EXHIBIT J

The origins of DOS: DOS creator gives his view of relationship between CP/M, MS-DOS. (Tim Paterson) (Letter to the Editor)

Tim Paterson
2446 words
3 October 1994
Microprocessor Report
23
Vol. 8, No. 13, ISSN: 0899-9341
English
COPYRIGHT 1994 MicroDesign Resources Inc.

DOS Creator Gives His View of Relationship Between CP/M, MS-DOS

The following letter comes from Tim Paterson, one of the two engineers who created MS-DOS for Microsoft back in 1981. Mr. **Patterson** is responding to comments made by author John Wharton in his article on **Gary Kildall** (see MPR 8/1/94, p. 10). Although this letter is longer than those that we usually print, we hope that you will find it of historical interest.

Dear Editor:

John Wharton's tribute to **Gary Kildall** is full of well-deserved praise for Mr. Kildall's contribution to the industry. Unfortunately, Mr. Wharton also got briefly side-tracked onto other issues that he demonstrated he knows nothing about. Through his occasional pot-shots in the past, he has made clear his contempt for MS-DOS and its author (me), and everyone is entitled to an opinion. But to present such distortions---even completely made-up stories--about MS-DOS as facts is irresponsible.

Let's start with the characterization of MS-DOS as "an unauthorized 'quick and dirty' knockoff of CP/M from Seattle Computer Products." Back in 1980, what we now call MS-DOS was sold by Seattle Computer Products (SCP) as 86-DOS with their 8086-based computer system, one of the first to use that microprocessor. At that time, CP/M ran only on the 8080/Z80 microprocessors, although a version for the 8086 was known to be under way. Before starting development of 86-DOS, SCP had been shipping its computers since 1979 and was in desperate need of standard software. The uncertain outlook for CP/M-86 led to the internal development project in April 1980.

SCP was a small company with no clout in the industry. To get major software developers to port their products from the 8080/Z80 to the 8086, I decided we had to make it as easy as possible. I had already written a Z80-to-8086 source code translator (hosted on the 8080 and CP/M). My plan was that running an 8080 CP/M program through the translator would be the only work required by software developers to port the program to the 8086. In other words, the interface used by applications to request operating system services would be exactly the same as CP/M's after applying the translation rules.

So 86-DOS generally had all the same application-visible elements as CP/M--the function codes, the entry point address, part of the File Control Block layout, etc. I used the 1976 CP/M Interface Guide for my description of the requirements. I also provided some similar commands from the console such as DIR, RENAME, ERASE--although any system would have such functions, regardless of name chosen.

But that's where the influence from CP/M ended. (Isn't that enough, you say?) Consider that 86-DOS used a completely different file-storage mechanism than CP/M (representing maybe 80% of the 86-DOS code). Once the functions for translation compatibility were done, I immediately added the "real" file interface--allowing the application to read or write any number of records of any size in a single request, rather than one 128-byte record at a time. I also added rudimentary built-in editing (maybe 15% of the code).

The point is, 86-DOS is completely different from CP/M inside. It is an entirely original work within the confines of providing the translation-compatible interface. Because of the completely different file storage format, none of the internal workings has any corresponding relation to anything within CP/M. I never used CP/M source or disassembly at any time while I was developing 86-DOS. It wouldn't have made sense to; there was nothing I could learn from it, since my tasks were different. And finally, if there had been a CP/M for the 8086 microprocessor, 86-DOS would never have been developed.

Contrast this scenario with some other cases of software "cloning." One example is the ROM BIOS that each IBM-compatible PC must have. The ROM is required to be fully compatible, yet IBM was unwilling to license it to others. Compaq and Phoenix Technologies were two of the first to clone the ROM with independently developed code that performed exactly the same functions and even had the same internal addresses (since some

Paterson
EXHIBIT NO. 40
1-18-07
C. HAMMER

software relied on this). Another example is Digital Research's DR-DOS (now Novell DOS), a perfect functional clone of MS-DOS with improvements added. Unlike 86-DOS, each of these clones performs exactly the same function for the same microprocessor as the original program being cloned. 86-DOS provided wholly new functionality (a completely different, faster storage format) for a microprocessor which had no alternative.

Referring again to Mr. Wharton's article, he states that "these protocols [for memory allocation, file sharing, etc.] were removed [from 86/MS-DOS], since Microsoft programmers didn't understand why they were needed." This is utter nonsense. The fact is, neither 86-DOS nor CP/M 2.2 (the 8080/Z80 version of that day) had any facilities "for memory allocation, file sharing, process switching, and peripheral management." As single-task systems, they generally had no need for it. Both simply gave all available memory to the single running task there was no other task to switch to or share files with.

For reasons unrelated to 86-DOS, I left SCP and went to work for Microsoft just in time to put the finishing touches on the adaptation of 86-DOS to the IBM PC (May 1981). I joined Bob O'Rear, who had done all the hard work of getting 86-DOS up and running on the IBM machine. Bob and I must be the programmers that Mr. Wharton refers to, since we were the only ones working on it. I'm pretty sure I didn't hack anything out of my program because I didn't understand it.

Finally, there's the discussion of machine independence. The original CP/M was a bit weak on this point, because it assumed a specific disk format: 77 tracks, 26 sectors, one head, 128 bytes per sector. CP/M 2.2 generalized this into a table-driven approach, although it still assumed 128-byte sectors. The basic problem with CP/M was that. it had no internal buffering--the application program was required to read/write files in physical 128-byte sectors. 86-DOS did have internal buffering and separated the logical record of the application from the physical sector on the disk, allowing either to be any size.

To quote from an 86-DOS manual from late 1980: "In order to provide the user with maximum flexibility, the disk and simple device I/O handlers of 86-DOS are a separate subsystem which may be configured for virtually any real hardware." This was put to the test, because the 86-DOS that ran on SCP's S-100 Bus 8086 computer using 8-inch floppy disks and a serial terminal for I/O was exactly the same binary as used on IBM's 8088 with 5-inch disks and memory-mapped video. Other early users of MS-DOS included Zenith and Sirius (later Victor), each of whose computers were unique.

BIOS (for Basic Input/Output System) was the name given to CP/M's hardware dependent layer so that the BDOS (Basic Disk Operating System) and all applications could be hardware independent. But it is not true that "Microsoft lifted the term 'BIOS' for MS-DOS but wrote the software to be machine-dependent anyway." IBM used the term BIOS to refer to ROMs resident in their machine. I used the term I/O System (never abbreviated) to describe the hardware-dependent layer of 86-DOS, and Microsoft even keeps this layer in a separate file on the disk (IO.SYS) from the machine-independent cede (MSDOS.SYS).

Responsible journalists check their facts. I don't know if Mr. Wharton is just so gullible that he believes any anti-Microsoft story he hears or if he just makes up the stories himself. To get some of the basics, he could start by reading Gates by Manes and Andrews, or Hard Drive by Wallace and Erickson. Then, as authors of both books did, just ask me if there are any questions.

--Tim Paterson, Microsoft

Mr. Wharton replies:

I couldn't agree more with Mr. Paterson's observation that responsible journalists check their facts. In the 42 hours available from when I began writing the Kildall piece until I had to deliver a publishable draft to production, I managed to run the column past several of Gary's closest friends and work associates dating back to his consulting days for Intel. They found several errors and omissions; all were corrected before publication. I stand by the piece as printed.

Mr. Paterson is clearly disturbed by my characterization of 86-DOS as an "unauthorized 'quick and dirty' knock-off of CP/M." Yet Mr. Paterson readily admits to having "cloned" the software from a CP/M specification document that was, incidentally, copyrighted and marked "proprietary to Digital Research." His conversion was certainly quick; Mr. Paterson's stated goal was to finish his code before DRI could finish theirs. And it was dirty: 86-DOS supported just 27 of the 37 OS calls implemented by CP/M at the time. In fact, according to the books Gates and Hard Drive, Mr. Paterson named the first version of his software QDOS, for Quick and Dirty OS.

And Paterson's work was most definitely not authorized. In 1980, the jury was still out concerning the conditions under which it was permissible for one program to appropriate the calling conventions, look, and feel of another. In the end, IBM spent more to head off a copyright-infringement lawsuit from DRI than it spent to acquire the rights for MS-DOS in the first place. I doubt IBM would have done so had it not felt legally exposed. Just last year, Microsoft publicly denounced Sun for promoting a "clone" of the Windows ABI that could run on SPARC workstations.

I can empathize somewhat with the bind in which SCP found itself: unable to sell its 8086 hardware for lack of software and unable to buy the software it wanted. But for Mr. Paterson to cite the unavailability of CP/M-86 as justification for appropriating the "look and feet' of a competing OS and its utilities seems to me to be analogous to telling a judge, "I needed a car, Your Honor, and the plaintiff wouldn't sell me his, so I was forced to take it."

And whereas Mr. Paterson argues that 86-DOS had to be functionally compatible with CP/M to allow CP/M-80 programs to be mechanically ported, in fact it was not. By my count, 86-DOS failed to implement at least nine of the required CP/M 2.2 function calls, altered the functions performed by two others, and "enhanced" the capabilities of several more. According to the book Undocumented DOS (the first edition of which was co-authored by Mr. Paterson himself), "Even in the beginning there were crucial differences between the two systems. MS-DOS did not, as widely claimed, mimic every last CP/M function call. For example, MS-DOS did not implement CP/M function 12 (0CH) to get the system version number. Somewhat unaccountably, MS-DOS instead used (and still uses) function 0CH to read the keyboard." This one change would likely have caused most CP/M 2.2 programs to malfunction, if translated according to the procedure Mr. Paterson describes.

Mr. Paterson is absolutely correct that both 86-DOS and (eight-bit) CP/M 2.2 were single-tasking systems, and neither had facilities for memory allocation, file sharing, etc. But CP/M was the basis for a family of compatible OS products that already supported multitasking (MP/M) and networking (CP/Net) functions. The protocols needed to support these additional features had all been defined by DRI long before 86-DOS was developed. CP/M-86 itself supported memory allocation, 8086 memory segmentation, and multitasking capabilities sufficient for printer spooling. 86-DOS and MS-DOS did not.

As to machine independence -- although complete 86-DOS documentation is somewhat hard to come by these days, it's my recollection that both 86-DOS and MS-DOS imposed implicit constraints on memory layouts as well as disk size and configuration, constraints not imposed by CP/M-86.

Mr. Paterson points out that for me to know Microsoft's thought processes, I would have had to talk to him or Bob O'Rear at the time. Well, as it happens, I did. In August of 1981, soon after Microsoft had acquired full rights to 86-DOS, Bill Gates visited Santa Clara in an effort to persuade Intel to abandon a joint development project with DRI and endorse MS-DOS instead. It was I--the Intel applications engineer then responsible for iRMX-86 and other 16-bit operating systems--who was assigned the task of performing a technical evaluation of the 86-DOS software. It was I who first informed Gates that the software he just bought was not, in fact, fully compatible with CP/M 2.2. At the time I had the distinct impression that, until then, he'd thought the entire OS had been cloned. (This visit was recounted briefly in the book Gates, by the way, which also lists my name as a source.)

First impressions die hard: at the time, the only documentation that existed for MS-DOS was an "86-DOS Programmer's Reference," on the cover of which the word "Programer" was printed in bold one-inch type--with just one "m." In the weeks that followed, I spoke repeatedly by phone with Tim, Bob, and their cohorts, trying to understand their work. I even made a pilgrimage to Bellevue and spent a day with them in their offices. It was a Monday in September, I believe.

The strong impression I drew 13 years ago was that Microsoft programmers were untrained, undisciplined, and content merely to replicate other people's ideas, and that they did not seem to appreciate the importance of defining operating systems and user interfaces with an eye to the future. In the end it was this latter vision, I feel, that set **Gary Kildall** so far apart from his peers.

Document mcpr000020011029dqa300077

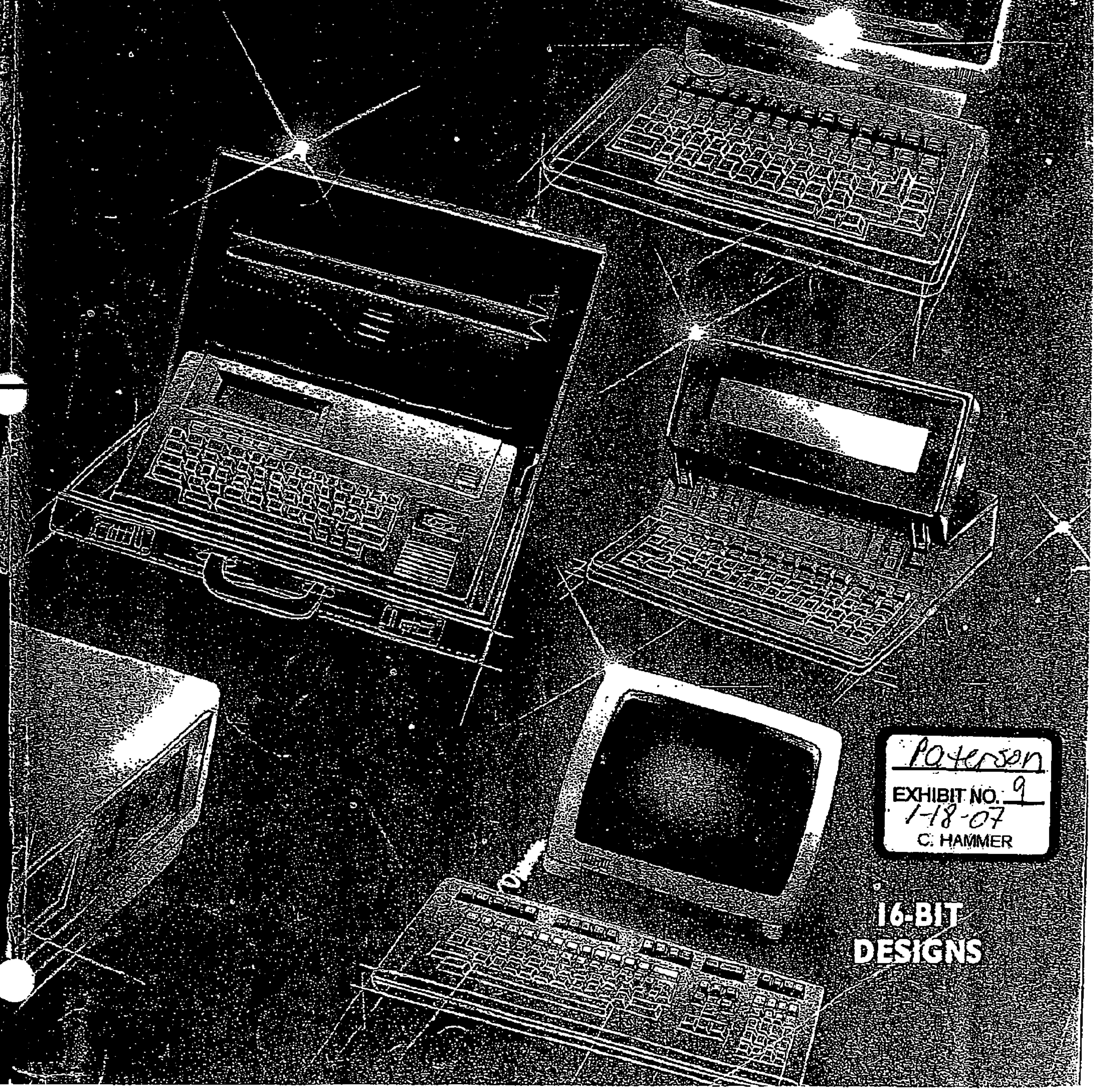
EXHIBIT K

12490

BYTE®

the small systems journal

JUNE 1983 Vol. 8, No. 6
$2.95 in USA
$3.50 in Canada/£2.10 in U.K.
A McGraw-Hill Publication



Paterson
EXHIBIT NO. 9
1-18-07
C. HAMMER

16-BIT DESIGNS

An Inside Look at MS-DOS

The design decisions behind the popular operating system

Tim Paterson
Seattle Computer Products
1114 Industry Dr.
Seattle, WA 98188

The purpose of a personal computer operating system is to provide the user with basic control of the machine. A less obvious function is to furnish the user with a high-level, machine-independent interface for application programs, so that those programs can run on two dissimilar machines, despite the differences in their peripheral hardware. Having designed an 8086 microprocessor card for the S-100 bus and not finding an appropriate disk operating system on the market, Seattle Computer Products set about designing MS-DOS. Today MS-DOS is the most widely used disk operating system for personal computers based on Intel's 8086 and 8088 microprocessors.

MS-DOS Design Criteria

The primary design requirement of MS-DOS was CP/M-80 translation compatibility, meaning that, if an 8080 or Z80 program for CP/M were translated for the 8086 according to Intel's published rules, that program would execute properly under MS-DOS. Making CP/M-80 translation compatibility a requirement served to promote rapid development of 8086 software, which, naturally, Seattle Computer was interested in. There was partial success: those software developers who chose to translate their CP/M-80 programs found that they did indeed run under MS-DOS, often on the first try. Unfortunately, many of the software developers Seattle Computer talked to in the earlier days preferred to simply ignore MS-DOS. Until the IBM Personal Computer was announced, these developers felt that CP/M-86 would be *the* operating system of 8086/8088 computers.

