls both 5¼" and 8" drives simultaneously)
- Two RS232 ports with hand shaking
- Centronics compatible parallel port
- 256K bytes of RAM
- Full IEEE-696 COMPLIANCE
- One year warranty

THUNDER 186 provides the earth shanking performance you have come to expect from LOMAS DATA PRODUCTS. THUNDER 186 is a complete SIXTEEN BIT SINGLE BOARD COMPUTER on a standard size S100 bus board. THUNDER's low price includes Concurrent CP/M-86* version 3.1. This allows the user to run PC-DOS**** programs, as well as, CP/M-86* programs on the same system. And, for those users that prefer, MS-DOS*** is also available.

TEST: THUNDER 186 BENCHMARK.

In our April '84 advertisement, we compared the benchmarks of our other advanced CPU products, against the COMPUPRO CPU86/87 10 Mhz. The test consisted of assembling the example BIOS (BIOS A86) distributed with standard CP/M-86. The operating system used was CP/M-86. Now, we put THUNDER 186 to the same test.

	LIGHTNING ONE™ 10 MHZ	COMPUPRO CPU86/87 10 MHZ	LIGHTNING 286™ 6 MHZ	THUNDER 186 8 MHZ
ASSEMBLY-TIME (SEC.)	47	55	41	48
RATIO TO 10 MHZ LIGHTNING ONE	1.00	1.17	.88	1.02
BOARD SET COST LIST PRICE AS TESTED	$2545.00	$3040.00	$3290.00	$1595.00
COST PERFORMANCE	$2545.00	$3556.80	$2895.20	$1629.00
RATIO	1.00	1.40	1.14	.64

RESULT: THUNDER 186 THE LEADER.

The benchmark comparisons clearly make THUNDER 186 the leader in the 16 bit price/performance race. Other THUNDER 186 advantages are too numerous to mention here. CALL US and we'll be glad to discuss the other advantages with you.

S100 bus board products & support for the system integrator ...

■ LIGHTNING ONE * 8086/8088 CPU**
8086 or 8088, with 8087 and 8089 coprocessors. Up to 10 MHZ operation .... **PRICES start at $425.00**

■ HAZITALL SYSTEM SUPPORT BOARD
2 serial, 2 parallel ports, battery protected clock calendar, Hard disk controller host interface .... **PRICE $325.00**

■ LDP 128/256K DYNAMIC RAM
Advanced dynamic RAM with LSI controller for failsafe operation, parity ...... **PRICE** 128K—**$495.00,** 256K—**$795.00**

■ RAM67 HIGH PERFORMANCE STATIC RAM
High speed (100ns) low power CMOS static RAM. 128K bytes, extended addressing .... **PRICE $995.00**

■ LDP72 FLOPPY DISK CONTROLLER
Single/double density, single/double sided disks, both 8" and 5 1/4" inch drives simultaneously .... **PRICE $275.00**

■ LIGHTNING 286—80286 CPU BOARD
Offers 4 times the performance of a 5MHZ 8086 CPU while maintaining software compatibility. ...... **PRICE $1395.00**

■ OCTAPORT 8 PORT SERIAL BOARD
8 serial ports 0 to 19200 baud operation real time clock interrupt. Ideal for multi-user systems such as MP/M-86.* . . .

All of LDP boards are fully tested to exacting standards and carry a one year warranty. We specialize in 16-bit products & support the four major operating systems for 16-bit processors: CP/M-86*, MP/M-86, CONCURRENT CP/M-86*, and MS-DOS (PC-DOS).

Dealer inquiries invited.

LDP

LOMAS DATA PRODUCTS, INC. □ 66 Hopkinton Road, Westboro, MA 01581 □ (617) 366-6434 □ Telex: 4996272

**CP/M-86, MP/M-86 and CONCURRENT CP/M-86 are trademarks of Digital Research.*
***MS-DOS is a trademark of Microsoft*
****Lightning One is a trademark of Lomas Data Products, Inc.*
*****PC-DOS is a tradmark of IBM*

CIRCLE 216 ON READER SERVICE CARD

The MS-DOS Window

Bulletin boards for the PC

by Hank Kee

A very popular application of the personal computer today is bulletin board access. The use of the PC to gather in data from remote databases and public domain programs, as well as to chat with fellow hobbyists, has now grown very rapidly. Very often these bulletin boards are used for posting technical queries to anyone who may have the right answer.

You can meet many interesting people this way. The electronic bulletin board has developed into a new means of making social contacts. There is now a multitude of bulletin board systems (BBS) operated by individuals and local computer clubs across the country.

The Bulletin Board System

The electronic bulletin board was originally developed as a way of posting general information of interest for anyone to read. They were quickly enhanced with a host of new functions and communications protocols. The primary functions of a PC bulletin board are to allow remote users to:

a) read and write both broadcast and private messages
b) download public domain software
c) upload new public domain software
d) upload messages for the system operators (SYSOPs)

Concurrent with the growth of electronic bulletin boards was the development of public domain program libraries. The most common method of distributing public domain software continues to be the volume diskette, obtainable at computer club meetings or through the mail. However, for those who can't wait and need only a few individual files from different volumes, the electronic bulletin board has proven to be a favorite means to transport these files.

A number of commercial bulletin boards serve public as well as private interests. CompuServe and The Source are those which come most readily to mind. But these are run on a subscription basis—meaning that the revenue clock ticks while you are on the system. They are multiuser systems that allow many users, from all over the world, to be logged on concurrently.

Sophisticated bulletin board systems today include password control, timeouts, and file attribute controls. These features have had to be developed to protect the systems against the very few who find it a technical challenge to willfully crash any and all bulletin systems, or think it funny to leave offensive messages. This type of behavior has slowed the growth of personal computing communications.

The wide availability of the IBM PC has made it possible to install a bulletin board system without the arduous task of customizing the I/O drivers that is required on CP/M systems and on low-end systems such as Atari or Commodore. On the IBM PC, the implementation of a bulletin board for local club activity is almost as simple as loading a program. Today, the functions of a bulletin board system are quite diverse. Most are single-user systems, serving primarily a local area.

RBBS-PC

There are many bulletin boards now in operation using the IBM PC. Most are using a package called (R)emote (B)ulletin (B)oard (S)ystem-PC (RBBS-PC). It is a single-user system compatible with the XMODEM protocol. RBBS-PC was developed by, at last count, 19 members of the Capitol PC Club in Washington, DC. This particular package is quite popular with various IBM PC user groups and clubs, and in a very short period of time it has grown enormously in functionality and sophistication.

The package is very well documented, and source code written in IBM Basic has been provided. A minimum of 128K is required. It is also possible to run this system with a single diskette drive, though that is not very practical. All the desirable features of a bulletin board have been included. It operates at the program level—the user does not have access to the command level.

The communications protocol for uploading and downloading binary files is XMODEM/MODEM7. A multiuser version called Multi-link is also available from the Capitol PC Group.

Implementation of RBBS-PC is straightforward. Since source code is available in BASICA, individual clubs can modify it to meet their specific requirements. The Capitol PC Group asks only that its initial sign-on and original code be left intact for distribution. RBBS-PC is treated as "freeware"—meaning that it is in the public domain, but a contribution or donation would be appreciated if the software is found useful. This means of distribution is quite persuasive, in this case—few individuals or clubs are hard-hearted enough to refuse a donation for such a good package that includes source code!

I recently implemented RBBS-PC for use by the local computer club to

We thought about calling it MacSimplex . . . after all it makes your IBM®PC behave like a Macintosh™ and much more . . .

and with over two years in the making, the Simplex Database Management System has features like 32-megabyte virtual memory and the most powerful networked/relational database in the microcomputer industry. Simplex was designed around how you think and the Macintosh way, so that you can use your favorite mouse to handle those mundane tasks like menu selection and data manipulation. And, if you don't have a mouse, you can use our keyboard mouse simulator, MouSim™.

Pop-up and pull-down menus, dialog and alert boxes are not just added features, they are the heart of the Simplex way. In addition, Simplex gives you both a software and a hardware floating point capability, each with 19-digit accuracy. It permits login, password, privilege, and can be used on a local area network. Simplex has full communications and a remote or local printer spooler. Above all, Simplex is modular and grows with you! Simplex also has a full-featured, English-like language which is simple to use.

You can't buy Simplex™, but it is now available as an integral part of *it's my* ***Business***™ and will be used by *it's my* ***Word***™, *it's my* ***Graphics***™, . . .

Businessmen! *it's my* ***Business*** will revolutionize the way that you handle your business. It saves time, money, and standardizes your system for all who use it. *it's my* ***Business*** comes with applications like accounting, interoffice or intraoffice mail, editing, invoicing, inventory managment, mail list, calendar, scheduler, forms and more. You can modify each of these to create applications specifically designed for you... maybe we should have called it "it's your Business".

Professionals! *it's my* ***Business*** has over 200 pages of examples and demonstrations to show you how to solve your everyday professional problems. And if these examples aren't enough, we give you a complimentary one-year subscription to Questalk™, our hands-on Simplex applications magazine.

System integrators and consultants, beware! If you are not using *it's my* ***Business*** with Simplex to solve your problems, don't be surprised when more novice programmers solve that complex math, industrial engineering, or business problem faster. We think that you can cut your concept-to-development time by an order of magnitude!

it's my ***Business*** (includes *it's my* ***Editor***) - $695.00
it's my ***Business*** Demo Disk - $20.00
it's my ***Editor*** - $100.00.

Quest Research software is available through your local computer store or through mail order from Quest Software Corporation at (205) 539-8086. 303 Williams Avenue, Huntsville, AL 35801.

Value added resellers and dealers please contact Quest Research, Incorporated at (800) 558-8088. 303 Williams Avenue, Huntsville, AL. 35801.

Quest™

Quest Research Inc.

CIRCLE 21 ON READER SERVICE CARD

MS-DOS Window

Continued from page 24

provide access to public domain software as well as a message center for club members. This particular version can display up to six bulletin messages and store up to 250 messages, which may be either global or private.

RBBS-PC provides both novice and expert modes. The novice user has available the total menu for ease of use. I have found that its usage was straightforward and no "training" or tutorial was really needed for the first-time user of the system. For the more experienced users, the command display is abbreviated to speed up operation.

The functionality of RBBS-PC is quite rich and the documentation supplied is comprehensive. In the past, many other bulletin board systems have been implemented for various CP/M and S-100 systems—and all required extensive customization for each sysop's particular hardware configuration. The RBBS-PC package is unique in that no customizing is necessary to make it operational.

The use of the IBM PC for bulletin board service goes a long way toward standardization. There is no need to program for foreign component boards in the system. RBBS-PC will no doubt become the de facto bulletin board system standard, just as XMODEM/MODEM7 has become one of the de facto protocol standards. Its success is attributed to unselfish sharing by the authors.

With the source available, it becomes very easy to customize RBBS-PC for individual usage, such as number of messages allowed, security measures, and so on. The comprehensive documentation includes the necessary instructions to configure the system by parameter specification. There are at least 70 different levels of options to tailor RBBS-PC, and there are default values within the CONFIG program for those who don't want to be bothered. This version of RBBS-PC works well with fixed disk as well as RAM disk.

The interpretive version works quite adequately, but is not recommended for normal operation; however, the coding is compatible with the IBM BASIC compiler,and the compiled version is very, very fast. The only drawback to the compiled version is its inability to list files, whereas the interpretive BASICA has the FILES command to list the directory. On the other hand, the compiled version has much more memory available for larger communication buffers, and a directory listing function can always be added by the user.

The recommended minimum PC (or clone equivalent) systems hardware and software configuration for RBBS-PC CPC12 is as follows:

- IBM PC or IBM PC/XT with 128K RAM and two double-sided disk drives
- 80-column monitor
- Asynchronous communications adapter (serial port)
- Hayes Smartmodem 300, 1200, or 1200b or Rixon PC212A with cable
- Voice-grade telephone connection for modem
- Printer
- PC-DOS 2.0/BASICA 2.0

RBBS-PC can be obtained from the *Capital PC Software Exchange, P.O. Box 6128, Silver Spring MD 20906* by sending a check for $6. RBBS-PC is distributed on one double-sided 8-sector diskette. RBBS-PC CPC12 can be a lot of fun to operate and can be a useful tool for information exchange. RBBS-PC seems to be meeting a need within the IBM-compatible PC user community. It has had a rapid evolution in the past year of its existence.

There are some future directions that RBBS-PC may take. KERMIT presently exists for IBM's VM/CMS and DEC's VAX and UNIX, and adds an error-correction protocol that doesn't rely on control character sequences being exchanged. This may very well be incorporated in a future release of RBBS-PC. KERMIT, which was developed by Columbia University, is quite popular because coding for KERMIT has generally been available on the mainframes.

GW-BASIC seems to have been adopted as a standard for the IBM-compatible PCs. Modifying the RBBS-PC source to be hardware independent would further widen the use of RBBS-PC—the current source has PEEKs and POKEs that may have to be changed on some clones. The rewriting of RBBS-PC in C is already underway, and may be completed by the time you read this article.

Getting started

Getting started with a public bulletin board system is not all that difficult, but at the same time some points should be noted. Activity, at first, will be slow

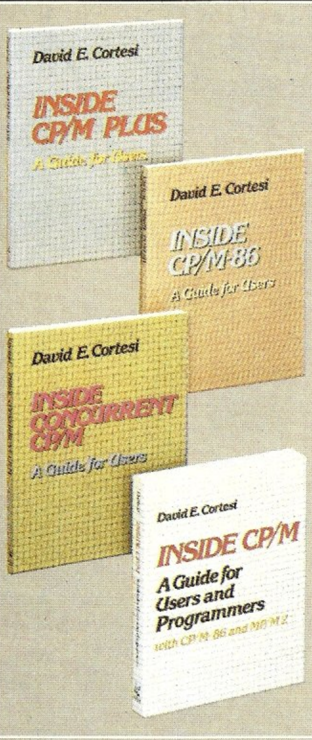
RBBS-PC will no doubt become the de facto bulletin board system standard.

until the telephone number for the board becomes established. A primary use of the bulletin board is for making available the downloading of public domain software. This is also its primary problem. Callers with autodial telephone features could camp on a line for excessively long periods of time, thereby locking out others who may wish to access the system only for messages.

No matter how sincere and trusting the majority of the public may be, there will always be a very few who think it is their duty to crash a system or to put offensive language onto a public board. We have not so far had any government regulations on personal computers. But we are surely inviting government intervention if we don't regulate ourselves. Many bulletin boards operated by clubs are now requiring registration before the user is permitted to enter messages. It is to be hoped that all of this will be sorted out, and that there will be no big brother legislators to oversee PC bulletin board usage.

CBS Computer Books— as advanced as you are

NEW!
A powerhouse trio of MACHINE-SPECIFIC CP/M GUIDES by the #1 author in the field
DAVID E. CORTESI



Inside CP/M Plus: A Guide for Users
With this clear guide to the updated version of CP/M for 8-bit machines, you can take advantage of all the features offered by one of today's hottest operating systems. Complete with reference section of CP/M PLUS commands. $18.45

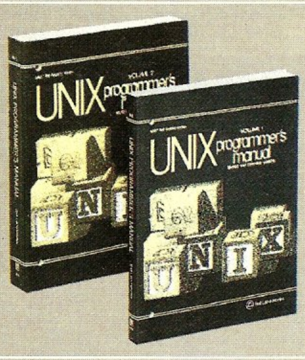
Inside CP/M-86: A Guide for Users
The first complete guide to CP/M for 16-bit computers. Emphasis is on quicker start-up times and better use of automated command sequences. Includes detailed coverage of CP/M-86 system commands. $17.45

Inside Concurrent CP/M: A Guide for Users
Everything you need to know about the new multi-tasking system for 16-bit machines—presented in tutorial sections that will soon have you performing file transformations without special programming. Includes a complete reference table of CONCURRENT CP/M commands. $18.45

...And the Cortesi bestseller that started it all

Inside CP/M: A Guide for Users and Programmers
"That rare gem, the well done introductory computer book."—*Personal Computer Age.* "A landmark work...by far the best CP/M guide I've seen."—*Microcomputing.* $26.45

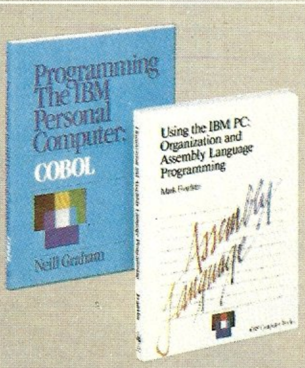
With all eyes on UNIX, we bring you Bell Labs' official documentation!



UNIX Programmer's Manual: Volume 1
By Bell Laboratories. This invaluable reference is your prime source for information on the use of system calls, special languages, compiler-writer tools, and all other aspects of the UNIX operating system. $35.45

UNIX Programmer's Manual: Volume 2
By Bell Laboratories. The companion to Volume 1 provides supplementary information conveniently organized into the areas of basics, editing, language tools, document preparation, and system maintenance. $35.45

Two musts for sophisticated use of your IBM PC



Programming the IBM Personal Computer: Cobol
By Neill Graham. Only an expert like Graham could introduce the IBM PC version of Cobol so clearly. He shows users how to construct a Cobol program, compile it, link it to previously written routines, and execute it. $20.45
Also available with disk at $40.45

Using the IBM PC: Organization and Assembly Language Programming
By Mark A. Franklin. *The* Assembly Language book for 1984. From basic data types and hardware/software concepts to interfacing with higher-level languages, it contains a wealth of hard-fact information. $20.45

CBS Computer Books
FROM HOLT, RINEHART & WINSTON
When A Help Command Isn't Enough

CBS/HOLT, RINEHART & WINSTON, Division Marketing Unit, 383 Madison Avenue, New York, NY 10017
CIRCLE 120 ON READER SERVICE CARD

MS-DOS Window

Continued from page 27

The multifunctional PC

The use of personal computers for home budgeting has not really materialized. It has been faster and much cheaper to use paper and pencil for such a task. The vast popularity of the PC in the corporate environment has been primarily in the use of electronic spreadsheets.

There is no one reason to justify the purchase of a personal computer for the home. The PC should be viewed as a utility with many functions capable of serving the homeowner. The use of the PC will grow as new functions are made available—ones that are not just automation of manual tasks.

The media hype with personal computers contends that if you don't have one in the home, a deficiency in the education of your offspring would surely result. In view of this assertion, hat has been the dominant role of the personal computer in the home? On the whole, very little good educational software is available. However, games proliferate, though they are not necessarily copies of non-computer games like Monopoly or chess, but rather extensions of video games that have been so popular in the arcades. Word processing is another common use; students have taken advantage of it to write term papers and theses. PCs have made it easier to organize thought on paper.

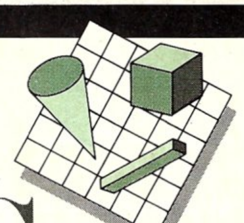
With the use of personal computers for bulletin board access, word processing, and education, the PC will be recognized as a home utility, serving many funtions. As the functionality of the PC grows, so will the number of households owning a personal computer. By the late 1980's it may be as common for families to have multiple PCs in the home, as it is to have multiple cars in the garage.

DisplayWrite 2 postscript

In my September column, I reviewed IBM's DisplayWrite 2 and commented on the limited number of printers that it can support. Since then, I have received from Koch Industries DW2's Em-U-Print, which provides expanded capabilties to DisplayWrite 2. This combination might just spell the end of production for the IBM DisplayWriter. The added printer drivers support the Qume Sprint 11 and the HP LaserJet printers.

For more information, contact: **Koch Industries Inc.,** *P.O. Box 812, Northbrook, IL 60062.*

Hank Kee, 42-24 Colden St., Flushing NY 11355



The Graphics Palette

Illuminating HALO

by Dave McCune

Graphics standards are inherently bureaucratic. The committees and subcommittees that produce them are political bodies, populated by representatives from corporations, trade groups and other highly partisan organizations. Standards evolve slowly and are hard to change.

Especially in a field as volatile as microcomputer graphics, standards are often obsolete before they are officially approved. And since official standards require bureaucratic consensus, rather than just a consensus of customers, they often try to be all things to all people. Instead, they are sometimes so nebulous that they are of little concrete use to anyone. Matching an application to a standard is often like asking a ballerina to dance with an elephant.

The gaps left by these de jure standards are sometimes addressed by unofficial de facto standards. These are generally private programs and protocols which are so useful that they gain widespread acceptance without intervention from official standards organizations. In data communications, the XMODEM and Kermit protocols are examples. CP/M, MS-DOS and UNIX are de facto operating system standards.

Computer graphics has its own de facto standards. The display protocol used by the Tektronix 4010 graphics protocol is one example. Another is HALO, a graphics programming interface built from a library of 134 drawing and graphics control subroutines.

HALO fills a gap

Aside from being a useful product, HALO is a wonderful lesson in the history of microcomputer graphics. HALO's authors make no secret of the fact that HALO resembles the Graphical Kernel System (see the last two months' columns on GKS). But during 1981 and 1982, while the IBM PC infiltrated millions of offices, ANSI (American National Standards Institute) and ACM (Association for Computing Machinery) were still arguing about *which* standard to champion (GKS or CORE), rather than encouraging standard implementations of either. And without 'official' standards, few companies were willing to produce implementations for micros.

In the meantime, bit-mapped color graphics became accessible to millions of users and programmers. And every programmer who sat down at, say, an IBM PC had to painfully encode a custom set of graphics display device drivers before writing the core of the application program.

As the IBM PC became a de facto standard micro in 1982, it was clear to graphics programmers that a standard, off-the-shelf set of routines to drive the common PC-compatible graphics devices would be a godsend. A full implementation of GKS or CORE for the PC would have been nice, but most of us just clamored for a simpler set of graphics drivers for the IBM graphics adapter, perhaps the Epson printer, and maybe two or three of the new PC-compatible enhanced graphics adapters. When you need to get somewhere fast, a Volkswagen now is better than a Rolls Royce later.

When HALO came out in late 1982, it was a welcome Volkswagen. It vaguely resembled GKS. It provided several graphics primitives—e.g., circle, polyline, and box—which could be filled with various hatch styles. It had simple text capabilities and primitive disk I/O. If you wanted to write a graphics program, HALO sure beat writing all the low-level stuff from scratch. HALO was snapped up by graphics programmers and quickly became something of a common tongue for IBM PC programming.

Much has changed since 1982. The Rolls Royces—or the Cadillacs, at least—have arrived in the form of rather complete GKS implementations for micros (see a review of one of these in last month's column). GKS has the vaunted advantage of existing on many different computers, from PCs to mainframes, thus providing a high degree of processor independence. HALO, on the other hand, provides an interface to many graphics display peripherals, but only on the IBM PC (and clones). And some of the new features of HALO (e.g., world and normalized device coordinates) are merely approximations of GKS functions.

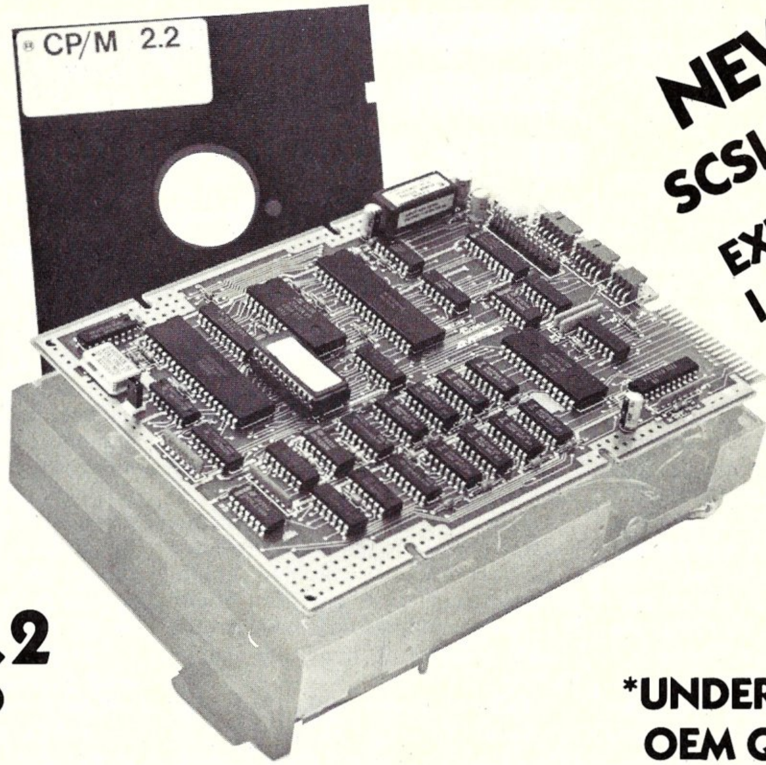
Table 1. Languages supported by HALO

Basic Interpreter (IBM)
Basic Compiler (IBM)
Macro Assembler (IBM)
Pascal (IBM and Microsoft)
Fortran (IBM and Microsoft)
C (Lattice, Microsoft, Computer Innovations)

Has HALO been eclipsed by GKS? No, I think not. While HALO lacks some of GKS' power (e.g., segment

The Little Board™...$349*

The world's simplest and least expensive CP/M computer



NEW SCSI/PLUS™ EXPANSION I/O OPTION

CP/M 2.2 INCLUDED

***UNDER $200 IN OEM QUANTITIES**

- 4 MHz Z80A CPU, 64K RAM, Z80A CTC, 2732 Boot ROM
- Mini/Micro Floppy controller (1-4 Drives, Single/Double Density, 1-2 sided, 40/80 track)
- Only 5.75 x 7.75 inches, mounts directly to a 5 1/4″ floppy drive
- 2 RS232C Serial Ports (75-9600 baud & 75-38,400 baud), 1 Centronics Printer Port
- Power Requirement: +5VDC at .75A; +12VDC at .05A/On-board -12V converter
- CP/M 2.2 BDOS • ZCPR3 CCP • Enhanced AMPRO BIOS
- AMPRO Utilities included:
  - read/write to more than 2 dozen other formats (Kaypro, Televideo, IBM CP/M86....)
  - format disks for more than a dozen other computers
  - menu-based system customization
- BIOS and Utilities Source Code Available
- SCSI/PLUS Adapter:
  - Mounts directly to Little Board • Slave I/O board control • Full ANSC X3T9.2
  - 16 bidirectional I/O lines • $99/Quantity 1

AMPRO
COMPUTERS, INCORPORATED

Distributor/Dealer/Reps
Inquiries Invited

Z80A is a registered trademark of Zilog, Inc.
CP/M is a registered trademark of Digital Research.

67 East Evelyn Ave. • Mountain View, CA 94041 • (415) 962-0230 • TELEX 4940302

CIRCLE 40 ON READER SERVICE CARD

Palette

Continued from page 30

storage and the *Pick ID* concepts), it provides some useful non-GKS functions (e.g., rubber-banding of primitives). HALO currently supports far more IBM PC video devices than any PC-based GKS version, while GKS handles more hardcopy devices (especially plotters) than HALO. And unlike GKS, HALO is extremely easy to learn and use.

Language support

The single-CPU HALO license buys the HALO library for one of six supported languages (Table 1). Note that HALO supports more languages than any current IBM PC implementation of GKS. I know of no Basic interpreter, Pascal or assembler GKS binding for the PC, so if you are wed to those languages GKS may not be an option.

I have used the Lattice C version of HALO 2.0. It comes on four floppies with over 200 pages of documentation. Three of the floppies contain the HALO subroutine library, which comes in five copies, one for each of the four Lattice memory models and one special model which allows HALO to be linked outside a 64K code segment. The library is approximately 150K in size. The library is specific to a particular display device, as well as to a language. So you buy the *IBM Color/Graphics Adapter Lattice C* library, say, or the *Scion Pascal* library.

Media Cybernetics, which sells HALO, also offers a MultiHALO version that allows the programmer to switch between display drivers at runtime. Media Cybernetics also sells a library of printer drivers for $100 and a library of text fonts for $100. They do not charge a royalty on programs you sell that contain HALO code. Instead, they negotiate a one-time multi-CPU license (a 'vertical' license) for applications programmers who want to sell HALO-based programs. The marketing director told me this up-front fee could range from $5000 to $125,000, depending on the application you have in mind. I guess it would pay to leave the fancy car and snazzy suit at home when you show up to cut a deal with these guys. . . .

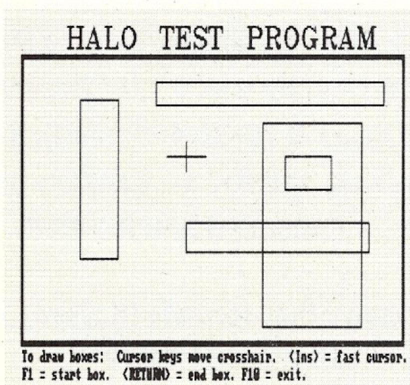
Learning HALO

The fourth disk contains a wonderful tutorial interface to HALO. When you crank up this LEARNHALO program, you can interactively type in HALO functions and argument lists and immediately see the effect on the screen. This program is a great timesaver, since you don't have to constantly modify, compile and link test programs to find out how to use a particular function. And the manual contains sample LEARNHALO sessions to guide you through the major features. As an added bonus, you can even store LEARNHALO command lines in a separate file and then run the batched commands. In other words, HALO graphics can be stored in an English-like display-list format and executed by LEARNHALO.

Complex images can be made to move quickly—a useful animation technique.

The manual is a good reference source, though it assumes you have some background in computer graphics. Each function is described briefly but clearly, and I had no trouble writing HALO programs with only the manual for help.

Table 2 lists some of the 134 HALO functions. Listing 1 shows many of these functions in actual use in a simple HALO program that allows the user to draw boxes on a screen.



HALO functions are called like any C functions. Since HALO pushes the addresses of passed arguments on the stack, though, you must always be sure to pass *addresses*, not values, to HALO. In C, then, arguments must be passed with the *address of* (&) operator. This is a bit awkward, of course, especially since you can't pass constants this way. One solution suggested in the HALO manual is to write macros which invoke HALO functions. For example, the line drawing function:

```
LNABS(X,Y)
```

could be included in a macro definition (Listing 2).

From now on, we can simply invoke the macro _lnabs(x,y). We can even pass constants: _lnabs(100,200). An added benefit of this construction is that the C preprocessor will now register an error if we pass an incorrect number of arguments to a HALO macro/function. When we pass arguments directly to HALO functions, we are never notified of this error.

HALO basics

HALO is a vector-oriented graphics interface. To draw a box, say, you specify only the coordinates of two diagonal corners, rather than describing all the points that make up the border of the box. In HALO, like GKS, the programmer inputs function calls and argument lists, and the graphics software interprets these and turns them into physical graphics displays. Compare this to some paint programs, in which the program simply places dots on the screen in response to an imaginary brush. Such paint programs often cannot recognize and manipulate higher-level graphic entities like circles, boxes and lines.

HALO has a complete set of graphic primitives: point, line, polyline (a series of joined lines), arc, pie wedge, circle and box. Objects can be filled with solid colors or hatch patterns defined by the programmer. (In fact, by defining a hatch pattern that alternates between pixels of two different colors, you can create new so-called 'dithered' colors.)

The concept of the graphics cursor is central to HALO. At any time, HALO keeps track of an invisible cursor. This is the current *drawing point*, analagous to the current location of a pen on a plotter. The graphics cursor may be moved around the display without leaving a trail, just as a plotter pen can be moved while the pen is up. Many HALO drawing functions use this invisible drawing point. The LNABS(X,Y) function, for example, draws a line from the current graphics cursor location to the point X,Y.

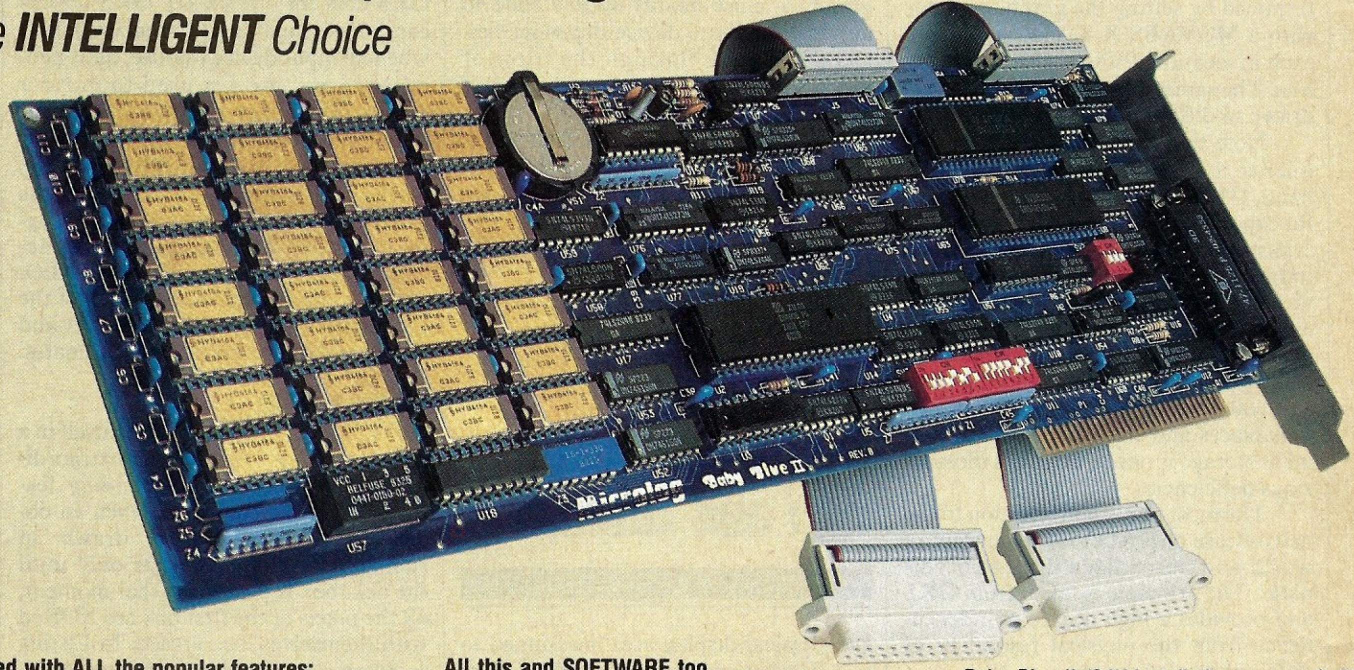
Coordinates

HALO deals with both absolute and relative coordinates. In the LNABS(X,Y) function, X,Y is an absolute location on the display coordinate grid. But in the LNREL(X,Y) function, X,Y are distances on the X and Y axes from the current drawing point. You could, for example, write a subroutine

Baby Blue II

Multifunction/Multiprocessing

*The **INTELLIGENT** Choice*



Packed with ALL the popular features:

- Up to a **256 Kilobyte Memory Expansion**, optional in 64K blocks.
- **Clock/Calendar** with long-life battery back-up (choose replaceable or optional rechargeable battery).
- **Parallel Port** for use with IBM/Epson and most other parallel printers (addressable as LPT1: or LPT2:)
- **Two (2) Serial Ports** addressable as COM1: and COM2:, OR use our unique extended port addressing to configure for serial devices other than COM1: and COM2: (i.e. pointing devices, graphics tablets, etc.)
- **Baby Blue II** is fully compatible with the IBM PC, the PC/XT and other compatibles, including the Texas Instruments Professional Computer.

Not just another DUMB multi-function board.

With all that hardware to manage, we think a board should have a brain. You won't find this feature on other boards – they may look pretty, but they can be pretty boring once you get them home. Baby Blue II is actually a second computer inside your PC, built around the high-speed Z-80B microprocessor.

- **Background Processing:**
  Compile, assemble, sort, calculate, communicate or print – all in the background, while you and your PC continue to work on other tasks in the foreground. Because it is a separate computer, Baby Blue II performs these functions without slowing down your PC, helping you to do more work in less time.
- **Autostart at Preset Time:**
  Baby Blue II can begin any task, even a background operation, at a specific time by consulting the on-board real-time clock.
- **CP/M Capability:**
  Baby Blue II offers instant access to the vast CP/M-80 library of mature, professional software for every conceivable application.
- **Dual Ported Memory and I/O:**
  You can use Baby Blue II's memory, ports and clock as ordinary enhancements to your PC. Or, you can let our Z-80 micro-processor control the board directly, for truly independent back-ground operation.

All this and SOFTWARE too . . .

Systems utilities

PDQ RAMdisk Software: create a FAST pseudo-disk drive of any size in available system RAM. The DOS 2.0 version lets you change the size of your RAMdisk from the keyboard.

Print Buffer/Spooler: a TRUE print spooler: the Z-80 buffers and manages printing independent of your PC. Unlike other so-called "spoolers", this one won't stop your printer or slow you down when you start another job.

Clock Software: sets Baby Blue II's clock and initializes the system clock at boot time – never type the time and date again!

Communications

Smart Terminal Emulator Package (STEP): talk to other microcomputers or connect to larger host computers, as an asynchronous terminal through Baby Blue II's serial ports. Unlike other "smart terminal" programs, STEP offers full emulation of popular video display terminals (the standard package includes Televideo 950 and Hazeltine 1500.
IBM 3101, DEC VT100 and many others are optionally available). You can send or receive text files, and with STEP's unique Sessions Menu, changing your configuration is a keystroke away.

BSTAM File Transfer Utility: Transmit and Receive text, HEX, and binary files (including .COM files) without errors and without fuss. BSTAM is easy to use, with all configuration parameters handled transparently under STEP.

CP/M-80 compatibility

Baby Blue Conversion Software: Microlog's famous CP/M Emulator turns CP/M-80 programs into PC-DOS programs for fast, efficient execution on Baby Blue II. Completely transparent operation using standard PC-DOS commands – freely mix PC-DOS with CP/M programs and text/data files on the same PC-DOS disks.

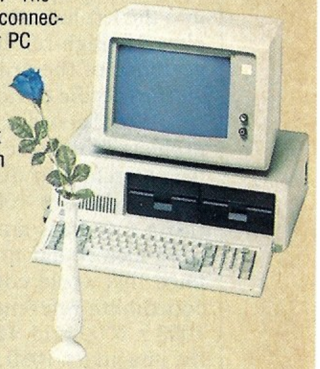
Convert: supports bidirectional file transfer between PC-DOS and popular CP/M disk formats.

Keyfix: automates your keyboard with 54 programmable function keys for CP/M programs, eliminating tedious typing chores (max. 80 characters per key).

Baby Blue II (64K installed) . . . . . . $ 695.

QUALITY you can count on . . .

Microlog is in the business of producing high quality, intelligent computer peripherals. Extensive factory testing gives our products one of the highest reliability rates in the industry. The finest components, connectors, and multi-layer PC board design, ensure years of trouble-free operation. We back all our products with a 90 day warranty and full technical support.
For more information on Baby Blue II and other fine Microlog products, see your micro-computer dealer or contact us:



Microlog™

A TU INTERNATIONAL COMPANY

Microlog, Inc.
222 Rte. 59
Suffern, NY 10901
(914) 368-0353

CIRCLE 116 ON READER SERVICE CARD

Palette

Continued from page 32

to draw a complex polygon using only relative coordinates. The location of the polygon in the display would then be determined by setting the graphics cursor with a MOVABS(X,Y) call just before each invocation of the polygon subroutine. The same object could easily be redrawn in different locations.

Note that GKS does not offer this relative coordinate feature. On the other hand, GKS provides a far more powerful capability called segment storage. It is possible to store any sequence of GKS output in an internal storage area and to identify the output as a segment. The entire segment can then be manipulated as an entity. By applying coordinate transformations (see below) to the GKS segment, it can be redrawn easily in various places and sizes. This lack of segment storage is one of HALO's most serious deficiencies.

The most important addition to the new release of HALO is the concept of world and normalized device coordinates. The principle is taken from GKS and provides a high degree of independence from the physical resolution of different display devices.

As an example, the IBM color graphics adapter offers a medium-resolution (320 x 200) and a high-resolution (640 x 200) mode. If we were to write a program which referenced physical pixels in medium-resolution mode, the program's graphics would be squashed into the left half of the screen when displayed in high-resolution mode. The appearance of the graphics thus depends on the specific device being used.

