The MS-DOS® Encyclopedia

Microsoft Press
Redmond, Washington
1988

Ray Duncan, General Editor

Foreword by Bill Gates

Foreword

Microsoft's MS-DOS is the most popular piece of software in the world. It runs on more than 10 million personal computers worldwide and is the foundation for at least 20,000 applications — the largest set of applications in any computer environment. As an industry standard for the family of 8086-based microcomputers, MS-DOS has had a central role in the personal computer revolution and is the most significant and enduring factor in furthering Microsoft's original vision — a computer for every desktop and in every home. The challenge of maintaining a single operating system over the entire range of 8086-based microcomputers and applications is incredible, but Microsoft has been committed to meeting this challenge since the release of MS-DOS in 1981. The true measure of our success in this effort is MS-DOS's continued prominence in the microcomputer industry.

Since MS-DOS's creation, more powerful and much-improved computers have entered the marketplace, yet each new version of MS-DOS reestablishes its position as the foundation for new applications as well as for old. To explain this extraordinary prominence, we must look to the origins of the personal computer industry. The three most significant factors in the creation of MS-DOS were the compatibility revolution, the development of Microsoft BASIC and its widespread acceptance by the personal computer industry, and IBM's decision to build a computer that incorporated 16-bit technology.

The compatibility revolution began with the Intel 8080 microprocessor. This technological breakthrough brought unprecedented opportunities in the emerging microcomputer industry, promising continued improvements in power, speed, and cost of desktop computing. In the minicomputer market, every hardware manufacturer had its own special instruction set and operating system, so software developed for a specific machine was incompatible with the machines of other hardware vendors. This specialization also meant tremendous duplication of effort — each hardware vendor had to write language compilers, databases, and other development tools to fit its particular machine. Microcomputers based on the 8080 microprocessor promised to change all this because different manufacturers would buy the same chip with the same instruction set.

From 1975 to 1981 (the 8-bit era of microcomputing), Microsoft convinced virtually every personal computer manufacturer — Radio Shack, Commodore, Apple, and dozens of others — to build Microsoft BASIC into its machines. For the first time, one common language cut across all hardware vendor lines. The success of our BASIC demonstrated the advantages of compatibility: To their great benefit, users were finally able to move applications from one vendor's machine to another.

Most machines produced during this early period did not have a built-in disk drive. Gradually, however, floppy disks, and later fixed disks, became less expensive and more common, and a number of disk-based programs, including WordStar and dBASE, entered the market. A standard disk operating system that could accommodate these developments became extremely important, leading Lifeboat, Microsoft, and Digital Research all to support CP/M-80, Digital Research's 8080 DOS.

The 8-bit era proved the importance of having a multiple-manufacturer standard that permitted the free interchange of programs. It was important that software designed for the new 16-bit machines have this same advantage. No personal computer manufacturer in 1980 could have predicted with any accuracy how quickly a third-party software industry would grow and get behind a strong standard—a standard that would be the software industry's lifeblood. The intricacies of how MS-DOS became the most common 16-bit operating system, in part through the work we did for IBM, is not the key point here. The key point is that it was inevitable for a popular operating system to emerge for the 16-bit machine, just as Microsoft's BASIC had prevailed on the 8-bit systems.

It was overwhelmingly evident that the personal computer had reached broad acceptance in the market when *Time* in 1982 named the personal computer "Man of the Year." MS-DOS was integral to this acceptance and popularity, and we have continued to adapt MS-DOS to support more powerful computers without sacrificing the compatibility that is essential to keeping it an industry standard. The presence of the 80386 microprocessor guarantees that continued investments in Intel-architecture software will be worthwhile.

Our goal with *The MS-DOS Encyclopedia* is to provide the most thorough and accessible resource available anywhere for MS-DOS programmers. The length of this book is many times greater than the source listing of the first version of MS-DOS—evidence of the growing complexity and sophistication of the operating system. The encyclopedia will be especially useful to software developers faced with preserving continuity yet enhancing the portability of their applications.

Our thriving industry is committed to exploiting the advantages offered by the protected mode introduced with the 80286 microprocessor and the virtual mode introduced with the 80386 microprocessor. MS-DOS will continue to play an integral part in this effort. Faster and more powerful machines running Microsoft OS/2 mean an exciting future of multitasking systems, networking, improved levels of data protection, better hardware memory management for multiple applications, stunning graphics systems that can display an innovative graphical user interface, and communication subsystems. MS-DOS version 3, which runs in real mode on 80286-based and 80386-based machines, is a vital link in the Family API of OS/2. Users will continue to benefit from our commitment to improved operating-system performance and usability as the future unfolds.

Bill Gates

Preface

In the space of six years, MS-DOS has become the most widely used computer operating system in the world, running on more than 10 million machines. It has grown, matured, and stabilized into a flexible, easily extendable system that can support networking, graphical user interfaces, nearly any peripheral device, and even CD ROMs containing massive amounts of on-line information. MS-DOS will be with us for many years to come as the platform for applications that run on low-cost, 8086/8088-based machines.

Not surprisingly, the success of MS-DOS has drawn many writers and publishers into its orbit. The number of books on MS-DOS and its commands, languages, and applications dwarfs the list of titles for any other operating system. Why, then, yet another book on MS-DOS? And what can we say about the operating system that has not been said already?

First, we have written and edited *The MS-DOS Encyclopedia* with one audience in mind: the community of working programmers. We have therefore been free to bypass elementary subjects such as the number of bits in a byte and the interpretation of hexadecimal numbers. Instead, we have emphasized detailed technical explanations, working code examples that can be adapted and incorporated into new applications, and a systems view of even the most common MS-DOS commands and utilities.

Second, because we were not subject to size restrictions, we have explored topics in depth that other MS-DOS books mention only briefly, such as exception and error handling, interrupt-driven communications, debugging strategies, memory management, and installable device drivers. We have commissioned definitive articles on the relocatable object modules generated by Microsoft language translators, the operation of the Microsoft Object Linker, and terminate-and-stay-resident utilities. We have even interviewed the key developers of MS-DOS and drawn on their files and bulletin boards to offer an entertaining, illustrated account of the origins of Microsoft's standard-setting operating system.

Finally, by combining the viewpoints and experience of non-Microsoft programmers and writers, the expertise and resources of Microsoft software developers, and the publishing know-how of Microsoft Press, we have assembled a unique and comprehensive reference to MS-DOS services, commands, directives, and utilities. In many instances, the manuscripts have been reviewed by the authors of the Microsoft tools described.

We have made every effort during the creation of this book to ensure that its contents are timely and trustworthy. In a work of this size, however, it is inevitable that errors and omissions will occur. If you discover any such errors, please bring them to our attention so that they can be repaired in future printings and thus aid your fellow programmers. To this end, Microsoft Press has established a bulletin board on MCI Mail for posting corrections and comments. Please refer to page *xvi* for more information.

Ray Duncan

The Development of MS-DOS

To many people who use personal computers, MS-DOS is the key that unlocks the power of the machine. It is their most visible connection to the hardware hidden inside the cabinet, and it is through MS-DOS that they can run applications and manage disks and disk files.