Other concerns crucial to the design of MS-DOS were speed and efficiency. Efficiency primarily means making as much disk space as possible available for storing data by minimizing waste and overhead. The problem of speed was attacked three ways: by minimizing the number of disk transfers, making the needed disk transfers happen as quickly as possible, and reducing the DOS's "compute time," considered overhead by an application program. The entire file structure and disk interface were developed for the greatest speed and efficiency.

The last design requirement was that MS-DOS be written in assembly language. While this characteristic does help meet the need for speed and efficiency, the reason for including it is much more basic. The only 8086 software-development tools available to Seattle Computer at that time were an assembler that ran on the Z80 under CP/M and a monitor/debugger that fit into a 2K-byte EPROM (erasable programmable read-only memory). Both of these tools had been developed in house.

MS-DOS Organization

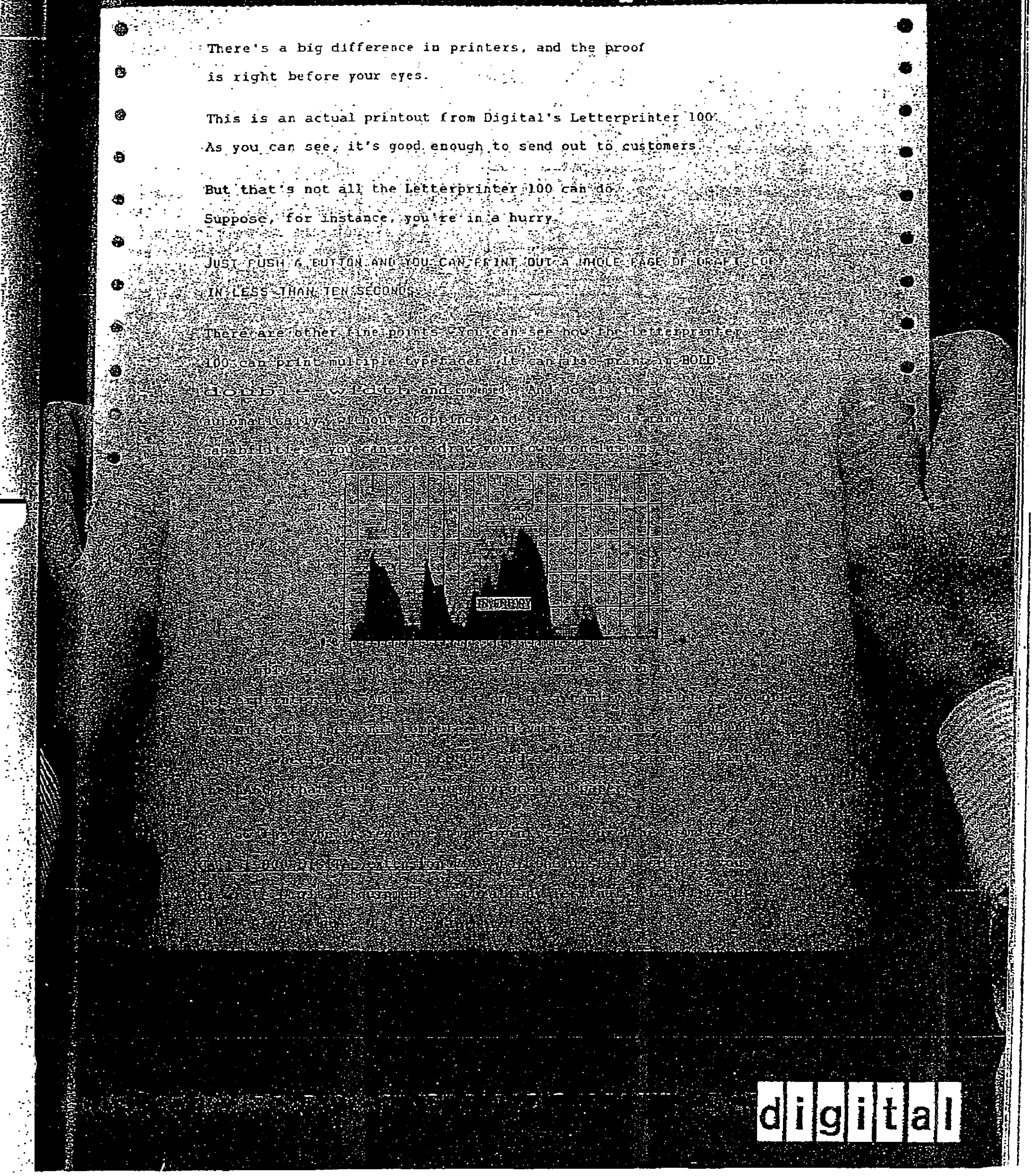
The core of MS-DOS is a device-independent input/output (I/O) handler, represented on a system disk by the hidden file MSDOS.SYS. It accepts requests from application programs to do high-level I/O, such as sequential or random access of named disk files, or communication with character devices such as the console. The handler processes these requests and converts them to a very low level form that can be handled by the I/O system. Because MSDOS.SYS is hardware independent, it is nearly identical in all MS-DOS versions provided by manufacturers with their equipment. Its relative location in memory is shown in figure 1.

The I/O system is totally device dependent and is represented on the disk by the hidden file IO.SYS. It is normally written by hardware manufacturers (who know their equipment best, anyway) with the notable exception of IBM, whose I/O system was written to IBM's specifications by Microsoft. The tasks required of the I/O system, such as outputting a single byte to a character device or reading a contiguous group of physical disk sectors into memory, are as simple as possible.

The command processor furnishes

Circle 122 on inquiry card.

Before you buy a printer look at the fine print.



There's a big difference in printers, and the proof
is right before your eyes.

This is an actual printout from Digital's Letterprinter 100.
As you can see, it's good enough to send out to customers.

But that's not all the Letterprinter 100 can do.
Suppose, for instance, you're in a hurry.

JUST PUSH A BUTTON AND YOU CAN PRINT OUT A WHOLE PAGE OF DRAFT COPY
IN LESS THAN TEN SECONDS.

There are other fine points. You can see how the Letterprinter
100 can print multiple typefaces. It can also print in BOLD,
double-width and condensed. And do all these styles
automatically, without stopping. And with its wide range of graphics
capabilities, you can even draw your own conclusions.

[illegible]

digital

the standard interface between the user and MS-DOS and is contained in the visible file COMMAND.COM. The processor's purpose is to accept commands from the console, figure out what they mean, and execute the correct sequence of functions to get the job done. It is really just an ordinary application program that does its work using only the standard MS-DOS function requests. In fact, it can be replaced by any other program that provides the needed user interface.

There are, however, two special features of the COMMAND file. First, it sets up all basic error trapping for either hard-disk errors or the Control-C abort command. MSDOS.SYS provides no default error handling but simply traps through a vector that must have been previously set. Setting the trap vector and providing a suitable error response is up to COMMAND (or whatever program might be used to replace it).

The second special feature is that COMMAND splits itself into two pieces, called the *resident* and *transient* sections. The resident, which sits just above MS-DOS in low memory, is the essential code and includes error trapping, batch-file processing, and reloading of the transient. The transient interprets user commands; it resides at the high end of memory where it can be overlaid with any applications program (some of which need as much memory as they can get). This feature is of limited value in systems with large main memory, and it need not be imitated by programs used as a replacement for COMMAND.

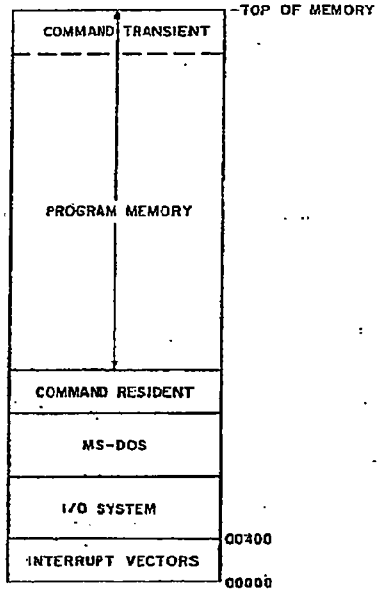


Figure 1: *Map of memory areas as assigned by MS-DOS.*

COMMAND provides both a useful set of built-in commands and the ability to execute program files located on the disk. Any file ending with the extensions .COM, .EXE, or .BAT can be executed by COMMAND simply by typing the first part of the file name (without extension). You can normally enter parameters for these programs on the command line, as with any of the built-in commands. Overall, the effect is to give you a command set that can be extended almost without limit just by adding the command as a program file on the disk.

The three different extensions

MODEMS/CRTS/PRINTERS/SWITCHES/MICRO COMPUTERS/CABLES

	PRODUCT DESCRIPTION	LIST	YOUR COST
MODEMS	UDS 103LP, 300 bps, Modem	145	130
MODEMS	UDS 202LP, 1200 bps, Half Duplex Modem	195	150
MODEMS	UDS 212LP, 1200 bps, Full Duplex (212A) Modem	445	CALL
MODEMS	U.S. Robotics Auto Dial 212A, 300/1200 Full Duplex Modem	599	475
MODEMS	Incomm Auto Dial 212A, 300/1200 Full Duplex Modem	699	450
MODEMS	Cermetec Auto Dial 212A, 300/1200 Full Duplex Modem	595	495
MODEMS	Microband Auto Dial 212A, 300/1200 Full Duplex Modem	695	495
MODEMS	Rixon Intelligent Modem, 300/1200 (10 Number)	495	CALL
MODEMS	Rixon PC 212A (IBM PC Modem Card) 300/1200	495	CALL
MODEMS	Incomm Multi Dial 300/1200 (10 Number)	795	550
MODEMS	U.S. Robotics Password 300/1200 (Auto Dial) Modem	495	CALL
MODEMS	U.S. Robotics Courier, Osborne, 300/1200 Modem	518	CALL
MODEMS	U.S. Robotics S-100, 300/1200 Auto Dial Modem	495	CALL
CRTS	Visual 50 CRT	695	550
CRTS	Freedom 100 CRT, w/20 F. keys & Editing	595	550
CRTS	Incomm Remote Station I, w/Integrated 300/1200 Modem	1295	995
PRINTERS	Epson MX80/Graftrax +	650	399
PRINTERS	Epson MX80 FT/Graftrax +	700	498
PRINTERS	Epson MX100/Graftrax +	900	700
PRINTERS	Epson FX-80	750	656
PRINTERS	Star Micronics Gemini 10	399	350
PRINTERS	Star Micronics Gemini 15	649	548
PRINTERS	AJ Letter Quality Printer, 30 CPS (KSR)	1450	1250
SWITCHES	Incomm AB Switch, 8 Pin	120	84
SWITCHES	Incomm AB Switch, 25 Pin	159	98
SWITCHES	Incomm ABC Switch, 25 Pin	198	138
BOB	IDS Breakout Box (Blue Box)	159	140
BOB	Incomm Breakout Box (Bob)	150	120
COMPUTERS	Epson HX-20-AA	795	CALL
COMPUTERS	Epson QX-10/256K	2995	CALL
COMPUTERS	Zenith Z-100	4000	CALL
CABLES	T-7MM, 7 Pin, 4 Wire, Telephone Cable (Modular Plugs)	10	7
CABLES	S-975, Modular Double Adapter	7	5
CABLES	EIA 9/5, RS 232, 9 Pin Cable, 5 FT MM/FF/MF	15	12
CABLES	EIA 25/5, RS 232, 25 Pin Cable, 25 FT MM/FF/MF	22	14
CABLES	EIA 50/5, Centronics Parallel Cable, 5 FT (36 Pin)	30	21
CABLES	MC 0050/10, Centronics 10 Ft. MM Cable, 36 Pin (10 FT)	38	25
CABLES	7010/5, IBM PC Printer Cable w/36 Pin	40	22
CABLES	8010/5 Apple II Printer Cable w/36 Pin	27	18
CABLES	9010/5, Atari Printer Cable w/36 Pin	30	21
SOFTWARE	U.S. Robotics Telpac	79	CALL
SOFTWARE	Rixon PC Com 1 (IBM PC Software)	69	CALL

ORDER TOLL FREE - 1-800-323-2666

TOUMAYAN & ASSOCIATES
115 N. Wolf Rd.
Wheeling, IL 60090
312-459-8866

We Welcome: ● Visa, Mastercharge ● Checks ● Company P.O. ● C.O.D. (Add. $1.50/Shipment)

ZX FORTH

Simplicity of BASIC with the speed of machine code!

A complete implementation of the FORTH language for the ZX81 and TS1000 computer. FORTH's most distinctive feature is its flexability. It can turn your computer into a "word processor". The basic unit is the WORD — the programmer uses existing WORDS to define his own which can then be used in further definitions. This makes program development much faster than other languages. FORTH is an interactive compiled language that expands the capabilities of your own ZX81/TS1000. Programs run up to 10 times faster than BASIC. The more programs you have written, the more words, therefore you can draw on those for further programs. FORTH is supplied on cassette and is accompanied by a 56 page users manual and an 8 page editors manual.

Z43 $29.95

Call toll free 1-800-833-8400
N.Y. (716) 874-5510
In Canada call toll free 1-800-268-3640

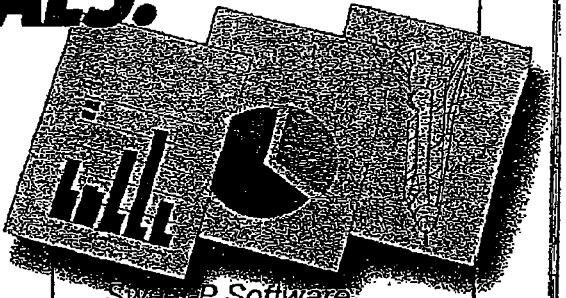
GLADSTONE ELECTRONICS

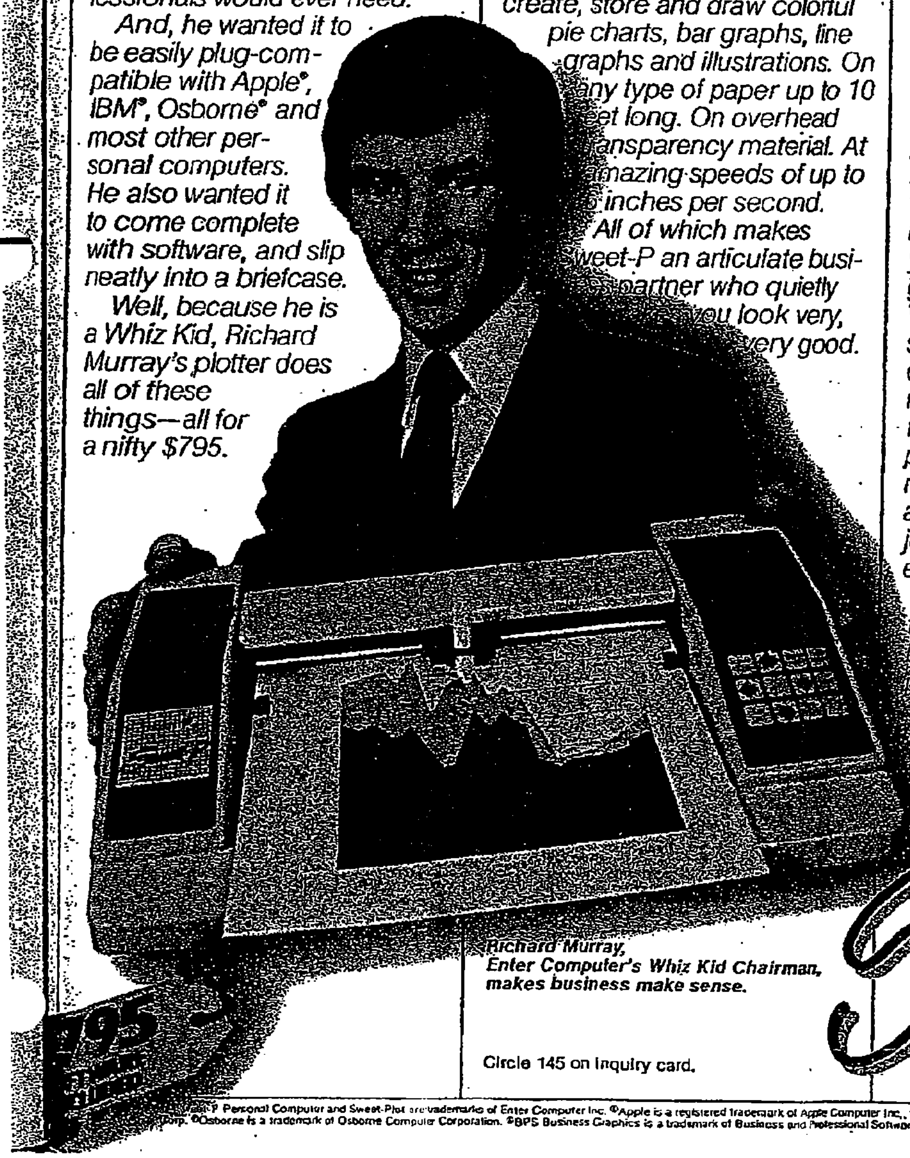
1585 Kenmore Ave., Buffalo, N.Y. 14217
Cheques or money orders. No C.O.D.s. Add shipping
In Canada: 1736 Avenue Rd., Toronto, Ont. M5M 3Y7. Canada: Prices may vary.