World coordinates free us from this problem. We define an arbitrary coordinate map for our display. In Listing 1 I chose a 1000 x 1000 display. I chose those numbers simply because they were nice round numbers. After I defined this world coordinate system, all coordinate references mapped to this 1000 x 1000 grid. HALO saw to it that the imaginary 1000 x 1000 grid mapped onto the full physical display of whatever physical device I used. Thus, world coordinate location 500,500 was in the exact center of the screen in both high-res and medium-res modes.

With the addition of this world coordinate system, HALO achieves device independence, at least with respect to IBM PC display devices. Unfortunately, though, HALO is annoyingly inconsistent. Some HALO functions still deal with physical pixels. SETLNWIDTH, for example, sets the line width in terms of physical pixels, not world coordinates. So a line width of 10 is twice as thick in medium resolution as in high resolution. And HALO bit-mapped text assumes a standard character size of 8 x 8 physical pixels. So the commands that draw 80 characters across the screen in high-res will only fit the first 40 on the screen in medium resolution. The programmer must modify source code to account for these device dependencies. GKS has gone through the filter of much public review and so is freer of such inconsistencies.

Another important coordinate concept borrowed from GKS is normalized device coordinates. In the NDC system, the physical display area is assumed to be 1.0 units wide and 1.0 units long. Particular locations on the display can be referred to in terms of fractions of this unit screen, e.g., location 0.5,0.5 is the middle of the display. The SETVIEWPORT function lets us set aside a subset of the screen and map our world coordinates to this subset. In Listing 1 our 1000 x 1000 world coordinate grid is mapped into a viewport that covers 0.8 of the unit screen in each direction. This subset of the screen is circumscribed with a border, and HALO will not let us move outside the viewport. By manipulating viewports we can set up various work areas in the display and alternately protect and unprotect them.

HALO provides a feature that lets you pan and zoom.

Finally, HALO provides a SETWINDOW function that allows us to map a subset of the world coordinate display to a particular viewport. We can use this feature to pan and zoom.

HALO provides relatively weak input functions. The only supported input devices listed in the manual are mice. I was disappointed to find that HALO doesn't even provide a cursor-key keyboard driver. Note that I was forced to write this driver myself in Listing 1. The GSS GKS implementation offers full support for the PC's keyboard as an input device.

HALO lacks the extremely useful GKS concept of *Pick ID*. GKS objects can be assigned an ID number. Later, when an input device (mouse, light-pen, cursor keys, etc.) is used to move a cursor, GKS will return the ID number of any object pointed to by the cursor. The programmer does not have to keep track of where icons are located on the screen, for example, in order to recognize that one has been picked. To accomplish the same function in HALO, the programmer must keep track of the coordinate locations of each object and compare them to the input coordinates.

Rubber-banding

HALO has the stage all to itself in a couple of areas. It offers an extraordinarily convenient *rubber-banding* feature. The principle is that when an object such as a box is drawn in rubber-band mode, it exists only until the next box is drawn. At that moment, all the pixels of the first box are XORed with themselves, i.e., erased. In Listing 1, once a user has selected the start coordinates of a box, a rubber-band box follows the cursor around the screen until the end coordinates are selected. This is a wonderful tool for interactive graphics applications and would be clumsy to reproduce in GKS.

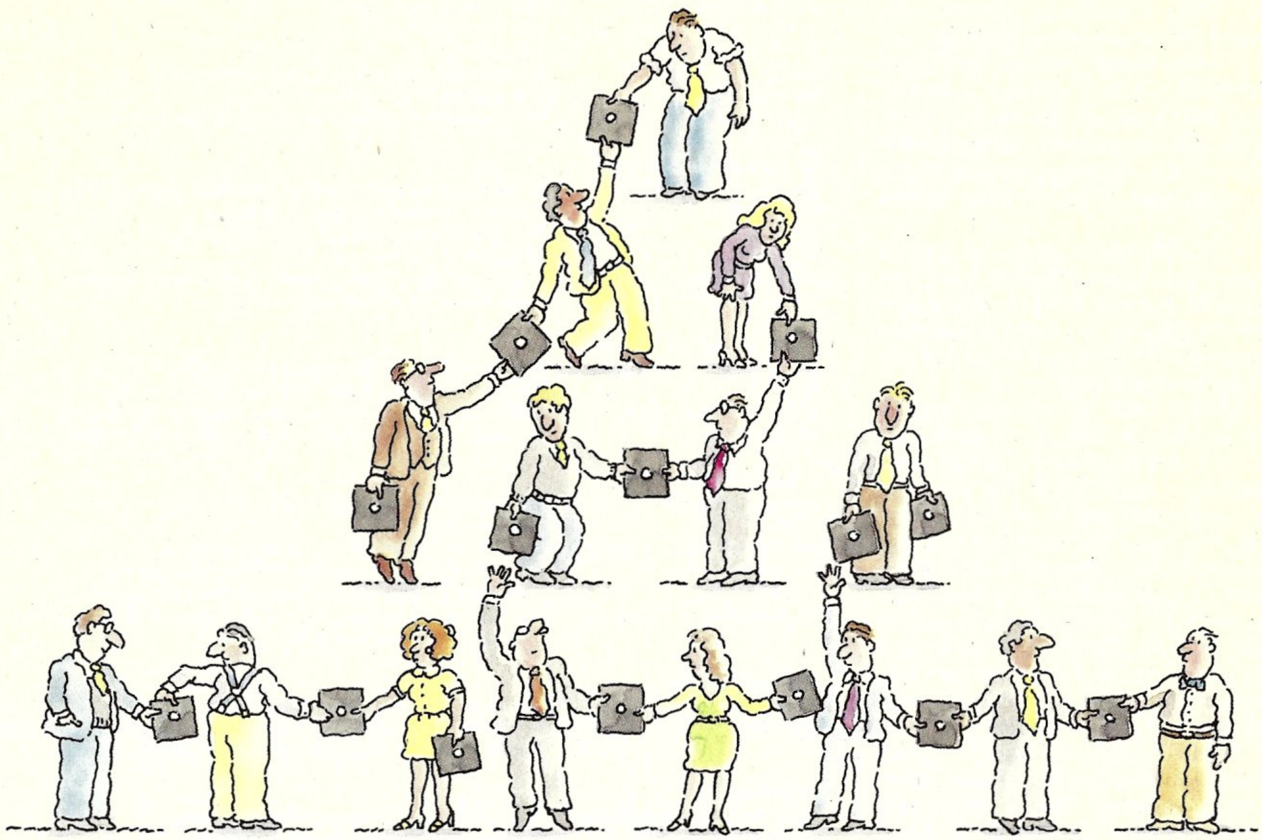
Further, HALO provides the ability to move whole rectangular areas of the screen quickly (MOVEFROM and MOVETO). As these blocks are moved, it is possible to XOR the old block with itself. The effect is that complex images can be made to move quickly. This is a useful animation technique.

HALO and GKS differ markedly in the way they store images to disk. The HALO GWRITE function stores the display bitmap in a compressed *run-length encoded* form. Instead of storing information for each individual pixel, HALO recognizes when adjacent pixels are of the same color and so stores the information for just one of the pixels, preceded by a *repeat count*. Bitmap compression of four to eight times is possible with this technique.

Table 2. Sample HALO functions

Initialization	
initgraphics(mode)	Clr scrn & init graphics; select mode (e.g., IBM hi-res)
closegraphics	Close HALO.
Color Manipulation	
setcolor(color)	Set current drawing color.
setpal(background, palette)	Set background and palette colors.

THE END OF SOFTWARE FREEBIES.



Finally there's a foolproof way to protect software against unauthorized duplication. And the technology is all on the disk itself.

The new Prolok™ disk doesn't need add-on hardware. Instead each diskette is marked with a unique, physical "fingerprint." No two are alike. A precise description of the individual print is encoded magnetically. The fingerprint AND the description must match exactly before the software is decrypted and released to the system. No match, no access.

Its genius is its simplicity and familiarity. Prolok looks like an unprotected disk, loads like an unprotected disk, works like an unprotected disk. The user feels immediately at home and in command. It's as easy as A>PROLOK B: filename.

Backups are easily made via normal system utilities. However, to be read they must be accompanied in the system by the original Prolok disk.

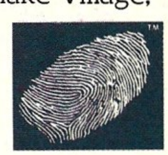
Prolok puts the casual copier—and even the deliberate pirate—out of business. It barely increases the price of your product, yet it makes sure your customers don't buy one program and copy ten.

Several command line slash (/) options are built into Prolok diskettes for customized security, depending on your needs.

Software can be loaded easily onto Prolok diskettes using any system from a PC to commercial mass duplication equipment.

Prolok is an engineering breakthrough of Vault Corporation, which has been successfully safeguarding software since the inception of security disk technology. Over 2000 businesses and organizations protect their valuable programs with Prolok.

Simply contact Vault Corporation at 2649 Townsgate Road, Suite 500, Westlake Village, CA 91361. Or phone us at 800-445-0193 (U.S.) or 800-821-8638 (California). And find out why software freebies are becoming a thing of the past.



PROLOK.
SOFTWARE PROTECTION, RIGHT ON THE DISK.

CIRCLE 107 ON READER SERVICE CARD

Continued from page 34

The problem with HALO's storage technique is that it is nearly impossible to identify high-level objects such as circles or boxes once a bitmap file has been read. GKS, in contrast, stores a compressed list of display commands to a so-called 'metafile.' This display list can be read by GKS almost as if a human being had entered the output commands. GKS can identify all higher-level objects in a metafile and so can let the user manipulate them once the file has been read.

If the ability to store this type of display list is important to you, you should be wary of HALO. While it is perfectly possible to maintain and store your own HALO display list, it is a significant programming job.

HALO's text functions are impressive. In addition to the 8 x 8 pixel standard text described earlier, HALO provides *stroke text*. This is text drawn in any number of fonts, in any size, aspect ratio, or angle. The standard package comes with one stroke-text font. A package of 12 additional fonts is now available, with 12 more in development. In addition, Media Cybernetics is willing to help programmers create their own font libraries. See Listing 1 for an example of both standard and stroke text.

HALO supports a wide range of video output devices, but a rather limited set of hardcopy devices. (Quite the opposite is true of the GSS GKS implementation; it is very strong on plotters but weak on video devices.) All of the major enhanced video boards for the IBM PC are supported.

Version 2.0 of HALO supplies an improved set of inquiry functions. It is easy to find the state of HALO at any time, including current cursor coordinates, colors, text attributes, and so on. One disappointment, though, is inadequate error handling: far too little detail about errors is supplied. If you pass invalid arguments, HALO will not offer much help in tracking them down.

Summary

All in all, HALO may not be as appropriate as GKS for some applications. Certainly, six months ago—before there was a GKS for the IBM PC—HALO was a winner. Today, though, GKS seems a better deal. Especially if you need to drive many plotters, or if you need to port your program to machines other than the IBM PC, HALO is not a good choice.

On the other hand, I enjoy working with HALO. It is easy and straightforward for the most part. The rubberbanding and animation capabilities are

inqclr(x, y, color)	Returns color at location x,y.
Crosshair, Text & Graphics Cursor Functions	
inithcur(height, width, color)	Initialize crosshair cursor.
inqhcur(x, y, color)	Returns location and color of crosshair cursor.
movhcurabs(x, y)	Move crosshair cursor to x,y.
inittcur(height, width, color)	Initialize text cursor.
deltcur()	Delete text cursor.
movtcurrel(Dx, Dy)	Move text cursor dx,dy from current location.
movabs(x, y)	Move graphics cursor (drawing pt) to x,y.
inqgcur(x, y, color)	Returns current graphics cursor location & drawing color.
Point and Line Functions	
deflnstyle(style, array(i))	Define a line style (e.g., dotted & dashed in magenta & white).
lnabs(x, y)	Draw line from current drawing pt to x,y.
lnrel(dx, dy)	Draw line to a point dx,dy from current drawing pt.
ptabs(x, y)	Draw point at location x,y.
polylnrel(xarray(i), yarray(i), n)	Draw n line segments, using element pairs of xarray & yarray.
setlnwidth(width)	Draw all future lines width pixels thick.
Circle, Arc and Pie Functions	
arc(r, start_ang, end_angle)	Draw arc of radius r from start_angle to end_angle
cir(r)	Draw circle of radius r with center at current drawing pt.
pie(r, angl, ang2, color)	Draw pie wedge of color with radius r from angle angl to ang2.
setasp(aspect)	Set aspect ratio of display (in order to make a circle round).
Area Fill Functions	
bar(x1, y1, x2, y2)	Draw solid bar with diagonal from x1,y1 to x2,y2.
clr()	Clear viewport
defhatchstyle(style, array(i))	Define hatch fill pattern with id style in array.
fill(color)	Fill bounded area with specified color or hatch pattern.
flood(color)	Flood complex region with specified color or hatch pattern.
polyfabs(xarray(i), yarray(i), n, color)	Fill polygon of n vertices, using element pairs of xarray & yarray, with specified color or hatch style.
Area Move Functions	
movefrom(x1, y1, x2, y2, array(i))	Copy bitmap defined by coordinates to array.
moveto(x, y, array(i), mode)	Move bit-map data in array to area with upper left at x,y, using given mode (e.g., destroy background, XOR new data with background).
Marker Functions	
initmarker(index, ascii_value)	Assign ASCII character in ascii_value to index.
markerabs(index, x, y)	Draw marker specified by index at location x,y.
Graphic Text Functions	
settext(height, width, path, mode)	Set text height & width in integer multiples of 8 x 8 pixels. Set path to up, down, right, left. Mode = no box around cell, or character in colored cell.
text(string)	Display string at current text cursor location.
Stroke Text Functions	
setfont(font_file)	Select font described in font file.
setstext(height, aspect, path)	Specify height, aspect ratio and path of stroke text.
setstang(angle)	Draw future stroke text at specified angle.
inqstext(ht,aspect,line,fill,path)	Returns current stroke text attributes.
Rubberband Functions	
rbox(x1, y1, x2, y2)	Draw box with diagonal x1,y1 & x2,y2. The box is XORed with itself as soon as the next RBOX is drawn.
rcir(r)	Draw a rubberband circle with radius r.
rlnabs(x, y)	Draw a rubberband line.
Coordinate Transformations	
setworld(x1, y1, x2, y2)	Define arbitrary world coordinates.
setviewport(x1, y1, x2, y2, border, background)	Define a viewport as a fraction of the display screen (i.e., as Normalized Device Coordinates), & map world coordinates into the viewport. Set border color and background color.
mapdtow(xd, yd, xw, yw)	Transform device coordinates to world coordinates.
mapntow(xn, yn, xw, yw)	Transform normalized device coordinates to world coordinates.
inqviewport(x1, y1, x2, y2)	Returns current viewport bounds.
Graphic I/O Functions	
gprint()	Print screen image on printer.
gwrite(file)	Write screen image into compressed DOS file.
gread(file)	Read and display file.
setgprint(device)	Select printer (currently, only Epson/IBM and IDS Prism supported).
setlocator(device, port)	Select graphics input device (currently, only certain mice supported).
readlocator(x, y, switch)	Get mouse coordinates & status of switches.

Listing 1. Simple HALO program for drawing boxes

```
/*****************************************************************************************
* FILE -        HALOBOX.C                                                                *
* FUNCTION -    Sample HALO program; draws boxes on video screen.                        *
* LANGUAGE -    C (Lattice, V. 2.0, Memory model "S").                                   *
* LINKAGE -     Link with HALO Version 2, HALOS.LIB.                                     *
* DESCRIPTION - HALOBOX illustrates several HALO features, such as text fonts, rubber-banding, device-  *
*               independent world coordinates, normalized device coordinates (viewports), cursors, etc.  *
*               The program outputs descriptive text and instructions to the video screen, and then sets *
*               up a drawing area in which the user can draw boxes of varying sizes.     *
* HISTORY -     001     84.09.08        David McCune                                     *
*****************************************************************************************/
#include (stdio.h)
#include (dos.h)
/************************************************************
* Define keyboard scan codes for cursor keys and function keys  *
************************************************************/
#define CUR_RT          77
#define CUR_LT          75
#define CUR_UP          72
#define CUR_DN          80
#define CUR_UL          71
#define CUR_UR          73
#define CUR_DL          79
#define CUR_DR          81
#define INSERT          82
#define ENTER           0x0D
#define F1              59
#define F10             68
/*****************************************************************************************
* HALO requires the ADDR of all fn arg's to be passed rather than any values. Since this is unwieldy, we  *
* define macros for commonly called HALO functions, in order to pass immediate values or variable names.  *
*****************************************************************************************/
#define _initgraphics(mode)     {int initvar; initvar=mode; initgraphics(&initvar);}
#define _setworld(x1,y1,x2,y2)  {float x1v,y1v,x2v,y2v; setworld(&(x1v=x1),&(y1v=y1),&(x2v=x2),&(y2v=y2));}
#define _box(x1,y1,x2,y2)       {float x1v,y1v,x2v,y2v; box(&(x1v=x1),&(y1v=y1),&(x2v=x2),&(y2v=y2));}
#define _rbox(x1,y1,x2,y2)      {float x1v,y1v,x2v,y2v; rbox(&(x1v=x1),&(y1v=y1),&(x2v=x2),&(y2v=y2));}
#define _movabs(x,y)            {float xv,yv; movabs(&(xv=x),&(yv=y));}
#define _movhcabs(x,y)          {float xv,yv; movhcurabs(&(xv=x),&(yv=y));}
#define _movhcrel(dx,dy)        {float dxv,dyv; movhcurrel(&(dxv=dx),&(dyv=dy));}
#define _movtcurabs(x,y)        {float xv,yv; movtcurabs(&(xv=x),&(yv=y));}
#define _setstext(h,a,p)        {float hv,av,pv; setstext(&(hv=h),&(av=a),&(pv=p));}
#define _settext(h,w,p,m)       {int hv,wv,pv,mv; settext(&(hv=h),&(wv=w),&(pv=p),&(mv=m));}
#define _inithc(h,w,c)          {float hv,wv; int cv; inithcur(&(hv=h),&(wv=w),&(cv=c)),}
#define _inqhc(x,y,c)           {float xv,yv; int cv; inqhcur(&(xv=x),&(yv=y),&(cv=c));}
#define _setlnwidth(w)          {int wv; setlnwidth(&(wv=w));}
/*****************************************************************************************
* Define maximum and minimum world coordinates and one normalized device coordinate viewport.            *
*****************************************************************************************/
#define X_MIN_WORLD     0       /* We define an arbitrary */
#define Y_MIN_WORLD     0       /* 1000 x 1000 workspace */
#define X_MAX_WORLD     1000
#define Y_MAX_WORLD     1000
#define X_LEN_WORLD     (X_MAX_WORLD - X_MIN_WORLD)
#define Y_LEN_WORLD     (Y_MAX_WORLD - Y_MIN_WORLD)
#define X_CEN_WORLD     (X_MIN_WORLD + (X_LEN_WORLD/2))
#define Y_CEN_WORLD     (Y_MIN_WORLD + (Y_LEN_WORLD/2))
#define X_MIN_VIEWP     0.10    /* Our program uses one NDC */
#define Y_MIN_VIEWP     0.09    /* viewport, 8/10s of the */
#define X_MAX_VIEWP     0.90    /* screen wide and 8/10s */
#define Y_MAX_VIEWP     0.89    /* high. */
#define HCX_WORLD       (X_LEN_WORLD / 20)
#define HCY_WORLD       (Y_LEN_WORLD / 20)
/************************************************************
* Miscellaneous definitions and global variables.           *
************************************************************/
#define CUR_FAST        (X_LEN_WORLD / 50)
#define CUR_SLOW        (X_LEN_WORLD / 200)
#define MODE_0          0
#define MODE_1          1
#define STA_NORMAL      1
#define STA_EXIT        2
#define NO_PRIM         0
#define BOX             1
int prim_mode, color;
float x, y, x1, y1;
```

Palette

Continued from page 37

great. HALO also seems to draw its graphics faster than the GSS version of GKS. And HALO supports more IBM PC video devices than GKS.

Perhaps most important, though, is that HALO is not forced to adhere to a standard. If there is a feature of GKS you would like to see modified or added, you face an awesome and turgid standards establishment. But, if you have a suggestion for improving HALO, you can go straight to the source. HALO may well be better able to meet the changing microcomputer graphics market than GKS.

Price: HALO: $200; additional language (each): $100; printer library: $100; font library: $100; MultiHALO: negotiable vertical license fee.

Available from: *Media Cybernetics, Inc., 7050 Carroll Ave., Takoma Park, MD 20912, (301) 270-0240;* or from *Lifeboat Associates, 1651 Third Ave., New York, NY 10128; (212) 860-0300* or *(800) 847-7078.*

Dave McCune, Proteus Group, Inc., 195 Garfield Pl., Brooklyn, NY 11215. CompuServe user no. 71046,1647

```
/***************************************************************
* MAIN                                                         *
***************************************************************/
main()
{
int ah, al = 0;
int status;
int border, bckgrnd;
float cur_step = CUR_SLOW;
   _initgraphics(MODE_1);       /* Init HALO; IBM high-res mode */
   /* Invoke device-independent world coordinate system, & set up an arbitrary 1000 x 1000 work area */
   _setworld(X_MIN_WORLD, Y_MIN_WORLD, X_MAX_WORLD, Y_MAX_WORLD);
   /* Pick a HALO font, & then use "stroke" text to print a title line on the screen */
   setfont("halo104.fnt");
   _setstext((Y_MAX_WORLD/12),1,0) /* ht=1/12 of dsply; aspect ratio 1/1; path hz, lft-to-rt */
   _movtcurabs(185,910);        /* Move text cursor */
   stext("HALO TEST PROGRAM"); /* Print stroke text */
   _settext( 1,1,0,0); /* print text using simpler "dot-matrix" graphics text: ht & width = 1 x "base size"
                        (8 x 8 pixels); path=hz, lft-to-rt; mode="unboxed" (cell not colored) */
   _movtcurabs(100,50);
   text("To draw boxes:  Cursor keys move crosshair.  <Ins> = fast cursor.");
   _movtcurabs(100,0);
   text("F1 = start box.  <RETURN> = end box. F10 = exit.");
   /* Now, define a smaller subset of the scrn for our drawing area by defining a "viewport." We will    */
   /* use a fraction of the unit screen defined as being from 0.1 to 0.9 on the x-axis and from 0.09 to  */
   /* 0.89 on the y-axis. We define a white viewport outline & a transparent background in the           */
   /* "setviewport" function. The 3-pixel line width gives a thick border line.                           */
   /* The clumsiness of the address-passing mechanism is demonstrated by making a direct call to the    */
   /* SETVIEWPORT fn (after assigning the parm variables 1st) instead of using a macro.                  */
   _setlnwidth(3);    x  = X_MIN_VIEWP;     x1 = X_MAX_VIEWP;      y  = Y_MIN_VIEWP;     y1 = Y_MAX_VIEWP;
   border = 1;
   bckgrnd = -1;
   setviewport(&x, &y, &x1, &y1, &border, &bckgrnd);
   _setlnwidth(1);
   _inithc(HCX_WORLD,HCY_WORLD,1); /* init & draw crosshair cursor in the middle of the display */
   _movhcabs(X_CEN_WORLD,Y_CEN_WORLD);
/*************************************************************************************************
* MAIN LOOP:  If the user presses a cursor key, the X and Y coordinates of the cursor are updated. The  *
* function MOVE_OR_DRAW determines whether the user has chosen to draw a box yet. If not, the crosshair *
* cursor is drawn at the new X,Y location. If BOX has already been chosen, then hitting a cursor key    *
* causes a rubber-band box top be drawn between the original X,Y and the new X,Y. If BOX is selected,   *
* then pressing ENTER causes a permanent box to be drawn at the final X,Y coordinates. The crosshair    *
* cursor is turned off while a rubber-band primitive is being displayed.                                 *
*************************************************************************************************/
   while(status != STA_EXIT)
   {
      get_keyscan(&ah, &al);
      switch(al)
      {
         case 0:
            switch(ah)
            {
               case CUR_RT:
                  x = ((x+cur_step)>1000) ? 1000 : (x+cur_step);
                  move_or_draw();   break;
               case CUR_LT:
                  x = ((x - cur_step) < 0) ? 0 : (x - cur_step);
                  move_or_draw();   break;
               case CUR_UP:
                  y = ((y+cur_step)>1000) ? 1000 : (y+cur_step);
                  move_or_draw();   break;
               case CUR_DN:
                  y = ((y - cur_step) < 0) ? 0 : (y - cur_step);
                  move_or_draw();   break;
               case CUR_UL:
                  y = ((y+cur_step)>1000) ? 1000 : (y+cur_step);
                  x = ((x - cur_step) < 0) ? 0 : (x - cur_step);
                  move_or_draw();   break;
               case CUR_UR:
                  x = ((x+cur_step)>1000) ? 1000 : (x+cur_step);
                  y = ((y+cur_step)>1000) ? 1000 : (y+cur_step);
                  move_or_draw();   break;
               case CUR_DL:
                  x = ((x - cur_step) < 0) ? 0 : (x - cur_step);
                  y = ((y - cur_step) < 0) ? 0 : (y - cur_step);
                  move_or_draw();   break;
               case CUR_DR:
                  x = ((x+cur_step)>1000) ? 1000 : (x+cur_step);
                  y = ((y - cur_step) < 0) ? 0 : (y - cur_step);
```

Palette

Continued from page 39

```
            move_or_draw();   break;
         case INSERT:
            cur_step = (cur_step == CUR_SLOW) ? CUR_FAST : CUR_SLOW;
            break;
         case F1:                 /* user want a box! */
            prim_mode = BOX;
            inqhcur(&x1,&y1,&color);
            delhcur();   break;
         case F10:                /* user want to quit */
            status = STA_EXIT;   break;
         default:  break;
      }
      break;
   case ENTER:
      switch(prim_mode)
      {
         case BOX:               /* draw the permanent box */
            delbox();
            _box(x1,y1,x,y);
            prim_mode = NO_PRIM;
            break;
         default:  break;
      }                          /* Turn cursor on again: */
      _inithc((X_LEN_WORLD/20),(Y_LEN_WORLD/20),1);
      _movhcabs(x,y);
      break;
   default:  break;
   }
 }
 closegraphics();  /* Exit from graphics mode */
}

/**********************************************************************************************
* RBAND_PRIM: Draws a rubber-band primitive. HALO draws the primitive & keeps track of its location. Then,*
* when the next identical primitive is drawn in another location, the old object is XORed with itself      *
**********************************************************************************************/
rband_prim()
{
   switch(prim_mode)
   {
      case BOX:
         _rbox(x1,y1,x,y);
         break;
      default:  break;
   }
}

/**********************************************************************************************
* MOVE_OR_DRAW: redraws the crosshair cursor if user has not chosen to draw an object. If drawing in     *
* progress, draws rubber-band version of object.                                                          *
**********************************************************************************************/
move_or_draw()
{
   if(prim_mode == NO_PRIM)
   {
      _movhcabs(x,y);
   }
   else
   {
      rband_prim();
   }
}

/**********************************************************************************************
* GET_KEYSCAN:  uses the Lattice V 2.0 "int86" software interrupt interface to use BIOS interrupt 16h.    *
* Returns scan-code of keyboard hit.                                                                      *
**********************************************************************************************/
get_keyscan(ah,al)
int *ah, *al;
{
   union REGS inregs;
   union REGS outregs;
     inregs.x.ax = 0;
     int86(0x16, &inregs, &outregs);
     *al = outregs.h.al;
     *ah = outregs.h.ah;
}
```

WIN AN APPLE IIc IN THE A+ SWEEPSTAKES

AND SAVE UP TO 33% ON A+ MAGAZINE

The A+ Sweepstakes is open to all our readers—no purchase is necessary—and you can win the new $2,000* Apple IIc System!

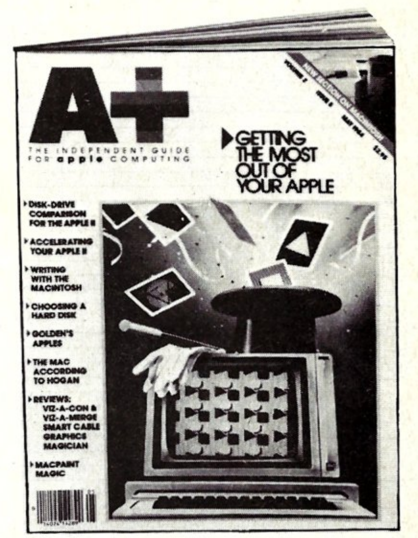
It's Easy to Enter...

Just mail the attached card or coupon below after filling in your name and address—and be sure to indicate whether you're also subscribing to A+ at our special rates, *saving up to 26%*

Just imagine, if you're the lucky winner you can pick up your 7½ pound system and carry it home with you! You'll get over $2,000* worth of equipment including the Apple IIc with a built in disk-drive, 128K RAM and 16K ROM, Applesoft BASIC and Mouse Text, PLUS the new flat panel LCD display and Mouse! *But why dream? Send in your Entry Card today, and save up to 26% on A+!*

Make All Of Your Computing Dreams Come True With A+!

A+, The Independent Guide For Apple Computing makes computing imaginative and educational. Every month you'll receive new business, education, game and personal applications that will help you expand the use of your Apple II, II+, IIe, III, Lisa, Macintosh or the new *IIc* in your home, office or school! In-depth reviews (and previews) of new hardware, software and peripherals makes A+ the only magazine you'll ever need to make your Apple grow!



Enjoy a year or more of A+ at our low prices—save up to 26% and enter the A+ Sweepstakes today! It's an opportunity you won't want to pass up.

*based on manufacturer's suggested list price.

OFFICIAL RULES—NO PURCHASE NECESSARY

1. On an official entry form or 3" X 5" piece of paper, handprint your name,address and zip-code. Enter as often as you wish, but mail each entry separately to: A+ SWEEPSTAKES, P.O. Box 2911, Boulder Colorado 80322. Entries must be received no later than January 31, 1985, and the drawing will be held by February 28, 1985. All entries become the property of A+ Magazine, which reserves the right to reprint the name and address of the winner.
2. The winner will be selected in a random drawing from among all entries received, under the supervision of the publishers of A+ Magazine, whose decision will be final. Only one prize will be awarded in this Sweepstakes. Winner will be notified by mail and may be required to execute an affidavit of eligibility and release. Odds of winning will depend on the number of entries received. Ziff-Davis will arrange for winners to pick up their prize at a local Apple dealer. Any manufacturer's warranties will apply, but Ziff-Davis makes no warranties to any prizes. Prize is not transferable. No substitution for prizes. Taxes are the responsibility of the winner.
3. Sweepstakes open to all residents of the U.S., its territories and possessions, except employees (and their families) of Ziff-Davis Publishing Company, its affiliates and its advertising and promotion agencies. Void wherever prohibited or restricted by law.
4. For the winner's name, send a stamped, self-addressed envelope to A+ SWEEPSTAKES, Circulation Department, Ziff-Davis Publishing Company, One Park Avenue, 4th Floor, New York, NY 10016.

OFFICIAL ENTRY FORM

A+ SWEEPSTAKES

P.O. Box 2928, Boulder, CO 80322

☐ ▶ **YES** Enter my name in the A+ SWEEPSTAKES and start my subscription to *A+ Magazine* for the term checked: AM8Z045

☐ 9 issues for $15.97. SAVE 15%!
☐ One year for $19.97. SAVE 20%!
☐ Two years for $36.97. SAVE 26%!

Savings based on full one-year (12 issues) subscription price of $24.97.

☐ **NO** I don't wish to subscribe now, but please tell me if I've won the Apple IIc.

Mr/Mrs/Ms ____________________
please print full name

Company ____________________

Address ____________________

City ____________ State ________ Zip ________

Check one: ☐ Payment enclosed ☐ Bill me later **Charge my:** ☐ AMEX ☐ Visa ☐ MasterCard

Card No. ____________ Exp. Date ________

Please allow 30 to 60 days for delivery of first issue. Add $1 per issue in Canada and all other foreign countries.

THE WORLDS OF UNIX

This issue is the second part of a special presentation on UNIX. In the first part, we focused on the early evolution of UNIX—the research period at Bell Labs; we also presented reviews of a software implementation of UNIX, an affordable hardware system running System V, and an extensive C-subroutine library package.

In this second part, we examine the developments that have taken place since 1974. An article by Ian Darwin and Geoff Collyer gives an overview of the period and contains much fascinating and heretofore unpublished information; our interview with Andy Hall, Director of the AT&T UNIX System Development Laboratory (Bell Labs), tells of work in progress and speculates on the direction in which UNIX is moving; Bruce Hunter reviews an affordable Codata system running Unisoft UNIX; and Michael Tilson looks at problems that affect the portability of C.

During the period since *Microsystems* was first published, several principles have emerged that help explain how the microcomputer industry has developed. The first is that it is a market-driven industry. The second is that technological advances have occurred at a rapid rate during this period. The third is that there has been a significant shift in the major buying power in the industry. The motivations that drive the industry and the directions the industry has taken can all be explained and understood in terms of these principles.

The proliferation of powerful, yet inexpensive, hardware has made portability and standardization of the software development environment two of the key issues in today's marketplace. When hardware was expensive, the issue of portability did not arise; for each machine, the operating system was designed to take advantage of the particular hardware features of that machine. Such an operating system is, by design, static. When enhancements to it become desirable—or necessary—they often prove to be economically unfeasible because hardware dependencies were designed in.

But the micro industry is market-driven, and the needs of that market change rapidly. New technologies in both hardware and software have had a cause and effect impact on the market-driven nature of the industry. And the shift in the industry's target population from programmer-as-end-user to an environment where the major economic force is the end user community, with a much smaller community of system and software developers, has greatly altered priorities within the industry.

These factors have not only put pressure on the operating system to adapt, but have also caused two competing priorities to emerge: the need to protect the current investment in application and development software, and the need to allow for technological developments in both hardware and software.

It is fairly easy to see that portability affords the best solution to the problem of how to protect existing software investment. And the principles behind portability are similar to the those embodied in creating a standard—whether a specific standard, such as graphics, or the more general idea of a standard operating system environment.

However, by their very nature, standards tend to inhibit technological advances. A feature cannot be included in a standard if it is not universally applicable, and new technologies are generally machine dependent until they gain market penetration and become widespread. On the other hand, since the intuitive view of portability assumes certain standards, it becomes valuable to reexamine standards to see if there is a way for them to accommodate technological advances without destroying the underlying principles of a standard. This is what we propose to do.

A good standard is usually created by discovering features that are already in widespread use, and putting them together to form the standard. Thus, a standard becomes a collection of universal features treated in a systematic way. If, instead, we think of the foundation of a standard as being a minimum threshold machine capability (processor power plus peripheral capabilities), then our collection of features gains some flexibility: any feature a machine cannot perform in hardware, it must emulate in software. If the threshold capability is high enough, the degradation that results from software emulation rather than hardware functioning will not be perceptible to the end user. In addition, if the standard is designed properly, it can accommodate an increasing threshold capability, which is what has been happening in the micro industry.

CP/M, while it was powerful in that it afforded a degree of portability, was weak in that its design did not allow for increasingly powerful machine capabilities. UNIX, on the other hand, while it also still has some serious shortcomings, does have the advantage that it was designed to be both portable and modifiable. It therefore not only is designed for machine independence, but can accommodate development modifications and enhancements without seriously affecting its original design. Enhancements to meet changing needs are therefore economically feasible. How much they are desirable, and where in the operating system architecture they should be allowed to be made, on the other hand, is a question that is currently under wide discussion.

Two competing priorities have emerged: the need to protect the current investment in software, and the need to allow for technological developments.

In particular, UNIX purists claim that the commercial development interests are motivating changes to UNIX that run counter to its original design and intent, while the commercial interests claim that these changes are necessary to make UNIX viable outside of the academic environment.

The point is that the design of UNIX allows for both interests to be satisfied. UNIX, like CP/M, has a layered architecture. There is the kernel, which is the core of the operating system; the shell, which is the command level interface; and the program, or applications, layer. The interests of the academic community and UNIX purists will be served if the only modifications made to the kernel are bug fixes and performance enhancements. And the interests of the commercial community can be satisfied by making the necessary enhancements, but doing so only at the OS/application interface level.

A mechanism that would systematically allow for the latter would be to add a fourth system layer: a virtual device interface, similar to the VDI that is being developed in the graphics standard arena. And, once again, although this would add another level of overhead, as the threshold machine capability increases, this overhead will become imperceptible to the end user.

A simple, yet effective, example of a virtual device interface that could incorporate the idea of a threshold capability is that of the standard output device. Both UNIX and MS/PC-DOS have a redirected I/O feature which allows input intended to come from the 'standard-in' device—the keyboard—to come from a file or another device instead, and output intended to go to the 'standard-out' device—the console CRT—to go to a file or another device instead.

The problem is that the standard devices are byte-stream oriented, and this is designed into the operating system. Thus, you can't have an application program take advantage of the cursor addressibility of most modern CRTs if the output is liable to be redirected to, say, the printer, which won't know what to do with the cursor addressing. If, instead, the output went through a virtual device interface, the interface could be made smart enough to handle such a situation—without the need to rewrite the operating system.

As Andy Hall explains in our interview with him, the above is a typical example of the kind of problem the UNIX development effort is dealing with. So, join with us as we travel through the recent evolution of UNIX and branch out to look at other areas of—The Expanding UNIverse.

The Evolution of UNIX —1974 to the Present

We begin our exploration of the period by examining the tools

by Ian Darwin and Geoff Collyer

Nobody needs to be told that UNIX is suddenly popular today. In this article we will show you a little of where UNIX was yesterday and has been over the past decade. And, without meaning in the least to minimize the incredible contributions of Ken Thompson and Dennis Ritchie, we will bring to light many of the others who worked on early UNIX versions, and try to show where some of the key ideas came from, and how they got into the UNIXes of today.

Our title says we are talking about UNIX evolution. Evolution? That means different things to different people. We use the term—in a way that might make the biologist in one of us blush—loosely, to describe the changes over time among the many different UNIX variants in use both inside and outside Bell Labs. Ideas, code, and useful programs seem to have made their way back and forth—like mutant genes—among all the many UNIXes that have been living in the phone company over the last decade.

Part 1 looks at some of the major components of the current UNIX system—the text formatting tools, the compilers and program development tools, and so on. Most of the work described in Part 1 took place at "Research"—a part of Bell Laboratories (now AT&T Bell Laboratories; then as now "the Labs") and the ancestral home of UNIX. In the next part, we look at some of the myriad versions of UNIX—there are far more than one might suspect. This includes a look at Columbus and USG UNIXes and the Berkeley UNIXes. Columbus, or CBUNIX, is the UNIX from Bell Labs at Columbus, Ohio. USG is the UNIX support group; we are loosely including PWB (Programmer's Workbench, and System II and System V, as USG UNIXes. Berkeley, of course, is the University of California at Berkeley. You'll begin to get a glimpse inside the history of the UNIXverse.

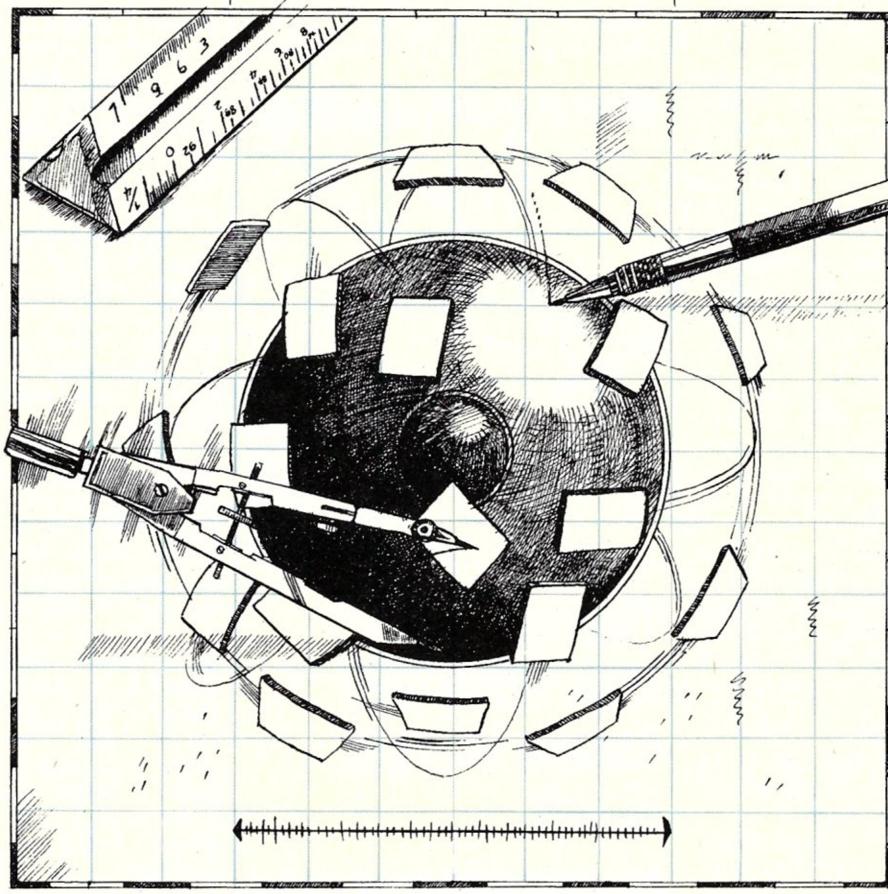
Basic sources

Since we can't say *everything* about UNIX in this article, we'll give some pointers for those who want to read more. Full acknowledgements will be found at the tail end of each installment. But first some basic sources must be mentioned.