In the sense that it opens the door to doing work with a personal computer, MS-DOS is indeed a key, and the lock it fits is the Intel 8086 family of microprocessors. MS-DOS and the chips it works with are, in fact, closely connected — so closely that the story of MS-DOS is really part of a larger history that encompasses not only an operating system but also a microprocessor and, in retrospect, part of the explosive growth of personal computing itself.

Chronologically, the history of MS-DOS can be divided into three parts. First came the formation of Microsoft and the events preceding Microsoft's decision to develop an operating system. Then came the creation of the first version of MS-DOS. Finally, there is the continuing evolution of MS-DOS since its release in 1981.

Much of the story is based on technical developments, but dates and facts alone do not provide an adequate look at the past. Many people have been involved in creating MS-DOS and directing the lines along which it continues to grow. To the extent that personal opinions and memories are appropriate, they are included here to provide a fuller picture of the origin and development of MS-DOS.

Before MS-DOS

The role of International Business Machines Corporation in Microsoft's decision to create MS-DOS has been well publicized. But events, like inventions, always build on prior accomplishments, and in this respect the roots of MS-DOS reach farther back, to four hardware and software developments of the 1970s: Microsoft's disk-based and stand-alone versions of BASIC, Digital Research's CP/M-80 operating system, the emergence of the 8086 chip, and a disk operating system for the 8086 developed by Tim Paterson at a hardware company called Seattle Computer Products.

Microsoft and BASIC

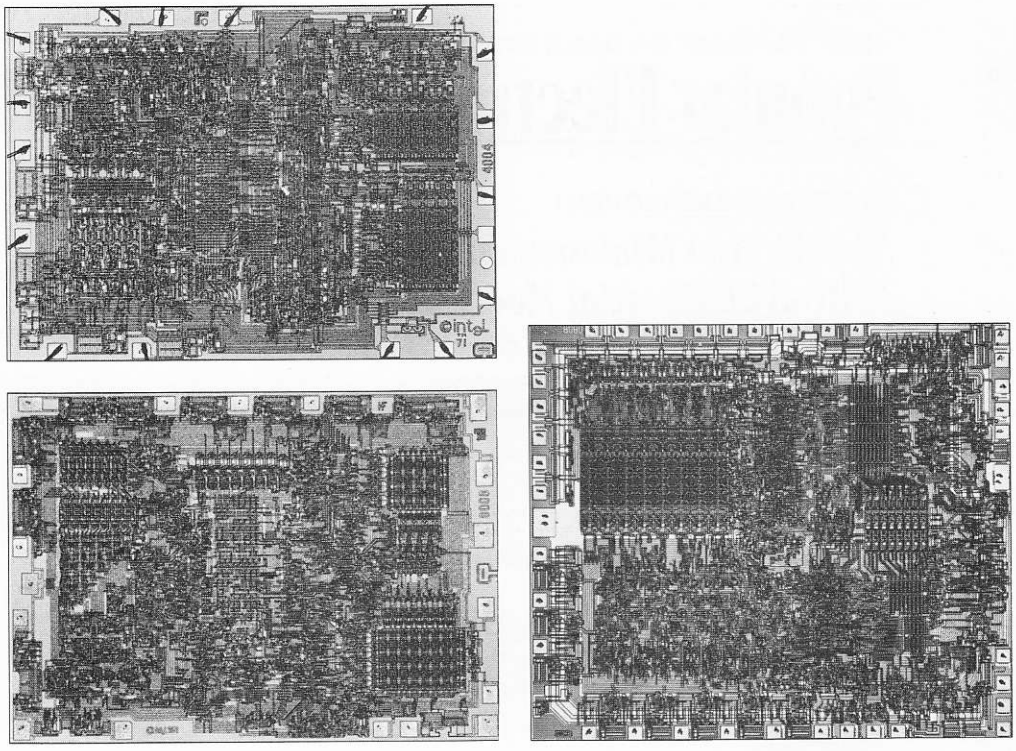
On the surface, BASIC and MS-DOS might seem to have little in common, but in terms of file management, MS-DOS is a direct descendant of a Microsoft version of BASIC called Stand-alone Disk BASIC.

Before Microsoft even became a company, its founders, Paul Allen and Bill Gates, developed a version of BASIC for a revolutionary small computer named the Altair, which was introduced in January 1975 by Micro Instrumentation Telemetry Systems (MITS) of

The Altair. Christened one evening shortly before its appearance on the cover of Popular Electronics *magazine, the computer was named for the night's destination of the starship* Enterprise. *The photograph clearly shows the input switches on the front panel of the cabinet.*

Albuquerque, New Mexico. Though it has long been eclipsed by other, more powerful makes and models, the Altair was the first "personal" computer to appear in an environment dominated by minicomputers and mainframes. It was, simply, a metal box with a panel of switches and lights for input and output, a power supply, a motherboard with 18 slots, and two boards. One board was the central processing unit, with the 8-bit Intel 8080 microprocessor at its heart; the other board provided 256 bytes of random-access memory. This miniature computer had no keyboard, no monitor, and no device for permanent storage, but it did possess one great advantage: a price tag of $397.

Now, given the hindsight of a little more than a decade of microcomputing history, it is easy to see that the Altair's combination of small size and affordability was the thin edge of a wedge that, in just a few years, would move everyday computing power away from impersonal monoliths in climate-controlled rooms and onto the desks of millions of people. In 1975, however, the computing environment was still primarily a matter of data processing for specialists rather than personal computing for everyone. Thus when 4 KB



Intel's 4004, 8008, and 8080 chips. At the top left is the 4-bit 4004, which was named for the approximate number of old-fashioned transistors it replaced. At the bottom left is the 8-bit 8008, which addressed 16 KB of memory; this was the chip used in the Traf-O-Data tape-reader built by Paul Gilbert. At the right is the 8080, a faster 8-bit chip that could address 64 KB of memory. The brain of the MITS Altair, the 8080 was, in many respects, the chip on which the personal computing industry was built. The 4004 and 8008 chips were developed early in the 1970s; the 8080 appeared in 1974.

memory expansion boards became available for the Altair, the software needed most by its users was not a word processor or a spreadsheet, but a programming language — and the language first developed for it was a version of BASIC written by Bill Gates and Paul Allen.