Circle 452 on inquiry card.

Circle 451 on inquiry card.

INSTANT BUSINESS GRAPHICS WOW PROFESSIONALS.





<u>$795. Personal Plotter Makes Waves.</u>

When Whiz Kid Richard Murray invented the Personal Plotter™, he wanted it to perform every graphic task professionals would ever need.

And, he wanted it to be easily plug-compatible with Apple®, IBM®, Osborne® and most other personal computers. He also wanted it to come complete with software, and slip neatly into a briefcase.

Well, because he is a Whiz Kid, Richard Murray's plotter does all of these things—all for a nifty $795.

Introducing the Sweet-P™ Personal Plotter.

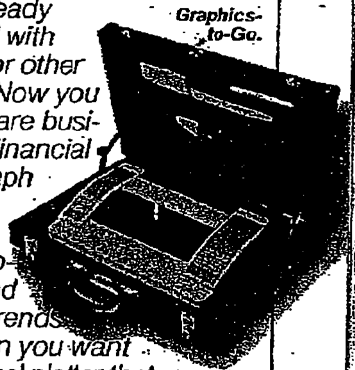
<u>More Presentable Presentations.</u>

With Sweet-P you quickly create, store and draw colorful pie charts, bar graphs, line graphs and illustrations. On ny type of paper up to 10 et long. On overhead ansparency material. At mazing speeds of up to inches per second.

All of which makes weet-P an articulate business partner who quietly ou look very, ery good.

<u>Sweet-P Software for Success.</u>

Not familiar with how a plotter works. Not to worry. Sweet-Plot™ I Software will have you drawing standard graphs in minutes. And, for complex business graphics there's Sweet-Plot II by Business and Professional Software, Inc. With a few keystrokes, it lets you extract and graph data you've already generated with VisiCalc® or other software. Now you can compare business and financial results, graph performance against projections and even plot trends.



So, when you want the personal plotter that means business, plug into the very professional Sweet-P.

Your local computer ore will show you y it makes good usiness sense, instantly.

Sweet-P™

Richard Murray, Enter Computer's Whiz Kid Chairman, makes business make sense.

Circle 145 on Inquiry card.

Enter Computer, Inc.
6867 Nancy Ridge Drive
San Diego, CA 92121
Tel: 619-450-0601

Sweet-P Personal Computer and Sweet-Plot are trademarks of Enter Computer Inc. ®Apple is a registered trademark of Apple Computer Inc., ®IBM is a registered trademark of International Business Machines Corp. ®Osborne is a trademark of Osborne Computer Corporation. ®BPS Business Graphics is a trademark of Business and Professional Software, Inc. ®VisiCalc is a registered trademark of VisiCorp.

allowed on program files represent different internal file formats.

- .COM files are pure binary programs that will run in any 8086 memory segment; in order for this to be possible, the program and data would ordinarily have to be entirely in one 64K-byte segment.
- .EXE files include a header with relocation information so that the program may use any number of segments; all intersegment references are adjusted at load time to account for the actual load segment.
- .BAT (batch) files are text files with commands to be executed in sequence by COMMAND.

File Structure

Disks are always divided up into tracks and sectors, as shown in figure 2. To access any particular block of data, the program first moves to the correct track, then has you wait while the spinning disk moves the correct sector under the head.

A somewhat more abstract view of disks was taken in developing MS-DOS. MS-DOS views the disk, not in terms of tracks and sectors, but as a continuous array of *n* logical sectors, numbered from 0 to *n*−1. Figure 2 shows the usual method of numbering the logical sectors. Logical sector 0 is the first sector of the outermost track; the rest of the track (and the next, etc.) is numbered sequentially. Logical sector *n*−1 is the last sector on the innermost track.

The mapping of logical sectors to physical track and sector is done by the hardware-dependent I/O System and is completely transparent to the MS-DOS file system. Any other method may be used, and MS-DOS wouldn't know the difference. Having a standard mapping, however, is essential for interchanging disks between computer systems with different peripheral hardware.

As shown in table 1, the MS-DOS file system divides the linear array of logical sectors into four groups. The first of these is the reserved area, whose purpose is to hold the bootstrap loader. Because the loader is usually very simple, only one sector is normally reserved.

The FAT (file allocation table), a map of how space is distributed among all files on the disk, comes next. Because it is so important, two copies are usually kept side by side. If one copy cannot be read because of a failure in the medium, the second will be used.

The directory follows the FAT.

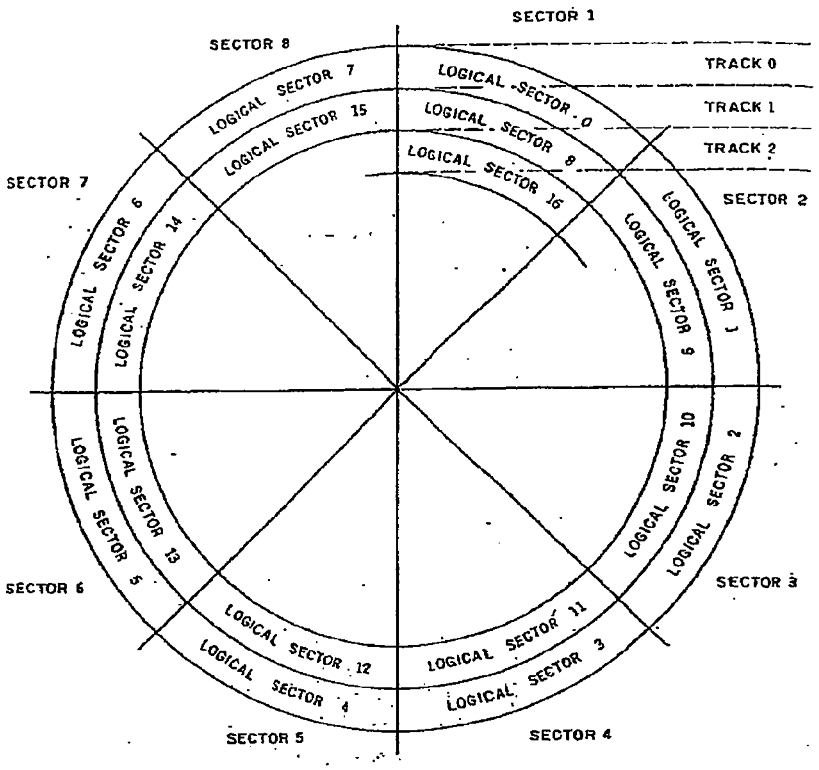


Figure 2: *Placement of disk sectors in IBM Personal Computer (single-sided) format.*

Logical Sector Numbers	Use
0	Reserved for bootstrap loader
1—6	FAT 1 } file allocation tables (FATs)
7—12	FAT 2 }
13—29	Directory
30—2001	Data

Table 1: *Map of disk areas on an 8-inch single-sided, single-density floppy disk.*

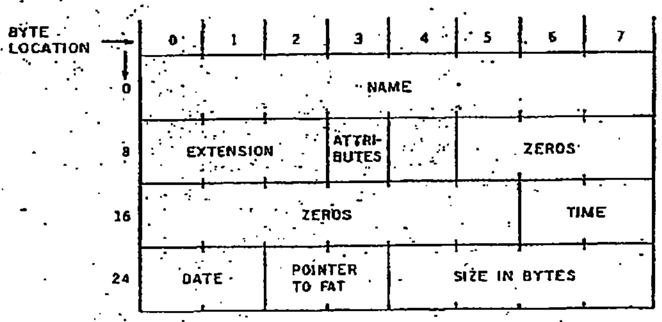


Figure 3: *Arrangement of bytes in disk directory entry.*

Match the 68000 Competition with UNIX™ operating systems.

Beat it with UniFLEX.™

We can't keep a good thing secret. Our new UniFLEX Operating System for the 68000 is better than the competitions UNIX operating systems or like offerings. Better in several ways. A whole new generation of users can now enjoy the efficiency, flexibility, and reliability that built our reputation.

Want the full story? Call us. Or watch for more details.

There *is* a superior alternative.

Technical Systems Consultants, Inc.
111 Providence Road
Chapel Hill, NC 27514 USA

Telephone: (919) 493-1451
Telex II: 510-920-0540

UNIX™ is a trademark of Bell Laboratories.
UniFLEX™ is a trademark of Technical Systems Consultants, Inc.

Logical Sector Numbers	Allocation Unit Number
30—33	2 (first allocation unit)
34—37	3
38—41	4
42—45	5
•	•
•	•
•	•
1998—2001	494

Table 2: *Allocation unit numbering for the 8-inch single-density format. To compute the logical sector number of the first sector in an allocation unit, you use the following equation: sector number = 4 × allocation unit number + 22.*

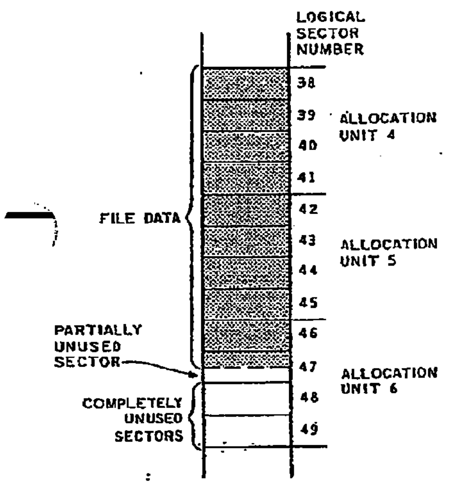


Figure 4: *Assignment of logical sectors to allocation units. Note that, in the file shown, more than two sectors are wasted because they are in an unused part of the last allocation unit.*

Each file on the disk has one 32-byte entry in the directory, which includes the file name, size, date and time of last write, and special attributes. Each entry also has a pointer to a place in the FAT that tells where to find the data in the file. Figure 3 shows the layout of a directory entry.

The rest of the disk is the data area. It is divided into many small, equal-sized areas called *allocation units.* Each unit may have 1, 2, 4, 8, 16, 32, 64, or 128 logical sectors, but the number is fixed for a given disk format. Allocation units are numbered sequentially. The numbering starts with 2; the first two numbers, 0 and 1, are reserved. Table 2 shows this numbering system applied to the 8-inch single-density disk format.

The allocation unit is the smallest unit of space MS-DOS can keep track of. The amount of space used on the disk for each file is some whole number of allocation units. Even if the file is only 1 byte long, an entire unit will be dedicated to it.

For example, the standard format for 8-inch single-density disks uses four 128-byte sectors per allocation unit. When a new file is first created, no space is allocated, but an entry is made in the directory. Then when the first byte is written to the file, one allocation unit (four sectors) is assigned to the file from the available free space. As each succeeding byte is written, the size of the file is kept up-dated to the exact byte, but no more space is allocated until those first four sectors are completely full. Then to write 1 byte more than those four sectors worth, another four-sector allocation unit is taken from free space and assigned to the file.

When writing stops, the last allocation unit will be filled by some random amount of data (figure 4). The unused space in the last allocation unit is wasted and can never be used as long as the file remains unchanged on the disk. This wasted space is called *internal fragmentation,* because it is part of the space allocated to the file but is an unusable fragment. On the average, the last allocation unit (regardless of size) will be half filled and, therefore, half wasted. Because each file wastes an average of one-half an allocation unit, the total amount of space wasted on a disk due to internal fragmentation is the number of files times one-half the allocation unit size.

The phenomenon called *external fragmentation* occurs when a piece of data space is unallocated yet remains unused because it is too small. This cannot happen in the MS-DOS file system because MS-DOS does not require files to be allocated contiguously. It is, however, present in more primitive systems, such as the UCSD p-System.

It would certainly seem desirable to minimize internal fragmentation by making the allocation unit as small as possible—always one sector, for example. However, for any given disk size, the smaller the unit, the more there must be. Keeping track of all those units can get to be a problem. Specifically, the amount of space required in the file allocation table would be quite large if there were too many small allocation units. For every unit, 1.5 bytes are required in the FAT; there are normally two FATs on the disk, each of which is rounded up to a whole number of sectors.

Now take a standard 8-inch single-density floppy disk that has 2002 sectors of 128 bytes. To minimize internal fragmentation, choose the smallest possible allocation-unit size of one sector. Two thousand allocation units will require 3000 bytes (24 sectors) per FAT, or 48 sectors for two FATs. If the average file size is 16K bytes (128 sectors), the disk will be full when there are 16 files on it. Waste due to internal fragmentation would be

16 files × 64 bytes per file = 1024 bytes (8 sectors)

Far more space is occupied by the FATs on the disk than is wasted by internal fragmentation!

To provide maximum usable data space on the disk, both internal fragmentation and FAT size must be considered because both consume data area. The standard MS-DOS format for 8-inch single-density disks strikes a balance by using four sectors per allocation unit. Two sectors per unit would have been just as good (assuming a 16K-byte average file size), but there is another factor that always favors smaller FATs and larger allocation units: the entire FAT is kept in main memory at all times.

The file allocation table contains *all* information regarding which allocation units are part of which file. Thus, by keeping it in main memory, any file can be accessed either sequentially

Powerful CP/M Software.

For Apple, Osborne, Xerox, Kaypro, North Star, SuperBrain, Heath/Zenith, and others.

Now only $29.95 each!

NEVADA COBOL

was $199.95 now only **$29.95.**

When we introduced Nevada COBOL in 1979, it was loaded with innovations. Today's Edition 2 is even better!

☐ Extremely Compact. You can compile and execute up to 2500 statements in 32K RAM, 4000 statements in 48K, etc.

☐ It's based upon the ANSI-74 standards with level 2 features such as compound conditionals and full CALL CANCEL.

☐ You can distribute your object programs royalty FREE!

☐ You get a diskette, 153-page manual with lots of examples and 16 complete COBOL source code programs.

Also available: COBOL Application Packages, Book 1 $9.95

NEVADA PILOT

was $149.95 now only **$29.95.**

☐ Perfect for industrial training, office training, drill and testing, virtually all programmed instruction, word puzzle games, and data entry facilitated by prompts.

☐ John Starkweather, Ph.D., the inventor of the PILOT language, has added many new features to Nevada PILOT. There are commands to drive optional equipment such as Video Tape Recorders. There's a built-in full-screen text editor, and much more.

☐ Meets all PILOT-73 standards for full compatibility with older versions.

☐ You get a diskette, 114-page manual and ten useful sample programs.

☐ See Review in *Microcomputing,* January 1983, page 158.

NEVADA FORTRAN

was $199.95 now only **$29.95.**

☐ Based on ANSI-66 standards with some 1977 level features.

☐ IF . . THEN . . ELSE constructs.

☐ A very nice TRACE style debugging.

☐ 150 English language error messages.

☐ You get a diskette, 174 pages of Documentation and five sample programs. Requires 48K RAM.

NEVADA EDIT

was $119.95 now only **$29.95.**

☐ High quality text editing for micros!

☐ A character-oriented full-screen video display text editor designed specifically to create COBOL, BASIC and FORTRAN programs.

☐ Completely customizable tab stops, default file type, keyboard control key layout and CRT by menu selection.

☐ The diskette comes with an easy to read manual.

To make our software available to even more micro users, we've slashed our prices. What's more, we're offering a money back guarantee. If for any reason you're not completely satisfied, just return the package—in good condition with the sealed diskette unopened—within 30 days and we'll refund your money completely.

This is a limited time offer, so order yours today!

Shipping/handling fees. Add $4.00 for first package and $2.00 each additional package. OVERSEAS Add $15.00 for first package and $5.00 each additional package. Checks must be in U.S. funds and *drawn on a U.S. bank!*

Trademarks: CP/M, Digital Research; TRS-80, Tandy Corp.; TeleVideo, TeleVideo Systems, Inc.; Apple II, Apple Computer Inc.; Osborne 1, Osborne Computer Corp.; Xerox 820, Xerox Corp.; Kaypro, Non-Linear Sys.; Heath/Zenith, Heath Co.; IBM, International Business Machine, Corp. © 1983 Ellis Computing.

MAIL TODAY! To: Ellis Computing
3917 Noriega St.
San Francisco, CA 94122
(415) 753-0186

ELLIS COMPUTING

The CP/M-80 Operating Systems and 32K RAM are required.