It is a truism that the final source of

information about UNIX is UNIX itself. And this, of course, requires that you have a source license. And to get a source license, you must sign in blood that you will not divulge the source code of UNIX in any way, shape or form. So, in preparing this article, we have stayed clear of looking at source code. But there are times when you want to do so, not so much to find out how some feature evolved as to see how it really works.

The UNIX Manuals are a prime source of information about UNIX. Now, it's trendy to deride these manuals, but for their intended purpose and audience they are for the most part examples of good technical writing. The *Programmer's Manual* or *User's Manual,* as it is variously called (more colloquially known as "Volume 1"), summarizes in a standard format each command, system call, library function, and many special files (in the technical sense), as well as system file formats, games, miscellany, and maintenance information. Comparing a series of manuals of different vintages offers the student of UNIX evolution a good view on changing conditions.



Volume 2 of the Manual set is a series of short papers ranging from notes on installing the system through compiler reference manuals to introductory tutorials. These papers, too, are typically well written but occasionally incomplete. They are concise and to the point; some people find them obscure. But remember the audience and the background—the papers are written for the benefit of sophisticated computer users. It was always assumed that you would "be a wizard" or have somebody around to help you. Or you would go to the conferences and ask others about problems. A careful reading of the manuals was (and is) required to become a wizard, along with hands-on time spent *using* (and eventually modifying) the system, learning by doing.

These papers, and others written at Research, established an interesting tradition, so counter to mainstream computerdom: *You write the program, you write the documentation.* In almost every case, the authors of the program are the authors of the paper describing its details. And in almost every case, acknowledgement is made to those who contributed significant ideas, advice or moral support to the project. This, of course, has made our work in this paper easier. It also speaks volumes about management and about programmers—both those programmers who write effective summaries of their programs, and those who don't condescend to.

The UNIX manuals are sometimes derided for the "BUGS" section. This is the place where the author(s) of a program list its design limitations. One UNIX critic said of this policy: "If they know about the bugs, why don't they fix them?" The point is that the early UNIX authors established the beneficial habit of documenting limits to the program, rather than always letting the end user find them. Dennis Ritchie comments: "Every other manual has bugs sections; they just aren't published." Many of the BUGS sections were intended as pointers for further development of the programs, rather than as warnings to the user. Ritchie adds: "Our habit of trying to document bugs and limitations visibly was enormously useful to the system. As we put out each edition, the presence of these sections shamed us into fixing innumerable things rather than exhibiting them in public. I remember clearly adding or editing many of these sections, then saying to myself 'I can't write this,' and fixing the code instead." [Ritchie, personal correspondence].

After the manuals, another important series of papers in a similar vein appeared in the *Bell System Technical Journal*, July/August 1978. This special issue—Part 2 of the July/August 1978 issue—is often referred to as "the blue book" because of its blue binding. (In reality all issues of the *BSTJ* from this time period are blue, so the handle is a bit misleading.) The magazine is now called *Bell Labs Technical Journal* and is doing another special issue on UNIX that should come into print around the same time as this article. Watch for it!

Many of the technical reports from Research are published as Computer Science Technical Reports (CSTRs); those still in print are available from AT&T Bell Labs.

Brian Kernighan has co-written several books containing interesting historical details. We will quote later from *Software Tools*, a book he wrote with P. J. Plauger, and *The UNIX Programming Environment*, which he wrote with Rob Pike.

Finally, access to a nearly complete collection of back issues of *;login:*, the journal of the USENIX Association, has been invaluable.

Text processing tools

One of the guiding lights of the UNIX utilities or software tools has

Evolution

Continued from page 45

been the deeply felt conviction that text should be stored in as simple, as general a format as possible, so that any program can easily process it. This idea (it seems to have been present from the beginning) has had the widest impact possible on UNIX in all its varieties. In recent times, however, there has been a regrettable tendency to move away from it, especially among commercial software developers.

We have rather arbitrarily divided the software tools into text processing tools and program development tools. Remember that UNIX makes no distinction between text files, program files and data files. Many of the same techniques can be applied to all three. But more on this later. First, an outline of the major tools and their development.

An old editor made new. The standard UNIX text editor **ed** has a lineage longer than many of us do. As early as 1969, the first assembly language version of **ed** was in place. Although later rewritten in C, the editor is fundamentally the same program as used then. Kernighan and Plauger wrote in 1976:

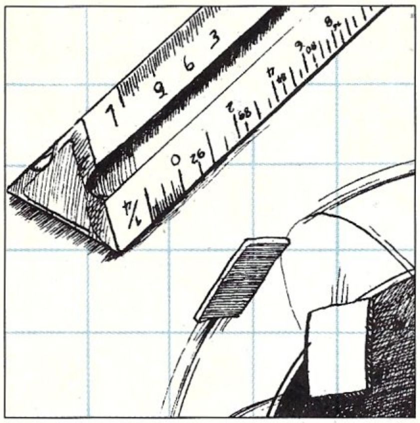
"The earliest traceable version of the editor presented here is TECO, written for the first PDP-1 timesharing system at MIT. It was subsequently implemented on the SDS-940 as QED, the "quick editor," by L. P. Deutsch and B. W. Lampson (see "An online editor," *CACM*, December, 1967). K. L. Thompson adapted QED for CTSS on the IBM 7090 at MIT, and later D. M. Ritchie wrote a version for the GE-635 (now HIS-6070) at Bell Labs.

"The latest version is **ed**, a simplified form of QED for the PDP-11, written by Ritchie and Thompson. Our editor closely resembles **ed**, at least in outward appearance." [*Software Tools*, page 217]. This is not to say that *ed* is the same as the TECO found on today's DEC computers; far from it. For one thing, TECO is character-oriented while *ed* is line-oriented. It seems rather a case of common ancestry.

During the 1970's, the editor went through countless revisions. Nearly every university had its own modified versions of **ed** and QED; some had several modified versions. Jay Michlin of Bell Labs wrote (in IBM assembler) a QED for IBM's mainframe TSO; this was released to Universities in the mid-70's. This was, in fact, one of my (Darwin) earlier exposures to the UNIX philosophy; around 1975, I heard about a "spiffy new editor" for TSO, so I ordered and installed it on the TSO system at the University of Toronto.

Did this wide variety of editor versions lead to massive confusion? Not really. For although most of the editors added new commands and features, they seldom deleted them. The result was that you could—and this is still true—learn a basic set of **ed** commands and special characters usable on every version.

Today the Seventh Edition, 4.xBSD and System III/V versions of **ed** are all sufficiently similar that one can move freely amongst them with only minor inconvenience. (Berkeley UNIX includes both the standard **ed**, and a different editor called **ex** and **edit**. This editor, which has common code with **vi**, has similarities to the standard editor, but is not close enough that one can freely move between it and normal UNIX **ed**.) The manual pages for every current version of **ed** are all recognizably derived from, say, the Sixth Edition document. System III/V extends the 'u' (undo) command, but most of the other commands are constant. If you've used **ed**, you've used an editor with a long history, and probably a long future.



The earliest UNIX formatter known to man is *roff*.

roff. Having a good text editor is only half the text-processing battle. Having entered your text, you still must format it neatly for presentation. That's the function of a text formatting program. The earliest UNIX formatter known to man is **roff**, a line-command formatter. Like **ed**, **roff** is part of a large and diverse family, one that includes the **runoff** package found on Digital Equipment computers (the latest release is called DSR, for DEC Standard Runoff). The earliest Runoff program is attributed by Kernighan and Plauger to J. Saltzer, who wrote it for CTSS. Runoff also is an ancestor of the Script programs available on IBM mainframe systems; that descent would be equally interesting for IBMers to trace (no doubt we'll get letters from those with information to SHARE with us).

roff was written by M. D. McIlroy at Research. Like **ed**, **roff** was well in place by the First Edition of UNIX. It was considered static by the time of the Sixth Edition and obsolescent by the Seventh, then was dropped altogether from System III.

nroff—the assembler of text. Computerists are never satisfied. So after **roff** came 'New Roff', or **nroff**, written by the late Joseph Ossanna, who throughout his career was concerned with improving the way text was handled. Ossanna's **nroff**, as Kernighan and Pike relate:

". . . was much more ambitious [than **roff**]. Rather than trying to provide every style of document that users might ever want, Ossanna made **nroff** programmable, so that many formatting tasks were handled by programming in the **nroff** language.

"When a small typesetter was acquired in 1973, **nroff** was extended to handle the multiple sizes and fonts and the richer character set that the typesetter provided. The new program was called **troff** (which by analogy to 'en-roff' is pronounced 'tee-roff'). **nroff** and **troff** are basically the same program . . ." with divergent processing appropriate to the differences in output device. [*UNIX Programming Environment*, page 289].

They point out that **troff** is tremendously flexible, and indeed many computer books have been typeset using it. But it can be complex to use. As a result, most everyone uses one or another "macro package"—a series of pre-programmed formatting commands—and optionally one of the pre-processors (such as **eqn**, **tbl**, and more recently **refer**, **pic** and **ideal**). **troff** was originally written in assembler, but was redone in C in 1975. Joseph Ossanna wrote both versions and maintained them until his death in 1977.

Macro packages. The earliest macro package to come into wide use was 'ms' for "manuscript." Written by Mike Lesk, the 'ms' macros provide a powerful but easy-to-learn (by comparison with bare **nroff**) approach to document formatting. The 'ms' macros were distributed with the Sixth and Seventh Edition UNIX and most subsequent releases. The package was modified at Berkeley, which also gave us the 'me' macro package.

The USG versions of UNIX include a macro package called 'mm' for "memorandum macros." These do most of the same things as 'ms', in slightly different ways, with the addition of numbered lists and a few other bells and whistles, but are about half again as big as 'ms'. The startup time with 'mm' is such that USG in 1979 had to resort to a compacted form of the macro packages; this found its way into System III.

There are two versions of the 'man' macro package used to format the manual pages in Volume 1 of the UNIX Manual Set. One was used up to V6, and the other from V7 on. If you see a manual page beginning with '.th' instead of '.TH', it's from V6. System III has an (undocumented) command **mancvt**, and 4.1BSD has **trman**, to convert manual pages from the old to the new format.

There is also the 'mv' macro set for producing viewgraph or slide presentations. This is a USG product, and versions of the USG manuals from PWB 1 up to just before System III carried the now-famous line: "The PWB/UXIX document entitled *PWB/UNIX View Graph and Slide Macros* is not yet available. *Viewgraph Macros* is in preparation." System III manuals appeared with scarcely a mention of 'mv', and it was first documented in the System V manuals.

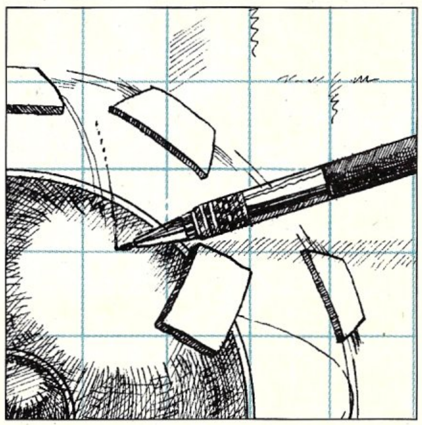
tbl, eqn. One view of **troff** is as an assembler language for text processing. If this be true, then **eqn** and **tbl** are the high-level compilers that go with it.

Mathematics has always been an inconvenience to traditional typesetting. This observation led Brian Kernighan and Lorinda Cherry to develop **eqn** for UNIX, and would later lead Donald Knuth to write his TeX typesetting package with math capabilities built in.

The **eqn** program reads an entire **nroff/troff** input file and passes it unchanged except for "equation specifications" delimited by .EQ and .EN requests. Material inside these requests is used to construct equations of considerable complexity from simple input. English words such as 'sum', 'x sub i', and 'infinity' produce the expected results (a large Sigma, x with a subscript i, and the infinity symbol respectively). In most cases no typesetter wizardry is required. A list of some 40 extra character definitions lives in the file **/usr/pub/eqnchar**; these can be copied, extended, or altered by the knowledgeable user.

eqn was written by Brian Kernighan and Lorinda Cherry. The "new graphic symbols" in **/usr/pub/ascii** are the work of Carmela L'Honmedieu (formerly Scrocca) at Bell Labs. The first public write-up of **eqn** appears to be a paper by Kernighan and Cherry in the *CACM*, March 1975. The software was included in the Sixth Edition UNIX.

Like **eqn**, **tbl** is a preprocessor that passes over a formatter input file looking for special requests (here .TS and .TE). The material between these requests is expected to be a series of special commands to **tbl** and some tabular



***troff* is tremendously flexible, but it can be complex to use.**

data. To greatly oversimplify how this program works, **tbl** replaces tab characters with explicit horizontal and vertical moves to make the rows and columns in the table align exactly under control of the table specification. It is invaluable for putting tabular material of any kind into documents.

Mike Lesk wrote **tbl** at Research; the idea for it came from an earlier table formatting program by J. F. Gimpel. **tbl** first appeared outside the Labs with the V6 release of UNIX. It appeared in its present form on the "Phototypesetter Version 7" (interim V7) PWB tape and in Seventh Edition UNIX distributions, and in all systems since then.

Lesk also wrote **refer**, a bibliography citation and reference package, which first appeared in V7.

Typesetter-independent troff. New! Improved! Yet again! That's right. **troff** is infinitely perfectable. In 1979, Brian Kernighan at Research set out to modify **troff**. Rather than rewrite it completely and be incompatible with the tens or hundreds of thousands of documents in existence, he chose to "clean up" **troff**. The task turned out to be rather more akin to cleaning the Augean Stables than he had imagined, but resolve did not desert Kernighan. Finally he emerged with a tape for the Device Independent **troff**, along with revised **tbl/eqn** and two new preprocessors, **pic** and Chris Van Wyk's **ideal**. **pic** (as the name implies) draws pictures. It is useful for drawing flowchart-like drawings, but there is much more to it than that. **ideal** also draws pictures, but is somewhat more mathematical in usage than is **pic**.

This set of products forms the basis of the commercialised "Documentor's Workbench" package from AT&T. And work continues, of course. Kernighan has been working on cleaning up the appearance of **eqn** output. Recently, Kernighan and John Bentley have written **grap**, a graph plotting preprocessor for **pic**. The program will not likely be released for some time, but a report on it should appear shortly as a CSTR.

Of Mice and Blits. Tired of typing at a dull, boring 24 x 80 screen, Rob Pike and Bart Locanthi had a better idea. Integrating the ideas of the Alto project and related work at Xerox PARC (Palo Alto Research Centre) with the UNIX approach to things, they built a special terminal called the Blit (not an acronym) with a 68000 processor, high-resolution screen, good keyboard, a mouse, *and* software shared between the host and the terminal. Their particular combination of these ingredients makes possible a form of interaction with the computer that is not yet understood by 99% of the people working in the computer field. Pike, in addition to being a radical advocate of the UNIX approach to software development, is quite visionary. Some of his work has been described at recent USENIX conferences. And most of the audience didn't seem to grasp the essentials of what he was saying. In 1969 Steve Munro described to me somebody with what he called "Unit Record Mentality," meaning somebody firmly attached to the (even then obsolescent) card punches, readers, and impact printers. By analogy with this, I could describe those who don't dig Pike's terminal interaction as being possessed of a "preinteractive mentality." But that would be premature.

True Blits are available only inside Bell Labs. The Blit has been commercialised by AT&T/Teletype, and is sold

Evolution

Continued from page 47

(with a different processor) as the 5620 terminal.

Style, Diction, Writer's Workbench. One of us (Darwin) has long been interested in the computerised processing of text, a term I take to mean more than is commonly included as "word processing." So I was quite interested to read a paper by L. E. McMahon, Lorinda L. Cherry and R. Morris entitled "Statistical Text Processing" in the 1978 special *BSTJ* issue on UNIX. I would later use several of the techniques mentioned in the paper.

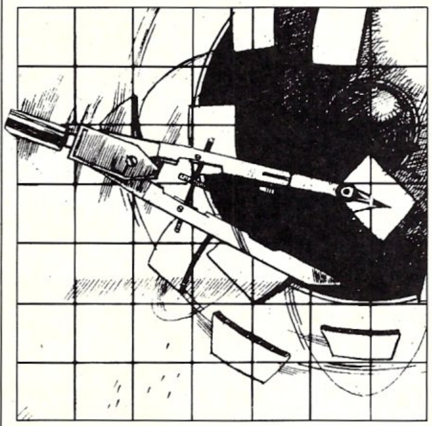
At the end of the paper, Ms. Cherry describes a program **parts** for finding parts of speech in English text. This was written to be the first pass of a system to add inflection to the **speak** program written by Doug McIlroy, but the person doing the stress part left the company. Ms. Cherry wasn't interested in the stress assignment, so she documented the work done so far and went on to other things.

In the spring of 1979, W. Vesterman of Rutgers approached Doug McIlroy at Research about computerizing one of the techniques Vesterman used in teaching writing. The students had to count surface features in their text and in a sample of text written by a professional writer. That summer, Ms. Cherry expanded **parts** considerably, and added the code that turned it into **style**, a program to analyse the readability and other characteristics of a textual document. She also developed **diction** to check for awkward word uses, overused words, and other problems facing everyone who composes text for others to read. In addition, she modified **deroff** to find the real text in a document. Vesterman consulted on this work.

And when the 4.1BSD release of the system came out, I was pleasantly surprised to see that **style** and **diction** were present. Bell Labs has a policy of sometimes releasing software to educational institutions; this probably explains the release at Berkeley.

While this was going on, the Human Factors group at Piscataway (now at Summit) was getting interested in automating document review, and Nina Macdonald of that group called Ms. Cherry about using **parts**. She had worked at Murray Hill in a Linguistics group and was familiar with the program. Ms. Macdonald took **style** and **diction**, and WWB evolved from there. Writer's Workbench (WWB) consists of **style, diction** and a dozen or so related programs for finding problems in written work. The "chattiness" level of the programs is set for the beginning user, but can easily be adjusted by the advanced user. The ideas for this work came from the Piscataway group, the Murray Hill group, and from Colorado State University, where extensive use of the Writer's Workbench (described at USENIX, Toronto, July 1983) currently puts several thousand undergraduates on WWB each year. The use of WWB is perceived to improve significantly the students' writing skills.

Many writers will be thankful to all who contributed, because these programs have proven themselves useful many times over. If buying a 4.1 or 4.2BSD system, insist on **style** and **diction**. If you get a System V UNIX, consider getting the WWB add-on if you'll be doing any document preparation. Writer's Workbench is one product that should survive and prosper as UNIX continues to evolve. The next major release of WWB (3.0) is scheduled for the spring of 1985.



awk is not at all awkward; it is a great simplification.

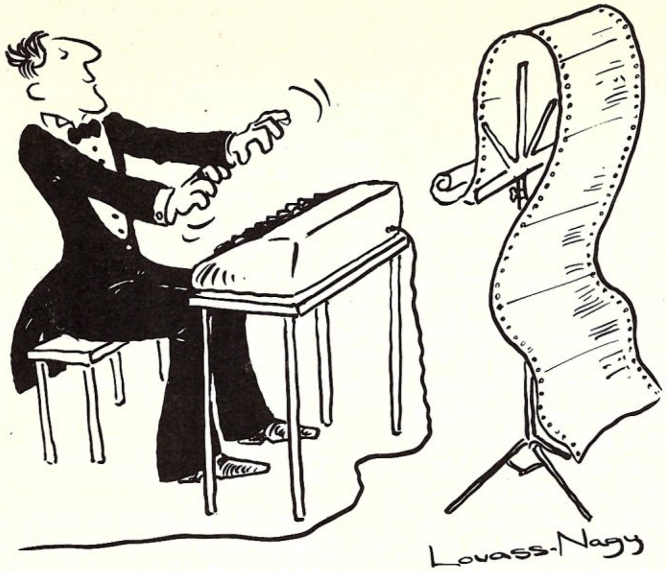
Compilers, languages, tools

What is an operating system without languages and utilities? Despite its limited support for Fortran, UNIX has always been known for the diversity of languages and tools that it provides. Some of these are well known; others are perhaps less well known than they ought to be.

The C programming language. The early evolution of the C language has been described elsewhere (see the reprint of Dennis Ritchie's paper in *Microsystems,* October, 1984. Dennis is rather modest, and doesn't tell you that the UNIX world has named the C compiler described there "the Ritchie compiler" (to distinguish it from other C translators). As we pick up the threads of the story, Fifth Edition UNIX has been in the field for some time. It is May, 1975, and the new improved Sixth Edition is about to be released. Ritchie has added some support for a new datatype, 'long integers' referred to with the keyword 'long', but this will not be documented. Not all the runtime support has been installed, and the tape goes out without it. Later Ken Thompson will announce that the support for 'longs', limited though it be, was there all along in V6. There is no support for 'short integers', or 'shorts'.

There follows a succession of releases of the C compiler. The PWB 1.0 release of UNIX, the first outside the Labs of a non-Research UNIX from Bell, goes out in 1977. And shortly thereafter a special-release tape known only as "Phototypesetter Version 7" includes a new release of **troff** as well as the C compiler, assembler, loader, archiver and bits of the C library including he first release of 'stdio'. These compilers seem to be from about the same vintage. Both compilers support another new datatype modifier, 'unsigned', which causes all bits of an integer to be treated as magnitude (on the PDP-11, for example, signed ints are from -32768 to +32767, while unsigned ints are from 0 to 65535). These compilers add typedefs, which allow you to generate your own names for existing datatypes, for a degree of independence from the machine datatypes. One of these compilers is somewhat buggy—the concept of 'cast' is in the code but doesn't work properly. Bit fields exist but are buggy; this is documented. The Phototypesetter Version 7 was primarily a release of **troff**; the compiler was included because it was necessary for **troff** (very convenient, since Research wanted to get the latest C out into the field anyway).

Finally the Seventh Edition of UNIX is released. Of course, it has another C compiler. This one, for the most part, is a "shaken down" version of the "Phototypesetter C" compiler. It is a lot more solid, although bit fields are still broken and now the bug is not documented. And there is a special kludge for **uucp** whereby casting an expression involving a character pointer to type unsigned treats the character referenced by the pointer as an unsigned character, a concept not yet in the compiler or lan-



Before Johann Sebastian Bach developed a new method of tuning, you had to change instruments practically every time you wanted to change keys. Very difficult.

Before Avocet introduced its family of cross-assemblers, developing micro-processor software was much the same. You needed a separate development system for practically every type of processor. Very difficult and very expensive.

But with Avocet's cross-assemblers, a single computer can develop software for virtually any microprocessor! Does that put us in a league with Bach? You decide.

The Well-Tempered Cross-Assembler

Development Tools That Work

Avocet cross-assemblers are fast, reliable and user-proven in over 3 years of actual use. Ask NASA, IBM, XEROX or the hundreds of other organizations that use them. Every time you see a new microprocessor-based product, there's a good chance it was developed with Avocet cross-assemblers.

Avocet cross-assemblers are easy to use. They run on any computer with CP/M* and process assembly language for the most popular microprocessor families.

5¼" disk formats available at no extra cost include Osborne, Xerox, H-P, IBM PC, Kaypro, North Star, Zenith, Televideo, Otrona, DEC.

Turn Your Computer Into A Complete Development System

Of course, there's more. Avocet has the tools you need from start to finish to enter, assemble and test your software and finally cast it in EPROM:

Text Editor VEDIT -- full-screen text editor by CompuView. Makes source code entry a snap. Full-screen text editing, plus TECO-like macro facility for repetitive tasks. Pre-configured for over 40 terminals and personal computers as well as in user-configurable form.

CP/M-80 version $150
CP/M-86 or MDOS version $195
(when ordered with any Avocet product)

EPROM Programmer -- Model 7128 EPROM Programmer by GTek programs most EPROMS without the need for personality modules. Self-contained power supply ... accepts ASCII commands and data from any computer through RS 232 serial interface. Cross-assembler hex object files can be down-loaded directly. Commands include verify and read, as well as partial programming.

PROM types supported: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 27C32, MCM8766, 2564, 2764, 27C64, 27128, 8748, 8741, 8749, 8742, 8751, 8755, plus Seeq and Xicor EEPROMS.

Avocet Cross-assembler	Target Microprocessor	CP/M-80 Version	CP/M-86 IBM PC, MSDOS** Versions
XASMZ80	Z-80		
XASM85	8085		
XASM05	6805		
XASM09	6809		
XASM18	1802		$250.00 each
XASM48	8048/8041		
XASM51	8051	$200.00 each	
XASM65	6502		
XASM68	6800/01		
XASMZ8	Z8		
XASMF8	F8/3870		$300.00 each
XASM400	COP400		
XASM75	NEC 7500	$500.00	
Coming soon: XASM68K...68000			

(Upgrade kits will be available for new PROM types as they are introduced.)

Programmer $429
Options include:
Software Driver Package -- enhanced features, no installation required.
CP/M-80 Version .... $ 75
IBM PC Version .... $ 95
RS 232 Cable .... $ 30
8748 family socket adaptor . . . $ 98
8751 family socket adaptor . . . $174
8755 family socket adaptor . . . $135

G7228 Programmer by GTek -- baud to 2400 ... superfast, adaptive programming algorithms ... programs 2764 in one minute.

Programmer $549

Ask us about Gang and PAL programmers.

HEXTRAN Universal HEX File Converter -- Converts to and from Intel, Motorola, MOS Technology, Mostek, RCA, Fairchild, Tektronix, Texas Instruments and Binary formats.

Converter, each version $250

Call Us

If you're thinking about development systems, call us for some straight talk. If we don't have what you need, we'll help you find out who does. If you like, we'll even talk about Bach.

CALL TOLL FREE 1-800-448-8500
(In the U.S. except Alaska and Hawaii)

VISA and Mastercard accepted. All popular disc formats now available -- please specify. Prices do not include shipping and handling -- call for exact quotes. OEM INQUIRIES INVITED.

*Trademark of Digital Research **Trademark of Microsoft

AVOCET SYSTEMS INC.™
DEPT. 1184-M
804 SOUTH STATE STREET
DOVER, DELAWARE 19901
302-734-0151 TELEX 467210

CIRCLE 201 ON READER SERVICE CARD

Evolution

Continued from page 48

guage. This will be quietly withdrawn later. The compiler has a bug in that it treats the right, not the left, side of an assignment as the value of the assignment expression. The V7 **stdio** exploits these two bugs, thereby making it nonportable. The semantics of casts will remain unsettled until slightly after V7.

Along with the Ritchie C compiler, V7 of UNIX includes the first release outside Bell Labs of a second C compiler, bearing the impressive name of "Portable C Compiler." Written by S. C. Johnson, this compiler has been in development since 1975 and uses the program development tool **yacc** but not **lex**. (A part of the **yacc** grammar for this compiler was published in the C manual with PWB in 1977.)

The portable C compiler turns out to be not as portable as desired, so a second version is developed over the next few years, called **pcc2.** At the ACM National Conference in 1983, Steve Johnson describes **pcc2** in some detail, and shows an example of its portability. Of the many "back ends" for it, one compiles a C language algorithm into the commands necessary to drive a VLSI fabrication process. So your program (if you work in the right part of the Labs!) can be compiled into a custom microprocessor, optimised to execute your program and nothing else! That sure outclasses the EPROM versions of Intel and Motorola microprocessors. **pcc2** has only recently been released; it is the C compiler for the Software Generation System.

One immediate beneficiary of the two-pass nature of **pcc** was the Fortran compiler, to which we will return shortly. But a second major fallout from **pcc** is a program called **lint**, which does partial compilation of C programs with much greater error checking. Like **pcc**, **lint** first appears with V7. We continue to recommend the use of **lint** to provide some reassurance of program correctness and portability.

Berkeley has taken the C language in some new directions. They have relaxed some restrictions on compiled programs. Most notably, variables can be almost any length and need not be unique in the first seven or eight characters. While this sounds handy, it is a major annoyance to the rest of the world, which has to change programs written with such "features" in order even to compile them. Berkeley programmers also tend to rely to an unprecedented extent on the 'asm' keyword, which allows you to interpolate assembler language code into the middle of the C program. 'asm' buys an increase in microefficiency, but only at the cost of a tremendous loss of portability. To preserve portability, the programmer should use #indef to include in the source code both an assembler version and a portable C version. But the latter is often omitted. A fine example was shown in a talk by Mike Tilson of Toronto's Human Computing Resources at the San Diego USENIX Conference in January, 1983. (See Tilson's article on page 84, based on his talk). Here's the code:

```
to = bp->b_ptr;
asm("movc3 r8,(r11),(r7)");
bp->b_ptr += put;
```

What it does is left as an exercise to the reader. The writer of this code left no clues as to how his mayhem works. As Mike says: "The variable 'to' is one of the registers used in this VAX assembly instruction. You guess which." Oh, we almost forgot. The three lines above are Copyright ©1980 by the Regents of the University of California.

Meanwhile, back at Research, B. Stroustrup has been busy adding "classes" to C. Classes (nothing to do with going back to school) are the interesting part of Simula 67. They provide for orderly interchange of data between modules, with no possibility of hidden de-

One reason for the success of UNIX is the consistency of the system in all its facets.

pendencies. A class consists of data (normally inaccessible from outside the class) and functions which are normally accessible from outside but which may be declared as inaccessible. One typically defines a class and publishes the names of the accessible functions. Functions outside the class cannot reference the data within that class except by calling the class's publicly accessible functions. This enforces modularity by hiding the details of a particular class's internals from other routines. Classes can be nested, of course, so you can develop such things as queues and stacks of objects. The C compiler encompassing all recent developments, including classes, declaration of function parameters for type checking, and other recent developments is given the name "C++". C programmers will recognize the pun; for others, it simply means "an incremented (augmented?) form of the C language, which retains the value of the old language." C++ has been in use for some time within the Labs, and may be cleared for external release soon.

In addition to the compiler, one needs a series of library routines to do Input/Output and some 'extra-linguistic' operations such as setexit( ) in V6 or longjmp( ) in V7. The first 'portable C library' was written by Mike Lesk and was implemented on the PDP-11, the IBM 370, and the Honeywell 6000 with the GCOS operating system. It set the style for subsequent development, and in Version 7 there was a "new portable I/O library" written by Ritchie. This has become known as 'stdio' (pronounced "stuh-DYE-oh") for the name of its header file, and is the I/O library distributed with all real UNIXes today.

The current C compilers for the PDP-11 continue to derive from the Ritchie versions. **pcc** for the PDP-11 never worked as well as the Ritchie compiler. Most other machines use **pcc**-based C translators, since the Ritchie compiler only works for PDP-11s. Many systems integrators wait earnestly for the release of **pcc2**, since porting **pcc** takes a non-trivial amount of work.

The future of the C language is not primarily in the hands of people like Dennis Ritchie and Steve Johnson and B. Stroustrup. Rather, it is in the hands of the ANSI C Language standards committee. But in the final sense, it is in the hands of programmers everywhere. This is partly because ANSI is a democratic agency, and any member of the committee has as much voice as a Dennis Ritchie or a Steve Johnson. It is also because C is a powerful language, and like all powerful tools it can be used or abused. This is not the place for a tutorial on C style, but the interested reader can refer to the article by Tilson cited above. Good use of C leads to rapid development of maintainable code; poor use of the language leads to code that looks like it was written in assembler. As we have seen, in a few cases it has been.

Fortran. Since UNIX comes from a Computer Science research background, it is perhaps natural that Fortran, that octogenarian, reptilian but ubiquitous language should be the object of some disdain among UNIXophiles. Indeed, the V6 *how to get started* document says that "no debugger is much help for Fortran." And the Sixth Edition manual set included the *C Reference* and the *C Tutorial*, but nothing on Fortran beyond the manual page for **fc** (1), a compiler for a slight variant of the ANSI Fortran-66 standard.

fc produced executable programs that used threaded code and floating-point instructions heavily; thus it ran slowly on machines without a floating-point processor, on which floating point had to be interpreted by the UNIX kernel.

The prime mover behind the next Fortran compiler was Stuart Feldman, who had been interested in compilers for some time. In 1976 he released a CSTR on "Fortlex—A General Purpose Lexical Analyzer for Fortran." This program reads a Fortran program and breaks it up into lexical tokens of the appropriate type. **fortlex** was used in the construction of various Fortran programming aids, such as a program to change all double-precision variables, functions and library calls to single precision. The paper also includes the **yacc** grammar for a Fortran scanner to be used with **fortlex**—not a complete compiler, but possibly a basis for one.

And the Fortran weakness of UNIX was remedied with a vengeance for the Seventh Edition. A compiler for the full ANSI Fortran-77 standard was included, the first implementation of

Evolution

Continued from page 51

the 1977 standard on any system anywhere, along with a paper detailing its use and implementation. One back end of this compiler was the same back end as the Portable C Compiler, so that it would be easy to adapt to new computers.

Although the Fortran compiler is part of all standard UNIX systems, most suppliers of 68000-based UNIX boxes do not include **f77**. Whether this is so they can charge extra for it, or because they couldn't figure out how to port it, is unclear. But commercial implementations are available for most micro-based UNIXes.

yacc+lex. One of the major tools used in compiler development is the **yacc** (yet another compiler compiler) program by Steve Johnson. When this program was developed in the early 1970's, compiler generators were being generated by many universities and other research institutes. As Kernighan and Pike remark, Johnson's choice of name for his program is ironic in that his has endured while most of the others have now been retired.

yacc reads in the specification of the syntax of a language and generates a program which parses that language. Note that this is not limited to "programming languages," but can be applied to any input that is structured. Many applications of **yacc** are mentioned in the **yacc** manual. **yacc** is also part of the **nrws** nroff-to-WordStar program used to translate some articles for *Microsystems.* The **yacc** manual mentions "compilers for C, APL, Pascal, RATFOR, etc. ..., a phototypesetter system, several desk calculators, a document retrieval system, and a Fortran debugging system" as programs that have been written in **yacc**. More recently, Cobol and Ada compilers have been constructed outside of the Labs.

But syntax analysis is only one part of compilation. Another part is lexical analysis, or scanning of the input to find certain kinds of tokens. For this, too, UNIX has an answer. The **lex** program by Mike Lesk and E. Schmidt provides this function. Since it is part of the UNIX tradition, of course, **lex** uses many of the same conventions. In particular, **lex** uses a variation on the notation for "regular expressions" as is used in the editors and elsewhere to describe the patterns to be looked for. If you've mastered commands like /[hH]e/ in the editor, you already know most of what you need to know to construct expressions for **lex**. And of course it works with **yacc**. The naming conventions of these two programs are such that both can be loaded together to form a working unit. Indeed many programs consist of **yacc** and **lex** outputs compiled and loaded together.

yacc was present in Version 6 (the manual page is dated late 1974); **lex** first appeared outside the labs in the PWB 1.0 release.

make. It's hard to imagine UNIX without the **make** utility. **make** is so taken for granted these days that the distribution of software in source form without a **makefile** is an event worthy of attention and inquiry. But there was a time when the name of the file with the instructions to build a system were chosen at random from the names 'build', 'rc', 'run', 'runfile,' and others. And these were shell files which built the entire component.

make builds a program or component from individual pieces, and recompiles only the minimum needed to rebuild it as changes are made. The edit-make-debug cycle is well known to UNIX programmers. Since the topic has been treated in detail in "The UNIX File" by one of us, we will not expand on it here. Suffice to say that **make** was written by Stuart Feldman at Research and first appeared outside Bell in the PWB release of the system. The barbarities of the Source Code Control System and the incompatability of this with most of UNIX including **make** led not

Classes provide for orderly interchange of data between modules.

to the correction of SCCS, but to an "enhanced" **make** that appears publicly in System III and System V.

ratfor, efl. Brian Kernighan at Research realized that Fortran would not go away, so he did something about it. He fixed it. He fixed it by adding the control structures of C and the definition and inclusion capabilities of the C preprocessor. The converter which takes in 'rationalised Fortran' and produces ugly conventional Fortran he called 'Ratfor.' Several versions were written; one in Fortran for bootstrapping onto other systems, another with

Evolution

Continued from page 52

yacc and **lex** as mentioned above. The meaning of Ratfor is best told in the book *Software Tools* co-written with P. J. Plauger. The source code for the programs in the book was made available on magnetic tape by Addison-Wesley in a move that was very foresighted for 1976. This led to the formation of the Software Tools User Group at Lawrence Livermore Labs in Berkeley; this group is still active and co-sponsors meetings with USENIX.

The *Software Tools* book would later be redone in Pascal (see 'Pascal' section). There are no plans announced for doing a "Software Tools in C" book; most of what you need is in Kernighan & Pike's book, *The UNIX Programming Environment.*

After the Fortran-77 compiler, Stuart Feldman turned his attention to Fortran extensions, and produced the **efl** language. This combines the control structures of Ratfor (which in turn derive from C) with the data-structuring capabilities of C, including the aggregates to group related data items, analagous to Pascal's **record** capability. **efl** is included in System III and some 4.xBSD systems. Some microcomputer ports (i.e., UniSoft) include **efl** even though they don't have a Fortran compiler.

awk. Aho, Weinberger, and Kernighan. The initials of three authors put together in the most pronouncable

True Blits are available only inside Bell Labs.

way. That's what they did when they couldn't think of a more imaginative name for a wonderful program they'd devised. A is Al Aho, of compiler book fame. W is Peter Weinberger of Research. And K is Brian Kernighan, just mentioned for his work on Ratfor. (Kernighan and Pike's book remarks that "Naming a language after its authors also shows a certain poverty of imagination" [page 131]). **awk** is not at all awkward; it is a great simplification. You can think of it as the combination of most of the best ideas of the other tools all rolled into one. We use it all the time. For some examples, see the review of "Leverage" in the August 1984 issue. You can enter the **awk** commands from the command line if they are simple enough, so that

```
awk -F: '{print $1}' /etc/passwd
```

is a complete program to print the names of all the accounts shown in **/etc/passwd**, the standard place for the names of all accounts on the system. By all means learn about **awk** and use it. It will make life far less awkward.

awk was first described in *Software Practice and Experience* in July 1978, and first distributed with Seventh Edition UNIX.