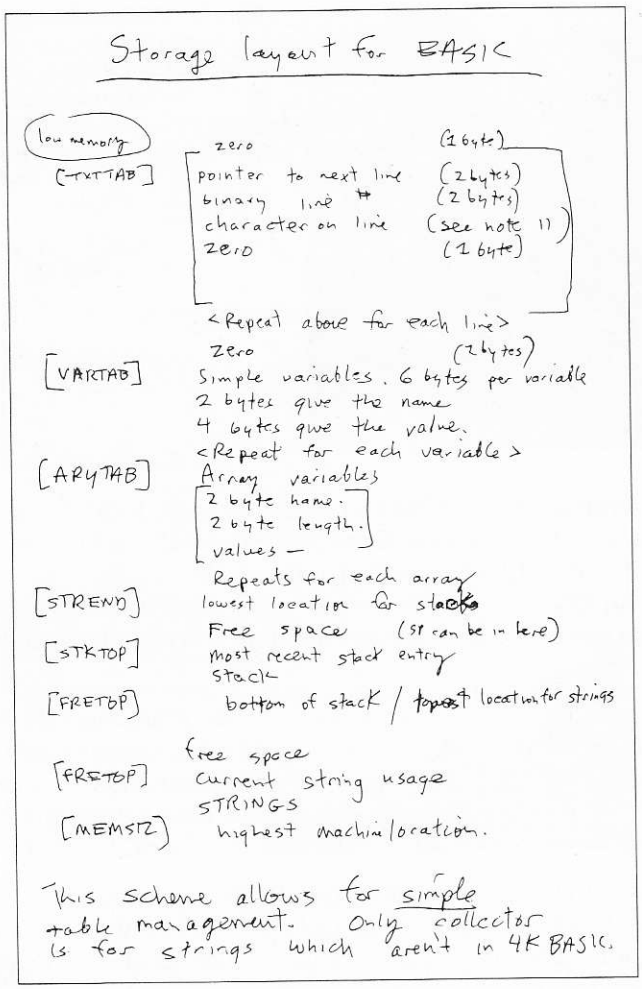
Gates and Allen had become friends in their teens, while attending Lakeside School in Seattle. They shared an intense interest in computers, and by the time Gates was in the tenth grade, they and another friend named Paul Gilbert had formed a company called Traf-O-Data to produce a machine that automated the reading of 16-channel, 4-digit, binary-coded decimal (BCD) tapes generated by traffic-monitoring recorders. This machine, built by Gilbert, was based on the Intel 8008 microprocessor, the predecessor of the 8080 in the Altair.

The January 1975 cover of Popular Electronics *magazine, featuring the machine that caught the imaginations of thousands of like-minded electronics enthusiasts — among them, Paul Allen and Bill Gates.*

Although it was too limited to serve as the central processor for a general-purpose computer, the 8008 was undeniably the ancestor of the 8080 as far as its architecture and instruction set were concerned. Thus Traf-O-Data's work with the 8008 gave Gates and Allen a head start when they later developed their version of BASIC for the Altair.

Paul Allen learned of the Altair from the cover story in the January 1975 issue of *Popular Electronics* magazine. Allen, then an employee of Honeywell in Boston, convinced Gates, a student at Harvard University, to develop a BASIC for the new computer. The two wrote their version of BASIC for the 8080 in six weeks, and Allen flew to New Mexico to demonstrate the language for MITS. The developers gave themselves the company name of Microsoft and licensed their BASIC to MITS as Microsoft's first product.

Though not a direct forerunner of MS-DOS, Altair BASIC, like the machine for which it was developed, was a landmark product in the history of personal computing. On another level, Altair BASIC was also the first link in a chain that led, somewhat circuitously, to Tim Paterson and the disk operating system he developed for Seattle Computer Products for the 8086 chip.



Storage layout for BASIC

low memory

[TXTTAB] zero (1 byte)
pointer to next line (2 bytes)
binary line # (2 bytes)
character on line (see note 1)
zero (1 byte)

<Repeat above for each line>
zero (2 bytes)

[VARTAB] Simple variables. 6 bytes per variable
2 bytes give the name
4 bytes give the value.
<Repeat for each variable>

[ARYTAB] Array variables
2 byte name.
2 byte length.
values —
Repeats for each array

[STREND] lowest location for stack

Free space (ST can be in here)

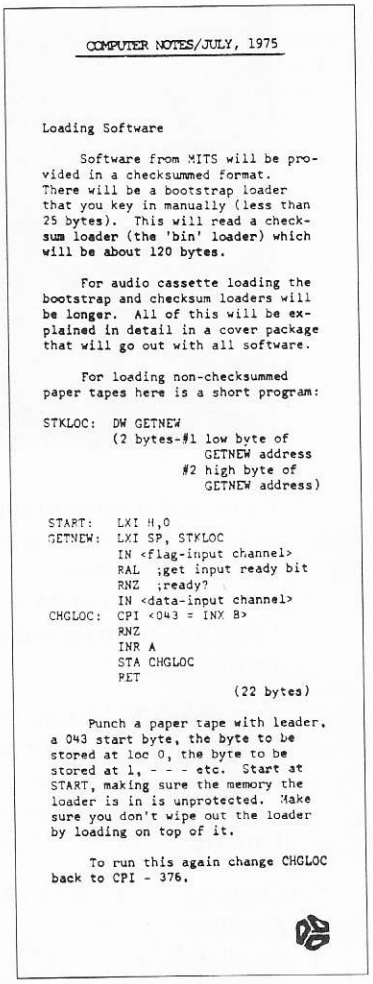
[STKTOP] most recent stack entry
stack

[FRETOP] bottom of stack / lowest location for strings

free space
current string usage
STRINGS

[MEMSIZ] highest machine location.

This scheme allows for simple table management. Only collector is for strings which aren't in 4K BASIC.



COMPUTER NOTES/JULY, 1975

Loading Software

Software from MITS will be provided in a checksummed format. There will be a bootstrap loader that you key in manually (less than 25 bytes). This will read a checksum loader (the 'bin' loader) which will be about 120 bytes.

For audio cassette loading the bootstrap and checksum loaders will be longer. All of this will be explained in detail in a cover package that will go out with all software.

For loading non-checksummed paper tapes here is a short program:

```
STKLOC:  DW GETNEW
         (2 bytes-#1 low byte of
                     GETNEW address
                  #2 high byte of
                     GETNEW address)

START:   LXI H,0
GETNEW:  LXI SP, STKLOC
         IN <flag-input channel>
         RAL  ;get input ready bit
         RNZ  ;ready?
         IN <data-input channel>
CHGLOC:  CPI <043 = INX B>
         RNZ
         INR A
         STA CHGLOC
         RET
                          (22 bytes)
```

Punch a paper tape with leader, a 043 start byte, the byte to be stored at loc 0, the byte to be stored at 1, - - - etc. Start at START, making sure the memory the loader is in is unprotected. Make sure you don't wipe out the loader by loading on top of it.

To run this again change CHGLOC back to CPI - 376.

On the left, Bill Gates's original handwritten notes describing memory configuration for Altair BASIC. On the right, a short bootstrap program written by Gates for Altair users; published in the July 1975 edition of the MITS user newsletter, Computer Notes.

From paper tape to disk

Gates and Allen's early BASIC for the Altair was loaded from paper tape after the bootstrap to load the tape was entered into memory by flipping switches on the front panel of the computer. In late 1975, however, MITS decided to release a floppy-disk system for the Altair—the first retail floppy-disk system on the market. As a result, in February 1976 Allen, by then Director of Software for MITS, asked Gates to write a disk-based version of Altair BASIC. The Altair had no operating system and hence no method of managing files, so the disk BASIC would have to include some file-management routines. It would, in effect, have to function as a rudimentary operating system.

Microsoft, 1978, Albuquerque, New Mexico. Top row, left to right: Steve Wood, Bob Wallace, Jim Lane. Middle row, left to right: Bob O'Rear, Bob Greenberg, Marc McDonald, Gordon Letwin. Bottom row, left to right: Bill Gates, Andrea Lewis, Marla Wood, Paul Allen.