Indicate diskette format:

8"	☐ SSSD (Standard IBM 3740 format)	
5¼"	☐ Apple CP/M	☐ Osborne
	☐ North Star DD	☐ North Star SD
	☐ TRS-80 Mod I (4200 hex)	☐ TRS-80 Mod I/Mapper
	☐ Heath, Hard Sector	☐ Heath, Soft Sector
	☐ Micropolis Mod II	☐ Superbrain DD DOS 3.X (512 byte sectors)
	☐ Xerox 820 (Kaypro)	☐ TeleVideo

Indicate software packages: ☐ COBOL ☐ PILOT ☐ FORTRAN ☐ EDIT

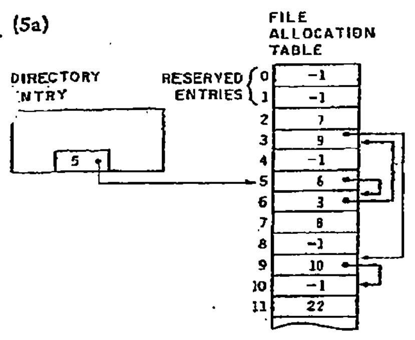
Send my order for ______ packages @ $29.95 each Total ______
COBOL Applications Package @ $9.95 each Total ______

☐ Check enclosed — In CA add sales tax ______
Shipping/handling ______
☐ MasterCard ☐ VISA — TOTAL ______

______ Exp. Date ______

Signature ______
Ship to:
Name ______
Street ______
City/St/Zip ______
Country ______

(5a)



(5b)

```
FF  FF  FF  07  90  00  FF  6F
00  03  80  00  FF  AF  00  FF -
6F  01
```

Figure 5: *Finding data via the directory and the file allocation table. Figure 5a shows how pointers are used to direct the operating system to the sequential parts of a file. The data stored in the sample file-allocation table is displayed in hexadecimal in figure 5b.*

or randomly without going to disk except for the data access itself. Schemes used in other operating systems (including CP/M and Unix) may require one or more disk reads simply to find out where the data is, particularly with a random access. In an application such as a database inquiry, where frequent random access is the rule, this can easily make a 2 to 1 difference in performance.

How the FAT Works

The directory entry for each file has one allocation unit number in it: the number of the first unit in the file. If, as in the previous example, an allocation unit consists of four sectors of 128 bytes each, then just by looking at the directory you know where to find the first 512 bytes of the file. If the file is larger than this, you go to the FAT.

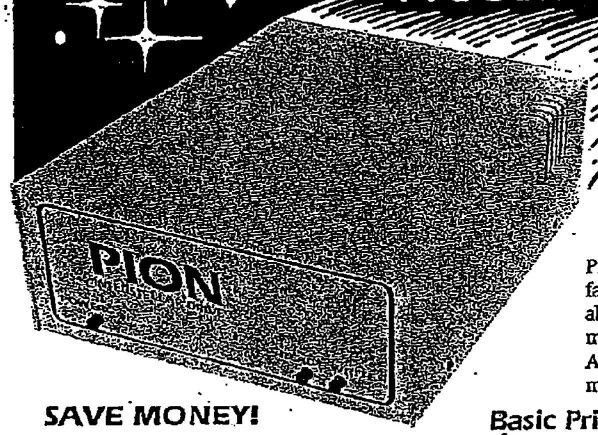
The FAT is a one-dimensional array of allocation unit numbers. As with any array, a given element is found with a numeric index. The numbers used as indexes into the FAT are also allocation unit numbers. Think of the FAT as a map, or translation table, that takes an allocation unit number as input and returns a different allocation unit number as output. The input can be any unit that is part of a file; the number returned is the next sequential unit of that file.

Let's look at the example in figure 5a. Suppose that the directory entry for a file specifies allocation unit number 5 as the first of the file. This locates the first four logical sectors (512 bytes). To find the next allocation unit of the file, look at entry 5 in the file allocation table. The 6 there tells you two things: first, the next four logical sectors of the file are in allocation unit number 6; and second, to find the unit after that, look at FAT entry number 6.

This process is repeated as you locate each allocation unit in the file. After number 6 comes number 3, then number 9, then number 10. In each case, the allocation unit number returned by the FAT tells you both

INTERSTELLAR DRIVE™

A SOLID STATE DISK EMULATOR



Save valuable time!
5 to 50 times faster performance than floppy disks and Winchester drives

PION'S INTERSTELLAR DRIVE is designed for use with a family of interfaces and software packages. Currently available are interfaces for IBM, S100, TRS80, Apple, SS50, and most Z80 uP, and software for most popular operating systems. Additional interfaces are continually being developed for the most popular computers.

SAVE MONEY!
Increase your computer's productivity

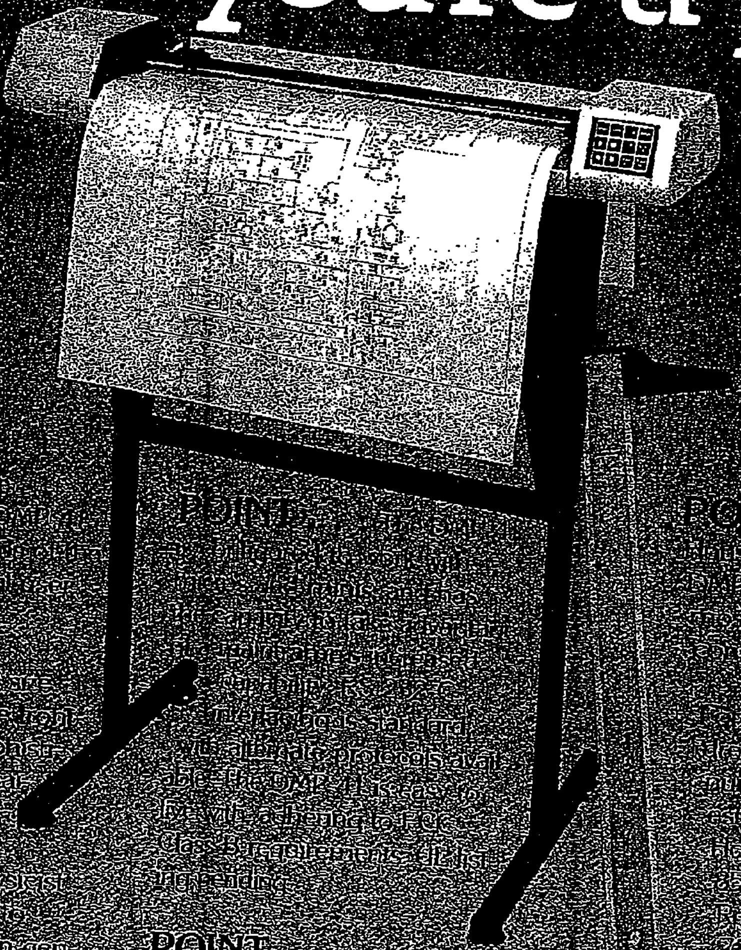
The INTERSTELLAR DRIVE is a high performance data storage subsystem with independent power supply, battery backup, and error detection. It has 256KB to 1 Megabyte of solid state memory integrated to perform with your operating system.

Basic Price for 256KB unit [includes interface and software]
$1095. *plus tax (where applicable) and shipping*
Visa and Master Card accepted

PION, INC. Tel.(617)923-8009
101R Walnut St., Watertown, MA 02172
TRS80 trademark of Tandy Corp. Apple trademark of Apple Computers
Interstellar Drive trademark of PION, Inc.

Circle 304 on inquiry card.

It's easy to make points when you're a pro.



POINT . . . The [illegible] DMP [illegible] [illegible] professional yet easy to operate.

POINT . . . [illegible] comprehensive front panel controls and sophisticated firmware are all [illegible] to the needs of the surveyor, drafter, cartographer, geophysicist and land developer. [illegible] You can generate superior architectural elevations, contour maps, circuit board layouts and assembly drawings quickly and accurately on bond, vellum or synthetic media.

POINT . . . [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] The [illegible] is standard with alternate protocols available. The DMP-41 is [illegible] [illegible] adhering to FCC class B requirements. UL list[illegible] pending.

POINT . . . Minutely defined step size and high resolution logic — combined with robust drives and optimized pen ballistics enable you to create plots of high precision and surpassing quality.

POINT . . . [illegible] [illegible] [illegible] [illegible] considerations.

For the name, address and phone number of your nearest distributor, write Houston Instrument, 8500 Cameron Road, Austin, Texas 78753. Phone 512-835-0900 or 800-531-5205 if outside Texas. In Europe contact Bausch & Lomb Belgium NV, Rochesterlaan 6, 8240 Gistel, Belgium. Tel 059-27-74-45 tx 846-81339.

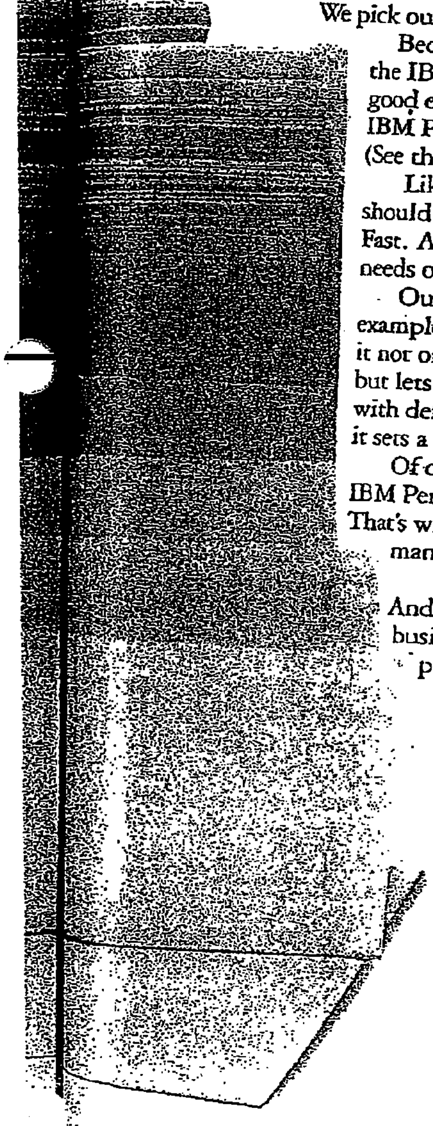
BAUSCH & LOMB
houston instrument division

Suggested US retail $2,995

The Chaplin character licensed by Bubbles, Inc., S.A.

For more ir

Congratulations. We published your program.



The envelope, please.

There's an acceptance letter inside. And a check that could have your name on it. (If we select your program, that is.)

But remember.

We pick our winners carefully.

Because the software we publish for the IBM Personal Computer has to be good enough to complement IBM Personal Computer hardware. (See the box at right.)

Like our hardware, this software should be simple to use. Friendly. Fast. And written to help satisfy the needs of the individual.

Our Personal Editor is a perfect example. A versatile text file editor, it not only helps the user save time, but lets him easily self-tailor a task with definable function keys. And it sets a standard of excellence.

Of course, every person will use the IBM Personal Computer differently. That's why we plan on publishing many different programs.

Entertainment programs. And educational programs. And business programs. And personal productivity programs. And graphics. And games. And more.

We'll also consider software written *by* programmers *for* programmers. For example, the BASIC Program Development System, Professional Editor and Diskette Librarian

IBM PERSONAL COMPUTER SPECIFICATIONS

User Memory
64K-640K bytes
Microprocessor
16-bit, 8088
Auxiliary Memory
2 optional internal diskette drives, 5¼" 160KB/180KB or 320KB/360KB per diskette
Keyboard
83 keys, 6 ft. cord attaches to system unit
10 function keys
10-key numeric pad
Diagnostics
Power-on self testing
Parity checking

Display Screens
Color or monochrome
High-resolution
80 characters x 25 lines
Upper and lower case
Operating Systems
DOS, UCSD p-System, CP/M-86†
Languages
BASIC, Pascal, FORTRAN, MACRO Assembler, COBOL
Printer
All-points-addressable graphics capability
Bidirectional
80 characters/second
18 character styles
9 x 9 character matrix

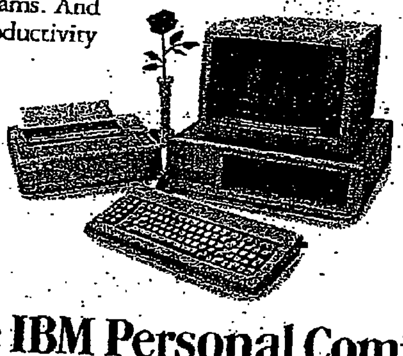
Permanent Memory
(ROM) 40K bytes
Color/Graphics
Text mode:
16 colors
256 characters and symbols in ROM
Graphics mode:
4-color resolution: 320h x 200v
Black & white resolution: 640h x 200v
Simultaneous graphics & text capability
Communications
RS-232-C interface
SDLC, Asynchronous, Bisynchronous protocols
Up to 9600 bits per second

are high-quality, full-function tools that were submitted by authors like you and subsequently published by us.

Now you might have the chance to win. Who knows? You could open the mailbox and find one of the envelopes shown here.

For information on how to submit your program, if completed and running, write: IBM Personal Computer External Submissions, Dept. 765.PC, Armonk, New York 10504.

IBM



The IBM Personal Computer
A tool for modern times

Circle 180 on inquiry card.

†For more information on where to buy the IBM Personal Computer, call 800-447-4700. In Alaska or Hawaii, 800-447-0890.

Circle 56 on inquiry card.

CARTRIDGE RIBBONS FOR
APPLE PRINTERS
NEC 8023A
C. ITOH PROWRITER
$9.95 EA $107.46 DOZ

CARTRIDGE RIBBONS FOR
EPSON
MX-80 MX-100
$6.99 EA $11.95 EA
$86.29 EA $129.08 DOZ

DUAL SPOOL RIBBONS FOR
OKIDATA PRINTERS

	EA	DOZ
80, 82, 83, 92, 93	$2.77	$29.92
84	$5.99	$64.69

LABEL SPECIAL
$2.99/K
(5K MIN)
1 ACROSS 3" x 15/16 CONTINUOUS LABELS

MAXELL DISKETTES
5¼" SINGLE SIDE DUAL DENSITY
MD-1
$29.90 10 PACK

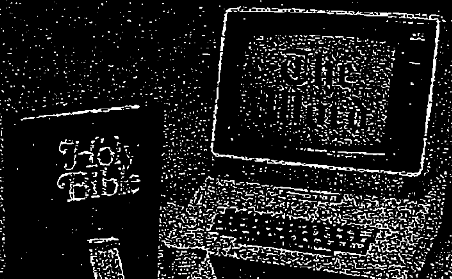
INNOVATIVE CONCEPTS
FLIP'N'FILE
DISC STORAGE BOX
HOLDS UP TO 60 DISKETTES
5¼" $24.95 EA
8" $29.95 EA

COMPLETE LINE OF OTHER RIBBONS AVAILABLE. PLEASE CALL
ALL ABOVE PRICES INCLUDE SHIPPING

Check-Mate™

51 DIAUTO DR. P.O. BOX 103
RANDOLPH, MA 02368
TOLL FREE 800-343-7706 IN MASS 617-963-7694
WE ACCEPT MASTER CARD & VISA
MASS. RESIDENTS ADD 5% SALES TAX
PHONES OPEN 9AM-7PM EASTERN TIME

For Those Who Seek.



[illegible]

TOPICS $49.95 • Bible Research System • THE WORD $199.95

[illegible]

Circle 43 on Inquiry card.

where the data is (i.e., in which unit) and where to look in the FAT for the next allocation unit number.

If you followed the example all the way through, you should have noticed that entry 10 in the FAT contains a −1. This, as you might have guessed, is the end-of-file mark. Allocation unit number 10 is last in the file, so its entry contains this special flag. (Another special value in the FAT is 0, which marks a free allocation unit.)

Now you know that this file occupies five allocation units, numbers 5, 6, 3, 9, and 10 in order. The file could be extended by finding *any* free unit, say 27, putting its number in entry 10 (where the −1 is now), and marking it with the −1 for end of file. That is, entry 10 in the FAT will contain 27, and entry 27 will contain −1. This demonstrates how you can extend any file at will and that you can use any free space on the disk when needed, without regard to its physical location.

If you ever want to look at a real FAT, there's one more thing you'll need to know. Each FAT entry is 12 bits (1.5 bytes) long. These entries are packed together, so two of them fit in 3 bytes. From a programming viewpoint, you would look up an entry in the FAT this way: Multiply the entry number by 1½, truncating it to an integer if necessary. Fetch the 16-bit word at that location in the FAT. If the original entry number was odd (so that truncation was necessary in the first step), shift the word right 4 bits; the lowest 12 bits of the word is the contents of the FAT entry. Reading a FAT from a hexadecimal dump isn't nearly as simple! Figure 5b shows the hexadecimal version of the sample FAT I've been using.

File System in Action

To put all this in perspective, you need to look at how MS-DOS handles a file-transfer request from an application program. With MS-DOS, application programs treat files as if they were divided into logical records. The size of the logical record is entirely dependent on the application and may range from 1 byte to 65,535 bytes. It is not a permanent feature of

SuperSoft FORTRAN

Now for CP/M-86®, MS DOS, and IBM PC DOS®

SuperSoft FORTRAN is the answer to the growing need for a high quality FORTRAN compiler running under CP/M-86 and IBM PC DOS. It has major advantages over other FORTRAN compilers for the 8086. For example, consider the benchmark program used to test the IBM FORTRAN in InfoWorld, p. 44, Oct. 25, 1982. (While the differential listed will not be the same for all benchmark programs, we feel it is a good indication of the quality of our compiler.) Results are as follows:

IBM FORTRAN:	38.0 Seconds
SuperSoft FORTRAN:	2.8 Seconds

In its first release SuperSoft FORTRAN offers the following outstanding features:

1. Full ANSI 66 standard FORTRAN with important extensions
2. Standard data types, double precision, varying string length, complex numbers
3. Free format input and free format string output
4. Compact object code and run time support
5. Special functions include string functions, dynamic allocation, time/date, and video access
6. Debug support: subscript checking, good runtime messages
7. Full IEEE floating point
8. Full 8087 support—available as option ($50.00)

Program developers:

SuperSoft's family of FORTRAN compilers means you can write your programs once and they will run under CP/M-80, CP/M-86, and MS DOS. This lets you get your applications running fast no matter what the environment.