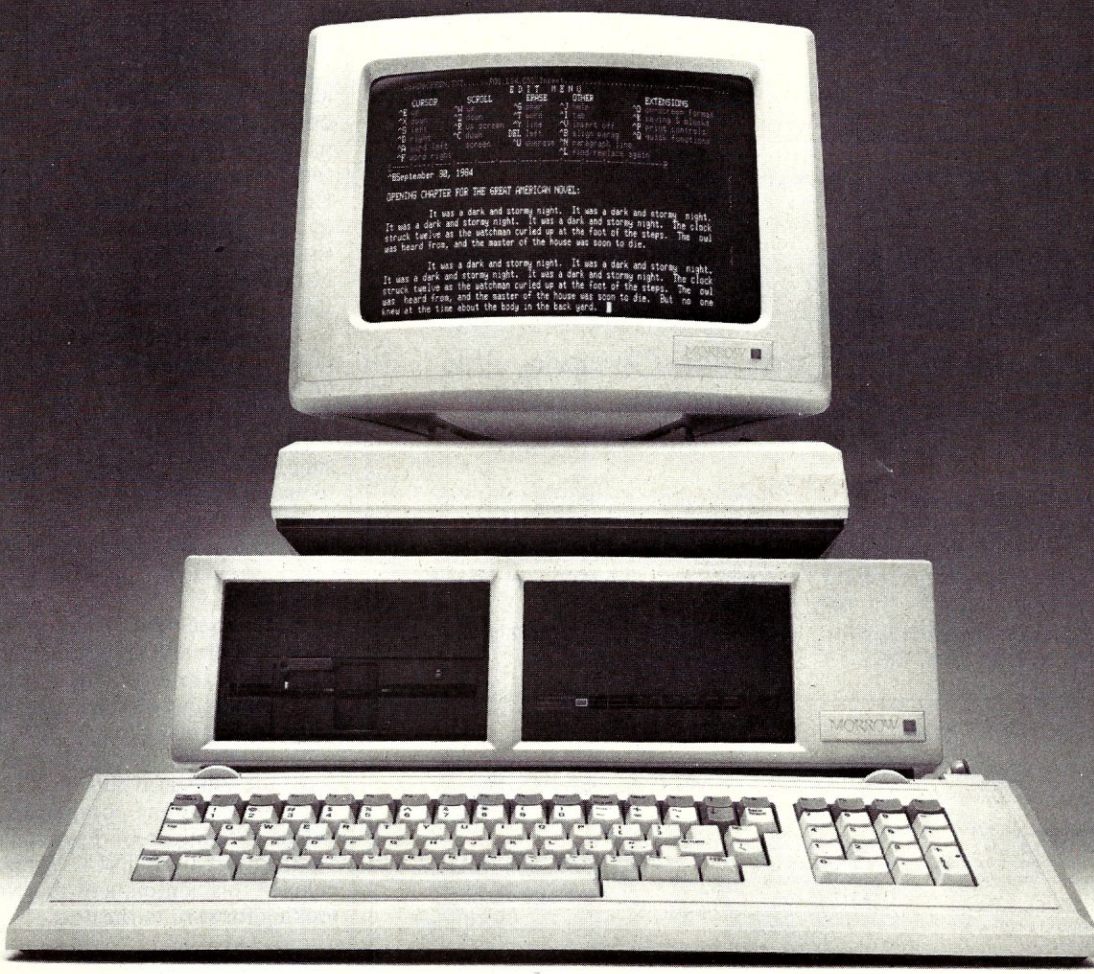
Pascal. Pascal did not catch on at Research. In 1981, Brian Kernighan did a paper published as a CSTR entitled "Why Pascal is not my Favorite Programming Language." The note was not based solely on introspection, for he and P. J. Plauger had just converted their book *Software Tools* into *Software Tools in Pascal,* including re-coding all the programs in Pascal. In the process they came to regard Pascal as their not favorite language.

Berkeley UNIX has included Pascal for a long time. Ken Thompson wrote the first version of Berkeley Pascal at Berkeley while working there as Visiting Mackay Lecturer in Computer Science in 1975/76. He spent the academic year at UC Berkeley, and taught several courses in Computer Science. He recalls: "When I arrived, the CS department shared an 11/45 with Statistics. It was 50-50 UNIX and RSTS. The first advance was an 11/70 dedicated to teaching. I put my first 155 [operating systems] course on it. Between the first and second quarters I wrote the Berkeley Pascal and talked Bob Fabry into using it on his 153 [data structures] class. It has been used for that ever since. By the time I left, there were several (2 or 3) 11/70s in the computing center providing UNIX service. CS had the 11/70 for teaching; they had almost completely taken over the stat 11/45 and there was a research 11/40" in an AI lab [Thompson, personal correspondence].

Pascal compilers can be had for most 68000-based UNIX boxes. These are available from commercial software firms and OEMs—see the annual UNIX software directory in the April *Microsystems.*

The S System. Finally, we cannot overlook an interesting "application" language from Research. *S: An Interac-*

Hard Disk Computer
$1999



■ If you thought you couldn't afford hard disk performance, think again. • The MD5-E hard disk Micro Decision computer with 128K RAM • Seagate 5¼″ Hard Disk with 5.4M bytes formatted (Second hard disk can be added) • 384K floppy disk backup • Superfast CP/M 3.0 operating system (compatible with most CP/M 2.2 software) • NewWord word processor • Correct-it spelling checker • New tilt & swivel monitor • Low profile keyboard • Morrow does it again. At $1999, this special introductory offer shatters the price barrier for hard disk computer systems ■

Call (800) 521-3493 (in California (415) 568-8599) for a dealer near you. Or write to Morrow, 600 McCormick Street, San Leandro, California 94577.

MORROW

Evolution

Continued from page 54

tive Environment for Data Analysis and Graphics, the title of the 1984 book by Richard A. Becker and John M. Chambers, puts it succinctly. We put it as follows: The S package is to conventional mainframe statistics packages as the UNIX shell is to batch Job Control languages. It provides interactive exploratory statistics, interactive and offline graphics, plus data modelling and time series manipulation.

The S language was developed at Research. The first public release was in 1981, which was for V7 and 32V. There were several interim releases; the next major release was in early 1984. This release was accompanied by a change in licensing and an order-of-magnitude cost increase for non-educational users, as part of the swing to the commercialisation of UNIX by AT&T Technologies.

The only remotely similar products that I know of in all of computerdom are APL and Speakeasy. APL was first implemented on IBM systems; at least one version for UNIX was developed. Speakeasy similarly arose on IBM hardware; a subset version called SpeakeC was developed at Purdue. Speakeasy was developed some time before S, but in quite different circles of influence. There appears to be no cross-pollination between the two, although many of the ideas are similar. S uses **yacc** to interpret its grammar; the **yacc** specification appeared in a *CACM* paper in 1984.

Interlude

The Computer Science Research Group, Centre 127, at Bell Labs has had an impact on computerdom far out of proportion to the number of people working there. A small group of talented people, started in motion by Ken Thompson and Dennis Ritchie with the original design of UNIX and Rudd Canaday's file system design, aided and abetted by those mentioned here and others, developed Research UNIX and its related tools. Many reasons are given for the success of UNIX, but one we'd like to add is the consistency of the system in all its facets. As a single example, the syntax used for pattern matching (a notation for the abstract concept of Regular Expressions) allows you to easily develop tremendous skill in pattern matching. This skill, once learned, can be applied in the editor to find text, in **awk** to find records to be acted on, in **lex** to specify partitioning of an input, (with simple modification) in shell commands to match filenames, and in a dozen or so contexts. This kind of consistency is a rare treat in any computer system; the extent to which it permeates UNIX is exemplary.

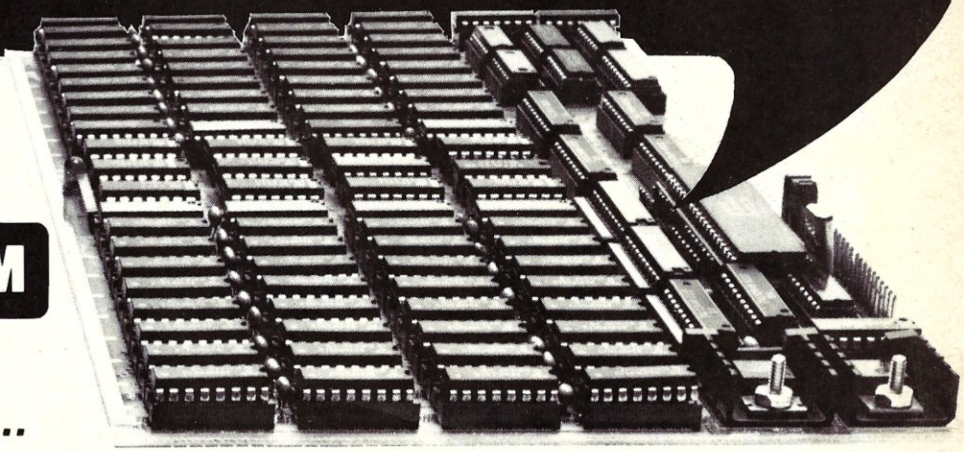
This concludes the first installment of our history of the natural creation of the UNIX timesharing system. The next installment will cover the relationships among many, many different versions of UNIX, most of them never released or publicised outside of Bell Labs and the telephone companies. There will be a "family tree" diagram illustrating the descent of UNIX. We hope you enjoyed this edition, and that you'll be looking forward to the next.

Acknowledgements

A myriad of UNIX experts helped with this paper. While it's not possible to thank everyone who offered to help, we should single out for special thanks Dennis Ritchie, who was there from the start and helped us to find it. We also got special help from Ken Thompson, who answered many questions about the past; Henry Spencer; and Laura Creighton for comments on the work in progress. Lorinda Cherry and Nina Macdonald helped by describing the evolution of Style, Diction and the Writer's Workbench. We as authors have final responsibility for the accuracy of the material presented here.

Ian Darwin, Box 603, Station F, Toronto ON, Canada M4Y 2L8

"We heard you have a memory board for ZENITH and CompuPro and Lomas and..."



UCI

EXPANDABLE RAM
FOR S-100 SYSTEMS

Look at these features...

- 256K up to 2MB of memory *without piggybacking* (9 bits wide)
- 8 bit or 16 bit systems (automatically selected)
- Operates with up to 8 MHz CPUs (with hidden refresh)
- On-board parity generation/detection
- Z-DOS Ram Drive Software
- Zenith System Memory and Ram Drive operational on same board
- Unique Switching Scheme — Zenith System can access entire 2MB of memory

Better yet, look at these amazing prices...!

256K	$ 795 list
512K	$1145 list
1MB	$2095 list
1.5MB	$3049 list
2MB	$3995 list

Also from UCI...

ZENITH 8 MHZ SPEED UP MODULE ... $49.95 list
Increase your system speed from 5MHz to 8 MHz

SIO-8 .... $449 list
Use for Centronic printers, IBM communication protocols, modems, terminals, etc. Occupies only 8 of available 256 I/O ports
- 8 serial channels • 2 parallel ports
- RS232

Call for our Dealer Pack and Compare!

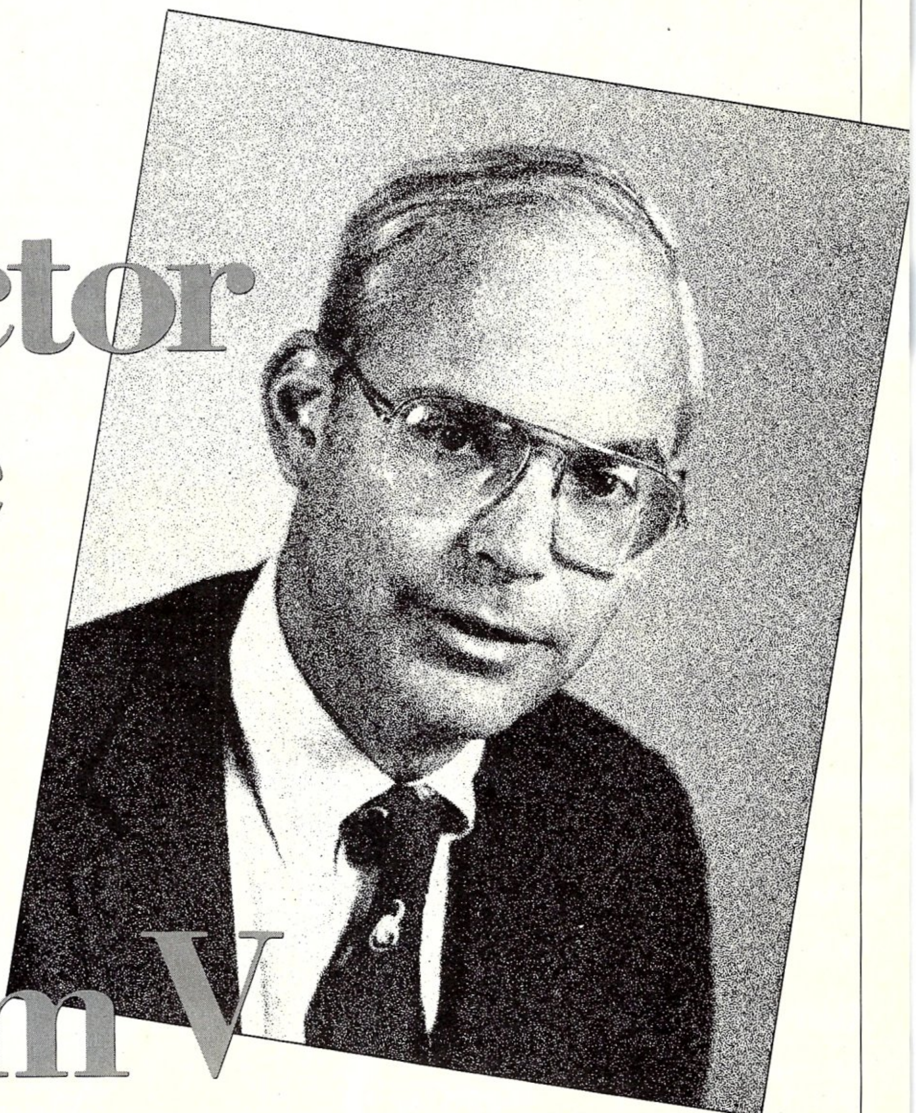
CALL TOLL FREE **1-800-824-2667**   IN OHIO CALL **216-673-5155**

UCI CORPORATION
an affiliate of ACME-CLEVELAND CORPORATION
948 CHERRY STREET
KENT, OHIO 44240

DISTRIBUTOR:
JOHN D. OWENS ASSOCIATES
12 SCHUBERT STREET
STATEN ISLAND, NEW YORK 10305
212-448-6298 212-448-6283
212-448-2913

CIRCLE 46 ON READER SERVICE CARD

A Conversation with Andrew D. Hall, Director of the Bell Labs System V Development



Transcript of an interview between Microsystems *(Mark, Chris) and Andy Hall, Director of the Bell Labs System V Development, at the AT&T Development Group's facility in Summit, NJ, on 22 August, 1984.*

Mark: At *Microsystems*, we support the type of standard environment created by the IBM PC. CP/M was powerful because it established a certain degree of portability for many programs. However, if you designed a large software project, one that was custom written for one client but designed well enough to be suitable for others, it could still easily require a good deal of modification effort before you could actually deliver it to another client. At least now, on the PC, if you write something for one client, there's a good chance you won't have to spend more than a minimal amount of effort to modify it for another. That's the kind of standard that we're talking about. It's also one of the primary reasons why we like UNIX and its philosophy, and the direction in which it's going.

I believe strongly that, very soon, end users won't be buying a particular vendor's machine just because of basic functionality issues. It's like the automobile: you don't buy one brand because it will take you further than another brand—they'll both get you where you want to go. You buy one over another because it's packaged a little nicer, or because it's the right company, or because of price/performance tradeoffs. Micros are going to get to the same place. Meanwhile, until they do, there are very definite basic functionality issues. The industry is still developing, and software vendors are concentrating on the end user market, because that's where the big dollars currently are, but are doing so in a rather short-sighted way. Many products are being developed quickly to get out into the market, but without all of the functionality that end users want, and without the capability for programmers to add that functionality. UNIX, with its portability, software development tools, and additional power that CP/M lacks, is a step in the right direction.

Andy: The analogy of the automobile has been pointed out to me before. You really don't care what the engine is, as long as it meets certain performance expectations. It's a fair analogy. The thing that's going to change the computer industry is that, like the automobile industry, if you want to be in the headlight business, you've got to make a headlight with a certain plug on it. UNIX has the potential for setting the standards for the plug-in market, in terms of both hardware and software. This is going to open up all kinds of opportunities—and markets—for third party vendors. Right now, third party software, for UNIX, is hindered by the lack of UNIX standardization. The standard needs to be defined to the extent that people are confident it will be maintained, so that they can have a viable interface.

Mark: In our interview with Dennis and Ken, they mentioned that they had not been directly supporting the development group since before version 7. This split between research and development is of historical interest, especially since Dennis's paper only covers the period through 1974. We'd like to know AT&T's intentions with respect to the operating system itself, as well as the issue of standardization.

There's a natural conflict between the commercial world saying they need features and the academic world saying they don't.

It's also interesting that, in the interview, Dennis and Ken said they developed UNIX because it was something *they* needed, and that it was just fine if it was developed however the development group and the end user community wanted. This is a real contrast to the complaints coming from the current core UNIX community that the encroachment of the commercial world into UNIX is somehow corrupting the philosophy of UNIX.

Andy: I think you probably sensed, from talking to Ken and Dennis, as well as other people, that UNIX gained in popularity through a sort of grass roots growth. AT&T made a serendipitous decision to release UNIX to Academia for a modest licensing fee, which got it into the universities. People used it and learned to like it, because they were in the same kind of environment for which Ken and Dennis developed it. So there was a natural synergy between our research efforts and the needs of the university laboratories doing research. The academics were suffering the same kind of productivity ills we had encountered; they, too, were expending large amounts of resources supporting vendor operating systems. And UNIX, despite the fact that it was unsupported, turned out to have many of the features that research people felt they needed in order to get their jobs done. It was easily modifiable, sufficiently reliable that they didn't have to expend a lot of resources on it, and sufficiently small that their people could learn it easily and modify it to suit their needs. So, it gained a lot of popularity in Academia.

Chris: Would you say that portability was another reason for its popularity in Academia, because of all the different kinds of machines being used?

Andy: Not in the early days. We'll come back to that, because it's an important part of the whole evolution. What happened is that people graduated from college and got into industry; there, they began to insist on having UNIX as their programming support environment—in other words, as a tool necessary to do their job. Then it started creeping into commercial products.

When UNIX hit the commercial marketplace, developers found that some other features were necessary before it could become viable. Things like file and record locking, real-time processing, a menu interface, and Apple/Lisa style displays. When you put UNIX into a desktop workstation, there are a number of deficiencies—it doesn't support database management really well. But, there's a natural conflict between the commercial world saying they need the features and the academic community saying they don't.

Mark: Do you think it's a question of the programmers saying, "Don't bother me with it," or is it something else, something more emotional?

Andy: It might just be system programmers. There's a psychology of programming that I don't think is very well understood, which leads to that. On the other hand, I think Ken and Dennis wouldn't exhibit that kind of emotion—I think they would say that changes have got to happen to take UNIX into

8086 and Z-8000 XASM includes Source Code Translators

	Z-80 CP/M®	ZILOG SYSTEM 8000 UNIX	IBM P.C. 8086/88 MSDOS	IBM P.C. 8086/88 CP/M 86	OLIVETTI M-20 PCOS
8086/88 ASM			$ 99.50	$ 99.50	
8086/88 XASM	$199.50	$750.00			$199.50
80186 XASM *new*	199.50	750.00	199.50	199.50	199.50
16000(all) XASM *new*	199.50	750.00	199.50	199.50	199.50
68000,08,10 XASM *new*	199.50	750.00	199.50	199.50	199.50
Z-8000™ ASM		750.00			299.50
Z-8000 XASM	199.50		199.50	199.50	
Z-80 ASM	49.50				
Z-80 XASM		500.00	99.50	99.50	99.50
Z-8 XASM	99.50	500.00	99.50	99.50	99.50
6301(CMOS) *new*	99.50	500.00	99.50	99.50	99.50
6500/11 XASM *new*	99.50	500.00	99.50	99.50	99.50
6502 XASM	99.50	500.00	99.50	99.50	99.50
65CO2(CMOS) XASM	99.50	500.00	99.50	99.50	99.50
6800,2,8 XASM	99.50	500.00	99.50	99.50	99.50
6801,03 XASM	99.50	500.00	99.50	99.50	99.50
6804 XASM *new*	99.50	500.00	99.50	99.50	99.50
6805 XASM	99.50	500.00	99.50	99.50	99.50
6809 XASM	99.50	500.00	99.50	99.50	99.50
8748 XASM	99.50	500.00	99.50	99.50	99.50
8051 XASM	99.50	500.00	99.50	99.50	99.50
8080 XASM	99.50	500.00	99.50	99.50	99.50
8085 XASM	99.50	500.00	99.50	99.50	99.50
8096 XASM *new*	199.50	750.00	199.50	199.50	199.50
1802 XASM	99.50	500.00	99.50	99.50	99.50
F8/3870 XASM	99.50	500.00	99.50	99.50	99.50
COPS400 XASM	99.50	500.00	99.50	99.50	99.50
NEC7500 XASM	99.50	500.00	99.50	99.50	99.50
NSC800	99.50	500.00	99.50	99.50	99.50
Subtotal	$ ______	$ ______	$ ______	$ ______	$ ______

Name ______________________

Company ______________________

Address ______________________

City __________ State ________ Zip ________

Phone ______________________ Ext. ________

Make and model of computer system ______________________

☐ C.O.D. *(2500AD pays C.O.D. charges)*

☐ VISA or MasterCard #, Exp. Date (mo./yr.)

____

Signature ______________________

TO ORDER. **Simply circle the product or products you want in the price columns above, enter the subtotal at the bottom of that column and add up your total order. Don't forget shipping/handling.**

Total $ ________

Check one:

☐ 8″ Single Density

☐ 5¼″ Osborne

☐ IBM P.C.

☐ Cartridge Tape

☐ Apple (Softcard)

☐ Kaypro DSDD

other formats available, please call!

shipping/handling ($6.50 per unit, $20.00 per unit for Int'l. airmail) $ ________

Total Order $ ________

CP/M is a registered trademark of Digital Research, Inc.

214

2500AD SOFTWARE INC.

P.O. Box 4957, Englewood, CO 80155, (303) 790-2588 TELEX 752659/AD

Andy Hall

Continued from page 59
the commercial marketplace.

Mark: They did say that real-time processing, for instance, went against the original intention of the time-sharing system. It could be done—it was just more work than they wanted to do. But if other people wanted to put it in, it was fine with them.

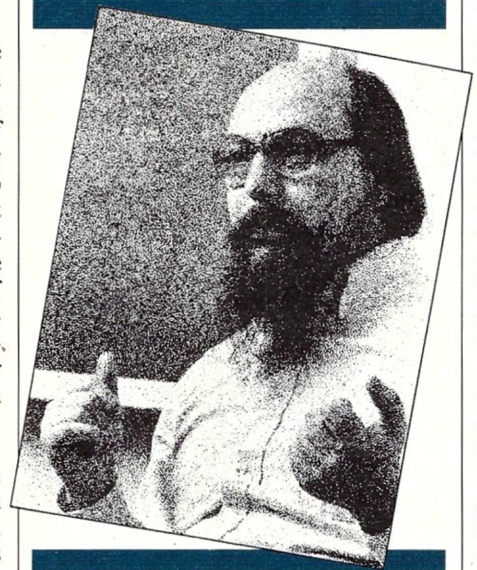
Andy: All they are really saying is, "We had some basic aims and objectives in creating UNIX, to meet our own needs. These were different from the needs of the commercial market place." It may well be that what has so far been done to meet the commercial needs—the first crack out of the box—isn't the best possible. It happens to meet the immediate needs, and from a business point of view, that's OK for now. But, we're being driven by natural forces in the market place to develop UNIX into a better commercial product, and AT&T as a whole is working quite hard to do that.

Mark: In terms of that development, what can you tell us about the intention for graphics standards, CRT handling, such as cursor addressing, and the kinds of problems that are involved with implementing independent devices into standards?

Andy: First of all, the concept of device independence, in the early days, was confined to devices that were byte-stream oriented: printers, TTY-like terminals, CRTs; all of them were very simple devices for which a simple stream model of the device would work. The concept of device independence was, therefore, fairly simple and articulate.

Since that time, various factors have all combined to allow us to build sophisticated devices that support complex graphics relatively inexpensively. If you want to get device independence back again, then you must virtualize those devices in order to present a general way of describing what happens on them. Below that level, depending on the particular output device, you've got to make compromises. For instance, you don't have colors but you have textures—how do you map colors to textures automatically? A lot of people are working on the problem of how to build an applications environment in which that kind of thing is fully supported.

Right now, what has happened with MS-DOS is that they push that responsibility out to the programmers. Even in UNIX, you'll find spooling programs that will take a file which is pure text, and will map it with the control sequences necessary to drive specific printers. UNIX provides a uniform way of writing byte streams to various devices, but has also pushed out to the applications level the responsibility of taking a text document containing italics, boldface, underlining, etc., and turning that into the specific control sequences to drive the printer. Typically, UNIX says, in the stored document file, "bold or italics go here;" then, there is a separate conversion program that maps the document to a specific output device. I think the same thing is going to happen when we start learning how to deal with bit-mapped displays, color displays, and so on.



UNIX gained in popularity through a grass-roots growth.

Mark: That's interesting, because eventually, we'll be treating all displays, as well as printers, as bit-mapped devices.

Andy: Technically, it's a difficult problem. It isn't as simple as the original concept of device independence, but I believe that the GKS standard, and the CORE standard, and some of the other graphics standards, are attempts to address these issues.

Mark: Is the development team here at AT&T also working on this, as part of the OS responsibility?

Andy: We are very interested in getting an applications environment defined for UNIX, so that people writing applications can have a uniform interface. But it's got to be part of the UNIX standard.

Mark: At what level? In the kernel, or at some interface level?

Andy: You're asking for implementation detail. I do believe that the separation between the kernel and the rest of the system is not relevant; the important thing is getting the standard there. Until you have that applications environment there, it's very difficult for people in the market place who are writing applications for UNIX to know what to do.

Chris: How far do the /usr/group standards conform to what AT&T feels is necessary?

Andy: Let's look at what /usr/group has done. They started with Version 7 to define what I would call a core standard—a minimum set of capabilities that has to be there for application developers to write software that would move from one vendor's UNIX system to another.

That standard doesn't address what we just talked about—the applications environment. The reason it doesn't is that they are taking what has already been done, and trying to standardize on that. Designing standards without having good examples to work from is not very successful. As a result, they haven't standardized on any applications environment, or on any menu interfaces, or that sort of thing. They have picked a smaller standard, and we are working quite closely with /usr/group to make sure that their standard is going in a direction that is consistent with AT&T, and to make sure that AT&T moves in a direction which supports the standard. We have a commitment to make our UNIX systems conform to whatever standard the /usr/group agrees on. It's important to the success of UNIX in the marketplace to get redefined standards, so that everybody who is developing software for UNIX can depend on their software running on different systems.

Chris: Where do you think the push for application environment standards is coming from? Mainly from you, or mainly from user groups?

Andy: There are really two user associations: there's USENIX, which represents the academic community, and /usr/group which really represents the commercial interests and end users and is responding to commercial needs.

Here is the problem. What we are responding to is the needs of the third-party and application people who are trying to develop applications for UNIX. CP/M, MS-DOS and PC-DOS already have their standards. If you write a PC-DOS application, you can depend on it running on any machine that runs PC-DOS.

Now UNIX, because it runs on so many different vendors' hardware, has the potential for each little instance of UNIX to be different in ways that could affect the livelihood of the applications software—you might have to make minor changes for every UNIX box you want to put your application on. That's a high cost, and /usr/group recognizes that. So their goal is to make sure that all UNIX systems, as they appear on different vendors' hardware, have a consistent interface to the applications. They don't want to have to invest a lot of hours rewriting an application for every different box—there's no leverage in that. And that, I think, is what is driving the standards, and the commercial world is what is driving the /usr/group.

Mark: There's also the problem of Version 7, System III, and System V. IBM is going after System III right now; that's what they are providing on the PC with PC/IX and Xenix, and even on the new AT, which is still System III, with Xenix.

Andy: I think too much is made of the differences between the versions. To the applications developer, the differences between System III and System V are relatively minor, and most of them are internal.

On the other hand, System V has major performance improvements over System III. Many of the internal algorithms to get performance out of the system are derived from larger minicomputers. It's interesting to know that earlier this year Gates committed the UNIX marketplace to bringing Xenix up to System V. The UNIX community is being drawn toward System V, and whoever is on Xenix is being moved in that direction, too.

Mark: And that means MS-DOS, too. To a user, MS-DOS looks much like UNIX now, with hierarchical directories, and as much device-independence as possible.

Andy: I think Microsoft has realized

Andy Hall

Continued from page 63

that they've got to get some kind of compatibility with UNIX in the market place, to take advantage of what the standardization efforts are going to do. It increases every software house's leverage to be part of that standard. The System III/System V issue is, I think, blown up out of proportion.

Mark: Is there anything more specific you can tell us about the differences?

Andy: An article on the Honeywell Scorpion, in *UNIX World* Vol. 1 No. 3 showed the differences in commands and tools diagrammatically. It looked as though System III was a subset of System V. In addition, System V has some major performance enhancements over System III.

Mark: What's the intention of the development group in terms of networking and talking to other non-UNIX machines? UNIX has uucp, but if I'm sitting on an MS-DOS machine, I can't talk to uucp. How do I get my files from PC-DOS or MS-DOS to UNIX and vice versa?

Andy: AT&T already has the UNIX bridge which allows you to connect an IBM PC to a 3B computer—it makes the 3B basically a remote file server for the PC.

Mark: Yes, but that's only for in-house where there's UNIX and a PC. How about remote communication over the telephone lines?

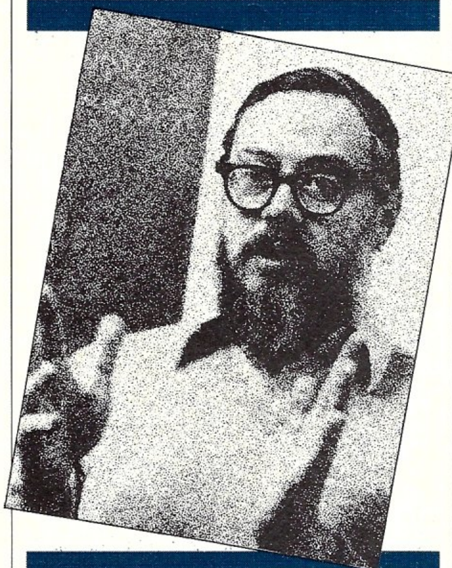
Andy: **Uucp** is totally outside the operating system. It's an applications program which happens to communicate UNIX-host to UNIX-host. But there are plenty of programs in the market place that will do the same thing between UNIX and a non-UNIX host.

The approach we're trying to take on networking and network architecture is to get to the open system architecture where third-party vendors can look at the architecture of UNIX and add their networking products to UNIX. So we are going to open up the operating system architecture and make it easy to add networking protocols and peripherals to the system.

But there are two kinds of networks to consider. If you are talking about small local area networks for point-to-point communication, there are plenty of UNIX networking packages available that will do the job. On the other hand, if you try to address a very big networking problem, along the lines of a corporation the size of an IBM or an AT&T, then the whole issue of network management comes into play; it's no longer a matter of simple point-to-point communication. Then, I think, we have in UNIX the same problem that we have with every other vendor—the available products really don't support that kind of network.

In AT&T we are addressing that

Changes have got to happen to take UNIX into the commercial marketplace.



problem, and are working quite hard to see what we can do within the UNIX architecture to make it feasible, though I don't think there is anything specific that I can talk about.

Networking is another area that depends on standards, but I think it would be a mistake to try to define a standard before having something in the marketplace about which people say, "Yes, that works." Networking has been a problem for twenty years and, to my mind, there is no good solution yet. But we certainly have to try to separate the two main issues, here. One is the question of media and transport; the other is the system architecture and how the applications see the network. One view says that the network should be independent of the transport. But that may be an overly simplistic view.

Mark: I'd like to bring up a question that is particularly important to *Microsystems*, and that's the question of whether UNIX can even run on micros. Ken and Dennis told us that the machine they first had was not very different from a micro in terms of power and resources. I think the micro industry is very confused on this question. For example, when AT&T released the 6300 PC, they didn't release it with UNIX; the rumour mill says that somebody at AT&T decided they couldn't put UNIX on the 6300—or couldn't do it fast enough.

Andy: To the question, "Can UNIX run on a micro?" I think the answer is unequivocally "Yes!"—and there are plenty of examples of it. But the real issue here is, "What do you mean by UNIX?" It's a packaging issue. If you define UNIX as a software development environment with its operating system, its 400 commands, its C and Pascal compilers and the whole gamut of software that people think of as comprising UNIX—then you're certainly not going to get that into a TRS-100 lap-top in one go, unless you add a substantial amount of disk resources. But yes, there is evidence to support the notion that you can take that software development environment and put it into a micro-based system with primary storage of 512K of RAM, a floppy disk, and a 10M or 30M hard disk. PC/IX is a good example.

But, the real issue is whether you can drive UNIX down into a lap-top portable or a workstation. And I believe the answer to that, too, is "Yes," if what you mean by UNIX is the operating system, the core piece. It all depends upon what market you are trying to support. If you are going into the nonprofessional, commercial marketplace, then we are told over and over again that the software tools (which take up the bulk of the space) have no relevance. In that market they want spreadsheets, word processors, graphics tools, and so on. So it becomes important to standardize on what the base operating system is to be and to standardize the application interfaces, so that those products become viable.

Mark: Another important issue is, to whom does the third party software vendor apply for licensing of UNIX on machines that don't yet have it. If I'm a software house with a client who insists on having HP machines, but the client's turnkey system is going to require UNIX for development and operation, do I talk to AT&T about licensing? or

Andy Hall

Continued from page 64

do I go to HP and get them to work out a deal to put UNIX on their machine?

Andy: I know this is an issue in the industry, and I know that the industry is looking closely at AT&T to see how they will do it. But I don't know the specifics, and I can't comment on it. One issue is the 'unbundling' of UNIX, and AT&T is certainly working with the industry to resolve it. It comes down to getting workable standards for the partitioning of UNIX—not a very difficult problem from a technical point of view.

Chris: Is this question of partitioning part of the standards that you and /usr/group are working on right now?

Andy: That's one of the questions, but it's not the major purpose of /usr/group. /usr/group only defines the the core piece that might run on a micro.

Mark: Can you tell us anything specific about what the development group is doing in terms of such things as file and record locking, and real-time processing?

Andy: File and record locking are two of the commercial requirements, and they were announced as part of the 3B2 product. That feature is being included in the licensing agreements. There are two levels under discussion, both of which are advisory, rather than mandatory. Both involve system calls in accordance with the /usr/group standard.

Real-time processing is a more difficult issue. Several vendors are offering real-time extensions to UNIX, but there are several holes to be filled. Setting process priorities and locking processes are capabilities that are already there. But if you mean guaranteeing a response to a task's request within a certain time, or guaranteeing allocation of system resources, then no, we don't have an agreed-upon way of doing real-time processing that is accepted by the industry. It's a problem that is being addressed.

The UNIX RTR is a real-time operating system with exactly that capability, and it does in fact make those guarantees. But that's a separate product, and not part of the UNIX standard.

Mark: The one thing I really don't like about UNIX is the normal shell interface. It just doesn't seem logical to say 'ls' to get a directory, and still less to say 'cat' when you want to type a file. But of

Too much is made of the differences between versions.

course, UNIX makes it *very* easy to both change the commands directly, and to provide your own shell that a novice user would find easy to use.

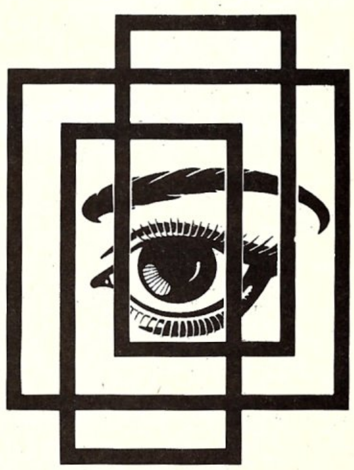
Andy: The shell is a classic case of a tool built by professionals for professionals. But the nonprofessional will be provided with an alternative shell that will give him more mnemonic command names, menu interfaces, etc. Dennis Ritchie's original paper specifically mentions alternative shells—it's one of the important features of the operating system that the shell and the user interface are *not* a built-in capability, but can be replaced. That's a key notion, and it's often forgotten. There are already systems on the market in which the shell has been replaced by a new shell that provides menu interfaces and a totally different feel, and yet have an escape for the UNIX aficionados who want to use the standard shell. That is what I consider a valid evolution of UNIX.

Mark: Going back to the question of computer size, it's my understanding that you couldn't run UNIX in less than 128K of memory.

Andy: That question of threshold is quickly becoming irrelevant. Technology is changing so fast that I would be hesitant even to guess at what the threshold is. On your desk today is the equivalent of a 7090 that filled a room in the 1960's. I don't see any end to that trend, so I'm not going to say what the threshold is in terms of the capability that you can cram into a lap-top computer.

Besides, I believe that the computer industry is driven by willingness to pay. When computers cost $1 million and up, corporations required approval at a very high level of the organization. When minicomputers came out, the level of approval was $250,000. And with the workstation at $4000 or so, a very low level of authority in the organization could approve it. I belive that price ranges are really driven by the people who make the decisions. The technology is such that when you find a willingness to pay $4000, then you're going to cram as much functionality as possible into that box. Look at what happened to pocket calculators: they started at $150 and went down to $50, $10, and now $5. I believe the same thing is going to happen with systems. When you talk of driving UNIX into 128K, it's a question of what market you are trying to penetrate and their willingness to pay. At each price level, you're going to find that the competition is going to be driven by technology, and how much functionality you can cram into that price range. I believe that soon you're going to find a full-function UNIX system, a software development environment, at the $500 level. The only question is when? Two years? Five years? I don't think it will be very long coming.

What I find exciting about UNIX is that there's a groundswell of acceptance. People who have used it are saying, "We've *got* to have it in our product." That feeling isn't restricted to one company, it's becoming general. And that's why I said earlier that the decision to license UNIX was serendipitous, because nobody could have predicted its success in the marketplace. The key was its portability. It was absolutely the right decision, because now every vendor sees a way to get a standard operating system without having to develop their own. And if it is really standard, the user gets access to all the software that has been developed for it, regardless of what machine he is running.



Codata 3300, Running Unisoft UNIX

An inexpensive UNIbox with a reliable implementation

by Bruce Hunter

I first met with Codata at the USENIX conference in Utah in June of this year, but I had read about them earlier. In the August 1983 issue of *Unique* (Volume 2, Number 12), David Fiedler claimed that Codata had the "most bang for the buck." Intrigued, I approached Jim Nakasuji, President of Codata, and Russell Cole, Codata's Director of Manufacturing, for a machine to review. They sent me a Codata 3300 with 1.25 meg of core and a 47-meg hard disk. The following is an extensive hardware and software review of this fine UNIX system.

Choosing a UNIX machine

Choosing a small UNIX machine is not a simple task. First of all, there are so many to choose from. Most "small" UNIX systems are 68000-based multibus units with Unisoft UNIX ports. Prices vary, but most fully implemented UNIX systems start at about $20,000 (non-OEM price). Codata is one of the few companies that offers UNIX machines at more affordable prices. The Codata 3300 is a remarkable machine that starts at an OEM price of $6,900 for a workable configuration of 320K of memory and 12 MB of Winchester storage, two serial ports and a megabyte of floppy storage.

Before you can accurately determine how much horsepower you want out of a UNIX computer, you need to analyze what you intend to do with it. My personal machines are used for both text processing and program development, with two to four users. On our Gifford MP/M 8-16 system (CompuPro), we have gotten by beautifully with 384K of memory and 22 meg of formatted hard disk, only getting in trouble when the modem was linked to the Amdahl at work while another terminal was engaged in CPU-intensive tasks, such as 16-bit program development. However, it is misleading to compare MP/M machines directly with UNIX machines. UNIX machines require more core and rotating memory. For full UNIX use, at least a half meg of core and 20+ meg formatted hard disk are needed. UNIX installations limited to either text work or program development could conceivably get by on 312K and 12 meg, unless database applications are involved. UNIX DBMs use a full meg of core, and the sky is the limit for hard disks.

Bringing up the Codata

Bringing up the Codata 3300 is an

extremely easy task. A bootstrap floppy loads the initiation phase of the start-up procedure. If the user wants to go directly into UNIX, he or she need only press return to get the unit into single-user mode. From there, striking Control D will bring the system into multiuser mode.

The initiation phase of the start-up procedure has an interesting feature. Normally the user boots the bootstrap floppy, and as soon as the first colon prompt appears, the user enters a return to go into UNIX. However, if any hard disk maintenance is required, it can be done *before* going into UNIX. Codata's bootstrap system allows for viewing the file system, hard disk formatting, tape booting, hard or soft disk booting, and other options before bringing up UNIX. The disks can be examined by logical disk number and block number, giving a granulation of 512 bytes. The ability to do system maintenance from the floppy disk allows a single, hard-disk system to be modified in any way, or even reconstructed from scratch. It also eliminates the need for a second hard disk. If you enjoy getting into heavy system reconstruction, Codata furnishes an optional stand-alone UNIX floppy for that purpose.