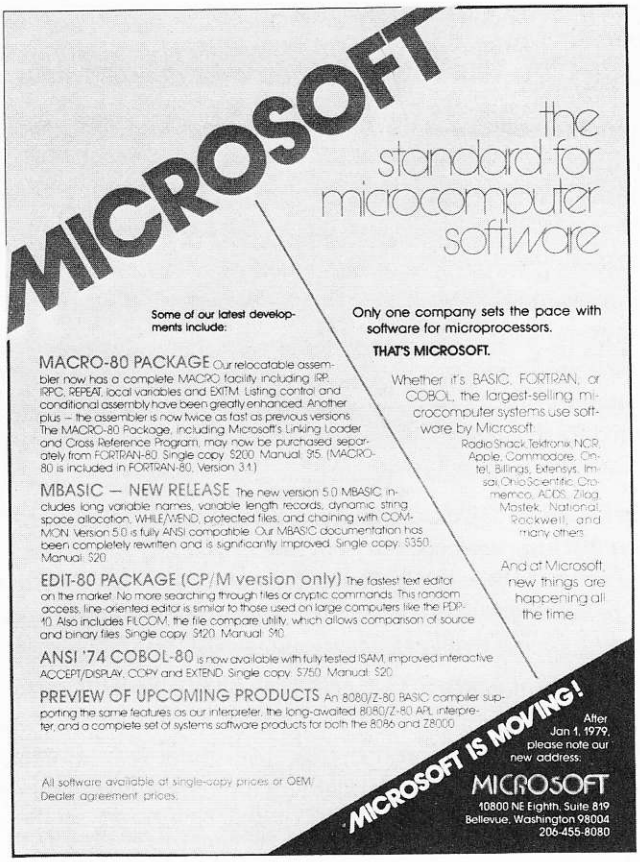
Gates, still at Harvard University, agreed to write this version of BASIC for MITS. He went to Albuquerque and, as has often been recounted, checked into the Hilton Hotel with a stack of yellow legal pads. Five days later he emerged, yellow pads filled with the code for the new version of BASIC. Arriving at MITS with the code and a request to be left alone, Gates began typing and debugging and, after another five days, had Disk BASIC running on the Altair.

This disk-based BASIC marked Microsoft's entry into the business of languages for personal computers — not only for the MITS Altair, but also for such companies as Data Terminals Corporation and General Electric. Along the way, Microsoft BASIC took on added features, such as enhanced mathematics capabilities, and, more to the point in terms of MS-DOS, evolved into Stand-alone Disk BASIC, produced for NCR in 1977.

Designed and coded by Marc McDonald, Stand-alone Disk BASIC, included a file-management scheme called the FAT, or file allocation table that used a linked list for managing disk files. The FAT, born during one of a series of discussions between McDonald and Bill Gates, enabled disk-allocation information to be kept in one location, with "chained" references pointing to the actual storage locations on disk. Fast and flexible, this file-management strategy was later used in a stand-alone version of BASIC for the 8086 chip and eventually, through an operating system named M-DOS, became the basis for the file-handling routines in MS-DOS.

M-DOS

During 1977 and 1978, Microsoft adapted both BASIC and Microsoft FORTRAN for an increasingly popular 8-bit operating system called CP/M. At the end of 1978, Gates and Allen moved Microsoft from Albuquerque to Bellevue, Washington. The company continued to concentrate on programming languages, producing versions of BASIC for the 6502 and the TI9900.



A Microsoft advertisement from the January 1979 issue of Byte *magazine mentioning some products and the machines they ran on. In the lower right corner is an announcement of the company's move to Bellevue, Washington.*

During this same period, Marc McDonald also worked on developing an 8-bit operating system called M-DOS (usually pronounced "Midas" or "My DOS"). Although it never became a real part of the Microsoft product line, M-DOS was a true multitasking operating system modeled after the DEC TOPS-10 operating system. M-DOS provided good performance and, with a more flexible FAT than that built into BASIC, had a better file-handling structure than the up-and-coming CP/M operating system. At about 30 KB, however, M-DOS was unfortunately too big for an 8-bit environment and so ended up being relegated to the back room. As Allen describes it, "Trying to do a large, full-blown operating system on the 8080 was a lot of work, and it took a lot of memory. The 8080 addresses only 64 K, so with the success of CP/M, we finally concluded that it was best not to press on with that."

CP/M

In the volatile microcomputer era of 1976 through 1978, both users and developers of personal computers quickly came to recognize the limitations of running applications on top of Microsoft's Stand-alone Disk BASIC or any other language. MITS, for example, scheduled

a July 1976 release date for an independent operating system for its machine that used the code from the Altair's Disk BASIC. In the same year, Digital Research, headed by Gary Kildall, released its Control Program/Monitor, or CP/M.

CP/M was a typical microcomputer software product of the 1970s in that it was written by one person, not a group, in response to a specific need that had not yet been filled. One of the most interesting aspects of CP/M's history is that the software was developed several years before its release date — actually, several years before the hardware on which it would be a standard became commercially available.

In 1973, Kildall, a professor of computer science at the Naval Postgraduate School in Monterey, California, was working with an 8080-based small computer given him by Intel Corporation in return for some programming he had done for the company. Kildall's machine, equipped with a monitor and paper-tape reader, was certainly advanced for the time, but Kildall became convinced that magnetic-disk storage would make the machine even more efficient than it was.

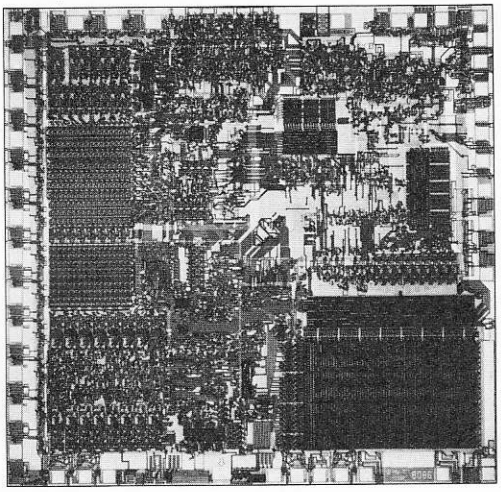
Trading some programming for a disk drive from Shugart, Kildall first attempted to build a drive controller on his own. Lacking the necessary engineering ability, he contacted a friend, John Torode, who agreed to handle the hardware aspects of interfacing the computer and the disk drive while Kildall worked on the software portion — the refinement of an operating system he had written earlier that year. The result was CP/M.

The version of CP/M developed by Kildall in 1973 underwent several refinements. Kildall enhanced the CP/M debugger and assembler, added a BASIC interpreter, and did some work on an editor, eventually developing the product that, from about 1977 until the appearance of the IBM Personal Computer, set the standard for 8-bit microcomputer operating systems.

Digital Research's CP/M included a command interpreter called CCP (Console Command Processor), which acted as the interface between the user and the operating system itself, and an operations handler called BDOS (Basic Disk Operating System), which was responsible for file storage, directory maintenance, and other such housekeeping chores. For actual input and output — disk I/O, screen display, print requests, and so on — CP/M included a BIOS (Basic Input/Output System) tailored to the requirements of the hardware on which the operating system ran.

For file storage, CP/M used a system of eight-sector allocation units. For any given file, the allocation units were listed in a directory entry that included the filename and a table giving the disk locations of 16 allocation units. If a long file required more than 16 allocation units, CP/M created additional directory entries as required. Small files could be accessed rapidly under this system, but large files with more than a single directory entry could require numerous relatively time-consuming disk reads to find needed information.