The current compiler allows 64K code space and 64K data space with expansion anticipated in future releases.

"At last, a FORTRAN compiler that works great on my 8086, 8087, and 8088 based systems!"

SuperSoft FORTRAN: available NOW and working great!

Requires:	128K with CP/M-86® and MS DOS
Price:	$425 (in each environment)

CP/M-80 version also available

In conjunction with SuperSoft, SuperSoft FORTRAN was developed by Small Systems Services, Urbana, IL, a leader in FORTRAN development.

CP/M and CP/M-86 are registered trademarks of Digital Research.

Japanese Distributor: ASR Corporation International, 4-23-8 Nishi-Shimbashi, Minato-Ku, Tokyo 105, Japan. Tel. (03) 437517, Telex 0242 2723

European Agent: Micro Technology Ltd, 51 The Pantiles, Tunbridge Wells, Kent, England TN2 5TE, Tel. 0892-45433, Telex 95141 Micro G

Circle 369 on inquiry card.

SuperSoft

FIRST IN SOFTWARE TECHNOLOGY P.O.Box 1628 Champaign, IL 61820 (217) 359-2112 Telex 270365

(6a)

$$\frac{\text{byte position } 1200}{128\,\frac{\text{bytes}}{\text{sector}}} = 9\text{, remainder } 48$$

Therefore, first byte to transfer is sector 9, byte 48

128 − 48 = 80 bytes to transfer in first sector
1200 − 80 = 1120 bytes left after first sector

$$\frac{1120 \text{ bytes}}{128\,\frac{\text{bytes}}{\text{sector}}} = 8\text{, remainder } 96$$

Therefore, transfer 8 whole sectors, then 96 bytes of last sector

(6b)

$$\frac{9 \text{ sectors}}{4\,\frac{\text{sectors}}{\text{allocation unit}}} = 2\text{, remainder } 1$$

Therefore, first sector to transfer is allocation unit 2, sector 1.

Figure 6: *Calculating physical, byte-level operations from logical definitions. The process outlined in figure 6a shows how the amount of data is calculated in physical terms. The actual position of data on the disk is computed in figure 6b.*

a file but in fact may vary from one file-transfer request to the next. The logical record size currently being used is passed to MS-DOS for each transfer made. It is, of course, completely independent of the physical sector size the disk uses.

To read a file, an application program passes to MS-DOS the size of a logical record, the first logical record to read, and the number of sequential logical records to read. Let's follow an example of how MS-DOS uses this information. Assume the application program is using 80-byte records and is set up to read a file 15 records at a time. Let's pick things up on its second read, that is, after it has already taken the first 15 records and is about to read the second 15. The request will be for an 80-byte record, the first record is number 15, and you want to read 15 records. Now pretend you're MS-DOS and analyze this request.

You immediately convert the request into byte-level operations. First multiply the logical record size by the record number, to get the byte position to start reading (80 × 15 = 1200). Then multiply the record size by the number of records, to get the number of bytes to transfer (also 1200).

Next, these numbers must be put in terms of physical disk sectors. This requires some divisions and subtractions involving the physical sector size and results in the breakdown of the transfer into three distinct pieces: (1) the position in the file of the first physical sector and the first byte within that sector to be transferred, (2) the number of whole sectors to transfer after the first (partial) sector, and (3) the number of bytes of the last (partial) sector to be transferred. The calculations and their results are shown in figure 6a. It is quite common for one or two of these pieces to be of length 0, in which case some of the following steps are not performed.

At this point, there is still no hint as to where the data will actually be found on the disk. You know that you want the tenth sector of the file (sector 9 because you start counting with 0), but you're not yet ready to check with the FAT to see where it is. The sector position in the file must be broken into two new numbers: the allocation unit position in the file and the sector within the allocation unit. For this example with single-density 8-inch disks (four sectors per allocation unit), this would be the third unit from the start of the file and the second sector within the unit (figure 6b).

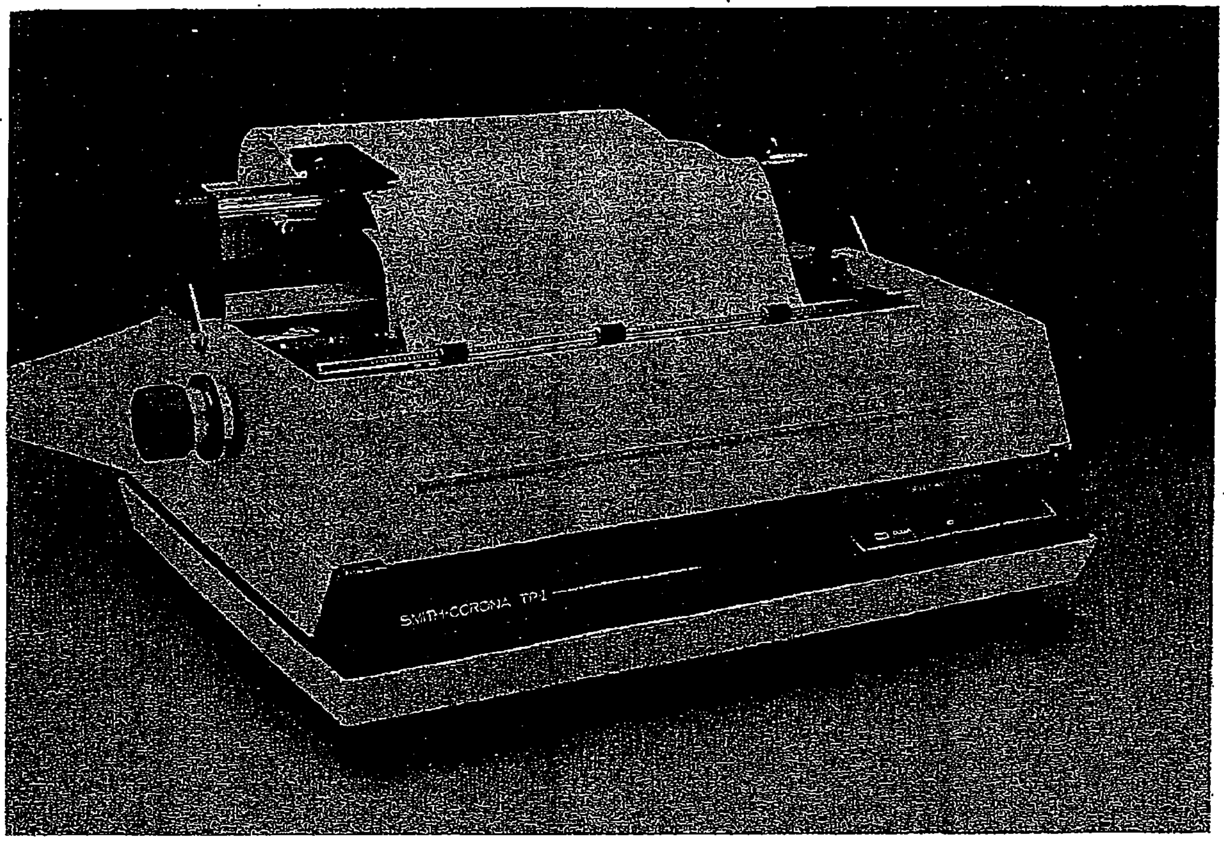
What you need to do now is skip through the FAT to the third allocation unit in the file. If your file is the same one shown in figure 5, then from the directory entry you learn that the first unit is number 5. Looking at entry 5 of the FAT, you see the second unit is number 6. And finally, from entry 6 of the FAT, you find the third unit of the file, number 3. Table 2 reminds you that allocation unit number 3 is made up of physical sectors 34, 35, 36, and 37; therefore, physical sector 35 is what you are looking for.

Now the disk reads begin. Physical sector 35, only part of which is needed, is read into the single buffer kept in MS-DOS solely for this purpose. Then that part of the sector that is needed is moved as a block into the place requested by the application program.

Next, the whole sectors are read. MS-DOS looks ahead in the FAT to see if the allocation units to be transferred are consecutive. If so, they are combined into a single multiple-sector I/O system transfer request, which allows the I/O system to optimize the transfer. This is the primary reason why MS-DOS disks do not ordinarily use any form of sector interleaving: a well-written I/O system will be able to transfer consecutive disk sectors if told to do it in a single request. The overhead of making the request, however, would often be too great to transfer consecutive sectors if it were done on a sector-by-sector basis.

Back to the example. MS-DOS will request that the I/O system read sectors 36 and 37 directly into the memory location called for by the application. Then, noting that allocation units 9 and 10 are consecutive, the corresponding sectors 58, 59, 60, and 61 from unit 9 and sectors 62 and 63 from unit 10 will be read by the I/O system in a single request. This completes the transfer of the eight whole sectors.

Smith-Corona makes a good deal better. With a $50 rebate.



The Smith-Corona® TP-I™ daisy wheel printer with optional tractor feed.

Ever since the Smith-Corona TP-I was introduced, it's been a great success with critics and users. And it's been a good deal at its low price.

Now, it's a better deal than ever. Because now you can get this high-quality, low-cost printer for even less.

Buy a Smith-Corona TP-I letter-quality printer any time between June 1 and July 31, and Smith-Corona will send you a $50 refund.

Of course, what you'll be getting is more than a good deal. Because the Smith-Corona TP-I is a printer with the same excellent print quality as found on the finest office typewriters.

The TP-I handles letter and legal sized paper. And with the new tractor feed option, the TP-I can handle both fanfold and single sheet paper—without ever having to remove the tractor feed!

The TP-I is very simple to operate. It's compatible with most personal and home computers and available with either standard serial or parallel data interface. And, unlike many printers, the TP-I is made in America.

There's a choice of state-of-the-art daisy wheels to give you a wide variety of fonts. (At $7.95 each, you can easily afford several.)

So if you're in the market for a high-quality, low-cost daisy wheel printer, get the Smith-Corona TP-I. Get it now and make a good deal a good deal better—with a $50 rebate.

Please send me more information on the Smith-Corona TP-I daisy wheel printer.

Name__________ Title__________

Company Name__________

Business Address__________

City__________ State_____ Zip_____

Type of Business__________

Mail coupon to:
Jerry Diener—V.P. Sales, Smith-Corona
65 Locust Avenue
New Canaan, Connecticut 06840

Smith-Corona B6

A Short History of MS-DOS

Known variously as Seattle Computer 86-DOS, IBM Personal Computer DOS, and Zenith Z-DOS, MS-DOS was developed by Seattle Computer Products for its 8086-based computer system. The MS-DOS history is intertwined with the general development of software for 8086-based computers.

In May 1979, Seattle Computer made the first prototype of its 8086 microprocessor card for the S-100 bus. There were brief discussions with Digital Research about using one of Seattle Computer's prototypes to aid in developing CP/M-86, which was to be ready soon. Although Seattle Computer was considering using CP/M-86 when it became available (expected no later than the end of 1979), there were only two working prototypes of the 8086 processor card and it was felt that both were needed in house. Therefore there wasn't one free for Digital Research.

Microsoft had already started a strong 8086 software development program. The firm was ready to try the 8086 version of Stand-Alone Disk BASIC, which is a version of its BASIC interpreter with a built-in operating system. During the last two weeks of May 1979, this BASIC was made completely functional using the hardware that Seattle Computer provided for Microsoft. Seattle Computer Products displayed the complete package (8086 running disk BASIC) in New York the first week of June at the 1979 National Computer Conference. (This was the first-ever public display of an 8086 BASIC and of an 8086 processor card for the S-100 bus.)

Seattle Computer shipped its first 8086 cards in November 1979 with Stand-Alone Disk BASIC as the only software to run on it. The months rolled by, and CP/M-86 was nowhere in sight. Finally, in April 1980, Seattle decided to create its own DOS. This decision resulted just as much from concern about CP/M's shortcomings as from the urgent need for a general-purpose operating system.

The first versions of the operating system, called QDOS 0.10, were shipped in August 1980. QDOS stood for Quick and Dirty Operating System because it was thrown together in such a hurry (two man-months), but it worked surprisingly well. It had all the basic utilities for assembly-language development except an editor. One week later, Seattle Computer had created an operating system with an editor—an absurdity known as EDLIN (editor of lines). A primitive line-oriented system, it was supposed to last less than six months. (Unfortunately it has lasted much longer than that as part MS-DOS.)

In the last few days of 1980, a new version of the DOS was released, now known as 86-DOS version 0.3. Seattle Computer passed this new version on to Microsoft, which had bought nonexclusive rights to market 86-DOS and had one customer for it at the time. Also about this time, Digital Research released the first copies of CP/M-86. In April 1981, Seattle Computer Products released 86-DOS version 1.00, which was very similar to the versions of MS-DOS that are widely distributed today.

In July 1981, Microsoft bought all rights to the DOS from Seattle Computer, and the name MS-DOS was adopted. Shortly afterward, IBM announced the Personal Computer, using as its operating system what was essentially Seattle Computer's 86-DOS 1.14. Microsoft has been continuously improving the DOS, providing version 1.24 to IBM (as IBM's version 1.1) with MS-DOS version 1.25 as the general release to all MS-DOS customers in March 1982. Now version 2.0, released in February 1983, has just been announced with IBM's new XT computer.

To finish the job, sector number 64 is read into the internal MS-DOS sector buffer. Its first 96 bytes are moved to the application program's area.

The Sector Buffer

This example shows the internal MS-DOS sector buffer being used in a very simple way. In reality, MS-DOS would normally perform the disk read in the example more efficiently than described here due to its optimized buffer handling. By keeping track of the contents of the buffer, disk accesses are minimized. The resulting speed improvement can be dramatic particularly when the requested transfer size is small (a fraction of a sector).

In the example, I assumed the application program was sequentially reading 15-record chunks (at 80 bytes per record) and had already completed the first such read. This would mean that sector 35 (the first one read in this example) would already be in the sector buffer because its first 48 bytes were needed for the previous read. MS-DOS would not reread this sector but instead would simply copy the remaining 80 bytes into the area designated by the application.

The presence of only one sector buffer in MS-DOS is a design inadequacy that is difficult to defend.

Likewise, when the application is ready to read the third chunk of the file, MS-DOS will find sector 64 already in the sector buffer. The last 32 bytes of the sector will be moved into place without a disk read.

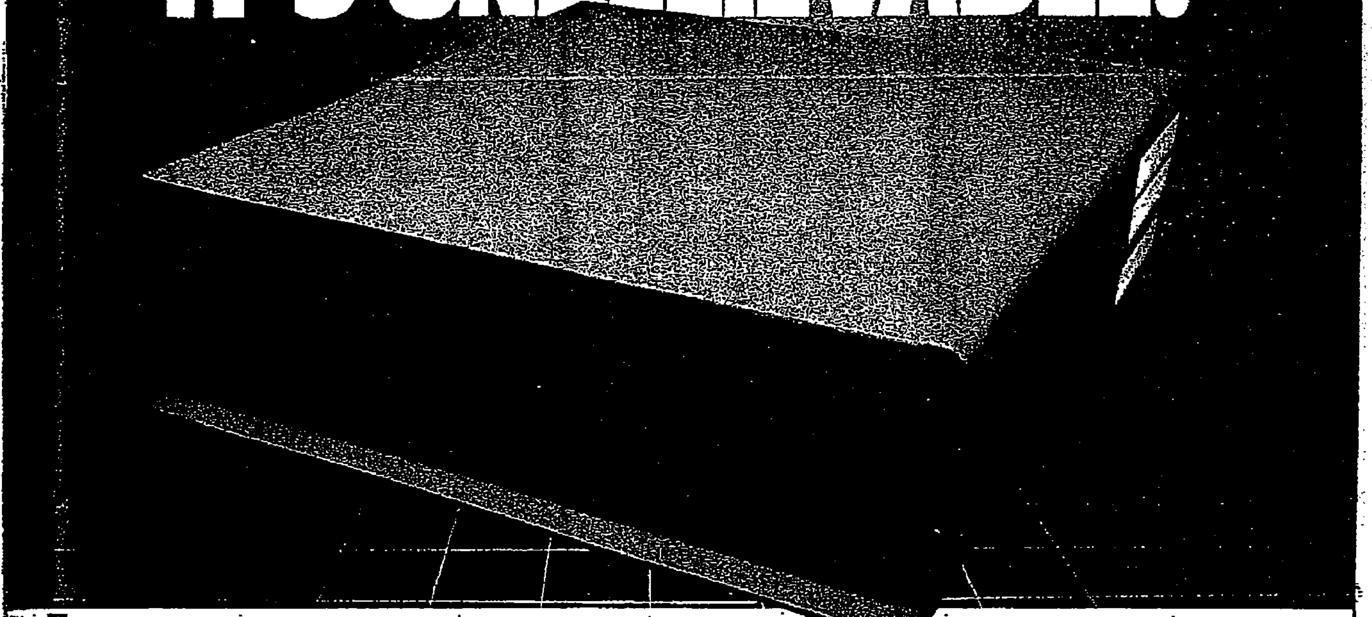
For its own internal simplicity, MS-DOS has only one sector buffer. Between the 15-record reads, should the application request some other transfer that requires use of the buffer, then the buffer contents will be changed, and these optimizations are not possible. In this particular case, in which most of the disk transfer does not need the buffer, there will be very little difference in speed either way. Let's look at a different case where this optimization is practically essential.