The Codata in the as-shipped condition is ready to go, a definite plus. Just take it out of the box, set your terminal for 9600 baud, connect the first terminal to the console port, and turn on the power to both units. Insert the bootstrap floppy after giving the hard disk a minute or so to come up to speed, then hit return after the first prompt. You are now in UNIX. If you don't want to do any super user work, hit Control D to get into multiuser mode and log in as *guest*. No parity is recognized or set, so whether odd, even, mark or space, you can't lose as long as the baud rate is 9600.

This is a good time to mention that bringing up most new systems is usually a horrendous chore unless you really enjoy working with hardware. There are baud rates to set, five flavors of parity to deal with, and making special RS-232 cables is the greatest nightmare of all. RS-232 cables require answers to questions like, "Which devices are DTEs (data terminal equipment)?" "Which devices are DCEs (data communications equipment)?" and "Will a null modem connection be required?" There are also special character delays and protocols to consider. One of the first things I look for before purchasing a new computer system is whether it is ready to run right out of the box. If it is,

Codata's chassis is reinforced by a heavy sheet-metal channel.

I know I'm going to save not only time and unnecessary hassles, I'm also going to benefit from the fact that the system has been burned in and tested. Both Codata and Gifford computer systems do this, and this feature adds considerably to the quality and value of their machines.

Once the Codata system is up and running, you're into straight UNIX. Codata uses a Unisoft 68000 port, a good UNIX port with many fine qualities. It is full AT&T V7 UNIX as outlined in the *UNIX Programmer's Manual* (Holt, Rinehart and Winston, 1983), except that FORTRAN is sold separately and RATFOR is missing. (Unbundling UNIX is a pet peeve of mine, but more on that later.) Unisoft also includes the more important Berkeley BSD enhancements. The system can be brought up in either C-Shell or Bourne Shell. It can look like V7 or BSD. The best thing about the Codata-Unisoft port is the extensive enhancements and bug fixes added by Yair Daon, Matt Perez, Gary Oates and others on the Codata staff. Whereas the original Unisoft UNIX port is a good, strong port in many ways, it is also quite fragile, and falls down easily. Codata's modifications keep it strongly on its feet.

Hardware

The supermicro is in a constant state of transition. The traditional S-100 computer had a horizontal box with 10 to 20 slots, and a power supply large enough for a welder. Today this kind of unit is being replaced either with smaller, less bulky, horizontal units such as the AT&T 3B2, or tall, slim, tower units, such as the Callan 300. The dual, 8" floppy drives of old are being replaced by a single, 5 1/4" minifloppy drive and a hard disk. Hard disks are not only more common nowadays, but are increasing in size from 10 to 12 meg on up to a mind-boggling 180 meg and beyond. Being aware of current technological developments like this helps you judge the quality of the hardware of a UNIX system. The most basic rule-of-thumb to remember is this: a hard disk is a necessity on multiuser UNIX systems.

Although the Codata 3300 looks like a traditional horizontal unit 22" long, 8" high, and 14" deep, it incorporates state-of-the-art technology inside and out. A single, 5 1/4" minifloppy drive is located on the right-hand side of the front panel. There are two indicator LEDs, one to indicate floppy disk activity and the other to indicate activity in the CPU. Part of the unit's rear panel houses eight DB-25 connectors for user terminals, printers, modems, and so on. This portion of the rear panel is fastened with a single screw, so it is easily removable. The DB-25 connectors have no more wires attached than necessary. The wires are tied neatly at one-inch intervals. On the fixed section of the rear panel, there are pin connectors for external disk drives and cartridge drives. The console even has its own separate DB-25 (RS-232) connector, and the chassis has provisions for external disk power, a printer, conventional tape drives, and auxilary I/O. The on/off and reset switches are also on the back panel.

Hardware buffs will be interested to know that the console port is a girl,

Codata

Continued from page 69

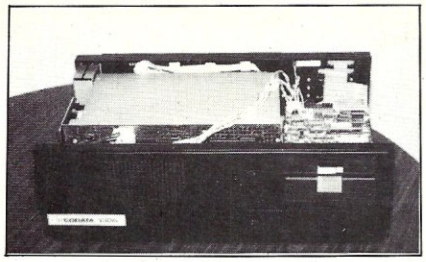
RxD (receive data) is asserted negative, and CTS (clear to send) and DSR (data set ready) are asserted positive. The unit runs with anything from three wires on pins 2, 3, and 7 to a full ribbon, all-pins-in cable. No special cables are needed as long as your console considers itself a DTE (TxD asserted).

At first glance, the Codata 3300 is not going to win any beauty contests. A plain-Jane black box, its tapered, sculptured sides appear to be plastic at first glance. However, a closer look reveals careful attention to detail and design. The housings are black plastic-coated sheet metal with a plastic bezel in the front holding the sheet metal faceplate. If you have ever owned a computer with a plastic housing, you are more than aware of the radiation effects the unit has on television. The Codata chassis is a much more effective shield.

I'm extremely impressed with the ruggedness of the Codata chassis, designed by Gary Oates. Like me, Gary was originally educated as a mechanical engineer, and this training certainly shows in the chassis design. The top of the housing is structurally reinforced by a heavy sheet metal channel to resist flexure. It has enough structural rigidity to support my weight, so those of you who like to stack additional equipment on the computer chassis can do so without fear. This is a fine indication of the quality of the Codata hardware, designed and built to last, inside and out.

Internally, the Codata is a precision machine, well designed and exquisitely built. The inside unit is dominated by an eight-slot, gold-anodized, aluminum card cage, with a cooling fan attached directly to the side of the card cage. Rather than the traditional vertical board arrangement, the card cage is laid horizontally so that airflow is directed across the boards without interruption. Air exits the cage to the rear and side (opposite the fan). The air exiting the end passes over the floppy and hard disk units before leaving the chassis through the slotted end panel, a clever design feature. Further attention to detail can be seen in the fine mesh screen fitted to the fan to remove particulates before they have a chance to coat the internals, particularly the boards.

The last four computers I have owned were S-100 mainframes. S-100 computers have very large power supplies, so when I saw the small power supply in the Codata, I was dumbfounded. The power supply is attached to the base near the front. All components are mounted on a heavy, black oxided bracket, which functions as a heat sink. The bracket is approximately 9″ x 4″ x 2″, and encompasses the entire power supply. I mention this in detail to emphasize the small size of the power supply, miniscule compared to the large power supply on my Gifford. The unit is fused for 5 amps, showing that it draws very little power, but my curiosity got the best of me. I instrumented the unit



Paged memory is used for very large programs.

for power consumption and temperature in order to give it a really thorough test. Starting up, the unit takes the most power, but the maximum power the Codata draws is a minuscule 1.8 amps for just a shade over 200 watts total. However, to be really sure that Codata's power supply was adequate, I wired the transformer and power supply bracket with thermocouples and monitored them throughout the day's operations. In a hot Los Angeles August, the hottest spot was the transformer, at a still-touchable and (electrically) cool 160°F. The power supply bracket was the next hottest item at 150°F. The balance of the unit ran very cool: it was quite touchable, and all readings were low.

After talking to Matt Perez about this intriguing power supply of theirs, I found out that it is a switching power supply, an interesting electrical feature. Switching power supplies differ from conventional power supplies in that they operate at high frequency rather than the conventional line frequency of 60 Hz. They do very little wave clipping (as do the conventional power supplies) and so run very efficiently. The main killer of power supplies is heat. When a power supply dies, it has a habit of taking all the boards with it. A switching power supply is more than adequate for all kinds of power use, and it is much more efficient than conventional power supplies, so it runs very cool. Again, this illustrates Codata's close attention to detail and innovative design. With heat the number one killer of computers, the Codata will live forever.

The technical details of the system's electronics are notable. Codata's hardware engineer, Matt Perez, has done an outstanding job of creating a machine that delivers a tremendous amount of power per dollar. Because the emerging new class of supermicro is constantly being compared to superminis (not exactly a fair comparison), supermicro designers can't afford to miss any tricks to gain throughput (speed). To maintain the fastest possible transfer speed from the CPU to memory, 256K of on-board memory is kept on the CPU card. This saves precious nanoseconds wasted in small transfers from the CPU card to the memory card by way of the bus. Their next magical trick is to have the main processor, the 68000, share as many of its duties as possible with other processors. Towards this end, Codata has added a separate processor for hard disk I/O and another intelligent processor on the I/O card.

Here's another Codata feature. Memory is furnished on cool-running, parity-checked dynamic 64K chips. Dynamic memory's main advantage is low power consumption. With the entire Codata machine drawing just a little over 200 watts, the entire machine can be protected by a very inexpensive UPS (uninterruptable power supply). A secondary benefit of dynamic memory is its lower initial cost, making memory expansion much easier on the pocketbook. Memory is paged (segment-over-page, a method of dual-memory granulation), and therefore very large programs can be run on the 3300 with minimal real memory. If you have not used paged memory yet, be aware that paged memory is slow. Use it as an extension to main memory, not as a substitute.

General impressions

The Codata 3300 is a very pleasant machine to use. The hardware is quiet. The only sound during hard-disk operation is made by the drone of the single cooling fan, with occasional percussive buzzes from the floppy disk unit for variety. The 68000 (MC68000L8), working in cooperation with the Codata system architecture, provides fast operation. In spite of UNIX V7's insistance on running to the hard disk for every command called, the system feels at least twice as fast as my Gifford 8-16 system. This is as it should be, since the registers and bus are twice as wide as the CompuPro's 8/16 architecture (32-bit processors for multiuser, supermicro systems represent the state-of-the-art for the immediate future). 256K of memory is resident in the CPU card, so there are no wait states in the processor's operation. Thus, full advantage is taken of the processor. Standard commands for listing, changing directories, and other pedestrian tasks are performed with blazing speed, almost as fast as in mainframe environments. No figures were available on throughput, but it's probably somewhere in the range of 1 MIPS, or in the same order of magnitude as a VAX 11/750.

CPU-intensive tasks like **find**, **man** and **nroff** are a good judge of performance because they really tax small UNIX systems. **nroff** works at 250 lines per minute, which is no speed record but not at all bad. This is no fault of the Codata—when formatting under UNIX Version 7, searching is CPU-intensive because it is not hashed. **find** works as fast as any machine in its class (although I didn't have time to heavily load the directories), and **man**, the online manual retriever, retrieves a manual page in 20 seconds. (An Amdahl does it in about 5 seconds, and a VAX 11/780 in 10 seconds, but Amdahl prices start at $2,000,000, and a VAX runs about $200,000.) A trivial **cc** compilation takes 20 seconds total for both compilation and linkage, and **yacc** compiles a small interpreter in approximately 4 seconds.

Testing the system with two users shows no system degradation whatsoever, but testing the system with at least six users is the place to start looking for potential system degradation. Superminis like Data Generals and VAXes generally show noticeable degradation at about eight users.

In general, the system software performs well. This article was written on the Codata in **vi** and formatted in the **ms** macro of **nroff**. Most **vi** commands work well, but a few do not. When I tested the machine, Control F would not scroll the screen, and the up arrow would not move the cursor up through the text. This is probably due to an error in the **termcap** entry for the Adds Viewpoint terminal. **nroff** and **ms** work well with no problems. **spell** also works like a charm, but the Unisoft port would certainly benefit from a program like Amdahl UTS's **mspell**, which allows misspelled words to be corrected where they lie.

I found the Bourne Shell to be a little weak in handling script errors. I suspect that Unisoft put much more effort into making C-Shell robust, and the Bourne Shell development took second priority. This is unfortunate because commercial UNIX system programmers use the Bourne Shell and accumulate a veritable encyclopedia of Bourne Shell scripts to maintain their systems.

It's unusual that /usr/bin is used to hold a large number of system commands that normally are stored in /bin. I like to reserve /usr/bin for 'homegrown' system commands, and leave all OEM-supplied commands in /bin. This method allows /usr/bin to be copied on nightly incremental backups. Meanwhile, /bin can be restored from the distribution storage media (tape or disk).

Finding SCCS (the source code control system) as part of the standard distribution was a pleasant surprise. SCCS is an invaluable tool for maintaining tight control for program or document source. It allows a copy of a controlled document to be taken from a locked file, but it does not allow a copy to be put back. As a result, documents or program source code cannot be modified without authorization. On multiple programmer projects, SCCS is invaluable.

The floppy disk has a capacity of 1 MB, a distinct help in a system with 10 to 80 MB on a single Winchester, because it stores more. When a UNIX system comes up to full loading, a tape of some sort is a necessity because there's so much more to store! We managed to get through a year and half of operation on our Gifford system with floppy disk backups until the day our Atassi hard disk drive went to Valhalla. If we had installed a tape backup system, we wouldn't have lost as much data.

The Unisoft UNIX port

Before going into the specific strengths and weaknesses of the Unisoft UNIX port, bear with me while I examine the strengths and weaknesses of AT&T UNIX implementations issued since the divestiture. In order to appreciate the Unisoft port for what it is today, it is necessary to lay the groundwork with some historical perspective.

A brief history of current UNIX implementations

UNIX Version 7

Whereas UNIX was created 15 years ago, commercial, AT&T-supported UNIX is less than a year old. Its roots go all the way back to Multics. UNIX actually falls into two time periods, BD (before divestiture) and AD (after divestiture). Before the divestiture of AT&T, UNIX was not a commercially viable operating system. Sheltered within the more tolerant, protective confines of Bell Labs and colleges and universities, UNIX did not have to withstand the immense rigors of commercial environments. It's a good thing, too, because it wasn't ready. UNIX Version 7 was the first official version after the divestiture, and it had

Codata

Continued from page 71

several limitations.

Although V7 incorporated the brilliant operating system design that makes UNIX unique and fascinating and was chock full of exciting tools, commands, and utilities, it had some problems. The problems were not what it lacked, but what it had in abundance in certain areas—bugs. Commonly used commands were just about bug-free, but others, like program-development tools, lacked sufficient error reporting and strength. On encountering an error, a robust programming utility stops and reports its location and the nature of the problem. A weak utility falls down, and in the case of most AT&T V7 ports (and consequently Unisoft V7 ports as well), when the utility fails, the system does a core dump. A core dump is of limited usefulness when you're debugging a program written in C, but no help at all if you're using a Shell or **awk** program.

UNIX System III

System III is the successor to UNIX V7. Its main claim to fame was the incorporation of numerous bug fixes, and a few minimal improvements to the system. In fact, System III was so unremarkable, most porting houses did not even bother offering it. UNIX porting houses supplying direct ports (like Amdahl, which offers an utterly superb mainframe version of UNIX called UTS) issued their own "new and improved" releases. In the last year, Amdahl's releases consisted of additions from UNIX System III, a number of Berkeley BSD 4.2 additions, and bug fixes. UNIX System III received little fanfare because there was nothing special enough to make noise about.

UNIX System V

The most recent addition to the UNIX family is System V. AT&T is very aware of its competition. In spite of all the space in the news media given to PC-DOS, micro operating systems are no competition to UNIX at all in the long run. In fact, PC-DOS is becoming more and more UNIX-like as time goes on. The operating systems AT&T must directly compete against are large, polished and robust, and they are hardly new guys on the block. I'm referring to mini and mainframe operating systems. DEC's VMS/OS and Data General's AOS/VS have a strong following, and their followers are resistant to being displaced by UNIX. IBM has many excellent operating systems, and the VM family of operating systems (MVS, CMS/VM, CP/VM, etc.) are the most formidable competition of all.

System V is AT&T's answer to this major-league competition. The design goals were speed and strength. Version 7 and System III had two major faults: all the commands are disk-resident, and searching algorithms are weak. (Disk-resident commands require the system to do a disk access at each invocation. Slow searching algorithms cause major utilities like **nroff**, which was used to write this article, **find** and **man** to spend a full minute to format two pages.) System V puts the most frequently called commands into high-speed memory as RSPs, resident system processes. [Note: The term RSP is used by Gifford Computer Systems, a VAR for CompuPro. Their MP/M 8-16 system, definitely a direct competitor of small UNIX systems, uses 90K of high-speed memory to house the system as its resident system process. Their system's responsiveness to this technique is substantial.]

Utilities requiring intensive searching such as **nroff** and **spell** now use hashing for exponential increases in speed. System V also has numerous bug fixes to make the system crash-resistant. AT&T failed to accomplish all that they wanted to in the first release of System V, so release 2 has additional improvements. As of this writing, UNIX System 5.2 is the latest version.

The 256K CPU does pedestrian tasks with blazing speed.

Unfortunately, AT&T may release a new version of UNIX, but the changes aren't automatically made on the Winchester drives of UNIX system owners. First of all, primary system porting houses (Intel, Motorola, Digital Research, Unisoft, Interactive Systems, etc.) must be willing to go ahead and make the changes for their particular processor (AT&T writes theirs for the 3B20s and VAX 11/780s only.) If they don't, the OEM is stuck with attempting the job. Some OEMs—like Codata, Amdahl, and Callan—continually provide numerous changes and updates. When purchasing a UNIX system, this is definitely something to consider.

Berkeley UNIX—BSD 4.2

Bell Labs not only created UNIX,

Codata

Continued from page 72

but contributed the majority of utilities and enhancements since UNIX's creation. However, there also have been many extremely significant contributions by people outside of Bell Labs. The most well-known contributions were made by some talented individuals at University of California at Berkeley. Some of Berkeley's more notable contributions to UNIX are **termcap** (a method of configuring terminals to respond to CRT-oriented software), **vi** (the visual editor), network communications, and helpful utilities like **more**, **head**, **tail** and many others.

The Berkeley contributions to UNIX are extremely popular, with good reason. They are significant enhancements to the UNIX system. A few OEMs go so far as to offer BSD 4.2 exclusively on their systems. More far-sighted porting houses like Unisoft perceived that standard UNIX was the mainstream staple to which the Berkeley enhancements should be added. The end result was a comprehensive UNIX port incorporating the best of both worlds. Unisoft doesn't call themselves "The Berkeley Port Authority" for nothing.

Unisoft entered the UNIX porting business in 1981, and they now have somewhere around 80 UNIX ports under their belts. Their primary emphasis

A hard disk is a necessity on multiuser UNIX systems.

was the porting of UNIX to the Motorola MC68000 family of processors. Having covered the development of UNIX up to the present, now let's take a in-depth look at the Unisoft UNIX port.

Unisoft System 7: strengths and weaknesses

On the whole, Unisoft has done an excellent job of assembling the standard AT&T UNIX distribution together with BSD 4.2 and SCCS, and porting it all to the 68000. It is one of the most powerful UNIX implementations for the money in terms of potential software horsepower. The inclusion of UNIX with Berkeley enhancements parallels the philosophy of Amdahl, whose excellent UTS UNIX port is far more comprehensive and polished than any other UNIX port I've examined to date. Unisoft has taken a similar philosophical approach to the porting of UNIX, cleverly including most of UNIX (outside of Fortran and RATFOR), and also incorporating the fine Berkeley enhancements.

I applaud Unisoft for taking this approach. Competitors of Amdahl and Unisoft, including AT&T, Interactive Systems, and Microsoft (Xenix), will have to come to grips with the fact that a fully implemented, bundled UNIX operating system with Berkeley enhancements is the minimum UNIX implementation necessary to compete effectively today. Supplying just the kernel is no bargain.

We in the industry should resist the efforts of some porting houses that want to unbundle UNIX and offer certain UNIX utilities and traditional UNIX languages as 'options.' On the other hand, traditional UNIX options are WWB (Writer's Work Bench), PWB (Programmer's Work Bench, including SCCS), and Documentor's Work Bench, the most recent arrival. These fine bundled packages have always been offered as options, and it's only fair that they remain so to bring their creators additional revenue.

Unfortunately, the Unisoft UNIX port has an Achilles heel, going back to the weaknesses in AT&T's UNIX V7 port. Unisoft's implementation has a way to go before it is sufficiently robust to survive in UNIX commercial environments. The performance of the programming tools and some of the more important filters such as **awk** and the Bourne Shell is flawed. Error reporting is a particularly weak area, and a core dump is performed at the drop of a hat. Let me emphasize that this is not limited to the Codata system, but to many other Unisoft ports. Larger OEMs with the luxury of a system programming staff, such as Codata, have put enough bandaids on the Unisoft UNIX port to cover the weak areas.

Bugs and weaknesses like those outlined above are not the exclusive domain of Unisoft ports. It is not widely known, but the Fortran compiler under AT&T's UNIX Version 7 had more bugs than a Sunday picnic. These bugs are simply a manifestion of what happens when UNIX is thrust into the hustle and bustle of commercial computer environments. Heretofore hidden bugs, warts, and gremlins will surface eventually in heavy commercial use. Large houses like Amdahl release new and improved UTS UNIX versions every few months. UNIX is still in a state of transition from a relatively fragile laboratory creation to a bulletproofed commercial operating system.

The design goals of UNIX System V were speed and strength.

Aside from the above weaknesses, the Unisoft port is very good. The day-to-day commands like **ls**, **cd**, **mkdir**, **cp**, **mv**, **pwd**, **rmdir**, and **rm** are solid as a rock. The text processing commands (such as **ed**, **ex**, **vi**, **nroff**, **troff**, to name a few) are a little buggy, but certainly strong enough in a pinch. In every other respect, the Unisoft port does very well.

Codata's software staff under Yair Daon has done a fine job of strengthening the Unisoft System 7 port to make it livable. I have talked to several other OEMs utilizing Unisoft UNIX ports, and they have had to do the same. The Codata/Unisoft port has been christened *Unisis*; the most notable Codata additions are the numerous routines added to bring up the system effortlessly and painlessly, and some excellent self-help features. The Codata system is capable of being both auto-configured and auto-booted.

Great efforts have been made to make the Codata system as maintenance-free as possible. Standard UNIX systems are time-intensive because they require knowledge of UNIX and skill administering UNIX. So far, very few OEMs have been able to come to grips with making system maintenance manageable enough for the system owners and operators not familiar with UNIX

Codata

Continued from page 75

system administration.

Codata has risen to the occasion. Rumors of future developments along the line of self-maintenance and automatic crash recovery are just a hint of some of the enhancements Codata has in store for release early next year. I am looking forward to reviewing Codata's 1985 machine. From what I've heard, most of the new features are in the "you are not going to believe this" category. [Unisoft has recently joined Digital Research in porting UNIX System V to the Intel 80286 processor.]

User support

When buying any system or software, the primary consideration after function should be support. If the vendor that sold you the system or software is not going to support it, you do not have very much. In many cases, the only deterrent to piracy is support alone. The vendor must be both willing to and able to support your package. My dealings with Codata showed that they were both willing and able to support the system. Several knowledgeable people at Codata repeatedly gave me patient, lengthy, online support.

Unisoft does *not* support their UNIX port directly. They leave the support of the system(s) to the OEM who manufactured the equipment. Because of hardware dependencies, this is probably a logical arrangement, except that frequently the OEM leaves the support of the system's software to the resaler. There are very few resalers knowledgeable enough to support a fully implemented UNIX system, so the end result is UNIX systems that are virtually unsupported. It behooves the buyer of any UNIX computer system to find out in advance how well he or she will be supported after he has purchased the system. Codata directly supports its UNIX port at no charge and does it well.

UNIX as it appears to the user is the same from system to system. UNIX as it appears to the individual doing system administration is quite different. At the user level, the hardware is completely user transparent. When you send a file to a device like a printer, all you have to know is that there is a printer and a command to get the file there. The system's administrator, however, must know the device by its internal name to access it.

This varies from system to system. A system programmer on a first-name basis with his devices on a piece of IBM or DEC hardware will be as lost a novice on a new 68000 if he has never seen one before. This is where user support is crucial. I have spent entire mornings on the phone with Amdahl while we installed a new version of UTS UNIX on the Amdahl mainframe at work, me typing and their support staff member walking me through the shuffling, expansion and installation. If more supermicro 68000-based UNIX OEMs followed suit, they would sell a lot more UNIX machines.

Codata offers something many similar OEMs do not—user support at no charge. This is a major consideration when buying a UNIX system, and it can't be taken for granted. Many 68000-based UNIX systems I know of cost twice as much as the Codata and offer no user support whatsoever. Of those that do offer support, I know of one company in particular that has one person just starting to learn UNIX as their user support 'staff.' Another company has the nerve to charge their customers over $1000 per year for 'optional support packages.' Naturally, Codata's user support is not intended to instruct novices, so don't go running to them to find out how to change directories or how to copy a file. Basic UNIX can be learned from any good introductory UNIX book, like *Introduction to the UNIX System* by Henry McGilton and Rachel Morgan (McGraw-Hill, 1983)

Codata is one of the few companies that offers UNIX machines at affordable prices.

and *The UNIX Operating System* by Kaare Christian (John Wiley and Sons, 1983). However, when you're stumped on something like figuring out how to make a correlation between the device files and the actual devices, then it's time to take advantage of Codata's able support group.

Documentation

UNIX documentation has always been drier than an army tech manual. Commands, subroutines and system calls are described as briefly as possible, with minimal elaboration. The wisdom of this approach is that the documentation is short enough in length to be

NEW from BORLAND!
TURBO TOOLBOX & TURBO TUTOR

"TURBO is much better than the Pascal IBM sells."
Jerry Pournelle, Byte, July 1984

"If you have the slightest interest in Pascal—buy it."
Bruce Webster, Softalk IBM, March 1984

BORLAND INTERNATIONAL GIFT PACK

ONLY **$99.95**
A SAVINGS OF $30!

What a gift for you and your friends! The extraordinary TURBO PASCAL compiler, together with the exciting new TURBO TOOLBOX and new TURBO TUTOR. All 3 manuals with disks for $99.95.

TURBO PASCAL Version 2.0 (reg. $49.95). The now classic program development environment still includes the FREE MICROCALC SPREAD SHEET. Commented source code on disk
- Optional 8087 support available for a small additional charge

NEW! TURBO TOOLBOX (reg. $49.95). A set of three fundamental utilities that work in conjunction with TURBO PASCAL. Includes:
- TURBO-ISAM FILES USING B+ TREES. Commented source code on disk
- QUIKSORT ON DISK. Commented source code on disk
- GINST (General Installation Program)

Provides those programs written in TURBO PASCAL with a terminal installation module just like TURBO'S!
- NOW INCLUDES FREE SAMPLE DATABASE

NEW! TURBO TUTOR (reg. $29.95). Teaches step by step how to use the TURBO PASCAL development environment—an ideal introduction for basic programmers. Commented source code for all program examples on disk.

30 DAY MONEY BACK GUARANTEE
These offers good through Feb. 1, 1985

For VISA and MASTERCARD order call toll free:
1-(800)-255-8008 1-(800)-742-1133
(Lines open 24 hrs., 7 days a week)
Dealer and Distributor inquiries welcome (408) 438-8400

CHOOSE ONE (please add $5.00 for handling and shipping U.S. orders)

_____	All Three-Gift Pack	$ 99.95 + 5.00	**SPECIAL!**
_____	All Three & 8087	139.95 + 5.00	**SPECIAL!**
_____	Turbo Pascal 2.0	49.95 + 5.00	
_____	Turbo Toolbox	49.95 + 5.00	
_____	Turbo Tutor	29.95 + 5.00	
_____	Turbo 8087	89.95 + 5.00	

Check _____ Money Order _____ VISA _____ MasterCard _____
Card #: _____________ Exp. date: _____________ Shipped UPS
My system is: 8 bit ____ 16 bit ____
Operating System: CP/M 80 ____ CP/M 86 ____ MS DOS ____ PC DOS ____
Computer: _____________ Disk Format: _____________
Please be sure model number & format are correct.

NAME: _____________
ADDRESS: _____________
CITY/STATE/ZIP: _____________
TELEPHONE: _____________

California residents add 6% sales tax. Outside U.S.A. add $15.00 (if outside of U.S.A. payment must be by bank draft payable in the U.S. and in U.S. dollars). Sorry, no C.O.D. or Purchase Orders. 69

BORLAND INTERNATIONAL
4113 Scotts Valley Drive
Scotts Valley, CA 95066
TELEX: 172373

Codata

Continued from page 76

stored on-line. However, it is not sufficient for learning purposes. Because it's neither comprehensive enough nor detailed enough to serve as a tutorial, it presupposes a previous knowledge of UNIX. Reference is the primary purpose of the on-line manual. This reflects the oral tradition of UNIX. At Bell Labs, or one of the colleges or universities that received the system before the divestiture of AT&T, the manual merely supplemented the learning process. Much of UNIX was taught by word of mouth or personal demonstration. This oral tradition does little to meet the needs of people trying to learn UNIX in business environments today.

Today's novice UNIX system owners and users are in a bit of a dilemma if they haven't learned UNIX at school. That UNIX may very well be the universal operating system of the future is a distinct possibility. Most of us in the computer and computer-related fields are going to have to learn UNIX eventually, because UNIX's influence is already responsible for enormous changes throughout the industry. If you don't know UNIX, you will never learn it from the on-line manuals, that's for sure. Fortunately, there is a good selection of UNIX books appearing on the market that supplement and enhance the manual with the necessary explanations in plain English and elaborate in detail on various concepts. Although most of these books are beginning tutorials that are designed to take the novice user through the learning phases, a few deal with advanced concepts as well, like Steven Bourne's book *The UNIX System* (Addison-Wesley, 1983) or Kernighan and Pike's *The UNIX Programming Environment* (Prentice Hall, 1984).

Codata directly supports its UNIX port at no charge and does it well.

The source of all UNIX manuals is the same, regardless of origin. *The UNIX Programmer's Manual*, published by Holt, Rinehart and Winston for Bell Labs, 1983, is the definitive version of the on-line UNIX manual. The Berkeley 4.2BSD version of UNIX adds pages to the manual to reflect the numerous significant contributions of William Joy and others. Porting houses like Unisoft, Interactive Systems or Digital Research take both the AT&T and Berkeley source codes and port them to a specific processor and machine architecture, adding (and sometimes deleting) commands and changing the manual to reflect the changes. The little-know ms macro **man** provides a template for UNIX manual pages, and when additions have been made, they are formatted under **nroff** or **troff** and stored in /usr/man.

Of the numerous UNIX manuals I've studied in the past year (including Amdahl's UTS, the manuals for Interactive Systems' UNIX port on the DEC VAX 11/780, the Unisoft port for the Codata 3300 manual, the Valid UNIX manuals, Mark Williams' Coherent, and others), the end result is variations on the theme of the same UNIX documentation as presented in *UNIX Programmer's Manual.*

Codata's documentation is 100% in the UNIX tradition. The manuals have been created by Unisoft from AT&T and Berkeley manuals. Codata manuals are similar to other UNIX manuals, although they are segmented a little differently. Unisoft puts all of the commands, subroutines, system calls, special files, file formats, games, macros, and maintenance commands in one book section, whereas other UNIX manuals divide these categories into three sections. The tutorials, which are usually in one massive volume, have been divided into two volumes, one for programming tools and the other for the remaining tutorials.

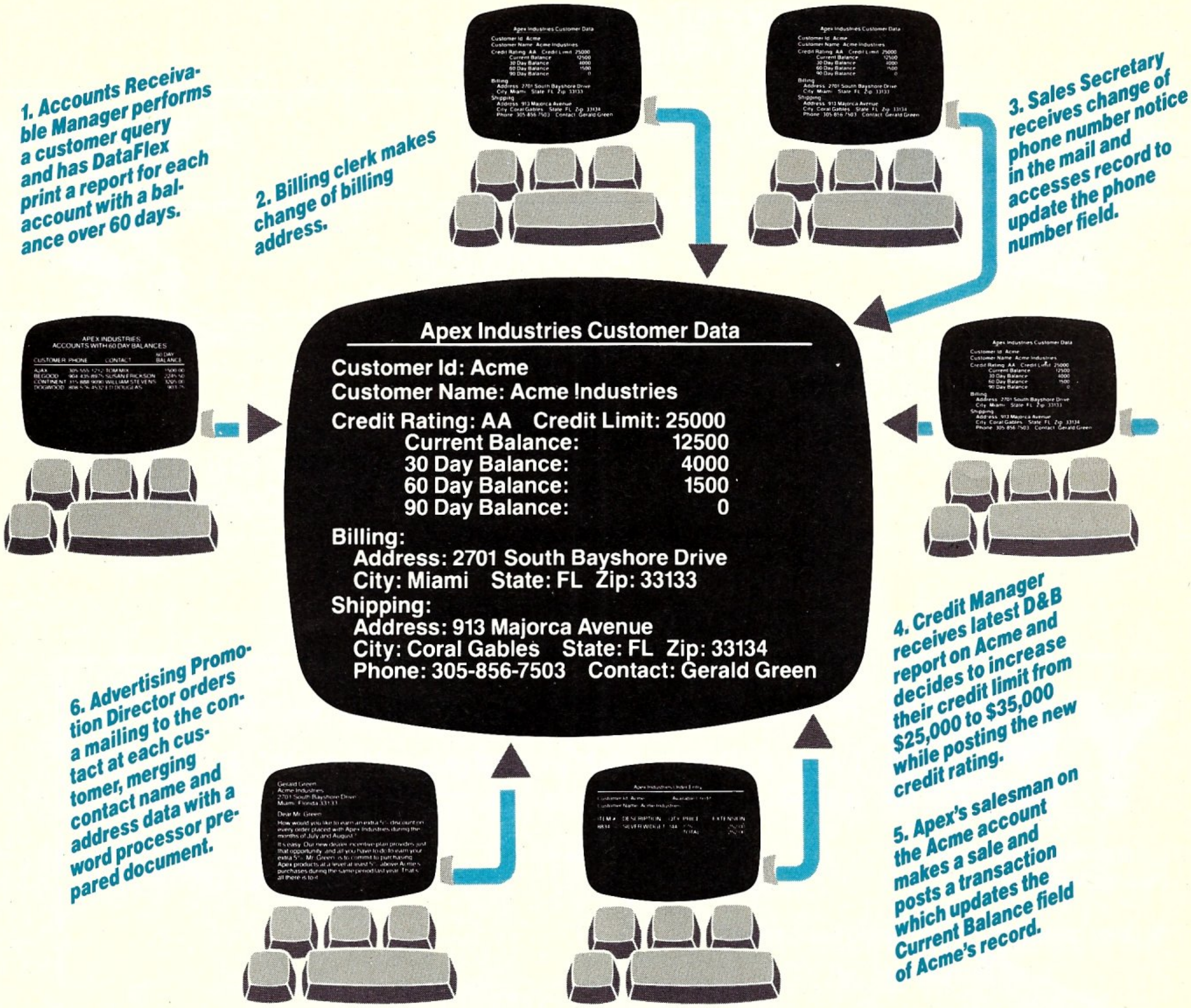
The documentation found in Volume 1 of Codata's manuals is also on-line. The **man** command followed by the command name brings up the page of the manual referring to that command. For example, the **ls** command is for listings. To find the manual pages that refer to **ls**, the following command is used:

```
man ls
```

This brings up that section of the manual to the console formatted by **more**. The on-line manuals are also traditional UNIX and is clearly one of the fringe benefits of a UNIX system. Amdahl's UTS has an enhancement called **findman** that searches a list for key words and returns any command

ALL AT ONCE!

AND NEVER A "LOCKED OUT" USER!



DataFlex is the only application development database which ***automatically*** gives you true multi-user capabilities. Other systems can lock you out of records or entire files for the full time they are being used by someone else. DataFlex, however, locks only the data being changed, and ***only*** during the micro-seconds it takes to actually write it to the file! The updated record is then immediately available. The number of users who can access, and change, records at the same time is limited only by the number of terminals on your system or network. Call or write today for all the details on DataFlex... the true multi-user database.

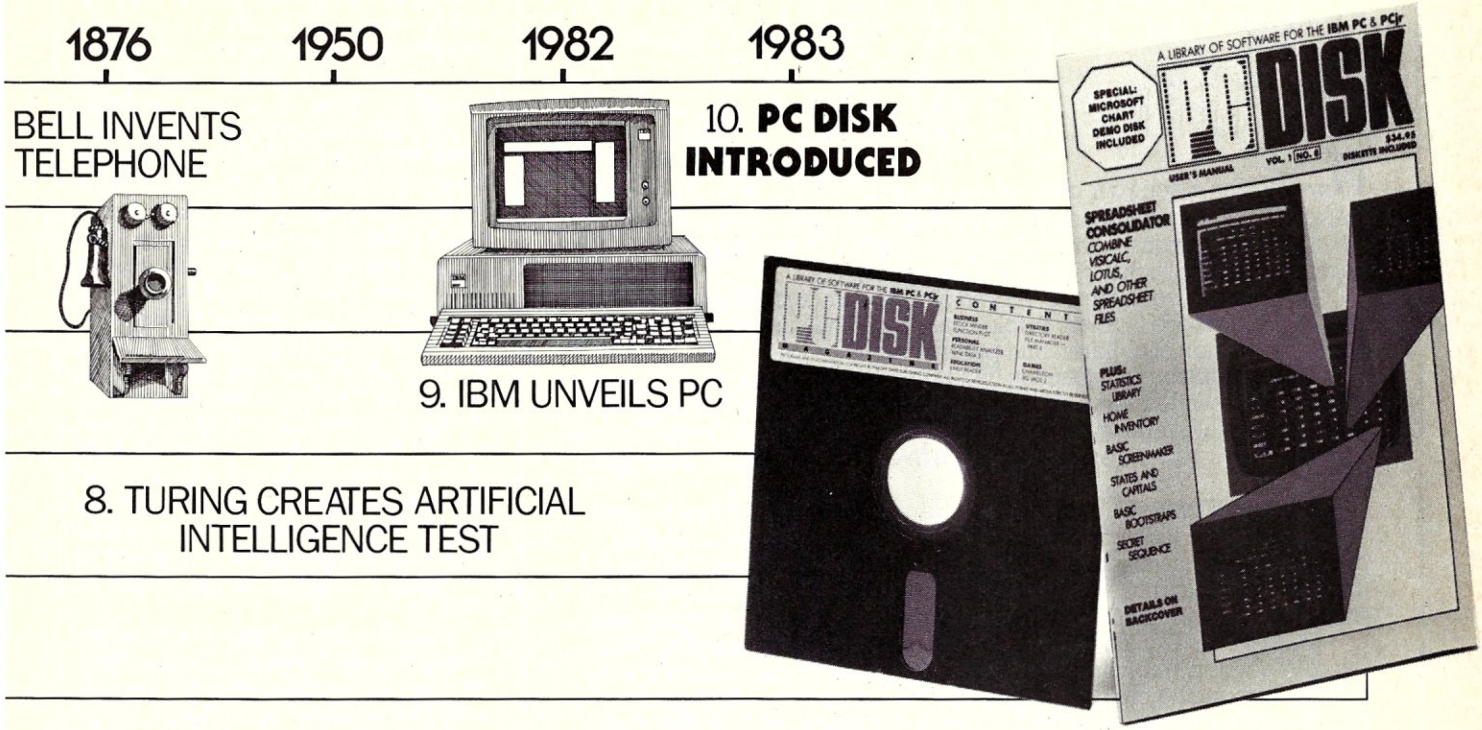
DATAFLEX™

DATA ACCESS CORPORATION
8525 SW 129 Terrace, Miami, FL 33156 (305) 238-0012
Telex 469021 DATA ACCESS CI

CIRCLE 215 ON READER SERVICE CARD

Compatible with CP/M-80, MSDOS networks, MP/M-86, Novell Sharenet, PC-Net, DMS Hi-net, TurboDOS multi-user, Molecular N-Star, Televideo MmmOST, Action DPC/OS, IBM PC w/Corvus, OMNINET, 3Com EtherSeries and Micromation M/NET.

MSDOS is a trademark of Microsoft. CP/M and MP/M are trademarks of Digital Research.



PC DISK, the continuous library of software for the IBM PC, PC XT, PCjr and PC AT, is the culmination of thousands of years of advances in the arts, sciences and communications.

Every PC DISK brings you up to 6 new, pre-tested programs that help you expand the functions of your IBM PC or compatible! With each issue, you'll receive new business, education, personal management and recreation programs that will add hundreds of dollars of new programs to your software library!