At the time, however, CP/M was highly regarded and gained the support of a broad base of hardware and software developers alike. Quite powerful for its size (about 4KB), it was, in all respects, the undisputed standard in the 8-bit world, and remained so until, and even after, the appearance of the 8086.



The 16-bit Intel 8086 chip, introduced in 1978. Much faster and far more powerful than its 8-bit predecessor the 8080, the 8086 had the ability to address one megabyte of memory.

The 8086

When Intel released the 8-bit 8080 chip in 1974, the Altair was still a year in the future. The 8080 was designed not to make computing a part of everyday life but to make household appliances and industrial machines more intelligent. By 1978, when Intel introduced the 16-bit 8086, the microcomputer was a reality and the new chip represented a major step ahead in performance and memory capacity. The 8086's full 16-bit buses made it faster than the 8080, and its ability to address one megabyte of random-access memory was a giant step beyond the 8080's 64 KB limit. Although the 8086 was not compatible with the 8080, it was architecturally similar to its predecessor and 8080 source code could be mechanically translated to run on it. This translation capability, in fact, was a major influence on the design of Tim Paterson's operating system for the 8086 and, through Paterson's work, on the first released version of MS-DOS.

When the 8086 arrived on the scene, Microsoft, like other developers, was confronted with two choices: continue working in the familiar 8-bit world or turn to the broader horizons offered by the new 16-bit technology. For a time, Microsoft did both. Acting on Paul Allen's suggestion, the company developed the SoftCard for the popular Apple II, which was based on the 8-bit 6502 microprocessor. The SoftCard included a Z80 microprocessor and a copy of CP/M-80 licensed from Digital Research. With the SoftCard, Apple II users could run any program or language designed to run on a CP/M machine.

It was 16-bit technology, however, that held the most interest for Gates and Allen, who believed that this would soon become the standard for microcomputers. Their optimism was not universal — more than one voice in the trade press warned that industry investment in 8-bit equipment and software was too great to successfully introduce a new standard. Microsoft, however, disregarded these forecasts and entered the 16-bit arena as it had with the Altair: by developing a stand-alone version of BASIC for the 8086.

At the same time and, coincidentally, a few miles south in Tukwila, Washington, a major contribution to MS-DOS was taking place. Tim Paterson, working at Seattle Computer Products, a company that built memory boards, was developing an 8086 CPU card for use in an S-100 bus machine.

86-DOS

Paterson was introduced to the 8086 chip at a seminar held by Intel in June 1978. He had attended the seminar at the suggestion of his employer, Rod Brock of Seattle Computer Products. The new chip sparked his interest because, as he recalls, "all its instructions worked on both 8 and 16 bits, and you didn't have to do everything through the accumulator. It was also real fast — it could do a 16-bit ADD in three clocks."

After the seminar, Paterson — again with Brock's support — began work with the 8086. He finished the design of his first 8086 CPU board in January 1979 and by late spring had developed a working CPU, as well as an assembler and an 8086 monitor. In June, Paterson took his system to Microsoft to try it with Stand-alone BASIC, and soon after, Microsoft BASIC was running on Seattle Computer's new board.

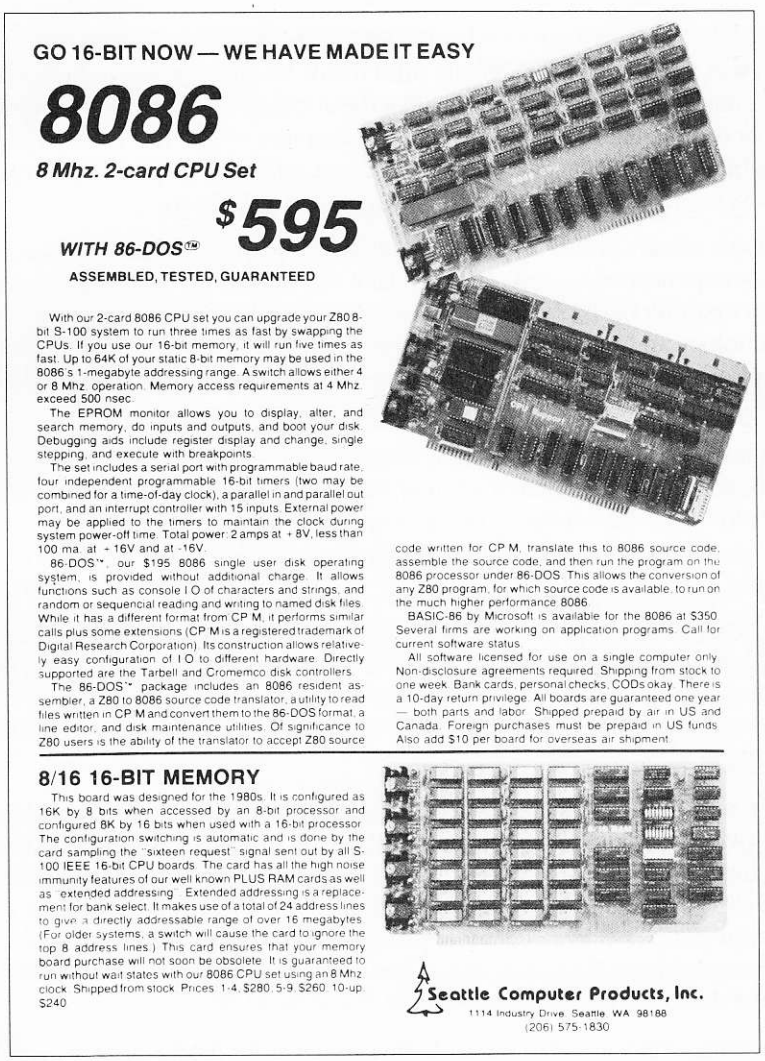
During this period, Paterson also received a call from Digital Research asking whether they could borrow the new board for developing CP/M-86. Though Seattle Computer did not have a board to loan, Paterson asked when CP/M-86 would be ready. Digital's representative said December 1979, which meant, according to Paterson's diary, "we'll have to live with Stand-alone BASIC for a few months after we start shipping the CPU, but then we'll be able to switch to a real operating system."

Early in June, Microsoft and Tim Paterson attended the National Computer Conference in New York. Microsoft had been invited to share Lifeboat Associates' ten-by-ten foot booth, and Paterson had been invited by Paul Allen to show BASIC running on an S-100 8086 system. At that meeting, Paterson was introduced to Microsoft's M-DOS, which he found interesting because it used a system for keeping track of disk files — the FAT developed for Stand-alone BASIC — that was different from anything he had encountered.

After this meeting, Paterson continued working on the 8086 board, and by the end of the year, Seattle Computer Products began shipping the CPU with a BASIC option.

When CP/M-86 had still not become available by April 1980, Seattle Computer Products decided to develop a 16-bit operating system of its own. Originally, three operating systems were planned: a single-user system, a multiuser version, and a small interim product soon informally christened QDOS (for Quick and Dirty Operating System) by Paterson.