Suppose the application wishes to write a file sequentially, one 16-byte record at a time. When the first record is written, MS-DOS simply copies the 16 bytes into the first part of the sector buffer. As each of the next seven records is written, it too

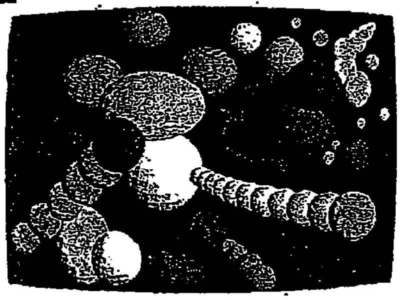
Circle 405 on Inquiry card.

512 SIMULTANEOUS COLORS DISPLAYED FROM A 16 MILLION COLOR PALETTE.

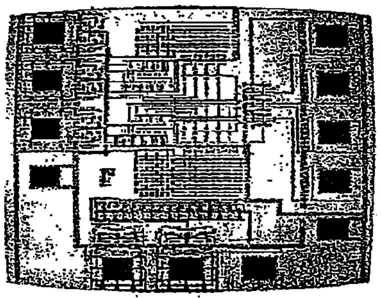
IT'S UNBELIEVABLE.



$4295 COLOR GRAPHICS.



"In The Beginning" By Richard Katz, Vectrix Corporation

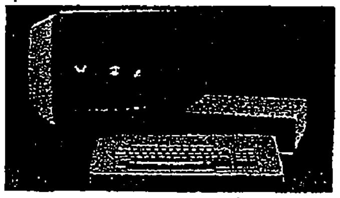


"Integrated Circuit Design" Courtesy of Floyd J. James, University of North Carolina at Chapel Hill

"Weather Satellite Image" Copyright WSI Corporation

VX384 • VERY HIGH RESOLUTION 672 by 480 pixels
• 512 COLORS PER PIXEL, 384K of graphics RAM
• COLOR LOOK UP TABLE with 16 million color palette
• ON-BOARD 16 BIT MICROCOMPUTER
• 3D GRAPHICS SOFTWARE PACKAGE, including rotation, scaling, translation, perspective, clipping, viewport, polygon and filled polygon
• HARDWARE LINE AND ARC GENERATION
• ON-BOARD AND USER DEFINABLE CHARACTER SET
• SERIAL OR PARALLEL INTERFACE to almost any host computer

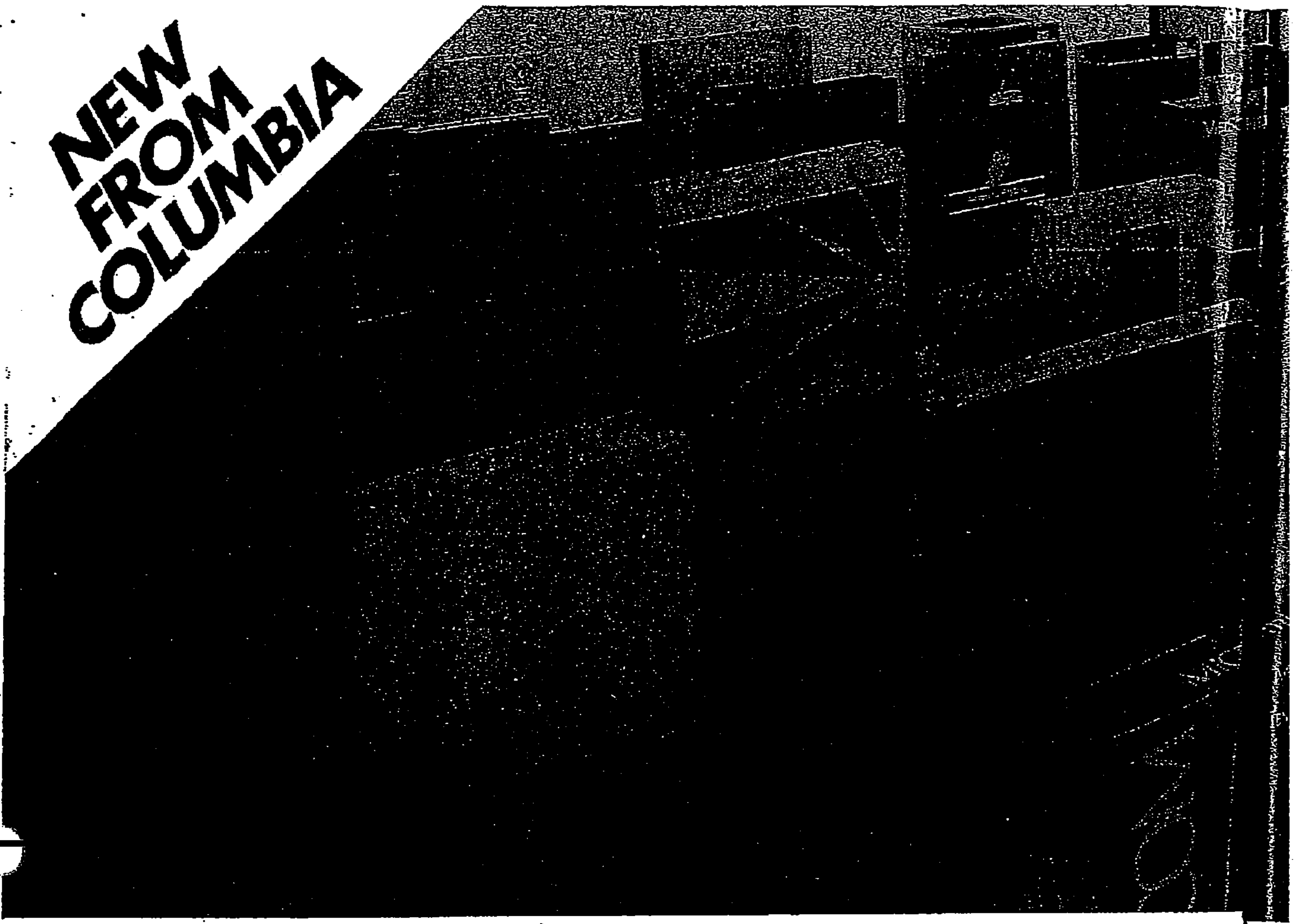
VX128 • 8 COLORS PER PIXEL, 128K of graphics RAM
• INCLUDES ALL FEATURES OF VX384 without look-up table for $2,495
• VXM HIGH RESOLUTION RGB COLOR MONITOR $1295
• COLOR GRAPHIC PRINTER with interface cable $1695
• FOR ADDITIONAL INFORMATION call Toll Free or write VECTRIX Corporation, 1416 Boston Road, Greensboro, NC 27407



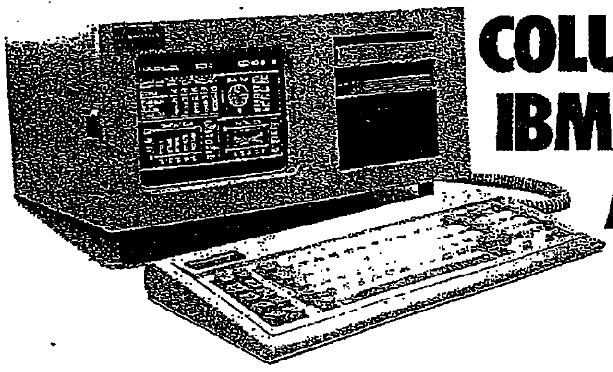
IF YOU STILL DON'T BELIEVE IT, CALL TOLL FREE:
1-800-334-8181

VECTRIX

NEW FROM COLUMBIA



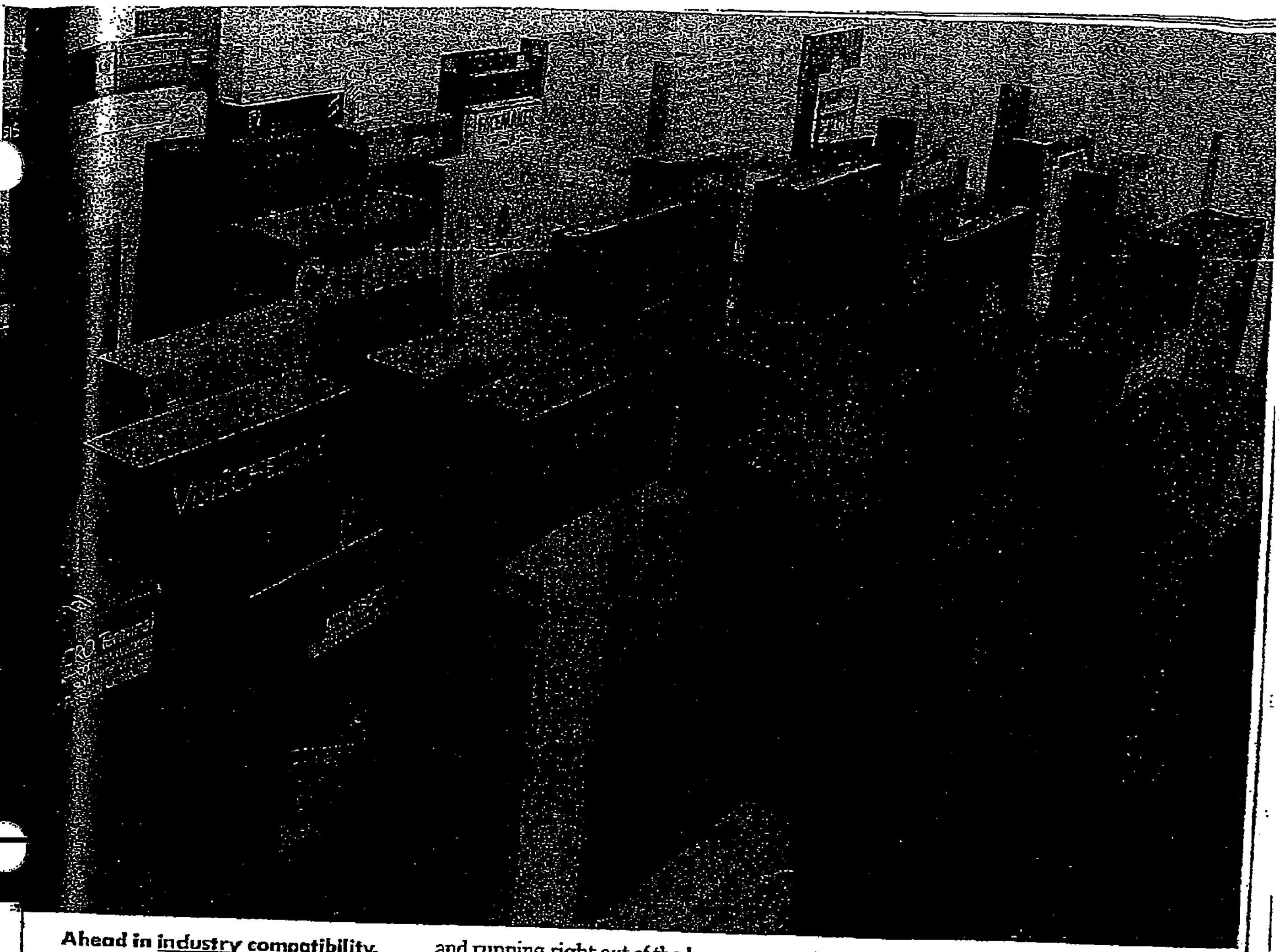
Shown are IBM-PC* compatible programs. The Columbia VP portable runs MS-DOS* plus five other operating systems.



INTRODUCING THE COLUMBIA VP PORTABLE. IBM-PC COMPATIBILITY AT AN AFFORDABLE PRICE.

World Headquarters:	**West Coast:**	**Europe:**	**Call our distributor nearest you.**		
9150 Rumsey Road	3901 MacArthur Blvd.	Limitenstr. 94	Access Systems	Advanced Management Systems	Peripherals Plus, Inc.
Columbia, MD 21045	Suite 211	4050 Moenchengladbach 2	Wellesley, MA	Aurora, CO	Montreal, Quebec, Canada
(301) 992-3400	Newport Beach, CA 92660	West Germany	(617) 237-7743	(303) 752-2972	(514) 849-7533
TWX 710-862-1891	(714) 752-5245	02166-47097			N.I.D.I. (National Instrument Distribution Inc.)
	Telex 277778	Telex 852452			Dayton, OH
					(513) 435-4503

Distributors in Australia, Austria, Belgium, Colombia, Denmark, Hong Kong, Israel, Italy, Malaysia, Netherlands-Antilles, Norway, Portugal, Spain, Sweden, Switzerland, United Kingdom, Venezuela.



Ahead in industry compatibility.

Today, the Columbia VP portable takes on hundreds of IBM-PC compatible software programs and IBM-PC add-ons or peripherals.

What's more, six other Columbia-supported operating systems are available—CP/M-80®, CP/M-86®, (Concurrent CP/M-86®, OASIS-16® and XENIX® available soon)—stretching the Columbia VP's software compatibility beyond any other personal computer you can buy.

Get started with thousands of dollars worth of software FREE.

Every Columbia VP portable is shipped with fully supported software that will save you thousands of dollars on your initial purchase. That means your Columbia is up and running right out of the box. Included is:

Perfect Writer™	Perfect Speller™
Perfect Filer™	Perfect Calc™
MS-DOS® with RAM disk	BASICA®
Macro/86 Assembler	Asynchronous Communications Support
MS-BASIC®	Columbia Tutor
Diagnostics	Home Accountant Plus™
Space Commanders™	CP/M-86®
Fast Graphs®	

Full feature performance at an affordable price.

The Columbia VP portable features an 8088 16-bit CPU, 128K RAM, (additional 128K, optional), 640K in dual disk drives, one IBM-PC compatible expansion slot, one serial and one parallel I/O, IBM-PC-compatible keyboard, and a 9" built-in monitor with graphics.

The best news of all is the price: $2,995—including software. And the Columbia is built with lasting value in mind. Rugged single board design plus the flexibility to expand and personalize your Columbia VP.

Made in U.S.A.—supported worldwide.

All Columbia hardware and software are backed by our dealers and distributors worldwide with national service provided by Bell & Howell. Call for the name of the dealer in your area. Let us show you a whole new world of performance and value.

COLUMBIA
DATA PRODUCTS, INC.

Circle 65 on inquiry card.

Commercial Computer Sales
Atlanta, GA
(404) 256-9190

Renaissance Technology Corp.
Concord, CA
(415) 676-5757

Compatible Data Systems, Inc.
Rochester, NY
(315) 437-3909

Southeastern Data Products
Lynchburg, VA
(804) 237-6286

MP Systems
Dallas, TX
(214) 385-8885

Commercial Computer Sales
Miami, FL
(305) 266-9569

Mid Tech Associates
Desoto, KS
(913) 441-6565

Tele-Terminals
Brooklyn Park, MN
(612) 536-6045

Mid Tech Associates
Florissant, MO
(314) 837-5200

Waybern Corporation
Garden Grove, CA
(714) 554-4520

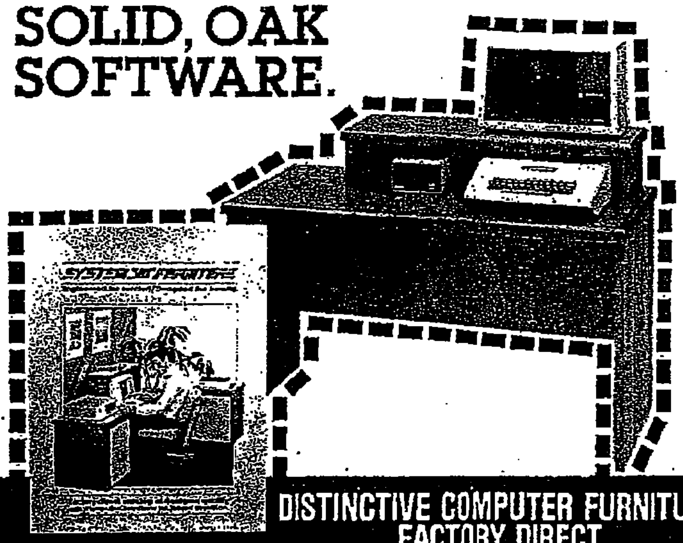
Micro Distributing, Inc.
Coquitlam, B.C. Canada
(604) 941-0622

Tele-Terminals
Milwaukee, WI
(414) 785-9221

Perfect Software, Perfect Writer, Perfect Speller, Perfect Filer, and Perfect Calc are trademarks of Perfect Software, Inc. Home Accountant Plus is a trademark of Continental Software Company. Fast Graphs is a trademark of Innovative Software, Inc. IBM and IBM-PC are trademarks of International Business Machines. CP/M and MP/M are registered trademarks of Digital Research, Inc. OASIS is a trademark of Phase One. MS-DOS, MS-BASIC and XENIX are trademarks of MICROSOFT.