When you subscribe to PC DISK you'll receive a new disk *every six weeks*, complete with a fully illustrated User Manual to prompt you through each new program and utility. Subscribe to PC DISK today and save over $60! You'll maximize the practical value of your PC and discover new uses that will help you get more out of your micro!

In just nine months you can add over 36 new programs and applications, adding new power and dimension to your PC for only $149! And if you use your credit card or enclose payment with your order, you'll save an additional 10%!

PC DISK, THE GREATEST SOFTWARE DISCOVERY FOR YOUR IBM PC OR COMPATIBLE!

Product Specifications:
- PC DISK will operate on IBM PC, PC XT, PCjr, and PC AT under PC or MS DOS 1.1, 2.0 and 2.1, requiring a minimum of 128K.
- Monitor vs. TV is recommended.
- All programs are fully tested and guaranteed to run. Damaged or faulty disks will be replaced at no charge.
- All programs and documentation are copyrighted by Ziff-Davis Publishing Company. All rights of reproduction in all forms and media strictly observed.

FOR FASTER SERVICE, PHONE TOLL FREE 1-800-932-0017 (in NJ only 1-800-624-0497)
Also available at your local dealer.

Codata

Continued from page 78

names that match the pattern. This and other enhancements like it would be a positive addition to the Unisoft port.

To date no UNIX OEM has furnished anything other than the traditional documentation set. Any other material pertinent to the particular piece of hardware, such as the section on how to bring up the Codata, is also still in the dry, technical stage. If UNIX is to continue its phenomenal growth rate, more readable documentation is necessary to understand the system. The only documentation that is exclusively Codata's is a very brief description of how to boot the system from the floppy disk, and it is still weak. However, the self-explanatory on-line help menus on the boot dialogue are adequate to understand how to boot the system, but a warm and furry booklet on how to bring the system up would be extremely helpful for those new to UNIX.

There is also nothing other than the Unisis manual (Volume 1, Section 8) on how to configure and maintain the system. This area also merits further documentation. This in no way detracts from the Codata system as a whole. I'm mentioning it in this context because it is a weak area on most UNIX systems offered today. UNIX system will sell a lot faster when there has been a chance to write the manuals as well as Digital Research manuals are written today, or even the Gifford Computer Systems MP/M 8-16 manual. There needs to be greater emphasis on communicating essential points in plain English to first-time UNIX users.

Conclusions

The beauty of a full UNIX distribution as offered by the Codata 3300 is that it contains just about all the software a writer, applications programmer or system programmer is ever going to need. The only missing software are a database manager (over half a dozen good ones are available) and a good accounting system. A good combination of both is HCR's Chronicle Accounting. Codata offers both Unify and Micro Ingres (a distinct improvement over Berkeley's Ingres).

UNIX has so many positive features. Once you become acquainted with UNIX, its unique portability is a tremendous advantage. To the user, UNIX is the same from supermicro (PC/IX on a PC) to mainframe (UTS UNIX on an Amdahl 580 front-ending a Cray II). UNIX **is** a universal operating system. If you can use the primitive editor **ed**, you will never be shut out of a

Codata

Continued from page 81

UNIX system regardless of port or manufacturer. Even knowing **vi** ensures compatibility with most UNIX machines. (**vi** is a Berkeley enhancement, the full-screen editor that has as many bells and whistles as WordStar. It will probably become a UNIX standard eventually.)

If you have used UNIX before, either AT&T or BSD, you're in business with a Codata in the first minute of operation. Most new users take only a day to get the hang of the system's dozen and a half necessary commands and editor.

CoData's future

A lot of promising developments will be coming from Codata this January. Among them are System V running on the 68010 with demand-paged memory, built-in diagnostices for all the disk systems, I/O and the CPU, and numerous memory-paging innovations exclusive to Codata, such as utilizing advanced methods of dealing with noncontiguous memory.

Codata has quietly been around for some time now, offering a distinctly superior system for a very reasonable price. They have been recently acquired by Contel, a major west coast communications firm with very large assets. You will be hearing a great deal about Codata in the future.

The areas of skill and expertise at Codata are impressive. The well-designed chassis and the outstanding heat transfer characteristics of the machine show knowledgeability and dedication to excellence on the part of their hardware staff. Their software staff is equally knowledgeable, and has made many contributions to the present Codata UNIX distribution. Unisoft supplies the port, but it still takes a staff of dedicated kernel-level system programers to make it work.

The Codata 3300 computer runs on the 68000 microprocessor, and is available with 12, 33, and 84 MB, priced at $6,900, $9,600, and $13,500 respectively. The 12 MB version includes 320K of RAM, two serial I/O ports, and a 1 MB floppy disk drive; the 33 and 84 MB versions include 10 serial I/O ports, in addition to 320K of RAM and a 1 MB floppy disk drive.

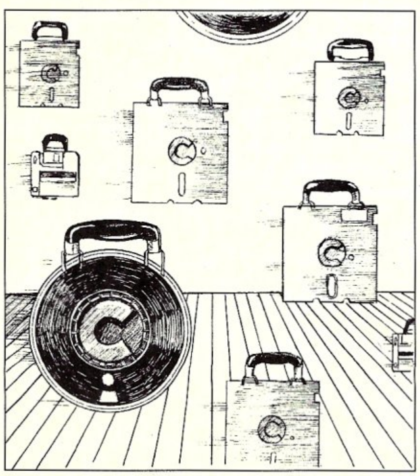
For more information, contact: *Contel Codata Systems, 285 No. Wolfe Rd., Sunnyvale, CA 94086; (408) 735-1744*

CIRCLE 328 ON READER SERVICE CARD

Bruce Hunter, 1020 S. Jenifer, Glendora, CA 91740

How Portable is C?

Problems that cause unpredictable results



by Michael Tilson

Imagine that you have just received a big program from your friend at X University. The program is written in C and runs on your friend's UNIX system. You also have C and UNIX, so you expect to have no problems, although you do recall that your friend uses a different brand of computer. You read the program off tape with no problem, and compile the program with no error messages. Then you run the program, and get this message:

```
Segmentation violation--core dumped
```

This isn't the desired result!

C language portability

This article focuses on machine-independent coding in the C programming language, illustrated by examples. All of the examples have arisen in practice. However, it does not cover a number of related issues, such as portable library routines, operating system interface, or the command interface, although these are also very important. This article assumes a working knowledge of the C programming language, and focuses on some of the less obvious portability problems.

Why do we want portability? Computers are becoming a commodity. When you buy a computer, you want to ask a few questions: Does it run UNIX (or your favorite generic portable system)? How fast is it? How reliable is it? How big a program can you run? How much does it cost?

You don't really want to ask very many other questions. In particular, you don't want to be forced to continue to buy from the same vendor if another computer is faster and cheaper. We want to reuse our software without any conversion cost. I might also add that portable software tends to be more reliable, since it tends to be less dirty and uses fewer "tricks". The secret of portability is to use only standard procedures known to have no side effects.

At my company, as at many others, there is a strong commercial interest in portability. We have supported or developed UNIX versions for the PDP-11, the VAX, the VAX under VMS, the PERQ workstation, the Computer Automation 4/95, the National 32000 and Motorola 68000 microprocessors. We have also worked with UNIX on several other processors, such as the 8086 and Z8000. We sell UNIX software products on these and other machines. We estimate that nonportable software has cost us well over $100,000 in the last year—and other companies have had similar experiences.

Levels of portability

There are two possible levels of portability: The first produces programs that are completely machine independent, type-safe and size-safe. The second level involves portability of resulting binary files from one "reasonable" machine to another—for example, between machines that have 8-bit characters, 16-bit short integers, and 32-bit long integers. For instance, you might have a C cross-compiler for the CA 4/95 that initially ran on a PDP-11, got moved to a VAX, and then got moved to the 4/95 itself. This kind of program may not be fully portable (since it generates machine code), but can still be made easy to move across an important class of machines.

Portability problems in C

C is not a safe language. The Kernighan and Ritchie reference book, *The C Programming Language*, describes a number of portability problems. To avoid these, you must be familiar with all of the footnotes in that book. Problems commonly arise from the careless use of pointers and the mismatching of types, from assumptions about the sizes (in bits) of various types, and from alignment assumptions (such as the order of chars within a short int). Type casts and the UNIX **lint** program checker don't solve all problems. Type casting can make **lint** shut up without solving the portability problem. There is no substitute for getting it right.

C has been described as a "high-level assembler". While this is an exaggeration, it is true that many C programmers visualize the resulting compiled code as they write C programs. This may result in greater efficiency on some machines, but it does not make for portable software.

Examples

The remainder of the article will focus on actual programming examples which have arisen in our work at Human Computing Resources. This is not an exhaustive list of what can go wrong, but rather an illustration of the care that you must take when writing portable programs. I should add that all of the examples are "lintable", at least with the default level of checking provided by some versions of **lint** (newer versions will catch some of the errors shown here). In the following examples, you will see calls to **abort**. On the machines most people are used to, **abort** will not be called; however, for each example there are also machines for which the **abort** call *will* happen.

```
Example 1:
      int *p
      char *q, *q2
      ...
      q2 = q;
      p = (int *)q;
      q = (char *)p;
      if(q !=q2)
            abort();
```

The coding fragment shown in Example 1 can fail because there is no guarantee that the conversion of a pointer from 'char *' to 'int *' will preserve all of the bits of precision if the 'char*' had not already been aligned to an int boundary. The code in Example 1 works on the VAX, PDP-11, and most other machines, but will fail on the Harris /6 and on some other machines. This kind of type mismatch is common in UNIX software.

```
Example 2:
      int *p
      char *q;
      int arr[50];

      p = &arr[20];
      q = (char *) &arr[0];
      q++;
      if(p < (int *)q)
            abort();
```

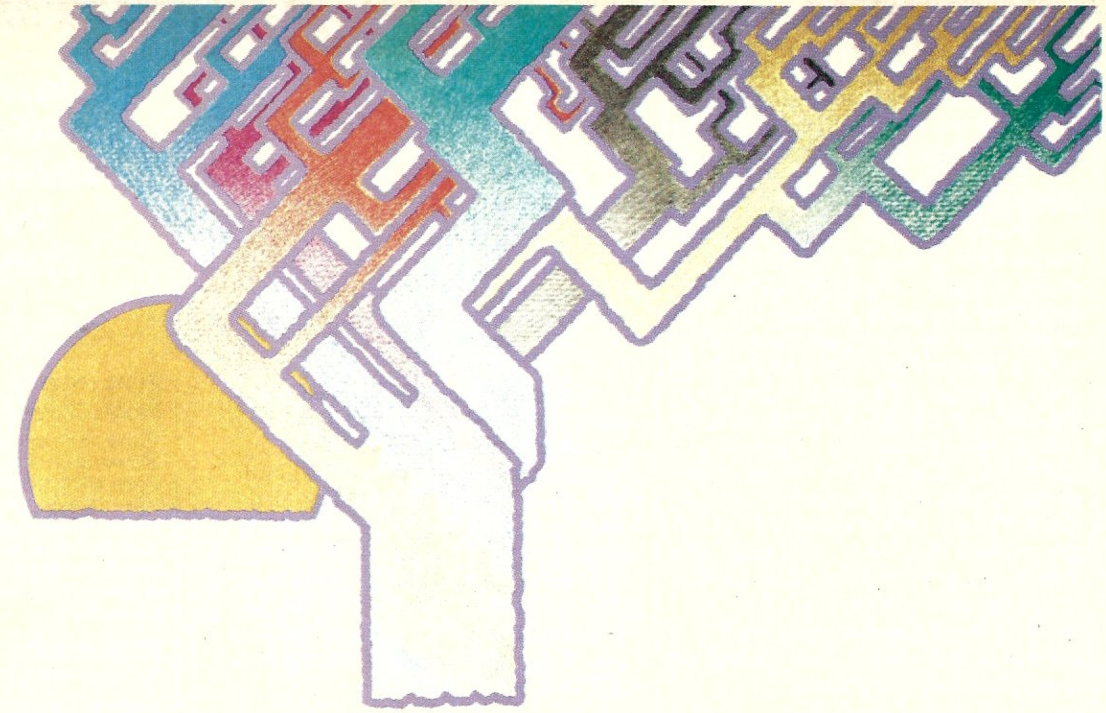
How can the coding fragment in Example 2 fail? Variable 'p' seems to be at a higher memory address than 'q'. Actually, it isn't. The 'q++' could produce an internal bit representation which, if considered as an 'int*' quantity, would look like a word pointer at a larger memory address. This example is adapted from the UNIX shell command interpreter and causes much trouble on the CA 4/95.

```
Example 3:
      int *a, *b;

      a = (int *)sbrk(0);
      b = a
      b--;
      if(a < b)
            abort();
```

What could be wrong with the coding fragment shown in Example 3? On a segmented-architecture machine, the UNIX memory allocator 'sbrk' might return an address at the start of a memory segment. Pointer arithmetic is valid only if the pointers remain within a known contiguous and properly aligned storage structure. The 'b--' could wrap around to the high end of a memory segment. This example comes from a version of the UNIX text editor.

```
Example 4a:
      execl("/bin/echo", "echo",
            "hello", 0 );
```



Btrieve and Xtrieve. B-tree file access for all your programming languages. With the query tool your users demand.

Introducing a powerful data base combination for PC application developers. Btrieve™, the most sophisticated file access method for your IBM™ PC. In single user and network versions. And Xtrieve™, a new menu-driven query tool that gives you—and your users—fast access to information.

Btrieve: for professional programmers. Btrieve provides fast, flexible file management for all your application development. All your programming languages—BASIC, Pascal, Cobol, C. With multikey access to records. Automatic file recovery. Unlimited records per file. Duplicate, modifiable, and segmented keys. Variable cache buffer.

Better applications faster. Based on the b-tree file indexing system, Btrieve provides optimal file management and eliminates writing file management routines. So you can develop *better* applications *faster.*

Xtrieve: easy window interface. The ideal complement to Btrieve, Xtrieve is the non-programmer's interface. Xtrieve's full relational capabilities let users define a virtual table of data from multiple files. Then Xtrieve speeds them through query building with a series of easy-to-follow windows.

No command language. Xtrieve is completely menu-driven, so there's no need to memorize command language. Or special syntax. Everything you need is on the screen. You and your users can perform unlimited queries—and get answers fast.

Xtrieve features a full range of restriction criteria. Online help messages. Automatic maintenance of information order by multiple indexes. And interfaces to access information from Lotus 1-2-3™ and dBase II®files.

Network capabilities. Network versions of Btrieve and Xtrieve allow data sharing in the most popular PC networks — PCnet™, NetWare™, EtherSeries™, MultiLink™, and OmniNet™.

For more information or to order, call or write:

SoftCraft Inc.

P. O. Box 9802 #590 Austin, Texas 78766 (512) 346-8380

Suggested retail prices: Btrieve, $245; Xtrieve, $195. Btrieve/N (network), $595; Xtrieve/N, $395. Dealer inquiries welcome. Btrieve requires PC-DOS or MS™-DOS 1.X or 2.X; Xtrieve, PC-DOS or MS-DOS 2.X.

Btrieve and Xtrieve, IBM, 1-2-3, dBase II, PCnet, NetWare, EtherSeries, MultiLink, OmniNet, and MS are trademarks of SoftCraft Inc., International Business Machines, Lotus Development Corp., Ashton-Tate, Orchid Technology, Novell Data Systems, 3Com Corp., Davong Systems Inc., Corvus Systems, and MicroSoft Inc.

CIRCLE 35 ON READER SERVICE CARD

C Portability

Continued from page 85

The error shown in Example 4a is from most versions of the UNIX Programmer's Manual. The 'excel' system call takes a list of character strings, terminated by a null pointer. The constant '0' is guaranteed to be a valid null pointer if used in an expression. However, C does no type checking across function calls. On a machine with 32-bit pointers and 16-bit integers, Example 4a could cause trouble. It would be better to use the code in Example 4b:

```
Example 4b:
    execl("/bin/echo", "echo",
          "hello", (char *)0); or
    execl("/bin/echo", "echo",
          "hello", NULL);
```

where NULL might come from a standard include file (e.g. **stdio.h**). Note that Intel 286 UNIX implementations commonly use 16-bit integers and 32-bit pointers.

```
Example 5:
     char *p = (char *) 0;

     if( *p !=0)
          abort();
```

Example 5 shows another problem common with null pointers such as 'p'. It is not valid to assume that a null pointer in turn points to a zero quantity, although this happens to work on many UNIX implementations—in fact, the VAX and PDP-11 implementations go out of their way to make it work. But this is highly non-portable, and will cause no end of trouble on other architectures.

Portable software is more reliable, since it tends to be less dirty.

```
Example 6:
     int x;

     x = 70000;
     if(x !=70000)
          abort();
```

The code in Example 6 works on a VAX but not on the PDP-11, because the number 70000 fits in an int on the VAX but not on the 11. Moral: always explicitly specify long if a number is going to be larger than somewhat. (A reasonable "somewhat" is 32767.)

```
Example 7:
     char *p, *q;
     int n;

     n = (int)p;
     q = (char *)n;
     if(p != q)
          abort();
```

The C language guarantees that there is a large enough int to hold the bit representation of a pointer. However, on a machine with 32-bit pointers and 16-bit ints, the fragment in Example 7 will fail because the conversion to int will overflow. Conversion to anything other than long is not portable. (By the way, conversion of pointers to integers is in general bad practice and is always non-portable if the value of the int is ever examined. The only safe thing to do is convert back to the same kind of pointer.)

Assume that the code fragment in Example 8 is running on a machine with a large (e.g. megabytes), contiguous, uniform address space.

```
Example 8:
     char *p, *q;
     long int n;

     n = 0;
     q = p;
     for(i=0; i<5; i++) {
          q += 32000;
          n += 32000;
     }
     if((q-p) != n)
          abort();
```

The subtraction of two pointers is defined by the manual to yield int, and not necessarily long int. Once again, int might not be large enough to hold the difference. This might be an unfortunate restriction in some C compilers, but it is dictated by the current C definition.

```
Example 9:
     struct {
               short int a_magic;
               long int a_text;
               long int a_data;
     } header;
     (void) write(f, (char *)
      &header, sizeof header);
```

What could be wrong with Example 9? Everything is nicely type cast, and even the return value from write is fastidiously voided. The problem is that a binary structure is being communicated to the outside world via the UNIX write system call. This always indicates a possible portability problem, since the resulting file can't be moved from machine to machine. Increasingly, we will

C Portability

Continued from page 87

see networks of dissimilar UNIX machines, but with network-wide file systems. One should be careful about producing binary files, and one should never assume that they will be portable.

```
Example 10:
      long int n;
      printf("%d\n", n);
```

The code in Example 10 is the bane of everyone who has ever had to move programs from the VAX to the PDP-11. The '%d' prints an int, which is the same as long int on the VAX, but not on the PDP-11. This is an example of the more general case of function argument mismatch, but one that is not checked by **lint** (although it really should be.)

```
Example 11a:
      double x = 1.234e30;
      printf("%d\n", x);
```

Example 11a shows a common programming error. A similar example was used in a very glossy advertising brochure sent out by a company promoting its C training courses. It looks as if the programmer wanted to print the number as decimal digits truncated or rounded to an integer. The fix proposed by the glossy brochure is shown in 11b.

```
Example 11b:
      printf("%d\n", (int)x);
```

On the VAX, this fix prints '0' rather than the correct number. On other machines, an overflow exception might be generated. This is an example of throwing in a type cast to patch a problem rather than getting it right from the first. Here a type cast was used to try to correct a datatype mismatch error. In other cases, casts are used to remove portability problems. In *all* cases, there is no substitute for making things match up correctly in the first place.

```
Example 11c:
      printf("%.0f\n", x);
```

The code fragment shown in Example 11c, which prints a floating-point number correctly rounded to the nearest integer, is a much better solution.

```
Example 12a:
      to = bp->b_ptr;
      asm("movc3 r8,(r11),(r7)");
      bp->b_ptr += put;
```

The next bit of program, shown in Example 12a, was supplied by a certain educational institution well known for its extensive UNIX modifications. This program fragment illustrates the extreme case of nonportability. 'asm' is a *keyword* in the UNIX portable C compiler. It causes the string argument to be emitted into the assembly source that results from compiling the program. In this example, there is no comment to say what is going on. The programmer *knows* what C variables are in particular registers. (The variable 'to' is one of the registers used in this VAX assembly instruction. You guess which.)

If you feel you *must* write such "efficient code," at least write it a bit more cleanly, as shown in 12b.

```
Example 12b:
      #ifdef vax
              asm("movc3 r8,(r11),
               (r7)");
      #else
              ---insert portable C
               equivalent here---
      #endif
```

```
Example 13a:
      int TheQuickBrownFox;
      int TheQuickGreyFox;
```

Example 13a causes no end of trouble when moving code from VAX UNIX systems (which typically allow this) to other systems. While meaningful variable names are laudable, names which are not unique in the first seven characters are not portable to other UNIX systems. (On non-UNIX systems, you might even be restricted to six

TURBODOS

The world's most advanced microcomputer operating system.

*** SIERRA * TELETEK * SD SYSTEMS ***

If you are using one of the above manufacturers board sets with TurboDOS, you are already our customer. If not, you should consider joining us. We have a great deal to offer.

IBM PC

BSI is now supplying TurboDOS on the IBM PC . . .

For information on upgrading your system to the TurboDOS operating system, contact your supplier or BSI. We have been implementing TurboDOS since 1981.

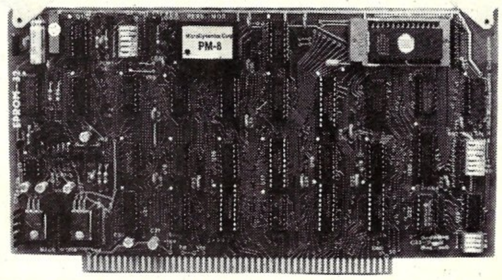
Dealer and Manufacturer queries invited.

BSI
1064 W. Ocean View Ave.
Norfolk, Va. 23503
Phone 804 587-3066
Telex 757518 BSIORF

Dealers and Manufacturers only Circle 60
End–Users Circle 61

S-100 EPROM PROGRAMMER

EPROM-32



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

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

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

$269.95*
(A & T)

MicroDynamics
Corporation

Suite 245 • 1355 Lynnfield Road • Memphis, TN 38119
(901)-682-4054

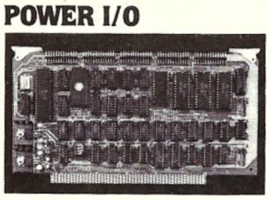
* Price includes EPROM-32, documentation and two Personality Modules(specify). Additional Modules—$7.95. Control software on 8" SSSD diskette—$29.95. UPS ground—$2.00, UPS air—$4.00, COD—$1.65, foreign add $15.00. VISA & MASTERCARD welcome.

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

CIRCLE 64 ON READER SERVICE CARD

Extended Processing S100 Boards

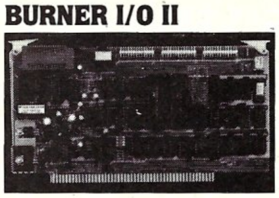
POWER I/O



High performance S100/IEEE-696 smart slave computer with 64K RAM, 3 serial ports, 1 centronic port, comprehensive 4K operating system in EPROM and 1 timer. Host access is through a high speed parallel I/O port. Accepts 256K RAMS when available. Optional ADD-ON board doubles I/O and RAM. Standard software and hardware supports 6 serial ports, 2 parallel ports and 512K of RAM. Entire board is software programmable including all I/O buffer sizes.

POWER I/O w/64K and 3S+P:	$375.00
64K RAM ADD-ON board:	$175.00
3S+P ADD-ON board:	$195.00
64K and 3S+P ADD-ON board:	$295.00

BURNER I/O II



Multifunction S100/IEEE-696 board. Complete EPROM programmer handles 5 volt EPROMS: 2508, 2758, 2516, 2716, 2532, 2732, 2732A, 2564, 2764, 27128, 27256. Fully I/O mapped. EPROM selected totally with software. No switches or program modules. Menu driven software supplied in 4K EPROM. 2 independent serial ports with baud rate to 19,200. 1 centronic type parallel port. Memory management for address lines A16-A23.

Option A: Full board	$355.00
Option B: Programmer	$220.00
Option C: I/O (2S+P)	$220.00
Option D: Programmer+I/O	$330.00
Option E: Memory management	$110.00
Memory management for B or C:	$ 25.00

All E.P. boards are built with quality components and are fully assembled and tested. Full documentation including schematics and source code listings.

ep Extended Processing 3861 Woodcreek Lane, San Jose, Ca, 95117 (408) 249-8248

CIRCLE 142 ON READER SERVICE CARD

INTRODUCING THE WORLD'S FASTEST

S-100 Z-80 SLAVE PROCESSOR

TurboSlave I

- 8 Mhz Z-80H
- Data transfers to 1 mbyte/second
- S-100 IEEE-696 compatible
- 4k Monitor rom
- Low parts count
- No paddle boards
- 128k Ram with parity
- 2 RS-232 Ports, 50-38.k baud
- F.I.F.O. communications
- On board diagnostics
- Low power consumption
- TurboDos compatible

INTRODUCTORY PRICE $495

Includes TurboDos drivers (a $100 value) and TurboSlave I with 128k ram.

EARTH COMPUTERS

P.O. Box 8067, Fountain Valley, CA 92728

FOR MORE INFORMATION CALL: (714) 964-5784
Registered trademarks: Z-80H, Zilog Inc.; TurboDos, Software 2000, Inc.

CIRCLE 101 ON READER SERVICE CARD

C Portability

Continued from page 88

characters.) With a little thought, you can always choose nice names that are also portable (Example 13b).

```
Example 13b:
        int TheBrownQuickFox;
        int TheGreyQuickFox;
```

These examples are only an indication of the kinds of portability problems that are often found in real C programs. You can avoid problems by viewing the machine as an abstract entity, and not making assumptions that depend upon actual bit representations. With a little bit of care, your software will be usable on a wide range of machines. And if your software is worth using more than once, then it should be portable, since it's worth using on more than one machine.

Michael Tilson, Human Computing Resources Corp., 10 St. Mary Street, Toronto ON Canada M4Y 1P9. As vice president of technical development at HCR, Mr. Tilson has directed a number of projects involving the creation of UNIX environments and C compilers on 16-, 32-, and 64-bit processors.

PFIX 86 from Phoenix

Symbolic debugging for MS-DOS/ PC-DOS

by Michael Olfe

There are times in the development of any program when it doesn't work. First, you suspect the compiler, the operating system, and the hardware of not working properly. Then, you go back and look at your own code, and make it a little better, but the bug still doesn't go away. Either you prove beyond a shadow of a doubt that Microsoft has a bug in their operating system, or you find out what's wrong with your code. To settle these issues, you reach for that modern-day leveller, the debugger.

What is a debugger?

Most of the time, the debugger finds that your code wasn't so perfect after all, because it wasn't doing what you thought it was. Maybe the hardware didn't work the way you thought it did, or the way the documentation *said* it did. Maybe you just coded it wrong and the compiler accepted it. Whatever it was, the magnifying glass of a debugger helps to make the problem clear.

My search

Way back in 1978, when CP/M was the dominant microcomputer operating system, my favorite debugger was one called BUG, and its little brother MuBug, from Phoenix Software Associates. BUG had no problem with Octal (used in some still-extant prehistoric listings), Binary, Decimal, or Hex—you could enter data into memory or read it out in any of those radices. BUG also loved multiple breakpoints with ANDed and ORed conditionals, e.g. "Break anywhere if memory location 55 goes above 100 hex AND the HL register is pointing to it OR the Program counter goes above E000 OR the parity flag changes." The manner of representing this statement might have been a little arcane (it went something like:

```
"0,( ((@M>55) & (HL=55)) |
     (PC>E000) | (P=!P))'",
```

if I remember rightly). But one got accustomed to it. In several years of using BUG I never saw anything that made me even consider changing.

Then along came the IBM PC and I went back to primitive tools like DEBUG. I was so debased that when DEBUG version 2 came along with PC-DOS 2.0, and included an assembly command, I actually felt grateful. Although products like Trace86 and Codesmith reminded me of what a glorious thing a good software debugger can be, I still yearned for the real thing—BUG on the IBM PC.

Of course, you can't go home again. Phoenix Software Associates has released a debugger for the IBM PC/XT, TI Professional, Wang Professional, and DEC Rainbow that has all the power of BUG, but doesn't look much like it. The new product, Pfix86, is easier to use and more powerful. It is not BUG reincarnated, but recreated. On the PC, Pfix86 is faster, friendlier, and more powerful than BUG was on CP/M. It has also been given additional hardware support and support for high-level language debugging.

Software vs. hardware debuggers

Pfix86 is a software debugger, which means it will *not* set breakpoints in ROM, although it will trace through ROM. It will not allow you to break if interrupts are disabled, nor is it the best tool for heavy debugging of interrupt-driven code—for this you should investigate the more expensive hardware debuggers. But if you are looking for a software debugger that is fully featured, Pfix86 is worth considering.

It offers in-line assembly, symbolic debugging of normal or overlayed programs (Pfix86 Plus), multiple windows, dynamic breakpoints, user-controlled data formatting, multiple radices, and absolute disk sector reads and writes. All this in 60K, which leaves room even in a small system for debugging large programs.

Pfix86 displays

When you run Pfix86, it first divides the display into STACK, CPU, DATA, FILE, PROGRAM, and BREAKPOINT windows, as shown in Figure 1. Pfix86 uses the IBM PC graphics characters to draw the windows, arrows, and non-ASCII bytes in the DATA1 window, so the actual screen is more readable than the one represented here. Highlighting and underlining are used to mark the current window, current data bytes, current instruction, and breakpoints.

The default sizes of these windows, and other default information, is set in an ASCII configuration file which can be edited or created with any editor. The configuration file can also specify whether Pfix86 should output to the color, monochrome, or default monitor. If you have two monitors on your PC, you can use one for the application code's screen output and the other for Pfix86's screen output. Even with one monitor, Pfix86 will prevent the application from overwriting the debugger display (see below). Pfix86 reads the configuration file named PFIX.CFG if no such file is named on the command line. Otherwise, it takes the information from the named file.

The DATA, FILE, PROGRAM, and BREAKPOINT windows can be shrunk to a minimum of one line, or grown to fill the maximum remaining space. They are 'tiled,' not overlapping windows, and their relative positions on the screen cannot be moved.

Pfix86's ability to go directly to I/O ports is invaluable.

When you are not in the menu system, but executing a command or tracing, the second screen line displays status information. The Evaluate expression command, for example, will display the value of the expression on line 2. Line 2 displays the current instruction's effective address and contents. Hence, if the DS register contained 0, the BX register 0200, the SI register 0100, if the contents of memory location 0300 were 0200A and the current instruction in the program window were MOV AX,[BX+SI], then DS:0300 = 0200A would be displayed on line 2. Instructions which use other segment registers or segment overrides are properly handled.

The disassembled listing in the program window makes use of WORD PTR and BYTE PTR prefixes to clarify the size of an operand.

Pfix86 commands

When a program does more, it has more commands, and keeping track of all the commands can become a problem for the user. Lotus 1-2-3 divided its commands into discrete menus with submenus, and Pfix86, with more than 65 distinct commands, has adopted the same approach. A complete list of all Pfix86 commands appears in Table 1.

Pressing F1 causes the main Pfix86 menu to appear on the top two lines of the screen as shown in Figure 2. The current selection is highlighted, with a list of subcommands below. There are two ways to select a command:

1. Use the space key or the arrow keys to highlight the command, then press Enter to select the command.

2. Press the first letter of the command to highlight and select it.

This method of selecting commands is a good combination of ease-of-use and speed of operation. Though the

```
PFIX86-PLUS rev. 1.02 Copyright(c) 1984 Phoenix Software Associates Ltd.
+-STACK-----------++-DATA1------------------------------------------------------+
| 1818:0012  04E2 || 1817:0000 CD 20 00 00 00 00 00 00  * ******                |
| 1818:0010  0542 || 1817:0008 00 00 00 00 00 00 00 00  ********                |
| 1818:000E  028C || 1817:0010 00 00 00 00 00 00 00 00  ********                |
| 1818:000C  F000 |+-FILE-------------------------------------------------------+
| 1818:000A  FFF0 ||                                                             |
| 1818:0008  9A00 |+-PROGRAM----------------------------------------------------+
| 1818:0006  0000 ||=>1817:0100  57                 PUSH DI                    |
| 1818:0004  0000 ||  1817:0101  56                 PUSH SI                    |
| 1818:0002  0299 ||  1817:0102  55                 PUSH BP                    |
| 1818:0000  20CD ||  1817:0103  CD21               INT  21                    |
|=>1818:0FFE 7C21 ||  1817:0105  890EBCB8           MOV  [B8BC],CX             |
+-CPU-------------+|  1817:0109  89168EB8           MOV  [B8BE],DX             |
| AX=0000 BX=0000 ||  1817:010D  5D                 POP  BP                    |
| CX=0000 DX=0000 ||  1817:010E  CB                 RETF                       |
| SP=0000 BP=0000 ||  1817:010F  0000               ADD  [BX+SI],AL            |
| SI=0000 DI=0000 ||  1817:0111  0000               ADD  [BX+SI],AL            |
| CS=0000 DS=0000 ||  1817:0113  0000               ADD  [BX+SI],AL            |
| SS=0000 ES=0000 ||  1817:0115  0000               ADD  [BX+SI],AL            |
| IP=0000 T:0     ||  1817:0117  0000               ADD  [BX+SI],AL            |
| A:0 C:0 P:0 S:0 |+-BREAKPOINT-------------------------------------------------+
| Z:0 D:0 I:1 O:0 ||  00:                                                        |
+-----------------++-------------------------------------------------------------+
```

Figure 1. Pfix display, broken into Stack, CPU, Data, File, Program, and Breakpoint windows.

Pfix86

Continued from page 93

menus go as deep as three layers, no command requires more than four keystrokes. The Breakpoint Set command, for example, can be accomplished by typing "< F1 > BS". A single keystroke —F10—will set a breakpoint at the current location in the Program window, and there are many other such keystroke shortcuts for commonly used commands. The ESC key always backs up one menu level, so there is no getting lost deep inside the menu tree and forgetting where you are.

At the root menu level, the ESC key gives a list of the function key command shortcuts, a nice enhancement to the Lotus-like menus (see Figure 3).

The in-line assembler is very close to the MASM assembler provided with MS-DOS. I noticed only two differences: the use of WO and BY prefixes to state the size of an operand, and the disallowal of constructs like PUSH 12[BP] (this must be written PUSH [BP+12]). The assembler automatically generates short, near, or far jumps, whichever is appropriate.

A nice feature of Pfix86 is the flexibility it allows in byte lists. Byte lists can be used with the Memory Set and Memory Find commands, or to enter data directly into the byte display areas of the Data or Program windows. A byte list can be a free mixture of hex bytes, assembler instructions, and character strings. The following is a valid byte list:

```
OFF 'This is a message' 0
              (CMP AX,20)
```

Character strings use single or double quotes, and doubling is used to insert the quote character in the string.

Data entry can be done at any time in any of three radices: hex, octal, or decimal. The radix can also be set for output to hex, octal, decimal, or binary, but Pfix86 does have a bias for hex. If you set the output radix to binary, for example, immediate operands and expressions will be displayed in binary, but the byte displays in the Program and Data windows will be in hex. The Data window can be selected to display in Word, Byte, or Long Word formats, but its data is always displayed in hex.

Pfix86's ability to go directly to the I/O ports is invaluable when you are working with unknown or poorly documented hardware. Often, a few INs and OUTs with the ports can uncover critical information. Similarly, being able to see what the program actually wrote to the disk, and repair it if necessary, is far more convenient than using an external disk editor.

Table 1. List of Pfix86 commands

The commands are presented as a menu tree. The indented commands are submenus of the command at the previous level of indentation. Thus, Breakpoint Enable Single is a command reached by selecting Breakpoint, then Enable, then Single, by typing BES. Where necessary, the command is explained in parentheses.

Breakpoint	
Clear	Initialize
Disable	Enable
All	All
Single	Single
Set	Temporary
Disk	
Disassemble	(Disassemble to disk)
File	(Read a text file into the file window)
Get	(Read data from absolute disk sectors)
Load	(Load a COM or EXE program from disk)
Name	(Set the filename to use for subsequent Disk Read/Write operations)
Put	(Write data to absolute disk sectors)
Read	(Read data from file, with optional offset)
Write	(Write data to file, with optional offset)
Evaluate	
Format Data	
Address	(Set the DATA window to display hex data in segment:offset format)
Byte	(Set the DATA window to display hex data as individual bytes)
Long	(Set the DATA window to display hex data as long words)
Word	(Set the DATA window to display hex data as words)
I/O	
In	(Word from port)
Out	(Word to port)
Read	(Byte from port)
Write	(Byte to port)
Memory	
Clear	Move
Find	Set
Program	
Arguments	(Set command line)
Execute	(Program name, command line)
Go	(Execute, starting at address)
Proceed	(Trace, skipping loops)
Restart	(Continue after break)
Symbol	
Add	(Add a symbol to the symbol table)
Initialize	(Remove all symbols from symbol table)
List	(List all symbols in symbol table to screen)
Size	(Set maximum display length of symbols)
Trace	(Execute single instruction)
Quit	
Yes	No
Radix	
Binary	Hex
Decimal	Octal
Screen	
Application	(Show application program screen's current appearance)
Color	(Send Pfix86 output to color monitor)
Monochrome	(Send Pfix86 output to monochrome monitor)
No update	(Do not update application program screen)
Update	(Update application program screen)
Tracing	
Clear	(Set one or more software interrupts not to be traced)
Disable	(Disable all interrupt

	tracing)
Enable	(Enable all interrupt tracing)
Set	(Enable one or more software interrupts to be traced)
Window	
Address	(Set origin address of DATA, PROGRAM, STACK, FILE, or BREAKPOINT window)
Change	
Break-point	(Select BREAK-POINT window as current window)
CPU	
Data	
File	
Program	
Stack	
Exchange	(Same as F3 key described in article)
Subwindow	
Add	(Add a subwindow to current DATA or PROGRAM window)
Next	(Same as ALT F4 described in article)
Remove	(Remove last-created subwindow)

Pfix86 supports DOS 2.x pathnames. Filenames can be fully specified (i.e., A:\ROOT\SUBDIR1\SUBDIR2\FILE). If the filename does not include a path, Pfix86 will search the current DOS path for the file.

Pfix86 breakpoints

There are 10 temporary breakpoints and 10 permanent breakpoints in Pfix86. Temporary breakpoints always cause the program to stop unconditionally the first time they are reached, and then they disappear. Permanent breakpoints can cause a program to stop, another breakpoint to be enabled or disabled, or an address to be called. In addition, they can have a count and a conditional expression. The count is decremented each time the breakpoint is reached. When the count reaches zero, the break is taken and the count reset to one, allowing re-use of the breakpoint. The conditional expression can be of arbitrary complexity.

The address associated with a breakpoint can be specified as a complete segment:offset, offset from the current CS, or made "global". "Global" means that the condition will be tested at each instruction. This makes it possible to monitor memory locations which you suspect are being clobbered by the application code.

Pfix86 uses the same operator symbols used in the C programming language, (excluding ++ and —), and uses them as C does in expressions.

```
[COUNTER]>500 && [COUNTER<1000]
                || [COUNTER]==10
```

evaluates exactly as it would in C. This expression can be used as the condition for a breakpoint, as in:

```
B S 0 * 3 S [COUNTER]>500 &&
  [COUNTER<1000] || [COUNTER]==10
```

The commands, broken down, are:

B Breakpoint—chooses "Breakpoint " from menu of commands

S Set—chooses "Set" from "Breakpoint" submenu

0 Breakpoint #0

* When used as an address, * means "global", i.e., test the breakpoint condition at the end of every instruction, and if true, break.

3 The breakpoint count. Break the third time breakpoint condition is met.

S What to do when the breakpoint condition is met. "S" means stop execution. Other actions which can be taken on breakpoint are: Enable another breakpoint, Disable another breakpoint, and Call an address.