Both Paterson (working on QDOS) and Rod Brock knew that a standard operating system for the 8086 was mandatory if users were to be assured of a wide range of application software and languages. CP/M had become the standard for 8-bit machines, so the ability to mechanically translate existing CP/M applications to run on a 16-bit system became one of Paterson's major goals for the new operating system. To achieve this compatibility, the system he developed mimicked CP/M-80's functions and command structure, including its use of file control blocks (FCBs) and its approach to executable files.



GO 16-BIT NOW — WE HAVE MADE IT EASY

8086

8 Mhz. 2-card CPU Set

WITH 86-DOS™ $595

ASSEMBLED, TESTED, GUARANTEED

With our 2-card 8086 CPU set you can upgrade your Z80 8-bit S-100 system to run three times as fast by swapping the CPUs. If you use our 16-bit memory, it will run five times as fast. Up to 64K of your static 8-bit memory may be used in the 8086's 1-megabyte addressing range. A switch allows either 4 or 8 Mhz. operation. Memory access requirements at 4 Mhz. exceed 500 nsec.

The EPROM monitor allows you to display, alter, and search memory, do inputs and outputs, and boot your disk. Debugging aids include register display and change, single stepping, and execute with breakpoints.

The set includes a serial port with programmable baud rate, four independent programmable 16-bit timers (two may be combined for a time-of-day clock), a parallel in and parallel out port, and an interrupt controller with 15 inputs. External power may be applied to the timers to maintain the clock during system power-off time. Total power: 2 amps at +8V, less than 100 ma. at +16V and at -16V.

86-DOS™, our $195 8086 single user disk operating system, is provided without additional charge. It allows functions such as console I/O of characters and strings, and random or sequencial reading and writing to named disk files. While it has a different format from CP/M, it performs similar calls plus some extensions (CP/M is a registered trademark of Digital Research Corporation). Its construction allows relatively easy configuration of I/O to different hardware. Directly supported are the Tarbell and Cromemco disk controllers.

The 86-DOS™ package includes an 8086 resident assembler, a Z80 to 8086 source code translator, a utility to read files written in CP/M and convert them to the 86-DOS format, a line editor, and disk maintenance utilities. Of significance to Z80 users is the ability of the translator to accept Z80 source code written for CP/M, translate this to 8086 source code, assemble the source code, and then run the program on the 8086 processor under 86-DOS. This allows the conversion of any Z80 program, for which source code is available, to run on the much higher performance 8086.

BASIC-86 by Microsoft is available for the 8086 at $350. Several firms are working on application programs. Call for current software status.

All software licensed for use on a single computer only. Non-disclosure agreements required. Shipping from stock to one week. Bank cards, personal checks, CODs okay. There is a 10-day return privilege. All boards are guaranteed one year — both parts and labor. Shipped prepaid by air in US and Canada. Foreign purchases must be prepaid in US funds. Also add $10 per board for overseas air shipment.

8/16 16-BIT MEMORY

This board was designed for the 1980s. It is configured as 16K by 8 bits when accessed by an 8-bit processor and configured 8K by 16 bits when used with a 16-bit processor. The configuration switching is automatic and is done by the card sampling the "sixteen request" signal sent out by all S-100 IEEE 16-bit CPU boards. The card has all the high noise immunity features of our well known PLUS RAM cards as well as "extended addressing". Extended addressing is a replacement for bank select. It makes use of a total of 24 address lines to give a directly addressable range of over 16 megabytes. (For older systems, a switch will cause the card to ignore the top 8 address lines.) This card ensures that your memory board purchase will not soon be obsolete. It is guaranteed to run without wait states with our 8086 CPU set using an 8 Mhz. clock. Shipped from stock. Prices: 1-4, $280; 5-9, $260; 10-up, $240.

Seattle Computer Products, Inc.

1114 Industry Drive, Seattle, WA 98188

(206) 575-1830

An advertisement for the Seattle Computer Products 8086 CPU, with 86-DOS; published in the December 1980 issue of Byte.

At the same time, however, Paterson was dissatisfied with certain elements of CP/M, one of them being its file-allocation system, which he considered inefficient in the use of disk space and too slow in operation. So for fast, efficient file handling, he used a file allocation table, as Microsoft had done with Stand-alone Disk BASIC and M-DOS. He also wrote a translator to translate 8080 code to 8086 code, and he then wrote an assembler in Z80 assembly language and used the translator to translate it.

Four months after beginning work, Paterson had a functioning 6 KB operating system, officially renamed 86-DOS, and in September 1980 he contacted Microsoft again, this time to ask the company to write a version of BASIC to run on his system.

IBM

While Paterson was developing 86-DOS, the third major element leading to the creation of MS-DOS was gaining force at the opposite end of the country. IBM, until then seemingly oblivious to most of the developments in the microcomputer world, had turned its attention to the possibility of developing a low-end workstation for a market it knew well: business and business people.

On August 21, 1980, a study group of IBM representatives from Boca Raton, Florida, visited Microsoft. This group, headed by a man named Jack Sams, told Microsoft of IBM's interest in developing a computer based on a microprocessor. IBM was, however, unsure of microcomputing technology and the microcomputing market. Traditionally, IBM relied on long development cycles — typically four or five years — and was aware that such lengthy design periods did not fit the rapidly evolving microcomputer environment.

One of IBM's solutions — the one outlined by Sams's group — was to base the new machine on products from other manufacturers. All the necessary hardware was available, but the same could not be said of the software. Hence the visit to Microsoft with the question: Given the specifications for an 8-bit computer, could Microsoft write a ROM BASIC for it by the following April?

Microsoft responded positively, but added questions of its own: Why introduce an 8-bit computer? Why not release a 16-bit machine based on Intel's 8086 chip instead? At the end of this meeting — the first of many — Sams and his group returned to Boca Raton with a proposal for the development of a low-end, 16-bit business workstation. The venture was named Project Chess.

One month later, Sams returned to Microsoft asking whether Gates and Allen could, still by April 1981, provide not only BASIC but also FORTRAN, Pascal, and COBOL for the new computer. This time the answer was no because, though Microsoft's BASIC had been designed to run as a stand-alone product, it was unique in that respect — the other languages would need an operating system. Gates suggested CP/M-86, which was then still under development at Digital Research, and in fact made the initial contact for IBM. Digital Research and IBM did not come to any agreement, however.

Microsoft, meanwhile, still wanted to write all the languages for IBM — approximately 400 KB of code. But to do this within the allotted six-month schedule, the company needed some assurances about the operating system IBM was going to use. Further, it needed specific information on the internals of the operating system, because the ROM BASIC would interact intimately with the BIOS.

The turning point

That state of indecision, then, was Microsoft's situation on Sunday, September 28, 1980, when Bill Gates, Paul Allen, and Kay Nishi, a Microsoft vice president and president of ASCII Corporation in Japan, sat in Gates's eighth-floor corner office in the Old National Bank Building in Bellevue, Washington. Gates recalls, "Kay and I were just sitting there at night and Paul was on the couch. Kay said, 'Got to do it, got to do it.' It was only 20 more K

of code at most — actually, it turned out to be 12 more K on top of the 400. It wasn't that big a deal, and once Kay said it, it was obvious. We'd always wanted to do a low-end operating system, we had specs for low-end operating systems, and we knew we were going to do one up on 16-bit."