Circle 164 on inquiry card.

SOLID, OAK SOFTWARE.



DISTINCTIVE COMPUTER FURNITURE. FACTORY DIRECT. UNCONDITIONALLY GUARANTEED.

All the peripherals you can buy won't help the productivity of your computer system if you don't have an organized way to utilize them. System VII Furniture gives you an efficient, comfortable workspace. With the warmth and beauty of hand-rubbed Oak and the affordability of factory-direct delivery.

Write us, or call our toll-free number to order a full-color catalog. Send $1.00 plus $1.00 shipping and handling. *$2.00 will be credited toward first purchase.*

SYSTEM VII FURNITURE CALL: 1-800-547-8888 Dept. F
P.O. Box 427 • Lafayette, Oregon 97127

Introducing SPL the first multi-mode spooler for CP/M® computers

If you believed that your computer couldn't do better than a single task system think again. You can convert your machine into a dual-task computer with SPL, the amazing Spooler program developed by Blat R+D: SPL enables you to use hidden capacity available on your CP/M computer to print documents and run your ordinary programs, all at once.

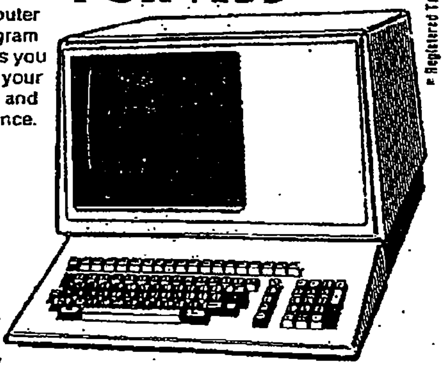
While printing, your regular programs won't stop processing, waiting for the printer to finish. SPL will store the information to be printed in internal or external (disk drives) memory until the printer is ready to receive the data. Result: your programs will run at full speed.

As SPL can use up to the full capacity of your disks for temporary storage, it's much more powerful than hardware spoolers, which are limited to 64k memory or less.

SPL is an advanced product with several modes of operation. In addition to intercepting the output to the printer, SPL can print your existing text files, or those that your programs will create from ɔw on. SPL will even take care of tab ɛpansion. As an added bonus, SPL needs no installation on most CP/M 2.x computers.

If you have a computer

Get A Second Computer FOR $139



You could get an equivalent increase in computing power by spending $1000 to $3000, but SPL is only $ 139, including disk and manual.

To order your SPL program call us today specifying what disk format you require. You can charge it to your VISA or Master Card if you prefer.

Blat Research+Development Corp.
8016 188th SW, Edmonds WA 98020
Call toll-free 1-800-LOBO-BAY
In WA call [206] 771-1408

® Registered Trademark of Digital Research

is just copied into the appropriate position in the sector buffer. Again with a 128-byte sector of an 8-inch single-density format, the sector buffer would be full at this point. Upon attempting to write the ninth record, MS-DOS would find it needs to put the record in a different sector from the one currently in the buffer. The current buffer contents are marked "dirty," meaning they must be written to disk rather than discarded. MS-DOS does this and then moves the ninth 16-byte record into the buffer.

Note that MS-DOS did not write the sector buffer automatically after 128 bytes had been written to it. This is because the DOS has no notion of a sequential file: every disk transfer has an explicitly specified record position and record size. Thus, it does not think of the buffer as "full"—for all it knows, the application program might back up and write the first 16 bytes over again. So the data is simply kept in the buffer until the file is closed or until the buffer is needed for something else.

Another optimization was taking place here that may have gone unnoticed. MS-DOS is always aware of the exact size of its files, and the assumption in the previous example was that this file was being newly written. Had it already existed, MS-DOS would have been forced to preread each sector into the sector buffer before copying any records into it. This is essential in case the program does random writes, intending to change only selected portions of the file. When the file is being extended (as in this case), the preread is not performed.

The possible outcome of this approach to buffer handling is that when the application program requests a write and is told it was successfully completed, the data may, in fact, not yet be written to the disk. The alternative approach is called *buffer write-through*, in which the data in the sector buffer would be written to disk each time the application requested a write. This would mean, in the example, eight rewrites of the same sector before moving on to the next, requiring a minimum of 1.2 seconds to write just 128 bytes! As

64

• Pa
• Ad
res
• Pin

• Addr
• Batte
• Funct

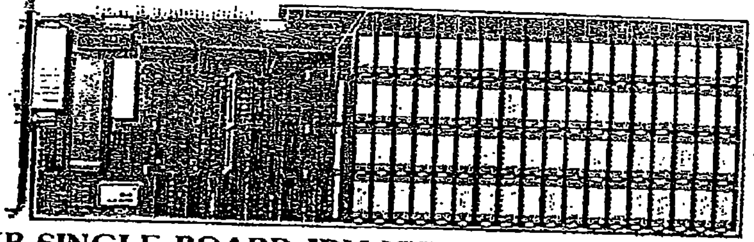
DON'T

Circle 44 on inquiry card.

Circle 58 on in

NEW UP TO 2MB ON A SINGLE CARD!

STATE OF THE ART MEMORY SYSTEMS

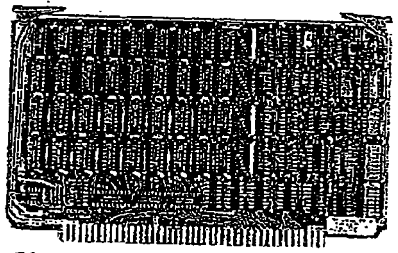


512KB SINGLE BOARD IBM MEMORY W/RS232-C PORT

- Addressable as a contiguous block in 64KB increments thru 1 megabyte.
- On board parity with interrupt on parity error.

SINGLE QTY. PRICE: $795.00

MEMDISK 1 Allows memory to emulate disks. Increases system performance!! FREE with purchase of memory.

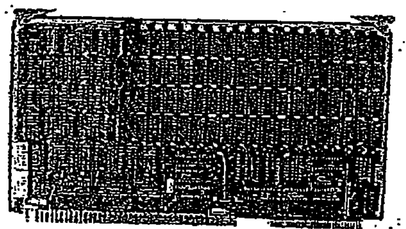


64KB SINGLE BOARD EXORCISER I, II, AND ROCKWELL SYSTEMS 65 MEMORY

- Parity checker on board.
- Addressable as a contiguous block in 4K increments with respect to VXA or VUA.
- Pin to Pin compatibility.

SINGLE QTY. PRICE: $250.00

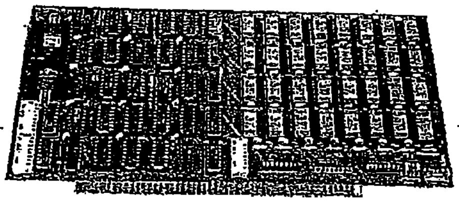
NEW



512MB TO 2MB SINGLE BOARD MULTIBUS MEMORY

- Pin to Pin MULTIBUS compatibility for both 8 bit and 16 bit systems.
- On board parity with selectable interrupt on parity ERROR.
- Addressable up to 16 megabytes.

SINGLE QTY. PRICE:		
	512KB	$ 895.00
	1MB	$4495.00
	2MB	$8700.00

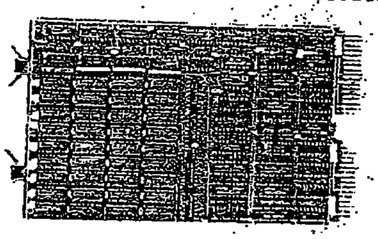


64KB SINGLE BOARD S100 MEMORY

- Addressable as a contiguous block in 4K word increments.
- Battery back-up capability.
- Functions with on-board refresh.

SINGLE QTY. PRICE: $250.00

NEW



256KB TO 1 MB SINGLE BOARD LSI 11 MEMORY

- On board parity generator checker.
- Addressable as contiguous block in 256KB increments through 4 megabytes.
- Battery back-up mode.

SINGLE QTY. PRICE:		
	256KB	$ 595.00
	512KB	$2650.00
	1MB	$3995.00

DON'T ASK WHY WE CHARGE SO LITTLE, ASK WHY THEY CHARGE SO MUCH.

Chrislin Industries, Inc.
31352 Via Colinas • Westlake Village, CA 91362 • 213-991-2254
TWX 910-494-1253 (CHRISLIN WKVG)

Circle 58 on inquiry card.

Circle 440 on inquiry card.

P&T CP/M®2 is GROWING

TRS-80 MODEL II $185

Still the best CP/M for the Mod II with features like 596 Kb per diskette, type ahead, full serial port support, and more.

TRS-80 MODEL 12/16 $220

Includes full support for thinline drives; gives 1.2 Mb per diskette for the Mod 16 (Z-80 mode), Mod 12 and Mod II's with double sided drives.

RADIO SHACK HARD DISK $250

Includes all the features of P&T CP/M 2 plus support for the 8.7 and 12 Mbyte hard disk drives.

CAMEO HARD DISK

Support for the standard Cameo hard disk system ($250) or the multiplexer (for multiple computers) system $400.

CORVUS HARD DISK $250

Support for a 5, 10, or 20 Mb Corvus hard disk system.

Start with a Model II floppy system and grow into a hard disk. Since all P&T CP/M 2 systems are fully compatible, you will have no conversion worries.

Special note: P&T hard disk systems allow you the user to configure logical drive assignments to your specifications. Write for more details.

Prepaid VISA, M/C, or COD orders accepted. All prices FOB Goleta and subject to change.

CP/M is a registered trademark of Digital Research. TRS-80 is a trademark of Tandy Corp.

PICKLES & TROUT
P.O. BOX 1206
GOLETA, CA 93116
(805) 685-4641

PICKLES & TROUT

the logical record size gets smaller, the time required to write becomes greater.

The presence of only one buffer does bring about the definite possibility of buffer "thrashing." Take the example of an application such as a compiler that will alternately read a small amount from one file and write a little bit to another. If both the reads and writes consist of a single 16-byte record, then the following sequence will be performed for each pair of records:

- read input file, get record from buffer
- read output file, put record into buffer
- write output file

For each record pair, three disk transfers are required. The result would be unbearably slow. The presence of only one sector buffer in MS-DOS is a design inadequacy that is difficult to defend (but it does help keep the DOS small). The practical solution is for applications that must access more than one file at a time to provide their own internal buffering. By requesting transfers that are at least half as big as the sector size, thrashing can be substantially reduced.

MS-DOS 2.0

Microsoft has now made available MS-DOS version 2.0 to all OEM (original equipment manufacturer) customers of previous versions. The 10 months put into version 2.0 by the MS-DOS team probably exceeds the total effort behind the previous 1.25 release, including the original development at Seattle Computer Products. While the changes have been substantial, the basic structure is still recognizable. I have been discussing the DOS at such a low level that most of what I've talked about applies directly to version 2.0 as well. Here are the three main differences, along with my personal comment as original author of the DOS. I was not involved in the MS-DOS 2.0 project.

MS-DOS 2.0 allows multiple-sector buffers. The number is determined by a configuration file when the DOS is loaded. It can be easily adjusted to the user's needs: for example, to accommodate more buffers to prevent thrashing (this is the ideal solution to the buffer thrashing problem previously discussed) and fewer buffers to make more system memory available.

The new MS-DOS does not keep the file allocation tables in memory at all times. Instead, the tables share the use of the sector buffers along with partial-sector data transfers. This means that at any one time, all, part, or none of a FAT may be in memory. The buffer-handling algorithms will presumably keep often-used sectors in memory, and this applies to individual sectors of the FAT as well. This change in the DOS goes completely against my original design principles. Memory is getting cheaper all the time, so dedicating a few thousand bytes to the FATs should be completely painless. Now we're back to doing disk reads just to find out where the data is. In the case of a random access to a large fragmented file (for example, when accessing a database that fills half of a small Winchester disk), it is possible that several sectors of the FAT would need to be visited, in random order, to find the needed allocation unit.

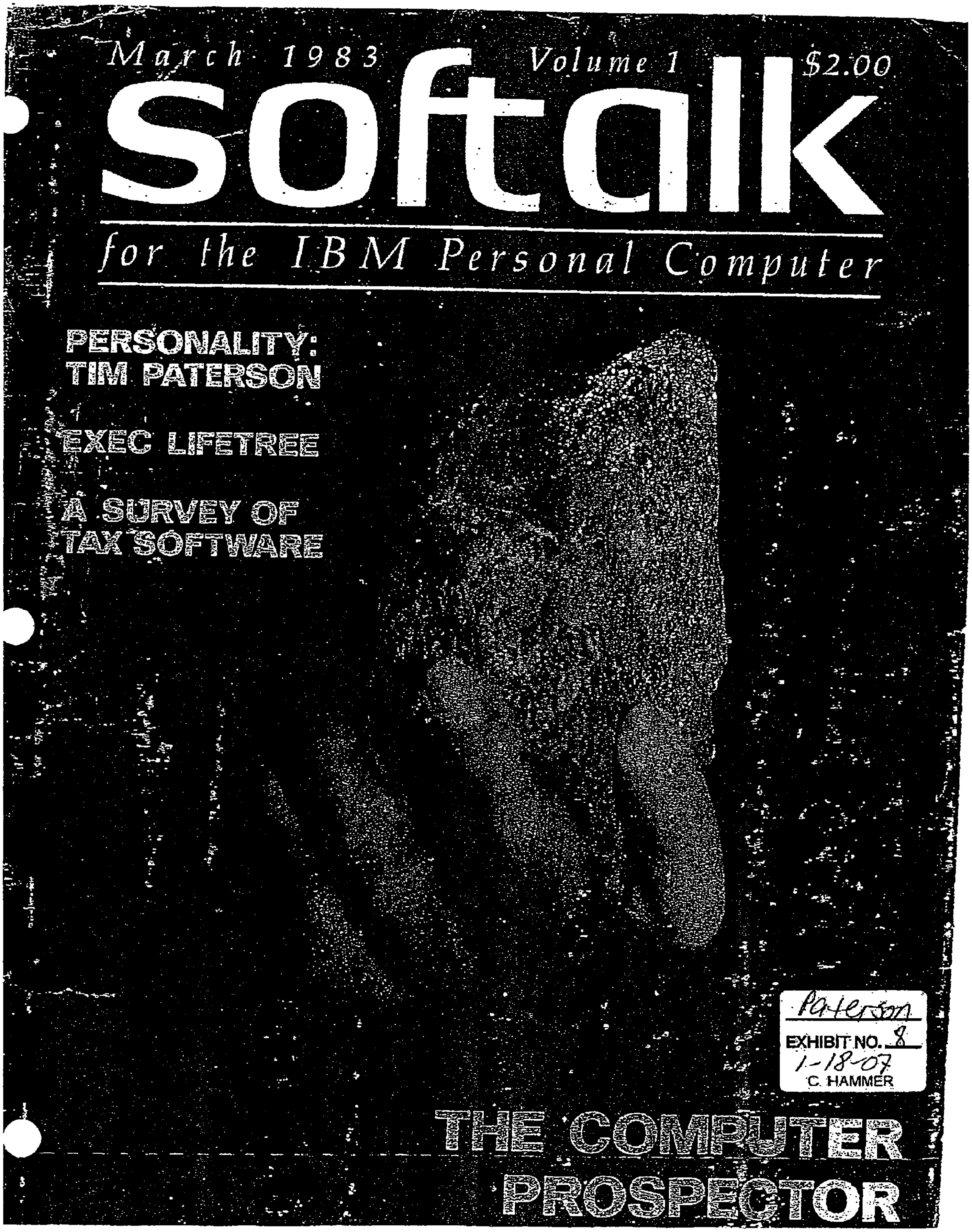
While MS-DOS retains the original fixed-size main directory, it now can have files as subdirectories. This hierarchical (tree-structured) directory system may be extended to any depth. This approach is nearly essential for users to keep track of all the files that might be on a hard disk.

MS-DOS version 2.0 is, on the whole, a substantial upgrade of the previous releases. The three preceding paragraphs are intended only to point out the way the 2.0 file structure differs from the file structure I've discussed, not to give you a complete product description. ■

About the Author

Tim Paterson worked for Seattle Computer Products on the design of its 8086 computer system and the operating system now called MS-DOS. He then worked for Microsoft for about a year. Since returning to Seattle Computer Products as director of engineering, he has been primarily involved with new hardware development.

EXHIBIT L



March 1983 Volume 1 $2.00

Softalk

for the IBM Personal Computer

PERSONALITY: TIM PATERSON

EXEC LIFETREE

A SURVEY OF TAX SOFTWARE

THE COMPUTER PROSPECTOR

TIM PATERSON THE ROOTS OF DOS

by David Hunter

"Life begins with a disk drive."
—Tim Paterson

One does not write an operating system and fail to pick up a little wisdom in the process, particularly when that operating system becomes the property of Microsoft Corporation.

Tim Paterson, who has changed jobs and companies fairly often in the past few years, is satisfied with his position at present. He's at Seattle Computer, a company that made its name in the S-100 market and has since developed its own microcomputer—the Gazelle. It's been almost a year since he quit Microsoft. His experiences constructing an operating system that eventually would be central to IBM's Personal Computer and many other computers is quite a story.