Symbolic debugging

There are two versions of Pfix86: Pfix86 and Pfix86 Plus. Pfix86 Plus allows symbolic debugging of EXE files that have been created with recent versions of the Phoenix linker Plink86. Plink86 appends a symbol table to the end of the EXE file. Pfix86 Plus reads this table into memory and allows the use of symbols wherever normal expressions are allowed. If you have never used a symbolic debugger, consider the differences in readability of the two code fragments listed in Figure 4.

Using symbols as expressions is also a great timesaver. For example, you can set a breakpoint at any label by typing its name followed by the F10 key. You can see the value of a symbol at any time by evaluating it—just type its name and the F8 key. Similarly, symbols can be added and used freely in arithmetic expressions. At any time, the complete list of symbols can be displayed, added to, changed, or erased. Or, you can write out the current disassembly to a file that includes symbols.

Symbols can also be useful in conjunction with the file window. Suppose you are debugging a program that was written in C. Load the C source code into the file window, and the EXE file into the program window. As you step through the EXE code, you can position the file window to show the original source code that corresponds to your object. Since the symbols in the source and the disassembly are the same, you will find it possible to debug completely on the screen, without referring to a listing. This approach is particularly useful when you are looking for places to optimize in assembly language.

One particularly impressive feature of Pfix86 Plus is that it will allow you to use symbolic debugging with overlays, whether or not the overlay you're referencing is in memory. The debugger keeps track of the overlays and brings them in when they're needed.

Documentation

You can read through Pfix86's documentation in 20 minutes and learn all you have to know to use the program. Or, you can fire up Pfix86 and refer to the documentation to work through the sample debugging session printed in the back. This sample will teach you 90% of what you need to know to use Pfix86, in about 10 minutes. The manual is clear, concise, and presents all necessary information in the proper order. It doesn't tell you anything you don't want to know, and tells you everything you have to know. With a table of contents, glossary, and index, the Pfix manual is also nicely produced and bound into an IBM-size binder. Something must be happening to the software business—a debugger manual more professional than most wordprocessing manuals were only a couple of years ago!

```
@BOBreakpoint@RO Disk Evaluate Format I/O Memory Program Quit Screen
Tracing Window
   Clear,Disable,Enable,Initialize,Set,Temporary
```

Figure 2. The main Pfix menu as it appears on the two lines of the screen.

Pfix86 *Continued from page 95*

```
1:CMND 2:ORIGIN 3:EXCH 4:WINDOW 5:TRACE 6:PROCEED 7:GO 8:EVAL
9:SCREEN 10:BKPT
```

F1	displays the main command menu, as described above.
F2	sets the origin of a display window. To set the PROGRAM window to display a disassembly from location 200, you would type "200<F2>". The DATA and STACK window origins can also be set.
F3	changes to another area in a window. This key toggles between the byte displays and their ASCII or assembly source in the DATA and PROGRAM windows. Byte, word, long word, or string values can be entered directly into the byte display portions of the DATA or PROGRAM windows.In the STACK window, F3 changes the order of the listing from high-to-low to low-to-high stack addresses.
F4	cycles through the DATA, PROGRAM, BREAKPOINT, CPU, and STACK windows, making each one in turn the active window.
ALT F4	Selects the next sub-window. There may be any number of PROGRAM or DATA windows, but only one PROGRAM and one DATA window can be displayed at a time. If you are monitoring different areas of code and data, you can set different subwindows on each area, and toggle between these "subwindows" with the ALT-F4 key.
F5	Causes a single-step trace.
F6	Traces one instruction, skipping all LOOPs, JCXZs, REPs, CALLs, and INTs.
F7	Continues execution after a breakpoint, or starts execution after a program has been loaded.
F8	Evaluates an expression. To see the value of [BX+SI]-OCDF, type "[BX+SI]-OCDF<F8>". Since the value is displayed in the current radix, F8 can also be used for quick radix conversions. If the current radix is hex, typing "20.<F8>" will display 014, which is the hex value of 20 decimal.
F9	Shows the application program screen. Pfix-86 traps the screen output of any program running under its control. Thus, a program which does screen output will not overwrite the Pfix-86 windows, but you can see what its screen looks like at any time by pressing F9.
F10	Toggles a temporary breakpoint at the current instruction in the PROGRAM window. The breakpoint is marked in the PROGRAM window by an asterisk.

Figure 3. A list of function-key command shortcuts.

At Last!

The Premier Data-Comm Package from the CP/M World is Now Available for MS-DOS & CP/M-86

MITE

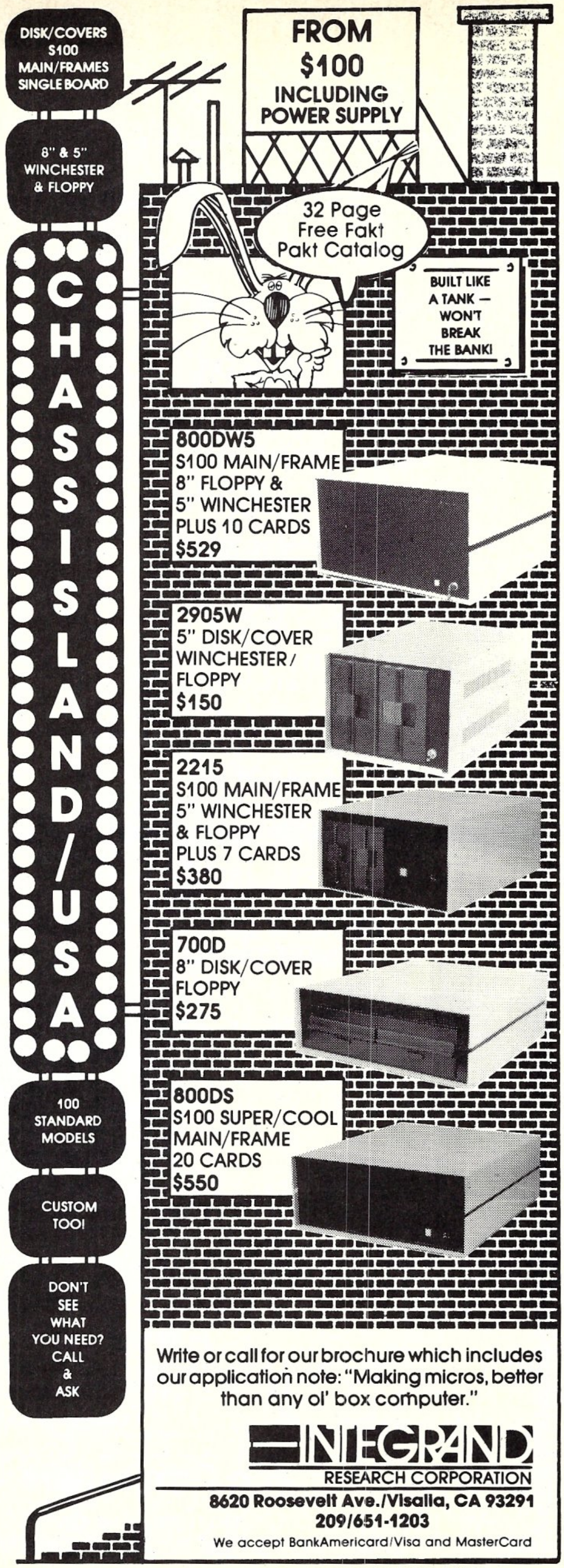
MITE allows access to virtually any information utility such as CompuServe and Dow Jones. MITE can exchange files with a large number of mainframes and microcomputers. MITE is pre-configured for over 100 microcomputers and supports most popular modems. MITE features an easy-to-use menu system and supports multiple protocols.

2639 North Monroe Street
Box 68 Suite B-188
Tallahassee, FL 32303
Mailing: P.O. Box 6045
Tallahassee, FL 32314 • Telephone (904) 385-1141

MYCROFT LABS INC

Dealer and distributor enquiries encouraged.

CIRCLE 171 ON READER SERVICE CARD



CIRCLE 49 ON READER SERVICE CARD

Pfix86

Continued from page 96

Limitations

Every debugger has limitations, and it is to Phoenix Software's credit that a chapter in the manual is devoted to explaining Pfix86's limits clearly. The limitations are mainly due to the fact that Pfix86 allows free access to all memory locations, disk sectors, and I/O ports in the system. The manual warns against overwriting the FAT sectors on the disk, writing over the DOS interrupt vectors, or writing to the disk controller's I/O ports—all are likely to make a cold boot necessary.

Another limitation is part of the design of every software debugger. On the 8086/88, all software debuggers use one of the interrupt vectors for trapping. Pfix86 uses Interrupt 1 for tracing and Interrupt 3 for breakpoints. Programs which set these interrupts will cause Pfix86 to loose control. There is no mechanism for changing vectors.

The basic design of Pfix86 makes it pleasurable and efficient in use. Its flaws are sins of omission rather than design flaws. Here's a nitpicker's list of enhancements for the next version:

- There is no simple way to append symbols that you've added by hand inside Pfix86 to the symbol table on disk. All you can do is disassemble the whole program to disk.
- The use of symbols in the PROGRAM window should be extended to immediate loads as well as JUMPS and CALLS.
- There should be a command to free all available memory. This would simplify the running of a copy of COMMAND.COM inside Pfix86, to do DIR, REN, etc. It would also make it easier to debug multiple programs in succession, without running out of memory. The current version of Pfix86 requires you to manually do a DOS call 47 in order to run COMMAND.COM from within Pfix86.
- Currently, only one data format applies to all DATA windows and subwindows. It would be more useful to allow each window to have a different format—one data window could then display byte format, while another could display long word format.

Summary

Pfix86 supplies the software developer with all the functionality required for debugging the vast majority of code. Its attention to ease-of-use and keystroke shortcuts means shortened debugging time, and the inclusion of symbolic debugging in a standard MS-DOS environment is a significant advantage. While some improvements could be made in Pfix's consistency in the use of radices and symbols, these are features which are hard to find at all in any other similar product. Pfix86 in its current incarnation is an advance over most software debuggers available for MS-DOS/PC-DOS.

Pfix86, $195. Pfix86 Plus (all features of Pfix86 plus symbolic debugging), $395. Plink86 (required for symbolic debugging with Pfix86 Plus), $395.

Requires DOS 2.x. Special version needed for DEC, TI, and Wang machines.

For more information, contact *Phoenix Software Associates Ltd., 1416 Providence Highway, Suite 220, Norwood, MA 02062 (800) 344-7200.* In Massachusetts, *(617) 769-7020*

CIRCLE 326 ON READER SERVICE CARD

Michael Olfe, 400 Rutgers St., Rockville, MD 20850

```
SAVEREGS:
          PUSH  CX                                PUSH CX
          PUSH  SI                                PUSH SI
          PUSH  DI                                PUSH DI
DIALUP:
          CALL  FAR DRAWSCREEN                    CALL 3146:00C3
          CALL  FAR DIALNUMBER                    CALL 3159:0005
          CALL  FAR SENDLOGON                     CALL 182C:6E9D
LOOP:
          CALL  FAR CHECKLINE                     CALL 3159:0014
          CALL  FAR CHECKKBD                      CALL 30CC:0083
          CALL  FAR UPDCLOCK                      CALL 3146:00B4
          JMP   LOOP                              JMP  00A9
```

Figure 4. Two code fragments showing symbolic names and hex addresses.

The MS-DOS PROMPT Command has more Power than You Think

Learn what PROMPT can do

by Charles Petzold

We usually think of *using* a disk operating system rather than *programming* it. But DOS, on most of today's computers, offers many powerful and often overlooked programming tools. PC DOS 2.0 and later, for example, lets you change the system prompt to specify anything you like to replace it. Consider the following:

```
Enter a DOS command ==>
```

Not impressed? Perhaps not, if that's all the command could do, but this command can do much more than put a substitute for A > on the screen. It can also:

- Display the time and date
- Display the current directory path (for tree-structured directories)
- Put this information anywhere on the screen
- Change the color/graphics display mode and color (or monochrome display attributes)
- Redefine the keyboard.

It's all a matter of understanding PROMPT, and the intricacies of DOS.

The PROMPT syntax

The PROMPT command is described on pages 10-18 through 10-20 of the DOS manual. However, as the manual indicates, it is an *internal* rather than an *external* command. The general format of the command is:

```
PROMPT prompt-text
```

where *prompt-text* stands for a wide range of commands. Executing this command in DOS makes the text following the word PROMPT the new system prompt, until you change it or boot up again. Only one blank between PROMPT and the prompt-text will become part of the prompt, but any number of trailing blanks following the prompt-text (before you press the Enter key) are included.

The word PROMPT, with no prompt-text following, resets the system prompt to the default drive and right angle bracket—what we are accustomed to seeing.

How long can the new prompt be? Normal command processing accepts strings of 128 characters, including the final return. This rule would allow a prompt of 120 characters (128 minus 6 for the word PROMPT, 1 for the space after the word PROMPT, and 1 for the after the word PROMPT, and 1 for the final Return). It is possible to create a longer prompt if the command is a line in a batch file.

The prompt becomes part of the command processor's environment. The environment memory (and hence any prompt that you set) can be viewed with the SET command if you include no parameters. If you have executed any programs that remain in memory (such as MODE, PRINT, ASSIGN, or GRAPHICS), you may have limited your environment storage. Using a long prompt may prevent you from specifying a long path name. Usually, the message OUT OF ENVIRONMENT SPACE results, but I've also encountered crashes with no messages when attempting to set long prompts.

Meta-strings

Certain characters cannot be used directly in prompt-text. These characters are the angle brackets, '<' and '>', the equal sign, '=', and the vertical bar, '|'. DOS uses these characters for other things, such as redirection of standard input and output. The equal sign works if it is not the first character of the prompt. Spaces, commas, semicolons, and equal signs are all ignored when they are the first character of the string.

To use such characters in the prompt, you can define *meta-strings* to take their places. These meta-strings consist of a dollar sign, '$', followed by a letter or symbol. The letter may be upper or lowercase. The PROMPT meta-strings are shown in Figure 1. To use the right angle bracket, for example, enter the meta-string $G. Thus, the command

```
PROMPT Enter a DOS Command--$G
```

causes the prompt shown at the beginning of this article. Since the dollar sign is used for these meta-strings, it takes two dollar signs to put one in a prompt display.

If you want to create a prompt that begins with a blank, comma, or semicolon, you must begin the string with a null character. Null characters are created by combining a dollar sign and any character not used for any other meta-string, such as X. For a prompt that begins and ends with three blanks, try

```
PROMPT $X   System Prompt   $X
```

Note the three spaces after the first $X and before the second.

For a noisy prompt, type Ctrl-G in the prompt-text. It will appear on the screen as ^G. But instead of showing up when the prompt is displayed, your PC will beep. This beep can be used as a signal that a long DOS job is done.

Displaying system information

The second group of meta-strings displays current information about the system. These allow you to show the current default drive (the normal prompt), the current directory path (if you are using tree-structured directories), the date, time, and version number.

How long can a new prompt be? Normal command processing accepts 128 characters.

The current directory path is what the CHDIR (or CD) command produces without parameters; the version number is the text that the VER command displays. For instance, the command

```
PROMPT $P$G
```

will display

```
A:\>
```

if drive A: is your default and you are in the root directory. If you are not in your root directory, then the directory path will be displayed:

```
A:\LEVEL1\LEVEL2>
```

The above display gives you a constant visual reminder of the current directory path, but the system must access the disk before displaying a directory-path prompt. This may cause a delay, depending on how deeply embedded in subdirectories you are. If your default drive is a diskette rather than the fixed disk, this frequent disk access may be annoying.

Other meta-strings position the cursor as part of the PROMPT. The cursor can be backspaced (with an erasure), or a carriage return/linefeed can send the cursor to the beginning of the next line.

The $H meta-string backspaces and deletes a character. You may want to combine this feature with the time display to get rid of the hundredths-of-seconds. For instance, the command

```
PROMPT $T$H$H$H$G
```

will produce the prompt:

```
10:52:30>
```

Three more $Hs will get rid of the seconds as well.

The meta-string $_ is a carriage return/linefeed sequence. The new prompt is displayed up to the $_, and then the cursor drops to the beginning of the next line, where anything after the $_ is displayed. The command

```
PROMPT OK$_
```

displays the same prompt you get in Basic—the OK prompt—and waits for input on the next line. This prompt may not be a wise choice since it looks like Basic, but you're still in DOS. Similarly confusing prompts should be avoided unless you have a special application for them.

Finally, the $E meta-string displays the ASCII Escape character (Hexadecimal 1B). Normally, this character shows up on the screen as a tiny left arrow. However, the ASCII Escape character, with the help of the DOS ANSI.SYS file, opens up a whole world of options with PROMPT.

The ANSI.SYS connection

ANSI.SYS was introduced in version 2.0 and is described in Chapter 13 of the DOS manual. The file includes a series of video control routines that begin with an Escape code. Since ANSI.SYS assists DOS in driving the display, it functions as a *device driver*—also new in DOS 2.0—a concept covered in Chapter 14. These chapters are two of the most arcane in the DOS manual, but using ANSI.SYS with the PROMPT command is simpler than these chapters imply.

ANSI.SYS is loaded from a CONFIG.SYS file when DOS is booted and becomes a part of DOS. You can create the necessary CONFIG.SYS file with the COPY command. Type

```
copy con: config.sys
```

press the Enter key, then type

```
device = ansi.sys
```

and push Enter, F6 (or Ctrl-Z) for End-of-File, and the Enter key again.

The boot disk must contain this CONFIG.SYS file and the ANSI.SYS file. Now, the $E meta-string gives you the whole range of extended screen con-

PROMPT

Continued from page 101

trol features documented in Chapter 13 of the DOS manual. ANSI.SYS can interpret any command string that begins with an ASCII Escape character and either position the cursor, move it around, set display attributes and colors, or redefine the keyboard.

Some of the more useful of these screen control sequences are shown in Figure 2. Notice that some characters are uppercase and some lowercase. This is an important distinction. ANSI.SYS interprets uppercase and lowercase characters differently.

Try something simple:

```
PROMPT $E[s$E[0;0H$T$E[K$E
    [u$N$G
```

It may not appear simple, but it is. Every $E meta-string is converted to an ASCII Escape code when the prompt is displayed, and thus indicates to the ANSI.SYS file that a control sequence is present. The $E[s ANSI.SYS string saves the current cursor position so you can put it back where it belongs after moving it around the screen. Next, $E[0;0H moves the cursor to position 0,0 on the display—the upper left-hand corner. The $T meta-string prints the time, while $E[K erases the rest of the top line, $E[u restores the cursor, and NG prints the normal prompt.

When DOS tries to display this prompt, the $E meta-string first sends an Escape character to the screen. ANSI.SYS recognizes the Escape code, intercepts the characters that follow, and carries out the command. The time display in the corner of the screen will be updated only when DOS displays the next prompt. In other words, it simply indicates the time when DOS last displayed the prompt.

If this command doesn't work and you see some tiny left arrows on the screen, then ANSI.SYS probably hasn't been loaded. You can find out by using the CHKDSK command to display the amount of available memory, since ANSI.SYS is a load-and-stay resident module.

Putting the time on the top of the screen creates a primitive status line. These status lines are fairly easy to create and customize with the PROMPT command, as long as you accept their limitations. Put status lines at the top rather than at the bottom of the display so that, when the screen scrolls, the previous status line will roll off the top and be replaced by the next one.

System status lines

Here's a prompt that displays all available status information on the top four lines of the display:

```
PROMPT $E[s$E[0;0H$V$E[K
$_Directory:$P$E[K
$_Date:      $D$E[K
$_Time:      $T$E[K$E[u$N$G
```

The $E[s saves the current cursor position (to be restored later). The $E[0;0H portion sets the cursor at line 0, column 0 (the upper left-hand corner). The $V meta-string prints the version line, and the $E[K sequence erases the rest of the line. The next three sequences begin with a carriage return/linefeed ($_) so they'll each have a line of their own on the display. Spaces are shown after Date and Time, so the information lines up nicely. Remember that the $E[K erases the rest of the current line. The cursor is restored from the initial save by the $E[u sequence. Finally, back where the prompt should be, the NG sequence shows the familiar

Meta-String	Definition Special Characters
$B	The ":" character.
$G	The " > " character.
$L	The " < " character.
$Q	The "=" character.
$$	The "$" character.
	System Information
$D	The date (14 characters: 3 character day-of-week, blank, 2 character month, dash, 2 character day, dash, 4 character year.)
$T	The time (11 characters: 2 digit hour, colon, 2 digit minutes, colon, 2 digit seconds, point, 2 digit hundredths-of-seconds).
$N	The default drive (1 character).
$P	The current directory path of the default drive (begins with default drive, colon, then a maximum of 63 characters of the path from the root to the current directory).
$V	The version number (currently prints 39 characters. "IBM Personal Computer DOS Version 2.00").
	Cursor Control
$H	Backspace and erasure of the previous character.
$_	(underline) A carriage return and line feed sequence.
	Other ASCII characters
$E	The ASCII Escape character.
$X	(where X is anything not used above) A null string.

Figure 1. PROMPT meta-strings.

Control String	Definition
$E[s	SCP (Save cursor position for a future RCP sequence)
$E[u	RCP (Restore cursor position from a previous SCP sequence)
$E[#;#H	CUP (Cursor position—sets cursor to the row and column substituted for the first and second # respectively)
$E[#A	CUU (Cursor Up # positions)
$E[#B	CUD (Cursor Down # positions)
$E[#C	CUF (Cursor Forward # positions)
$E[#D	CUB (Cursor Backward # positions)
$E[2J	ED (Erases whole display and homes cursor to upper-right hand corner)
$E[K	EL (Erases from cursor to end of line)
$E[#;...#m	SGR (Set graphics rendition—see page 13-8 of the DOS manual, S$E[7m will turn on reverse video; $E[0m will go back to normal.)
$E[0;#;"s" ;13p	Reassign Extended ASCII Keyboard Key # to the command string "s" followed by a carriage return. See page 13-10 of the DOS manual for more information.

Figure 2. Screen control sequence.

default drive and right angle bracket. For clarity these commands are shown on separate lines, but in reality this PROMPT sequence must be entered on a single line.

If you'd like this status information to stand out, try using reverse video or some other attribute with the control sequences shown on page 13-8 of the DOS manual. The $E[7m sequence turns on reverse video. Anything that follows will be printed black on green (on a monochrome display). The $E[0m sequence restores normal video, unless you want *all* DOS displays in reverse video. A $E[5m sequence makes characters blink. If you have a color display, you can switch background and foreground colors.

If you leave out the version number, you can put the current directory, date, and time all on one line at the top of the screen, provided that your directory path is 10 characters less than the maximum of 63.

My favorite prompt puts the date and time in the upper right-hand corner of the display in reverse video and replaces the normal prompt with the current directory path:

```
PROMPT $E[s$E[1;51H$E[K$E
     [7m$D/$T$E[0m$E[u$P$G
```

Once you find a prompt you like, put it in AUTOEXEC.BAT so that the prompt will be set every time you boot up. You can do this either by putting the prompt directly in the AUTOEXEC.-BAT file, or by putting the prompt in a batch file (mine is called PROMPTST.-BAT) and then putting the name of the batch file in AUTOEXEC.BAT.

Keyboard Reassignment

An ANSI.SYS command sequence that permits keyboard reassignment is described on page 13-10 of the DOS manual. The general format of this control sequence is:

```
$E[key;definitionp
```

The *key* specification is either the ASCII code in decimal for the key you are redefining—'65' for capital A, for example—or a zero followed by a semicolon followed by an extended ASCII key number. Extended ASCII key numbers are used for keys that do not have normal ASCII definitions. These include the function keys, cursor movement keys, and some others. The *definition* specification is what the key will become. This can be more than one character long and may be specified in pieces. The pieces are separated by semicolons. They may be either ASCII codes in decimal, or text in double quotation marks. The keyboard redefinition is terminated by lowercase p.

ANSI.SYS will store the reassignment and use it when the redefined key is pressed. Once a key is redefined, it stays redefined until you change it or boot up again. The PROMPT command that redefines a key can be used only once. The key is then redefined, and you can specify a different prompt to control what is displayed on the screen.

These keyboard reassignments will not work in any program that bypasses DOS to get keyboard information, such as the Basic interpreters. These programs go straight to the BIOS for their keyboard information. You'll have to experiment with various programs to find out which ones will accept the redefined keys.

The redefined keys will be the one most frequently used for DOS commands, following the operating system prompt. When you see the prompt, the COMMAND.COM file is running, and that program only uses DOS keyboard information.

What keys should you reassign? The function keys are your best bet. For instance, if you have a number of batch files set up, you can define function keys to run them. If you find yourself typing the same command over and over, let a

PROMPT

Continued from page 103

function key do the job.

You probably do not want to use functions keys F1 through F6. These keys have special meaning to DOS (See Chapter 2 of the DOS manual). If you think that leaves only four function keys, guess again. The 10 keys are really 40 function keys, since they can be pressed in combination with the Shift, Ctrl or Alt keys. So, you actually have 34 function keys left if you don't use F1-F6. The extended keyboard numbers for these keys are shown in Figure 3. If that's not enough, you can try using the Alt key in combination with letters or the top row of number keys. These are also shown in Figure 3. Other possibilities are noted in the *Technical Reference Manual.*

If you want to define F7 to run a directory of the A: drive, type:

```
PROMPT $E[0;65;"Dir A:";13p
```

The entire prompt-text is used for the keyboard reassignment, so nothing is displayed on the screen after this prompt sequence has been executed. The prompt-text consists of an Escape character, a left bracket, a zero (to indicate the use of an extended keyboard number), the number from Figure 3, another semicolon, the redefinition, another semicolon, a 13 (ASCII carriage return), and a lowercase p which tells ANSI.SYS that this sequence is a keyboard reassignment.

After the key has been reassigned, you can type PROMPT without an argument to restore the regular prompt. Pressing the F7 key causes the *"Dir A:"* to be treated as a command.

Reassign with batch files

You can reassign keyboard keys more conveniently with a two-line batch file called KEYDEFIN.BAT:

```
PROMPT $E[0;%1;"%2 %3 %4";13p
PROMPT
```

The first line is the PROMPT command to reassign a keyboard key. The percent signs followed by numbers are replacable parameters of the batch file. The second line restores the prompt to normal. Instead of executing the PROMPT command in the second line, you may want to put in the name of the batch file where your favorite prompt is stored.

Note that the Echo feature must be *on* for this file to work. DOS must actually try to display the prompt for ANSI.SYS to intercept it and reassign the keys.

To use this batch file, try:

```
KEYDEFIN 66 DIR B:
```

This sets %1 to 66, %2 to DIR, %3 to B:, and %4 to nothing, resulting in the following set of commands:

```
PROMPT $E[0;66;"DIR B:";13p
```

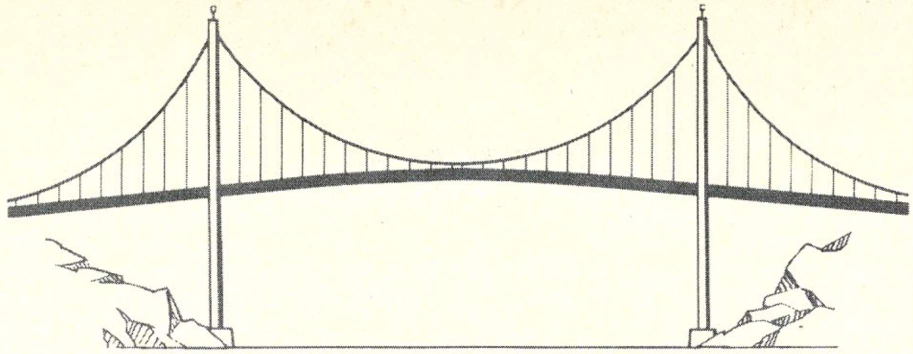
Funct. Key	Extended Keyboard No. Base	Shift	Ctrl	Alt
F1	(59)	84	94	104
F2	(60)	85	95	105
F3	(61)	86	96	106
F4	(62)	87	97	107
F5	(63)	88	98	108
F6	(64)	89	99	109
F7	65	90	100	110
F8	66	91	101	111
F9	67	92	102	112
F10	68	93	103	113

( ): Used for DOS line editing

Top row of keyboard	Extended Keyboard No. in combination with ALT
1	120
2	121
3	122
4	123
5	124
6	125
7	126
8	127
9	128
0	129
.	130
=	131

Letter Key	Extended Keyboard Number with ALT	Letter Key	Extended Keyboard Number with ALT
A	30	N	49
B	48	O	24
C	46	P	25
D	32	Q	16
E	18	R	19
F	33	S	31
G	34	T	20
H	35	U	22
I	23	V	47
J	36	W	17
K	37	X	45
L	38	Y	21
M	50	Z	44

Figure 3. Extended keyboard numbers for F7-F10.



PELADA TEXT ENGINEER

The first full featured word processor for the IBM personal computer capable of producing all mathematical, scientific and engineering symbols* including superscripts and subscripts for complete equation editing on screen.

Multi-lingual capabilities include the specific characters, accents and letters found in French, Spanish, German, Italian and all other western European languages.

Advanced Word Processing features include:

- Command line driven editor
- Comprehensive, easily accessible on-line help
- All features, including bold, italics, underlining and special characters appear on screen exactly as they are printed.

PELADA INFORMATICA INC.

87 Queen St., Kingston, Ontario,
Canada K7K 1A5
Tel. (613) 549-1747

*Requires Epson FX-80, FX-100 or similarly featured printer.

FREE SAMPLE DISK AVAILABLE UPON REQUEST.

CIRCLE 122 ON READER SERVICE CARD

PROMPT

Continued from page 104

appear on the command line and displays the directory.

By selecting extended keyboard numbers from Figure 3, you can define keys to activate your most-used commands. Note that the number of batch parameters you want the redefined key to accept is equal to the number of items (separated by blanks) within the quotation marks.

For example, the batch file discussed above can't be used to define a key that accepts four parameters. In order to do that, the command must be:

```
PROMPT $E[0;%1;"%2 %3 %4
       %5";13p
```

because the %1 parameter selects the key to be redefined. You can even go all out and try:

```
PROMPT $E[0;%1;"%2 %3 %4 %5
              %6 %7 %8 %9";13p
```

but since the parameters %2 through %9 must be separated by blanks (to preserve the spacing in the key reassignment sequence), this command adds unnecessary characters to key definitions that have less than eight parameters.

Because there's a zero after the left bracket, this batch file can only reassign keys with extended keyboard numbers. Another batch file without the zero and first semicolon would handle definitions for regular ASCII codes.

Without the semicolon followed by the number 13 in the PROMPT statement, the cursor will stop after typing DIR A: and wait for more input or the Enter key. The ';13' sequence puts in the carriage return. If you prefer the option of adding something to the line, then leave out the carriage return (but don't leave out the final lowercase p). If you want the function key to deliver several commands, separate each of them by a semicolon, a '13,' and another semicolon. The sequence inserts carriage returns between the commands.

You can put your favorite keyboard reassignments in an AUTOEXEC.BAT file to be set during the system boot. Each redefinition requires one PROMPT command. After all the key redefinitions are done, a final PROMPT command or the name of another batch file can set the display prompt.

You also can set up batch files to define particular keys for special functions, but with a parameter to define a filename. For instance, an assembly language programmer may want to define function keys that assemble, link, run an EXE2BIN utility, and then load DEBUG—with the same program name. A batch file to do this reassignment (named QUICKASM.BAT) can be:

```
PROMPT $E[0;65;"MASM %1";13p
PROMPT $E[0;66;"LINK %1";13p
PROMPT $E[0;67;"EXE2BIN %1;
       %1.COM";13p
PROMPT $E[0;68;"DEBUG
       %1.COM";13p
```

Then, using the command

```
QUICKASM progname
```

you can assemble the program by pressing the F7 key, link it with the F8 key, change it to a .COM file with the F9 key, and load it into DEBUG with the F10 key.

Not all programs get keyboard information from the standard DOS function calls. Also ANSI.SYS reserves only 200 bytes for the storage of keyboard reassignments—when your system crashes, you know you've defined one key too many.

However, you can use DEBUG to patch the ANSI.SYS file to allow more than 200 bytes of keyboard reassignments.

Charles Petzold, 63 E. 9th St., Apt. 10N, New York, NY 10003

Implementing X.25 Communications Protocol

Part 3: Intel 8274 and Western Digital WD2511 chips

by Eric L. Beser

In the last issue we described the implementation of X.25 with the aid of the Intel 8273 chip. In this last part of the series, we cover the Intel 8274 and Western Digital WD2511 chips.

8274 Multi-Protocol Serial Converter (MPSC)

The 8274 MPSC is the Intel version of the Z80-SIO and has the capabilities necessary for X.25. A comparison between the 8273 and 8274 shows that the 8274 is a much more complicated chip that can handle almost any method of communication. The 8273, on the other hand, is a more specialized device, used only for frame-oriented communication. The advantages of using the 8274 are that the 8274 is cheaper and easier to find, and that the registers are bit-for-bit compatible with the Z80-SIO. This means that the software drivers will work for the Z80-SIO without modification. Additionally, the 8274 has two channels that operate independently of each other, which means that one channel may be used for asynchronous transmission while the other channel is used for synchronous.

Figure 1 shows the block diagram for this device. If you are familiar with the register structure of the Z80-SIO, then you already know that of the 8274—they are identical. Figure 2 describes these registers and illustrates the function of each bit. The essential difference between the Z80-SIO and the 8274 is in the hardware interfacing. The 8274 is designed for Intel's family of microprocessors (8085, 8086, 8088) and requires signal translation in the read/write logic if it is to be used with a Z80 CPU. However, the interrupt capability and structure of the 8274 are identical to those of the SIO.

Interrupt structure

The 8274 has a very powerful interrupt structure that involves multiple sources of interrupts, the priorities being resolved within the device. Figure 2 shows the interrupt structure. Interrupt sources can be divided into three categories: receive, transmit, and external/status interrupts. The relative priority of the various interrupts is software-controllable and may be set by writing a 1 or a 0 into register 2, bit 2. The CPU sees only one interrupt, and the priority is resolved internally. However, the device may be set to do mode 2 interrupts (vector table) identical to those of the Z80-SIO.

Receive interrupt. In HDLC mode, this interrupt occurs on receipt of an address field that matches a preprogrammed address. In general receive using DMA, it signifies the start of a valid frame. With a DMA device, the procedure is minimal. Without the DMA, this procedure would interrupt on every character, and would read a byte into a buffer.

The special receive condition interrupt occurs at the end of the frame and may be used to set up the DMA, redefine the buffer for the next frame to be received, and alert the operating system. This procedure would analyze the contents of the results to determine if any error conditions exist that would have also caused this interrupt. These error conditions would be parity errors, buffer overrun (caused by interrupts not being serviced in time), framing error, or aborted frame.

Transmit interrupt. The transmit buffer empty condition generates a transmit interrupt. In DMA mode, this condition generates a DMA request and is transparent to the CPU. In a non-DMA mode, this interrupt would get the next character to transmit or reset if there are no more characters left to transmit.

Exernal status interrupt. Once the transmission of data has ended, the 8274 makes another DMA request, which will go unfilled. This request causes an underrun condition that interrupts the CPU. The reset of the external status command register causes the frame check sequence to be transmitted. It is important to reset the external status command directly after transmitting the first byte to ensure the transmission of the frame check sequence rather than interfill sync bytes.

The external status interrupt occurs because of the underrun condition, but it also occurs due to changes in the status of modem control lines. To determine the state of the DCD or CTS lines before the interrupt occurs, the register must be read before you reset the interrupt. Likewise, to determine the status after the interrupt occurred, the register may be read again after you reset the interrupt.

Module L1_8274

Shown in Listing 1 is the library of level 1 drivers for the 8274. As in all the other libraries, the procedures pertain only to the physical level functions.

Init_8274. This procedure initializes the 8274 in HDLC mode for channel A. Channel B is not used in our example. Note that register 4 is initialized before any of the other transmit or receive parameters, per Intel instructions. Channel A has been programmed for DMA operation.

Init_8274_HDLC. Upon acknowledgment of an 8274 interrupt, execution starts with this routine. Status register 2 is read, which contains the vector into the cause of the interrupt. Depending on the value of the bits (0-7), the appropriate interrupt routine is called.

The 8274 has a structure that involves multiple sources of interrupts.

Rx_A_8274. This routine is called after reception of the first character in DMA mode. Since the transfer mode is DMA, we have a maximum of three character times in which to service this interrupt and enable the DMA device. At very high data rates (400 Kbaud), there may not be enough time to read register 2 and enable the DMA controller without overrunning the receiver. In this case, the DMA controller would have to be pre-enabled. If non-DMA mode were to be used, this routine would have to read the character into a buffer.

L1_Xmit_Frame. This is the interface procedure between level 2 and level 1. Its purpose is to initialize the DMA device and frame pointers for transmission, transmit the first character (the address field), reset the external status register (remember to ensure that the frame-check sequence is transmitted when the information field has completed), set the Tx_In_Progress semaphore, and then exit.

Cha_External_Status_Change. This procedure is called when the transmit buffer underruns, or when external changes occur in the modem controls. If, in fact, transmission ended, the Tx_In_Progress semaphore is reset and the event is noted in the frame information field.

If the interrupt had occurred for other reasons, that too is noted and the appropriate error handler is notified through the operating system. If the frame was transmitted normally, the operating system is notified to release other processes from hibernation waiting for transmission to complete.

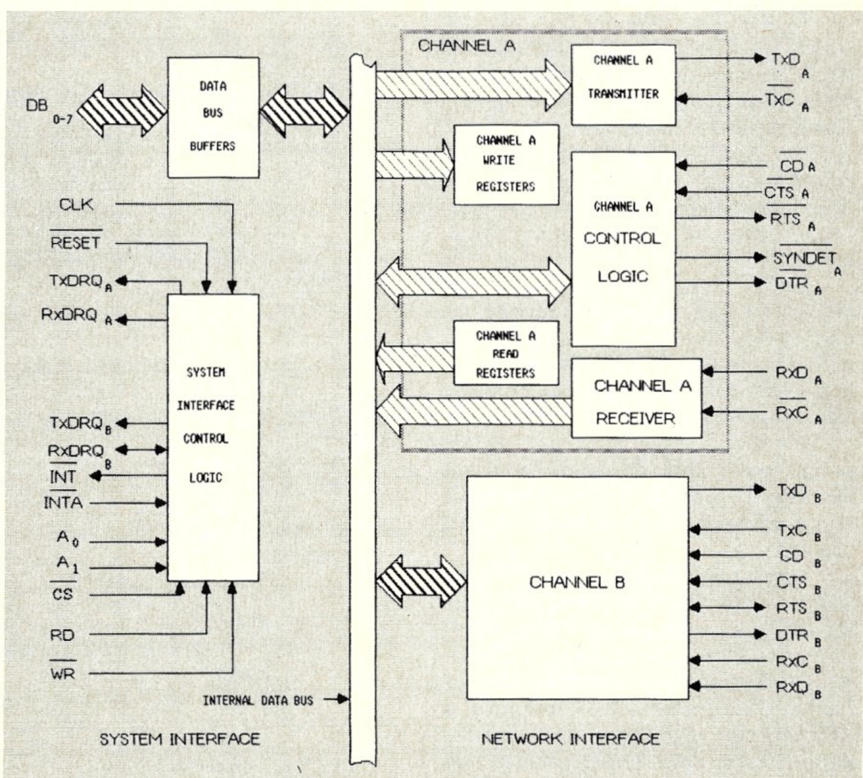


Figure 1. 8274 block diagram.

X.25

Continued from page 000

Rx_A_Special_8274. This procedure is called when the end-of-frame condition occurs while receiving a frame. It may also occur due to receive errors, and these errors must be taken into account. The status byte is read from register 1 and, in event of error, the operating system is notified. However, if all is well, the received frame is noted to the operating system, and the DMA device is preinitialized for the next frame by calling Start_Rx.

Start_Rx. Used for housekeeping purposes, this procedure sets up the DMA device with the proper address of the current buffer or local buffer before the first frame is received or at the conclusion of the end-of-frame interrupt.

LSI circuit simplifies X.25 interface

One of the benefits in the development of large-scale integrated (LSI) circuitry is the ability to squeeze more code and hardware in smaller spaces. The WD2511 Level 2 Controller is no exception. In a 48-pin package, you will find a microcontroller with 11K of ROM, DMA circuitry, and line interface with PLL and TTL level conditioning. What we have been discussing in terms of hardware and software (library included) can fit into this chip. Thus external timers, memory latches, and the system software (about 1000 lines of code) can be eliminated.