At that point, Gates and Allen began looking again at Microsoft's proposal to IBM. Their estimated 400 KB of code included four languages, an assembler, and a linker. To add an operating system would require only another 20 KB or so, and they already knew of a working model for the 8086: Tim Paterson's 86-DOS. The more Gates, Allen, and Nishi talked that night about developing an operating system for IBM's new computer, the more possible — even preferable — the idea became.

Allen's first step was to contact Rod Brock at Seattle Computer Products to tell him that Microsoft wanted to develop and market SCP's operating system and that the company had an OEM customer for it. Seattle Computer Products, which was not in the business of marketing software, agreed and licensed 86-DOS to Microsoft. Eventually, SCP sold the operating system to Microsoft for $50,000, favorable language licenses, and a license back from Microsoft to use 86-DOS on its own machines.

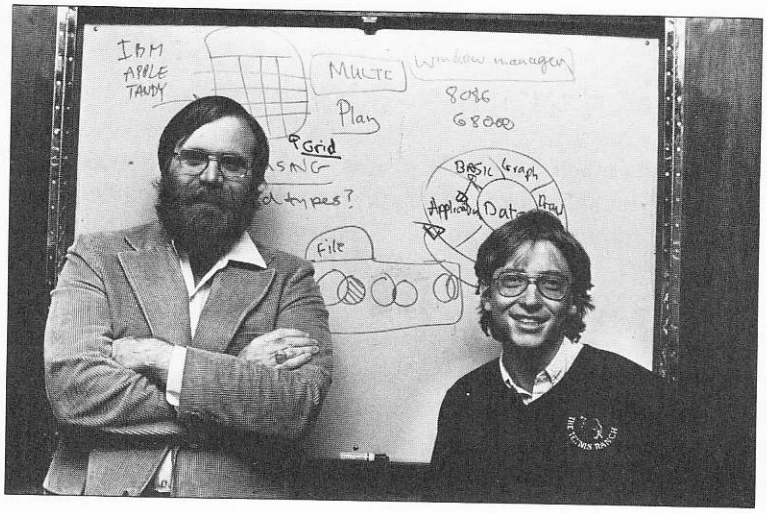
In October 1980, with 86-DOS in hand, Microsoft submitted another proposal to IBM. This time the plan included both an operating system and the languages for the new computer. Time was short and the boundaries between the languages and the operating system were unclear, so Microsoft explained that it needed to control the development of the operating system in order to guarantee delivery by spring of 1981. In November, IBM signed the contract.

Creating MS-DOS

At Thanksgiving, a prototype of the IBM machine arrived at Microsoft and Bill Gates, Paul Allen, and, primarily, Bob O'Rear began a schedule of long, sometimes hectic days and total immersion in the project. As O'Rear recalls, "If I was awake, I was thinking about the project."

The first task handled by the team was bringing up 86-DOS on the new machine. This was a challenge because the work had to be done in a constantly changing hardware environment while changes were also being made to the specifications of the budding operating system itself.

As part of the process, 86-DOS had to be compiled and integrated with the BIOS, which Microsoft was helping IBM to write, and this task was complicated by the media. Paterson's 86-DOS — not counting utilities such as EDLIN, CHKDSK, and INIT (later named FORMAT) — arrived at Microsoft as one large assembly-language program on an 8-inch floppy disk. The IBM machine, however, used 5¼-inch disks, so Microsoft needed to determine the format of the new disk and then find a way to get the operating system from the old format to the new.



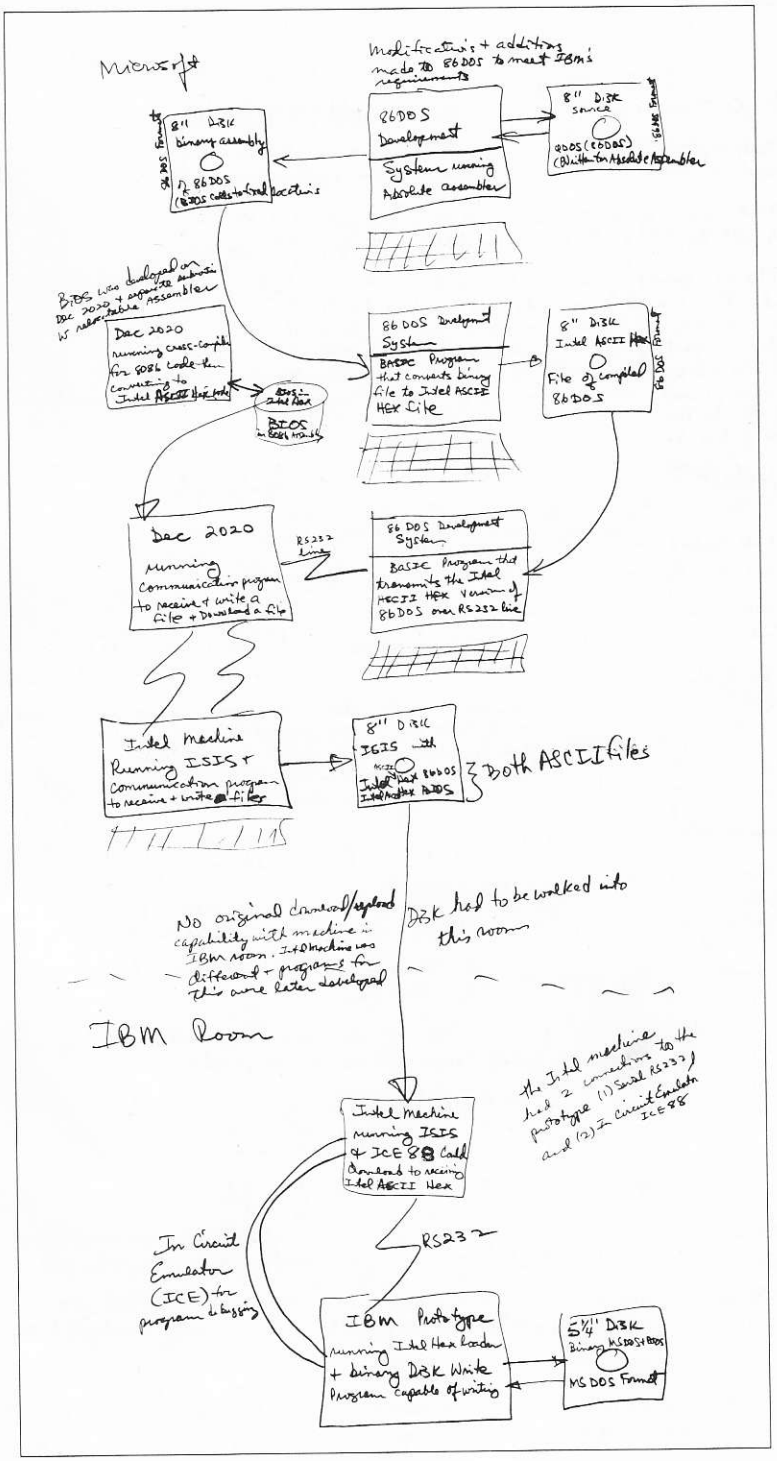
Paul Allen and Bill Gates (1982).