Seen and Not Seen. Paterson looks good. He wears faded jeans. He has a dark beard and moustache, and he often breaks into a mischievous grin. A terrific programmer and hardware engineer, at twenty-six Paterson is typically innocent-looking.

He and his kind are the backbone of computer companies. There is no shortage of business people and production people in the computer industry. There are only a few terrific programmers. Everyone else is replaceable. Yet, in a big company, programmers are sometimes the least known, least appreciated employees.

"With all the code being written out there, who gets the credit? People like Bill Gates get it all and he hasn't written anything in years," says Paterson.

Paterson has been a pawn, a very valuable pawn, in a gigantic game of corporate chess. He has been moved around the board and asked to perform numerous tasks, some of which he'd like to forget. Like any good pawn, he's been relatively straightforward and dependable. But deep down, he's always wanted to be his own master, to call his own moves.

"My dad was an electrical engineer and when I went to school it was the first thing I tried," Paterson says.

In high school, Paterson took a semester of Fortran. He worked with a 7400, one of the TTL series of small logic chips. He didn't learn good textbook design from classes. "I learned it by reading and playing with it. I got a lot of exposure to electronics stuff at home."

It was as a student at Seattle's University of Washington that Paterson first encountered personal computers. In early 1976, his roommate bought an IMSAI 8080. "He provided the cash. I selected and maintained it," recalls Paterson. "It had a 4K memory board with the 8080 chip and no I/O devices. You could play a few stupid games like 'chase the bit,' which was entertaining for fifteen minutes at a time."

Days of Wire and Solder. Later that year, Paterson got a job as a technician at a Seattle-area retail computer store. There he was an eyewitness to those early days when the only way to own a microcomputer was to buy components and assemble it yourself. He's not kidding when he says, "Life begins with a disk drive."

With his university courses and practical experience in the retail store, Paterson learned quickly. He started toying around with designing his own peripheral boards.

"I got to know Rod Brock of Seattle Computer when he came into the store periodically. We were selling his boards. Eventually he asked me to consult for Seattle Computer.

"They were having problems with a 16K static memory board for the S-100," Paterson continues. "Rod Brock hired me in June 1978, at fifty dollars a day, to make the board work. I left the retail computer store at that time." After a few weeks of consulting, he was made a salaried employee of Seattle Computer.

In his computer science classes, Paterson had become interested in operating systems, as well as hardware and compilers. After receiving his bachelor of science degree in June '78, he attended graduate courses off and on for about another year. But he gradually lost interest. "I thought they were too oriented towards theory and not what I needed."

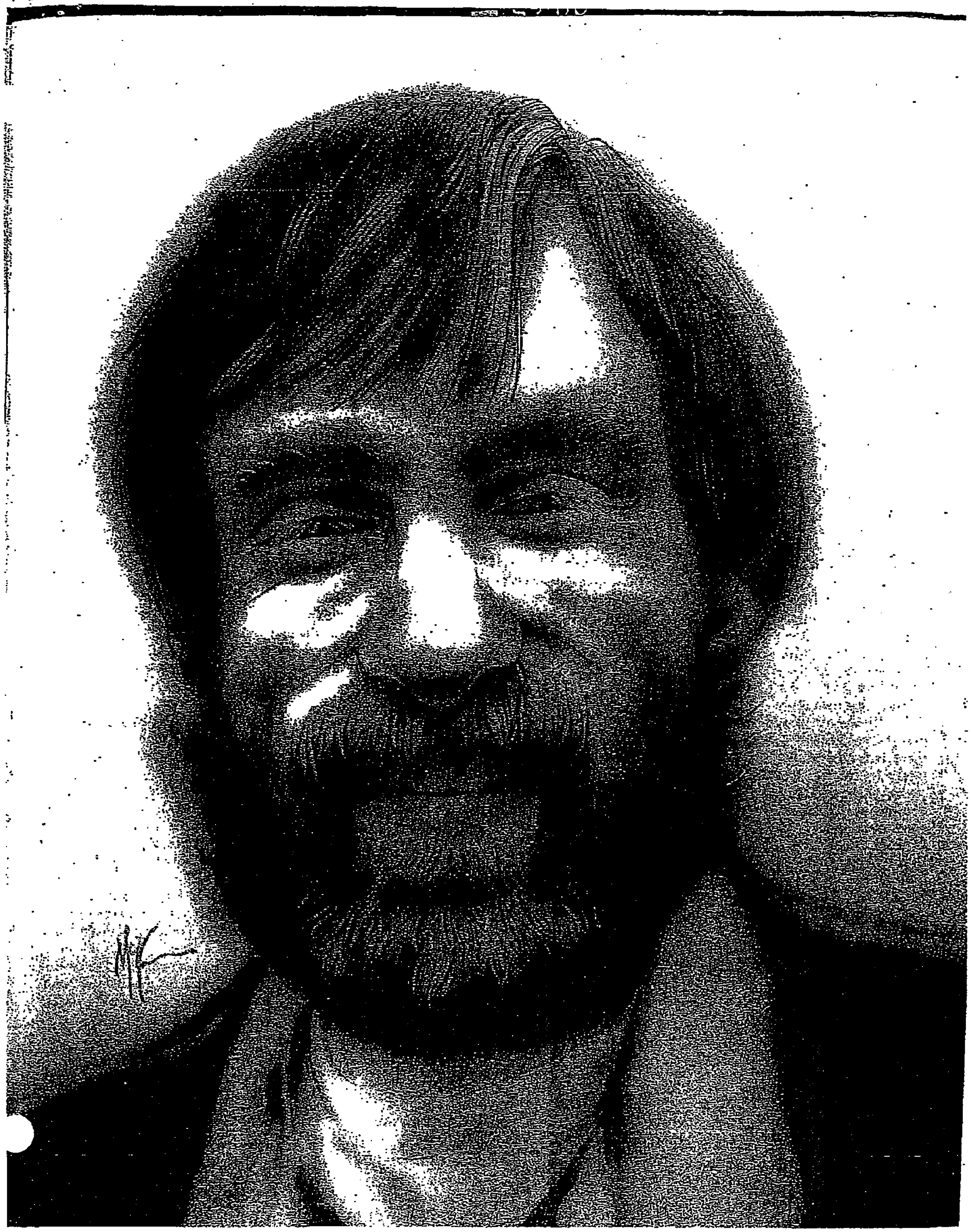
At Seattle Computer he at first worked on several projects—redesigning an S-100 memory board, which led to two new memory board designs. Things changed when he attended a local seminar on the Intel 8086 chip in late July 1978.

"I gained the respect of Rod Brock and made suggestions. I thought doing something with the 8086 would be neat and Brock gave it the go-ahead.

"The first design of the 8086 CPU card was finished by the end of January. We had a prototype by May 1979. We built three boards, but didn't wire them all up. There were both layout and design errors, but we got two prototypes working."

Rainy Day Computer. Seattle Computer was toying with the idea of eventually building its own computer, thus the CPU card project. They wanted to diversify, but had no firm business plan.

Once a prototype of the 8086 CPU was up and running, Seattle was approached by Digital Research to see if it could get CP/M to run on it. Microsoft, which had moved to the Seattle area in January 1979, wanted to see if some of its programs would work, too. At the end of May 1979, Paterson went to Microsoft to work with



Bob Orear there. In a week or so, Paterson cranked out all 32K of Microsoft's Basic onto the 8086.

"That's pretty remarkable," says Paterson. "Microsoft already had developed several good utilities, like a cross-assembler for use with the PDP-10. There were a few bugs, but basically the CPU worked and the development tools worked."

At the 1979 National Computer Conference in New York, Seattle Computer was the guest of Microsoft and Lifeboat. They showed off standalone Basic-86, then the only software for the 8086. Seattle Computer started shipping the product with its CPU card in November, primarily to software developers.

In April 1980, Paterson began work on an operating system.

"We needed an operating system at Seattle Computer for our own computers and I wanted to do one. So we decided to go for it.

"I was waiting for Digital to come out with CP/M-86. I thought they would have it real soon. If they had beat me I wouldn't have taken the trouble.

"I had always wanted to write my own operating system. I've always hated CP/M and thought I could do it a lot better."

Fast and Grimy. In the spring of 1980, Paterson started working on what would become MS-DOS. By July, he had finished roughly 50 percent of the operating system and called it QDOS 0.10 (for quick and dirty). He quickly found a bug and it became QDOS 0.11.

"Step one was to write down what CP/M-80 did. Step two was to design a file system that was fast and efficient."

One of the significant differences between MS-DOS and CP/M-86, when it finally appeared, was the file management system. CP/M usually provides a window to no more than 16K or 32K. With MS-DOS, the entire file is available. Paterson created QDOS's file management module using the same method found in standalone Basic-86.

"I'm into bottom-up programming. You know that you're going to need certain functions later in the program. I build tools on which to make the next layer.

"When you're programming top-down, it's stepwise refining—going from general actions to smaller actions. With my method there isn't a lot of diagramming. Bottom-up programming is definitely legitimate, it just doesn't get a lot of press."

By the end of August 1980, QDOS 0.11 worked well and was being shipped. It didn't stay QDOS very long, and not many copies were distributed. For the much improved update, Paterson worked hard to include all the necessary features of a complete operating system.

"There was an assembler, resident in the operating system, and debugging, but no editor. I wrote the quickest line editor I could imagine—quick in how fast I wrote it, two weeks.

"I was aghast," says Paterson, "when I heard that IBM was using it and not throwing it out the window."

Eighty-six on Cue. In December 1980, Paterson and company came out with 86-DOS, 0.33, which had significant improvements over QDOS. "86-DOS reflected pretty much everything we had learned so far. Just about the only thing it didn't have was variable sector record sizes. The assembler, originally written on the Z-80, was made faster. We also made some changes in the system calls. It was a pretty polished package when it was released."

Starting at the end of 1980, Seattle Computer sold 86-DOS to OEMs (original equipment manufacturers) and other companies like Microsoft.

"They [Microsoft] paid us a flat fee. It was not a per-copy royalty, but per OEM. Part of the contract said we couldn't ask them who they were selling it to or planning to sell it to."

In early 1981 the IBM Personal Computer had not yet been announced, but rumors were flying about Big Blue's micro. "We all had our suspicions that it was IBM that Microsoft was dealing with, but we didn't know for sure."

Around April 1981, while he was doing some internal changes on 86-DOS—modifying system calls and including error handling for hard disks—Paterson decided to quit Seattle Computer. In May, he went to Microsoft to work full-time on the PC-DOS version of 86-DOS.

"The first day on the job I walk through the door and 'Hey! It's IBM,' " says Paterson, grinning impishly. "I worked at Microsoft a neat eleven months. In May, June, and July I worked on things I hadn't quite finished, refining PC-DOS."

International Business Machinations. This was the beginning of an eleven-month hurricane. Almost daily, Paterson shipped stuff to Boca Raton for IBM's approval, and IBM would instantly return comments, modifications, and more problems.

"They were real thorough. I would send them a disk the same day via Delta Dash. IBM would be on the phone to me as soon as the disk arrived." Paterson pauses and winds up. He's remembering one request that clashed violently with his view of the project.

"IBM wanted CP/M prompts. It made me throw up." But when IBM asks, you comply if you're a lowly programmer, and that is what Paterson did.

He finished PC-DOS in July, one month before the pc was officially announced to the world. By this time, 86-DOS had become MS-DOS.

"Microsoft wanted to own it—pay all monies now and take it off Seattle Computer's hands. Both companies realized a good deal when they saw it. Seattle Computer really didn't have the marketing clout of Microsoft.

"So on July 27, 1981, the operating system became Microsoft's property. According to the deal, Seattle Computer can still see the source code, but is otherwise just another licensee. I think both companies were real happy. The deal was closed just a few weeks before the pc was announced. Microsoft was quite confident." Paterson pauses.

"Selling it was the right move. Seattle Computer is doing good without it. The timing was appropriate. I was invited to sit in on the meeting between Rod Brock and Paul Allen. I was flattered."

Is There Life after DOS? After the initial version of PC-DOS, Paterson went on to other programming tasks at Microsoft. He worked on an 8086 version of Microsoft's Basic compiler.

Paterson is like many programmers in the industry. Sure, he likes elegance. Sure, he likes simplicity. Sure, he likes to make things easy for the user. But what he likes more than anything else in a program or system is speed.

"I love assembly language. I'm a speed freak, especially with mathematical routines. If there's a loop that you want to repeat five times, I'll just rewrite the line that many times. Bam, bam, bam, woosh! The 8086 does normal loops real slow."

Paterson, still with Microsoft, did some consulting for Seattle Computer that fall, working on a RAM card for the IBM pc. Soon after he finished, he heard that Microsoft was working on a similar project.

"It was a real thrill to design a card for my employer's competitors. Microsoft was not too upset. They didn't chop my fingers off. By March 1982, Seattle's board had become quite popular. It came out months before anyone else came out with a multifunction board."

Late in 1981, Paterson and Microsoft got the word that IBM was looking for a 1.1 update. In November, he was made technical product manager of MS-DOS. He and his group delivered the initial version of 1.1 at the end of the year, a few days early. Then the Boca Raton deluge came.

"The whole process drove me crazy. A lot of bugs—PTRs [program trouble reports]—kept dribbling in and IBM would make a phone call a day. It really drove me nuts. I felt like a puppet on an IBM string."

In March 1982, after two months of going back and forth with Paterson, IBM accepted 1.1. Paterson spent the last weeks of that month planning 2.0. Then, on April 1, he suddenly quit Microsoft. (Mark Zbikowski became the group manager and has brought MS-DOS to the brink of releasing the 2.0 version, which Paterson had little to do with.)

Not a man to burn his bridges, Paterson left Microsoft on good terms and returned to Seattle Computer to work, at first, on Seattle's floppy disk controller.

Wrong-Way Paterson. Seattle Computer was doing quite well. Paterson had owned 10 percent of the company since 1979, and had been an officer and member of the board. Achieving such a position at Microsoft was unlikely.

"It was not what was wrong with Microsoft, but what Seattle Computer had to offer. At Seattle, I'm director of engineering. Hey! That's really motivating. It was the basis for my moving back. I was a little irritated with Microsoft, mainly having to work with IBM. Microsoft recognizes talent. If somebody complains, they try to move them around."

Paterson moved himself, though, out the front door.

At present, he and Seattle Computer are "catching up." They have new S-100 products in the works and will have "new IBM stuff" soon.

"We're working on our expansion cards, making them with four functions instead of just two. We want to catch up with those four-function guys. We're also working on a new enclosure for the Gazelle.

"I have the urge to do another operating system—to see if, under the circumstances, I couldn't do it better. I don't think I will, though, not at Seattle Computer. I don't have the time.

"Currently, my job does not include designing software, though I consider myself a software designer. I can picture all kinds of neat things—combined packages. Memory's cheap these days and it should be possible to have a spreadsheet, word processor, database, and more at fingertip control in one package.

"Still, the 8086/8088 looks like it for a while. It'll go through a software jump; it hasn't caught up with the Z-80 yet."

Speed Racer. When far from the world of programming and corporate politics, Paterson is something of a race-car nut. He codrives a Mazda RX-2 in pro rallies.

"The car has a roll cage and we wear helmets and all that. I have an RX-7 and, yeah, I'm kinda into cars."

Paterson is still looking for that elusive something. Independently minded, he seeks complete freedom. He doesn't want to work for someone else all his life. More properly put, he doesn't want always to be doing someone else's work.

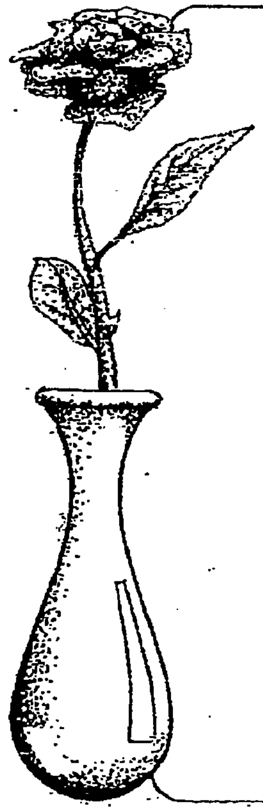
Some year Paterson would like to start his own company. When his Seattle Computer stock is worth enough, he just may sell it and go off on his own.

"Don't worry, the boss knows. Rod Brock said to me, 'Tim, in a few years you'll go.' There is the potential that I'll go all the way with Seattle, but I just don't know. Small companies that make it either become big or become part of a big company."

For the moment, Paterson is just another brilliant programmer. He's happy, but a little sad sometimes.

"Who cares who wrote the operating system?"

We'll remember, Tim. ▲



The RoseWare™ Directory

- ☐ More than 2,000 PC-compatible products and services
- ☐ Compare similar products from more than 900 companies
- ☐ Print product specifications and/or mailing labels
- ☐ Menu-driven, easy to use
- ☐ Requires 64K, one disk drive
- ☐ Diskette $30.00 (Add $1.95 in Calif.)

The RoseWare Directory · 1015 Gayley Ave., Ste. 323
Los Angeles, California 90024 · (213) 457-7110