One of the tradeoffs that you get with this type of controller is the lack of flexibility in protocols obtainable from other chips mentioned in this article. The firmware is restricted to the level 2 X.25 protocol. On the other hand, this controller starts the link, keeps track of the retransmission of frames, and disconnects the link—which other controllers do not do.

DMA circuitry included in the WD2511 is quite simple to interface, requiring only three pins for handshaking with the processor bus. Transfers are fast, often occurring in a single cycle, and are embedded in the CPU's clock cycle in such a way that they are transparent to the CPU. It is necessary to do this because the WD2511 must transmit and receive data simultaneously on two channels.

Additional hardware interfacing is quite simple. With the WD2511, one may build a bisync (or any other protocol) to X.25 converter using a Z80, a 6164 16K RAM chip, an EPROM, four latches used in the DMA interface, a Z80 SIO (for the bisync) or a PIA for parallel async data. This converter may be wirewrapped on a 22-pin vector board.

Software interfacing of this chip is as simple as the hardware connections. Two main data structures must be included to keep track of transmitted and received data. These structures are lookup tables and contain address and control bytes for each packet. The WD2511 is given the starting address of the transmit lookup table; the device assumes that the receive lookup table follows in contiguous memory. These tables allow up to seven frames to be outstanding in each direction, and are divided into eight segments of eight bytes.

A Pascal declaration of these lookup tables and other necessary data structures would be as shown in Listing 2.

The first data structure is the packet. This structure is an array of buffsize

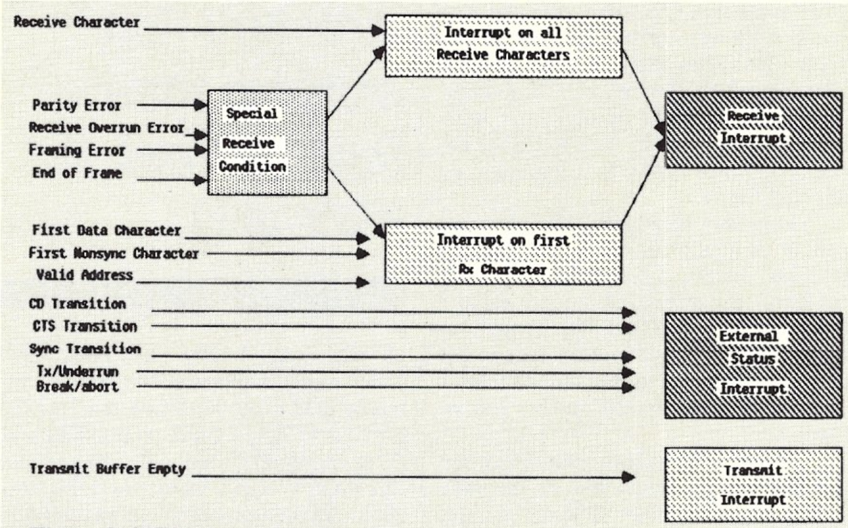


Figure 2. 8274 interupt structure.

```
                         LISTING 1
                  Finite State Machine PDL
ENTER
    WHILE event queue not empty DO
        EVENT:=decode
          COMMENT:  decode is a function that retrieves
                    events from the event queue in order of
                    occurrence or decodes the incoming frames
                    into an event code
            action:= action_table(event,current_state)
            PERFORM do_action(action)
    ENDDO
EXIT


                        Do_Action PDL

PROCEDURE do_action(action:integer);
BEGIN
     DO CASE action
     0: null
     1: do case 1
     2: do case 2
     3: BEGIN
          IF event good
              SET state to next state
          ELSE
              SET state to bad state
          ENDIF
        END
     |
     |
     n: do case n
     ENDCASE
END
```

LISTING 2

```
const

     looksize = 8;
     buffsize = 132;    {may vary with link quality}
          {  status bits in lookup table }
     message_acked = $80; {set by WD2511 to indicate that
                           the frame has been acknowledged;
                           check this bit before loading new
                           packet in buffer}

     packet_ready  =  $01;  { this  bit is set  by  CPU  to
                              indicate that this block may
                              be transmitted by WD2511 }
     rec_ready = $01;  { setting this bit informs the WD2511
                         that   the  buffer  is  ready   for
                         receive; reset by WD2511 when frame
                         has been received error free }
     frame_complete = $80; { set by WD2511 to indicate that
                              a frame has been received }

type
     packet_ptr = ^packet;
     packet = record
          frame : array [1..buffsize] of byte;
          chain_link : packet_ptr  {used to chain to longer
     end;                                     buffers }

     lookup = record
          status: byte;
          address,
          count : integer;
          user_defined : array [1..3] of byte
          end;
     look_table = record
          tlook : array [1..8] of lookup;
          rlook : array [1..8] of lookup;
          error_count:array [1..10] of integer
     end;

var
     lookup_table : look_table;
```

LISTING 3

```
ENTER

      RESET Device
      PROGRAM I/O
            COMMENT: Registers 1,8,9,A, B
      IF loopback then begin
            SET RTS OFF
            SET CTS ON
            RESET Receive data buffers
            SET Reg E = Reg F
      END
      ELSE
            IF dte then begin
                  SET REG E = 01
                  SET REG F = 03
            END
            ELSE begin
                  SET REG E = 03
                  SET REG F = 01
            END
      ENDIF
      IF CHAIN then begin
            load data into transmit buffers
            program amount of chain segments
                comment: reg C upper four bits
            program chain buffer size
                  comment: reg C lower four bits
```

```
            repeat
                  set all transmit xfer pointers
            until all pointers set
      END
      Program the lookup tables for the transmit data buffers
      IF loopback then
            set active loop test and recr bit
      If link = 0
            set send bit
END
EXIT
```

LISTING 4

```
FRAME COMPLETE INTERRUPT
ENTER

      READ SRI BIT REGISTER 3
      IF ((REGISTER 3 and PKR) = PKR)) THEN BEGIN
            comment:  packet  has been received error free and
                      in correct sequence
                 IF recr bit set THEN
                      reset Recr bit
                 ENDIF
      ENDIF
      IF ((Register 3 and XBA) = XBA) THEN BEGIN
            comment: packet previously transmitted
                      has been acknowledged by remote station
            IF Loopback THEN BEGIN
                  verify correct indication of the I/O
                       registers
                  verify transmitter look-up tables are
                       correct
                  verify receive look-up tables are
                       correct
                  verify receive data is correct
            END
            ELSE
                  IF Error bit set THEN BEGIN
                       READ ERROR REGISTER
                       If get_next_chain_segment then
                            If NOT(current segment xfr address
                                 loaded) THEN BEGIN
                                 read register 6 to establish
                                 which  chain segment is ' being
                                      loaded
                                 load xfr address pointer into
                                      current segment
                            ENDIF
                       ENDIF
                  ENDIF
            ENDIF
      ENDIF
      EXIT
```

bytes followed by a pointer to another buffer. The WD2511 may chain the packets together to form longer buffers. This chaining is transparent to the CPU. The second data structure is the lookup table. The first byte is a status byte whose bits indicate that the frame has been received, is ready to transmit, may be read from a buffer, or may be sent by the device. Following this status byte is a word that points to the packet. Next, the count of received or transmitted data is stored in an integer variable. The other three bytes are for user definition and may be used to contain other packet information. This record is organized as two arrays of type packet. Following the two arrays is another eight-byte array of integers which is used to store error statistics (T1 Timeout, frame retransmission, etc.).

Architecture

Figure 3 is the block diagram of this device. Mode control and monitoring of status by the CPU is performed by the read/write control circuit. This block maintains 16 registers used by the

RP/M T.M.

By the author of Hayden's "CP/M Revealed."

New resident console processor RCP and new resident disk operating system RDOS replace CCP and BDOS without TPA size change.

User 0 files common to all users; user number visible in system prompt; file first extent size and user assignment displayed by DIR; cross-drive command file search; paged TYPE display with selectable page size. SUBMIT runs on any drive with multiple command files conditionally invoked by CALL. Automatic disk flaw processing isolates unuseable sectors. For high capacity disk systems RDOS can provide instantaneous directory access and delete redundant nondismountable disk logins. RPMPIP utility copies files, optionally prompts for confirmation during copy-all, compares files, archives large files to multiple floppy disks. RPMGEN and GETRPM self-install RP/M on any computer currently running CP/M®2.2. Source program assembly listings of RCP and RDOS appear in the RP/M user's manual.

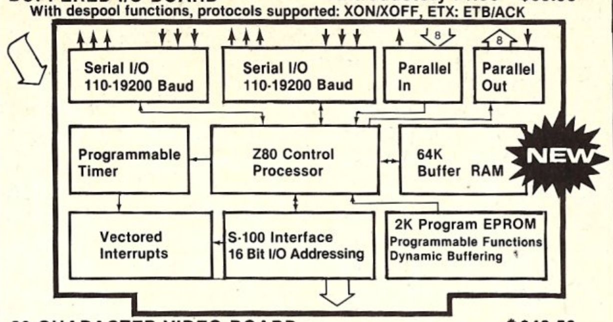
RP/M manual with RPMGEN.COM and GETRPM.COM plus our RPMPIP.COM and other RP/M utilities on 8" SSSD $75. Shipping $5 ($10 nonUS). MC, VISA.

microMethods

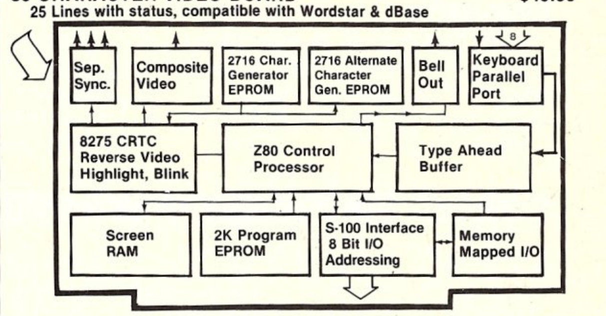
P.O. Box G, 118 SW First St.
Warrenton, OR 97146, (503) 861-1765

CIRCLE 42 ON READER SERVICE CARD

BUFFERED I/O BOARD Introductory Price *$59.95
With despool functions, protocols supported: XON/XOFF, ETX: ETB/ACK



80 CHARACTER VIDEO BOARD *$49.50
25 Lines with status, compatible with Wordstar & dBase



Includes Bareboard, Heatsink & Documentation Call or write for more information.

SPC
Simpliway Products Co.
P.O. Box 601
Hoffman Estates, IL 60195
(312) 359-7337

VISA
MasterCard

OEM dealer pricing available, $3.00 S/H, IL. Res. add 7% tax
dBase™ - of Ashton - Tate Corp. — Wordstar™ - of Micropro Int'l. Corp

CIRCLE 6 ON READER SERVICE CARD

Q/C

For only $95, Q/C is a ready-to-use C compiler for CP/M. You get complete source code for the compiler and over 75 library functions. Q/C is upward compatible with UNIX Version 7 C, but doesn't support long integers, float, parameterized #defines, and bit fields.

- Full source code for compiler and library.
- No license fees for object code.
- Z80 version takes advantage of Z80 instructions.
- Excellent support for assembly language and ROMs.
- Q/C is standard. Good portability to UNIX.

Version 3.2 of Q/C has many new features: structure initialization, faster runtime routines, faster compilation, and improved ROM support. Yes, Q/C has casts, typedef, sizeof, and function typing. The *Q/C User's Manual* is available for $20 (applies toward purchase). VISA and MasterCard welcome.

THE CODE WORKS
5266 Hollister
Suite 224
Santa Barbara, CA 93111
(805) 683-1585

Q/C, CP/M, Z80, and UNIX are trademarks of Quality Computer Systems, Digital Research, Zilog, Inc., and Bell Laboratories respectively.

CIRCLE 68 ON READER SERVICE CARD

Computer problems?

DON'T BLAME THE SOFTWARE!

Isolators prevent:

Pat. # 4,259,705

- CPU/printer/disk interaction
- Lightning or spike damage
- AC power line disturbances
- RFI-EMI interference

Commercial Grade Isolators

ISO-1 3 Isolated Sockets	$ 81.95
ISO-2 2 Isolated Socket Banks, 6 Sockets	$ 81.95

Industrial Grade Isolators

ISO-3 3 Double Isolated Sockets	$122.95
ISO-11 2 Double Isolated Banks, 6 Sockets	$122.95

Laboratory Grade Isolators

ISO-17 4 Quad Isolated Sockets	$213.95
ISO-18 2 Quad Isolated Banks, 6 Sockets	$180.95

Circuit Breaker, any model (Add-CB)	Add	$ 11.00
Remote Switch, any model (Add-RS)	Add	$ 20.00

ESP Electronic Specialists, inc.
171 S. Main St., Box 389, Natick, Mass. 01760 (617) 655-1532

Toll Free Order Desk 1-800-225-4876
MasterCard, VISA, American Express

CIRCLE 102 ON READER SERVICE CARD

Continued from page 112

software to set up and operate this chip. There are three internal microcontrollers: one for transmit, one for receive, and one to monitor the control functions. There is zero-insertion circuitry, CRC generation, a FIFO stack and DMA control lines.

Programming

Writing code to operate this device is easy. For example, an initialization sequence that initiates transmission or reception would have the PDL shown in Listing 3.

An interrupt is generated whenever the frame is sent or received. It would typically have the program logic shown in the example in Listing 4.

The PKR and XBA bits in the status register indicate that a packet has been received or a packet that had been transmitted has just been acknowledged. The other causes of the interrupt are that error conditions occurred, or that a recently received chained packet needs to have its pointer updated with the new buffer location.

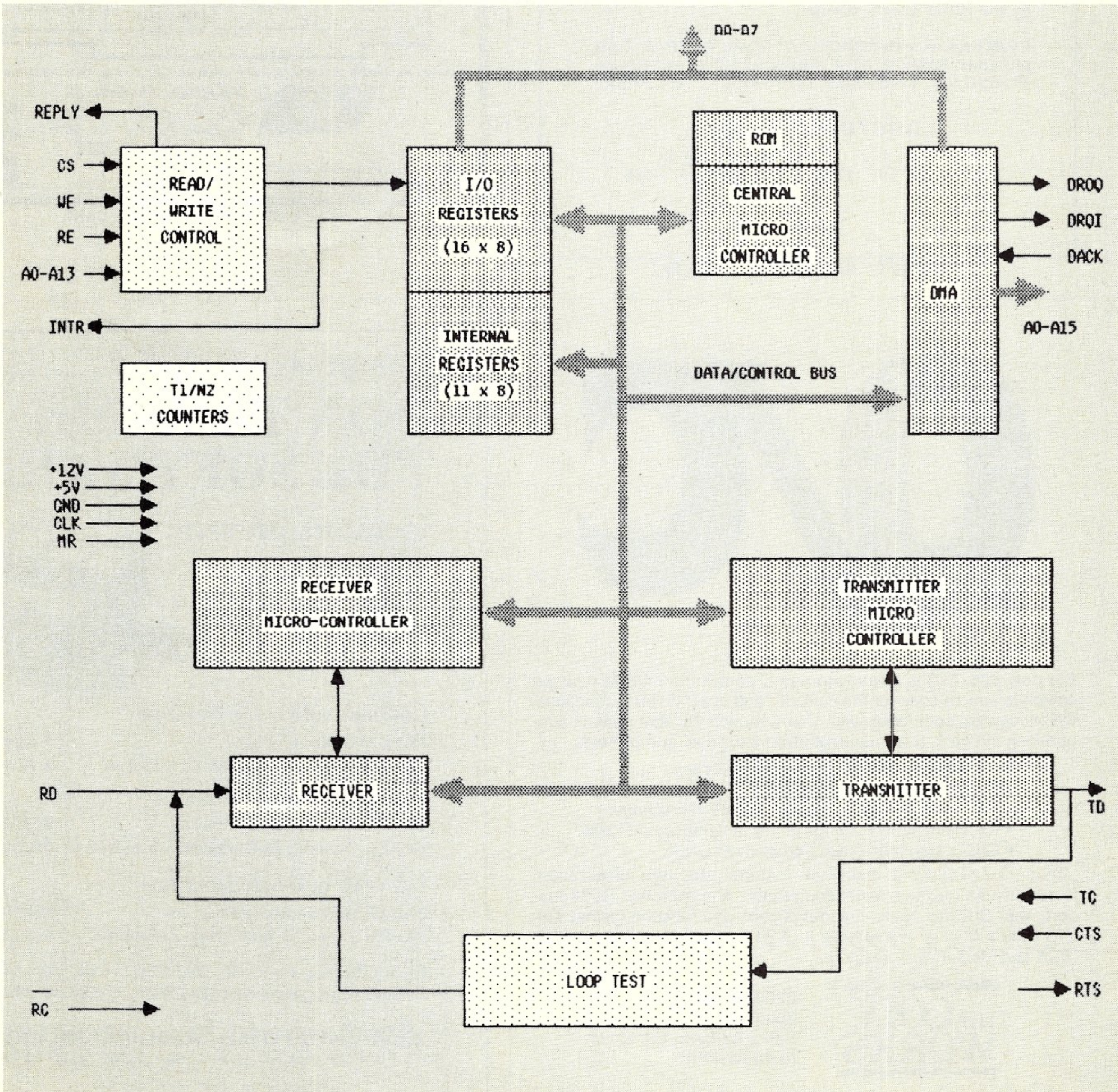
As you have probably realized, interfacing software to this chip is not much more than buffer management and I/O monitoring.

Summary

By now, you should have a more complete understanding of implementing a datalink protocol.

I have presented both the software solution to the problem, and a lower-cost hardware solution. In view of the number of calls I have been receiving since Part 1 was published, I can assume that many designers were involved with X.25 and, like me, had many unanswered questions. I hope that the second and third parts of this series have answered most of these.

Eric L. Beser, 2509 North Calvert St., Baltimore, MD 21218



WRITE

The Writer's Really Incredible Text Editor lives up to its name! It's designed for creative and report writing and carefully protects your text. Includes many features missing from WordStar, such as sorted directory listings, fast scrolling, and trial printing to the screen. All editing commands are single-letter and easily changed. Detailed manual included. Dealer inquiries invited. WRITE is $239.00.

BDS's C Compiler

This is the compiler you need for learning the C language and for writing utilities and programs of all sizes and complexities. We offer version 1.5a, which comes with a symbolic debugger and example programs Our price is (postpaid) $130.00.

Tandon Spare Parts Kits

One door latch included, only $32.50.
With two door latches $37.50.
Door latches sold separately for $7.00.

All US orders are postpaid. We ship from stock on many formats, including: 8", Apple, Osborne, KayPro, Otrona, Epson, Morrow, Lobo, Zenith, Xerox. Please request our new catalog. We welcome COD orders.

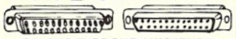
Workman & Associates
112 Marion Avenue
Pasadena, CA 91106
(818) 796-4401

WORKMAN & ASSOCIATES
USER-FRIENDLY SOFTWARE

CIRCLE 86 ON READER SERVICE CARD

QUALITY PARTS AT DISCOUNT PRICES!

SUB-MINIATURE D TYPE CONNECTOR



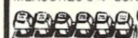
SOLDER TYPE SUB-MINIATURE CONNECTORS USED FOR COMPUTER HOOK UPS.

DB-15 PLUG	$2.75
DB-15 SOCKET	$4.00
DB-15 HOOD	$1.50
DB-25 PLUG	$2.75
DB-25 SOCKET	$3.50
DB-25 HOOD	$1.25

KEY ASSEMBLY

5 KEY $1.00 EACH

CONTAINS 5 SINGLE-POLE NORMALLY OPEN SWITCHES. MEASURES 3¾" LONG.



6 KEY $1.25 EACH

CONTAINS 6 SINGLE-POLE NORMALLY OPEN SWITCHES. MEASURES 4¼" LONG.

7 CONDUCTOR RIBBON CABLE

SPECTRA-STRIP RED MARKER STRIP. 28 GA STRANDED WIRE.
$5.00 PER ROLL (100 FT.)

"PARALLEL" PRINTER CONNECTOR

SOLDER STYLE 36 PIN MALE USED ON "PARALLEL" DATA CABLES.
$5.50 EACH

CRYSTAL

CASE STYLE HC33/U

2 MHZ **$3.50 EA**

COLORBURST 3579.545 KC **$1.00 EACH**

13 VDC RELAY

CONTACT: S.P.N.C.
10 AMP @ 120 VAC
ENERGIZE COIL TO OPEN CONTACT.
COIL: 13 VDC 650 OHMS
SPECIAL **$1.00 EACH**

FREE! FREE! FREE! SEND FOR FALL '84 48 PAGE CATALOG

MINIATURE TOGGLE SWITCHES

ALL ARE RATED 5 AMPS @ 125 VAC

S.P.D.T. (on-on)
P.C. STYLE, NON-THREADED BUSHING.
75¢ EACH
10 FOR $7.00

S.P.D.T. (on-on)
SOLDER LUG TERMINALS.
$1.00 EACH
10 FOR $9.00
100 FOR $80.00

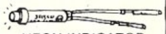
S.P.D.T. (on-off-on)
NON-THREADED BUSHING.
P.C. STYLE
75¢ EACH
10 FOR $7.00

S.P.D.T. (on-on)
P.C. LUGS, THREADED BUSHING.
$1.00 EACH
10 FOR $9.00
100 FOR $80.00

SOLID STATE BUZZER

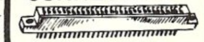
STAR #SMB-06L. 6 VDC. TTL COMPATIBLE.
$1.00 EACH
10 FOR $9.00

120V INDICATOR



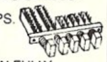
NEON INDICATOR. RATED 120 V 1/3 W. MOUNTS IN 5/16" HOLE
75¢ EACH RED LENS.
10 FOR $7.00
100 FOR $65.00

EDGE CONNECTORS



22/44 22/44 GOLD PLATED CONTACTS .156 CONTACT SPACING.
$2.00 EACH 10 FOR $18.00

5 STATION INTERLOCKING



MADE BY ALPS. 3 - 2PDT AND 2 - 6PDT SWITCHES ON FULLY INTERLOCKING ASSEMBLY. 3¼" BETWEEN MOUNTING CENTERS.
$2.50 EACH

5 STATION NON-INTERLOCKING

SAME AS ABOVE, EXCEPT EACH SWITCH OPERATES INDEPENDENTLY.
$2.50 EACH

ALL ELECTRONICS CORP.
905 S. VERMONT • P.O. BOX 20406 • LOS ANGELES, CA 90006

TOLL FREE ORDERS • 1-800-826-5432
(IN CALIFORNIA: 1-800-258-6666)
AK, HI, OR INFORMATION • (213) 380-8000

- QUANTITIES LIMITED
- MINIMUM ORDER $10.00
- USA: $2.50 SHIPPING NO C.O.D.!
- FOREIGN ORDERS: INCLUDE SUFFICIENT SHIPPING
- CALIF. RES. ADD 6 1/2%

CIRCLE 92 ON READER SERVICE CARD

Quelo™ 68000 Assembler

New 5.0 Release — First Commercial Release January, 1983

The **Quelo**™ portable 68000 assembler conforms to the Motorola resident assembler, publication M68KMASM[D4].

Quelo™ 68000 Assembler Package Features:

Input file concatenation, include function, macros, global parameter substitution from command line, listing date-time stamp, up to 31 character symbols, conditional assembly, structured programming directives, instruction optimization, 68010 instructions, relocation and linking, complex expression linking (all operators), DB-DW-DL directives for Z80 byte order data generation, object library utility, software configuration tracking, conditional linking, options for assembler and linker to write complete symbol table to a file, detailed symbol table listings, assembler symbol cross-reference, linker global symbol cross-reference, object library symbol cross-reference, superb linker load map, various HEX load formats produced by linker, error messages in English, extensive typeset manual with index, readily transported to any system with a C compiler and "UNIX like" system interface.

Ready to run in various disk formats for **CP/M-80, -86, -68K, MS-DOS and PC-DOS**	$595
Portable source version with detailed installation and testing instructions.	$750 and license agreement.

For more information or to order write or call Patrick Adams
COD, Visa, MasterCard.

Quelo™
2464 33rd Ave. W., Suite #173
Seattle, WA 98199
(206) 285-2528

CP/M-80, CP/M-86, CP/M-68K TM DRI. UNIX TM Bell Labs. MS-DOS TM Microsoft. PC-DOS TM IBM.

CIRCLE 55 ON READER SERVICE CARD

C Programmers: Program three times faster with *Instant-C*™

***Instant-C*™** is an optimizing **interpreter for C** that makes programming three or more times faster. It eliminates the time wasted by compilers. Many repetitive tasks are automated to make programming less tedious.

- Two seconds elapsed time from completion of editing to execution.
- **Symbolic debugging;** single step by statement.
- Compiled execution speed; 40 times faster than interpreted Basic.
- Full-screen editor integrated with compiler; compile errors set cursor to trouble spot.
- Directly generates .EXE or .CMD files.
- Follows K & R—works with existing programs. Comprehensive standard C library with source.
- Integrated package; nothing else needed.
- Works under PC-DOS*, MS-DOS*, CP/M-86*.

More productivity, less frustration, better programs. ***Instant-C*™** is $500. Call or write for more info.

Rational Systems, Inc.
(617) 653-6194
P.O. Box 480
Natick, Mass. 01760

Trademarks: MS-DOS (Microsoft Corp.), PC-DOS (IBM), CP/M-86 (Digital Research, Inc.), Instant-C (Rational Systems, Inc.)

CIRCLE 39 ON READER SERVICE CARD

Now available for the computer experimenter!

COMPUTER CONNOISSEUR'S DELIGHT!

NOW BE IN CONTROL WITH YOUR COMPUTER — THE ONLY PUBLICATION OF ITS KIND WRITTEN FOR THE USER. DISCOVER THE SECRETS AND LEARN THE VERSATILITY OF MODERN COMPUTER COMMAND CONTROL CONCEPTS. EXPERIMENT WITH COMPUTER AND TELEPHONE SYSTEMS, INTERFACE THEM, LEARN HOW THEY WORK, WHAT THEY DO... AND HOW TO GET THEM TO WORK FOR YOU! A COMPLETE TELEPHONE ENGINEERING COURSE IS INCLUDED IN MONTHLY CHAPTERS, BRINGING YOU THROUGH STEP, CROSSBAR, ESS, BUBBLE, AND ATOMIC SWITCHING SYSTEMS! EXCLUSIVE COVERAGE IN BIOLOGICAL COMPUTING SYSTEMS, TOO! COMPUTERS AND TELEPHONES ARE THE FUTURE. THIS PUBLICATION IS AN ABSOLUTE MUST FOR EVERYONE INTERESTED.

UNPUBLISHED MATERIAL
WIT
COMICS
DIRECTORY LISTING NETWORKS
ACCESS CODES

The one you've all been waiting for

NOW AVAILABLE — *Learn how to repair telephones and telephone systems, how they work, in monthly installments with the magazine for you.*

Computel™
PUBLISHED MONTHLY

ONE YEAR SUBSCRIPTION $14.00
(SAMPLE COPY $2.00)
SUBSCRIPTION & 2 PROGRAMS $20.00

Credit Card Only: 1-800-6-COMPUTEL (Everywhere EXCEPT California)
1-800-5-COMPUTEL (Calif. Calls Only)

COMPUTEL—the complete SOURCE for everyone. You can now do the things you've only heard about, right in the privacy of your own home. Indispensable reference to phreaks and hackers. Learn how to get all kinds of computer programs FREE. Get the inside story of big business systems—their quirks and flaws—and remain up to date with vital occurrences within the computer industry. Computel is a publication designed for everyone who has an intense curiosity of computer systems, containing a wealth of hard to find information, codes, and numbers. Published monthly.

VISA **Computel Publishing Society**
6354 VAN NUYS BL., #161-M/VAN NUYS, CA 91401

CIRCLE 279 ON READER SERVICE CARD

LETTER QUALITY

OPTIONS:
FORMS TRACTOR $85.00
RS-232C SERIAL $65.00
MULTI-LOAD SHEET FEEDER $375.00

DAISY WHEEL
QUME COMPATIBLE
PRINTER

NOW ONLY **$488**

HIGH RELIABILITY & PERFORMANCE

describes this 18cps Centronics Printer. 256 char. buffer w/optional tractor and sheet feed. Uses Qume 96 spoke Daisy Wheel & Qume Multi-Strike IV ribbon cartridges. Soft touch controls and Bi-directional print gives excellent quality on all types of stock. 8-level impression control prints original PLUS four copies.

DEALER INQUIRIES INVITED.

Call: **(714) 895-5033** *or Write:*
ORANGE COUNTY COMPUTER
15131 Triton Lane, Suite #122
Huntington Beach, CA 92649

VISA

CIRCLE 225 ON READER SERVICE CARD

Xerox Warranty
Service & Maintenance

Promotional Prices for COMPUTER PROFESSIONALS

We set up your sale/leasebacks.

CompuPro® SYSTEMS

MORROW DESIGNS *Tarbell*

PRINTERS, TERMINALS, SOFTWARE

Computer professionals, we want you as a customer, a resource and a reference. Because we sell as many computers by referral as directly from our ad, we want to expand our professional customer base. We are offering promotional prices **only** to knowledgeable users, for a short while.

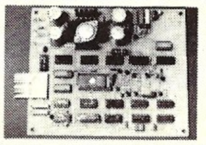
TERMS: Cash with order. You pay all freight. Add 5% for 2% 10/Net 30 AAA1 or Fed./State P.O. 2%/mo. carrying chg.

CALL US 9 AM to 10 PM
205 879-5976
205 879-4735

CostPlus COMPUTERS
P.O. Box 6114 • Birmingham, AL 35259-6114
Our bank is 1st National of Birmingham, (205) 326-5120

CP/M EPROM PROGRAMMING SYSTEM

2708	2732A
2758	27128
2716	27C16
2732	27C32
2764	27C64



- STAND ALONE BOARD - ELECTRONIC SWITCHING OF EPROM TYPES
- USES 24 VOLT XFMR FOR POWER - ALL SUPPLIES/TIMING ON BOARD
- DESIGNED WITH EASY TO GET PARTS - COMPREHENSIVE 28 PG MANUAL
- WORKS WITH ANY CP/M SYSTEM - INDEPENDENT OF SYSTEM SPEED

*** * INTERFACE TWO WAYS * ***

1 - ONE 8 BIT INPUT PORT AND ONE 8 BIT OUTPUT PORT - 16 WIRES AND A GROUND - NO HANDSHAKE LINES REQUIRED
2 - CENTRONICS PRINTER PORT - 8 OUTPUT DATA BITS AND ONE INPUT DATA BIT (BUSY LINE) - 9 WIRES AND A GROUND

*** * CONTROL PROGRAM COMMANDS * ***

- PROGRAM EPROM(S) FROM DISK - SAVE EPROM(S) TO DISK
- READ DISK FILE INTO RAM - PROGRAM EPROM(S) FROM RAM
- READ EPROM(S) INTO RAM - COMPARE EPROM WITH RAM
- VERIFY EPROM IS ERASED - COPY EPROM
- DISPLAY/MODIFY RAM - (MONITOR MODE) WITH 11 SUB COMMANDS FILL-DUMP-XFER-EXAMINE-MODIFY-BIAS-PROGRAM-VERIFY, ETC)

BARE P.C. BOARD WITH COMPLETE DOCUMENTATION AND SOFTWARE ON 8" SINGLE DENSITY DISKETTE $69

(ABOVE WITH COMPLETE PARTS KIT - $169)(A&T - $189)
SOFTWARE AVAILABLE FOR OSBORNE, KAYPRO OR ON 2764 EPROM

TO ORDER SEND CHECK, MONEY ORDER, WRITE OR CALL
ANDRATECH
P.O. BOX 222
MILFORD, OHIO 45150
(513) 752-7218

CALL OR WRITE FOR MORE INFORMATION -- ADD $3.00 FOR SHIPPING
OHIO RES ADD 5.5% TAX -- VISA/MC ACCEPTED -- $3.00 FOR COD

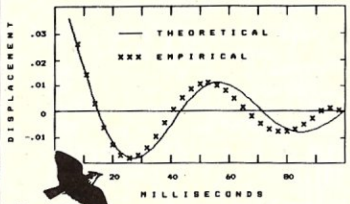
CIRCLE 276 ON READER SERVICE CARD

Graphs without Graphics?

No need for screen graphics. Publishable graphs on your dot matrix printer.

Easy to Use. No programming.
CP/M 80 or 86, MS-DOS, or PC-DOS.
Excellent Manual. Most disk formats.

DataPlotter™



Lark Software™
7 Cedars Road
Caldwell, NJ 07006

Line Graphs & Scatterplots . . $69
Bar Graphs & Pie Charts . . . $69
Both for $99
(Prices include manual)

Shipping . . . . . . add $3
Outside US and Canada . . . . add $6
Specify which Printer
Visa, M/C
(201) 226-7552

CIRCLE 250 ON READER SERVICE CARD

C Programmers
B-Trees
For
$75.00 + 2.00 Postage

Source Code Included

The Softfocus B-Trees record indexing library will help you develop sophisticated application programs. With Softfocus B-Trees you get:

- high speed file handling for up to 16.7 million records per file
- customizable BDS or K & R standard C source code
- no royalties on applications programs
- support random and sequential file access
- includes example programs

Softfocus
1277 Pallatine Dr., Oakville, Ont.
Canada L6H 1Z1 (416) 844-2610
CHARGEX VISA

CIRCLE 219 ON READER SERVICE CARD

NORTHSTAR AND MBSI REALWORLD USERS

New Products and Enhancements

THE FINANCIAL ANALYST:
a concise financial analysis program fully interfaced with general ledger.

MULTI-CO:
an enhancement to enable recording of multiple companies on a single disk.

FINSEC:
church contribution management and mailing program.

FUTURE ENHANCEMENTS FORTHCOMING . . .

Contact Victor Moffitt or circle reader service number.

DATA-CO.
978 TIOGUE AVENUE
COVENTRY, RI
(401) 828-7385
Dealer Inquiries Invited

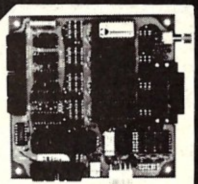
CIRCLE 231 ON READER SERVICE CARD

PC THE **BUYERS GUIDE**

THE MASTER DIRECTORY OF PRODUCTS FOR THE IBM PC, PC XT, PCjr AND MOST COMPATIBLES!

AVAILABLE AT YOUR LOCAL NEWSSTAND AND COMPUTER STORE.

NOW INCLUDES TINY BASIC
FORTH AVAILABLE



SIBEC 51
8051/52 DEVELOPMENT BOARD

8051-Based Single-Board Computer with Monitor/Debugger

- 4 28-pin byte-wide sockets; monitor will program EEPROMS.
- Perfect for System Development and Educational Applications

VISA
$335

Binary Technology
P.O. BOX 67 • MERIDEN, NH 03770 • 603/448-1285

classified ads

CLASSIFIED RATES: Per Word, 15 Word Minimum. REGULAR: $2.00. EXPAND-AD: $3.00 GENERAL INFORMATION: Prepayment discounts available. Payment must accompany order except credit card — Am. Ex., Diners, MC, VISA (include exp. date) — or accredited ad agency insertions. Copy subject to publisher's approval; must be typewritten or printed. First word set in caps. Advertisers using P.O. Boxes MUST supply permanent address and telephone number. Orders not acknowledged. They will appear in next available issue after receipt. Send order & remittance to: Classified Advertising, MICROSYSTEMS Magazine, 1 Park Avenue, New York, NY 10016. Direct sales inquiries to (212) 503-5115. For Customer Service Call: (212) 503-4506.

SOFTWARE

REL/MAC converts M80 REL files to MAC source files. $74.95 on 8" SSDD, CP/M-80. Free information. Microsmith Computer Technology, P.O. Box 1473, Elkhart, IN 46515.

ENGINEERING SOFTWARE-CP/M, MS-DOS, TRS-DOS, free flyer. PLOTPRO Graph Printing Program. Linear/log/semi-log, multiple plot, grid lines, labeling, $52.95. ACNAP Analyzes active/passive electronic circuits. MonteCarlo, Worst Case, Sensitivities, $52.95. SPP-Signal Processing, FFT, linear/non-linear. LaPlace, transient analysis, $62.95. BV Engineering, Box 3429, Riverside, CA 92519. (714) 781-0252.

ATTN: COMPUTER USERS. We do transferring of data from one computer format to another computer format. Most CP/M systems and some PC-DOS. Prices available on request. Please call the DISK LOADERS. (714) 855-9230.

forMath™ formats technical manuscripts: equations, matrices, integrals, footnotes, multiple fonts/sizes, symbols. Supports dotmatrix, daisywheel, laser printers, phototypesetters, CRTs. CP/M, PC-DOS, MS-DOS, mainframes. Shantha Software, 50 West 97th St. 11N, New York, NY 10025. (212) 222-SNIP.

CBASIC/CB80/CB86 program generator. Create program for displays or listings in minutes. Includes direct cursor positioning, input editing functions, 1/0 formats. All source included. $49.95. For more information: Warwick Information Technologies, 29 Hawthorne Ave, Warwick, NY 10990; or call Lu Elliot at (914) 986-5139 or Jana Goodman at (914) 446-5521.

COMPUTER EQUIPMENT/SUPPLIES

REPLACEMENT RIBBONS for computer printers and word processors. Fantastic savings! Thousands in stock. Quick delivery. Call or write: 1-(800) 292-6272. National Computer Ribbons Corp., 1114 Elbank Ave., Baltimore, MD 21239.

PUBLICATIONS

SCHULTZ'S BIMONTHLY reports best bargains: microcomputers, peripherals, applications, supplies, $9.95 single report, postpaid, guaranteed. Postroad, Box 1212C, Roanoke, VA 24006.

ADVERTISERS INDEX

Advertiser	Page
Advanced Digital Corp.	C2
Advanced Digital Corp.	1
All Electronics Corp.	116
American Planning Corp.	111
Ampro Corp.	31
Avocet Systems, Inc.	49
BD Software	56
Beck Manufacturing	18, 19
Borland International	14, 15
Borland International	77
Bush Systems Inc.	89
CBS Computer Books	28
C Systems	87
C Ware Corp.	39
The Code Words	113
CompuPro	C4
Computer Editype Systems	76
Computer Helper Indus. Inc.	120
Computer House	120
Computing	63
D & W Digital	65
Data Access Corp.	79
Digital Research	96
Digital Research	97
Earth Computer	89
Ecosoft, Inc.	67
Electralogics, Inc.	13
Electronic Specialists Inc.	113
Extended Processing	89
FairCom	38
G & G Engineering	7
Gifford Computer Systems	7
Greenleaf Software	51
HSC, Inc.	66
Harvard Softworks	99
Haven Tree	40
I-Bus	50
Illuminated Technology	78
InMac	29
Input/Output Techonology Inc.	20
Integrand	98
Intercontinental Microsystems Corp.	C3
Korsmeyer	38
LG Software	104
Lattice, Inc.	67
Logical Devices, Inc.	27
Lomas Data Products, Inc.	23
MBP	26
Macrotech International Corp.	9
Mark Williams Company	17
Martian Technologies	97
Mendocino Software	97
Micro Dynamics Corp.	89
Microlog	33
Micro Methods	113
Micro-Mikes	53
MicroType	115
Mitek	29
Morningstar Microcomputer	85
Morrow	55
Mycroft Labs, Inc.	98
New Generation Systems, Inc.	107
Nicolet	27
Paradynamics	65
Pelada Informatica	105
Pickles & Trout	4
Programmer's Connection	71
Programmer's Shop	107
Programs Int'l	10
Quelo	116
Quest Research	25
Quic-n-Easi Products, Inc.	74, 75
RH Electronics	103
Rational Sytems	116
Relational Memory Systems	22
STSC	52
SLR Systems	72
Semi-Disk	72
Simpliway Products	113
Softcraft	86
Software Horizons, Inc.	73
The Software Toolworks	65
Solaronics/disk plus	104
Southern Computer Corp.	54
System Engineering Tools	88
2500 AD	60, 61
Tecmar Inc.	11
Teletek	2
UCI	57
Unipress Software, Inc.	91
Unisource	5
Vault	35
Workman & Associates	116
Xyquest	37