This work, handled by O'Rear, fell into a series of steps. First, he moved a section of code from the 8-inch disk and compiled it. Then, he converted the code to Intel hexadecimal format. Next, he uploaded it to a DEC-2020 and from there downloaded it to a large Intel fixed-disk development system with an In-Circuit Emulator. The DEC-2020 used for this task was also used in developing the BIOS, so there was additional work in downloading the BIOS to the Intel machine, converting it to hexadecimal format, moving it to an IBM development system, and then crossloading it to the IBM prototype.

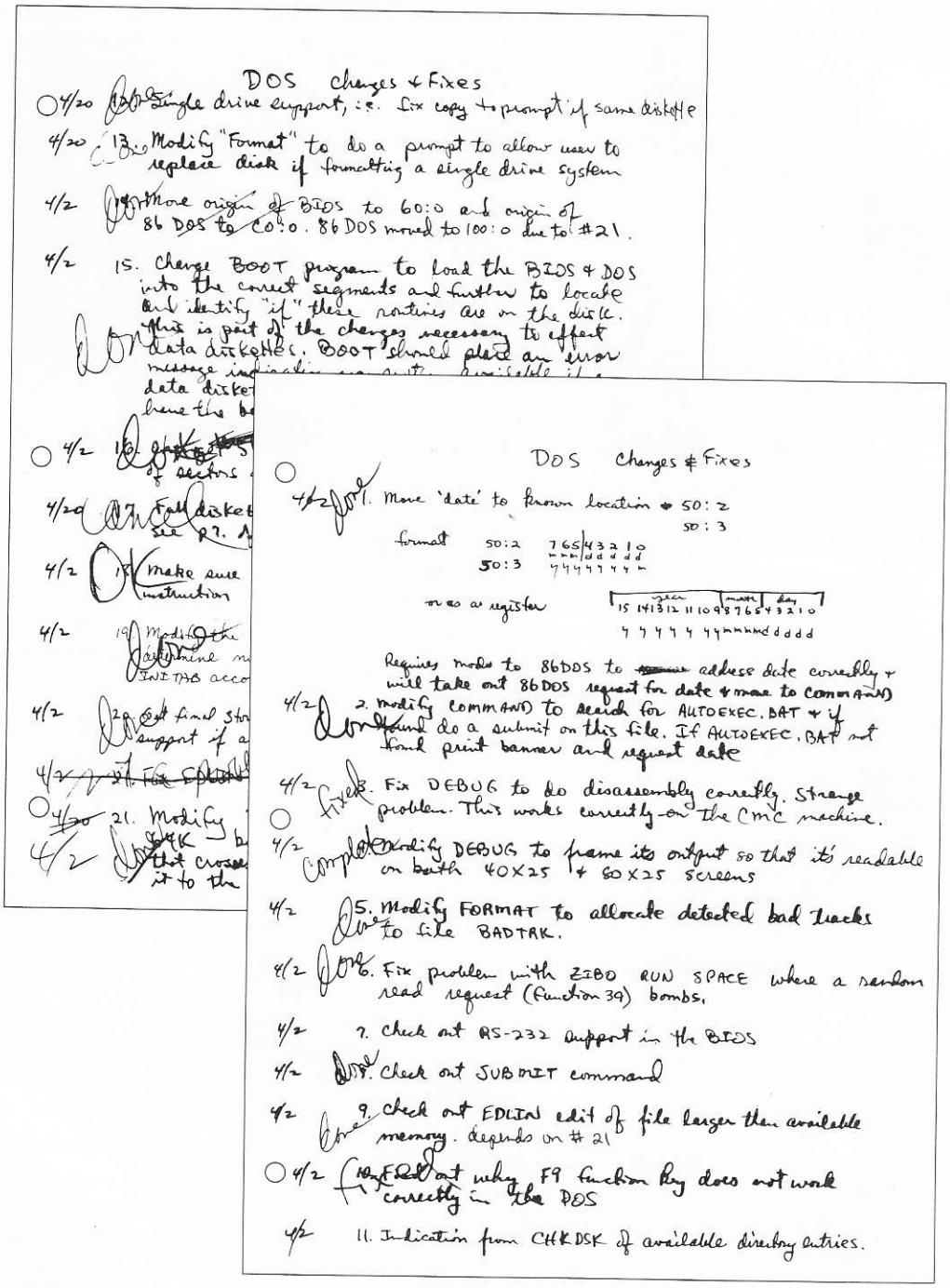
Defining and implementing the MS-DOS disk format — different from Paterson's 8-inch format — was an added challenge. Paterson's ultimate goal for 86-DOS was logical device independence, but during this first stage of development, the operating system simply had to be converted to handle logical records that were independent of the physical record size.

Paterson, still with Seattle Computer Products, continued to work on 86-DOS and by the end of 1980 had improved its logical device independence by adding functions that streamlined reading and writing multiple sectors and records, as well as records of variable size. In addition to making such refinements of his own, Paterson also worked on dozens of changes requested by Microsoft, from modifications to the operating system's startup messages to changes in EDLIN, the line editor he had written for his own use. Throughout this process, IBM's security restrictions meant that Paterson was never told the name of the OEM and never shown the prototype machines until he left Seattle Computer Products and joined Microsoft in May 1981.

And of course, throughout the process the developers encountered the myriad loose ends, momentary puzzles, bugs, and unforeseen details without which no project is complete. There were, for example, the serial card interrupts that occurred when they should not and, frustratingly, a hardware constraint that the BIOS could not accommodate at first and that resulted in sporadic crashes during early MS-DOS operations.



Bob O'Rear's sketch of the steps involved in moving 86-DOS to the IBM prototype.



DOS changes & Fixes

4/20 Single drive support, i.e. fix copy to prompt if same diskette

4/20 13. Modify "Format" to do a prompt to allow user to replace disk if formatting a single drive system

4/2 Move origin of BIOS to 60:0 and origin of 86 DOS to 80:0. 86 DOS moved to 100:0 due to #21.

4/2 15. Change BOOT program to load the BIOS & DOS into the correct segments and further to locate and identify "if" these routines are on the disk. This is part of the changes necessary to effect data diskettes. BOOT should place an error message in

4/2 Make sure instruction

4/2 Modify the ... INITTAB acco

4/2 20. Put final sto... support if a...

21. Modify ... that crosse... it to the

DOS Changes & Fixes

4/2 1. Move 'date' to known location @ 50:2 50:3

format 50:2 7 6 5 4 3 2 1 0 — m m d d d d d
50:3 y y y y y y y —

or as a register 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 — year, month, day — y y y y y y y m m m m d d d d d

Requires mods to 86DOS to address date correctly & will take out 86DOS request for date & mods to COMMAND

4/2 2. Modify COMMAND to search for AUTOEXEC.BAT & if found do a submit on this file. If AUTOEXEC.BAT not found print banner and request date

4/2 3. Fix DEBUG to do disassembly correctly. Strange problem. This works correctly on the CMC machine.

4/2 Modify DEBUG to frame its output so that it's readable on both 40X25 & 80X25 screens

4/2 5. Modify FORMAT to allocate detected bad tracks to file BADTAK.

4/2 6. Fix problem with ZIBO RUN SPACE where a random read request (function 39) bombs.

4/2 7. Check out RS-232 support in the BIOS

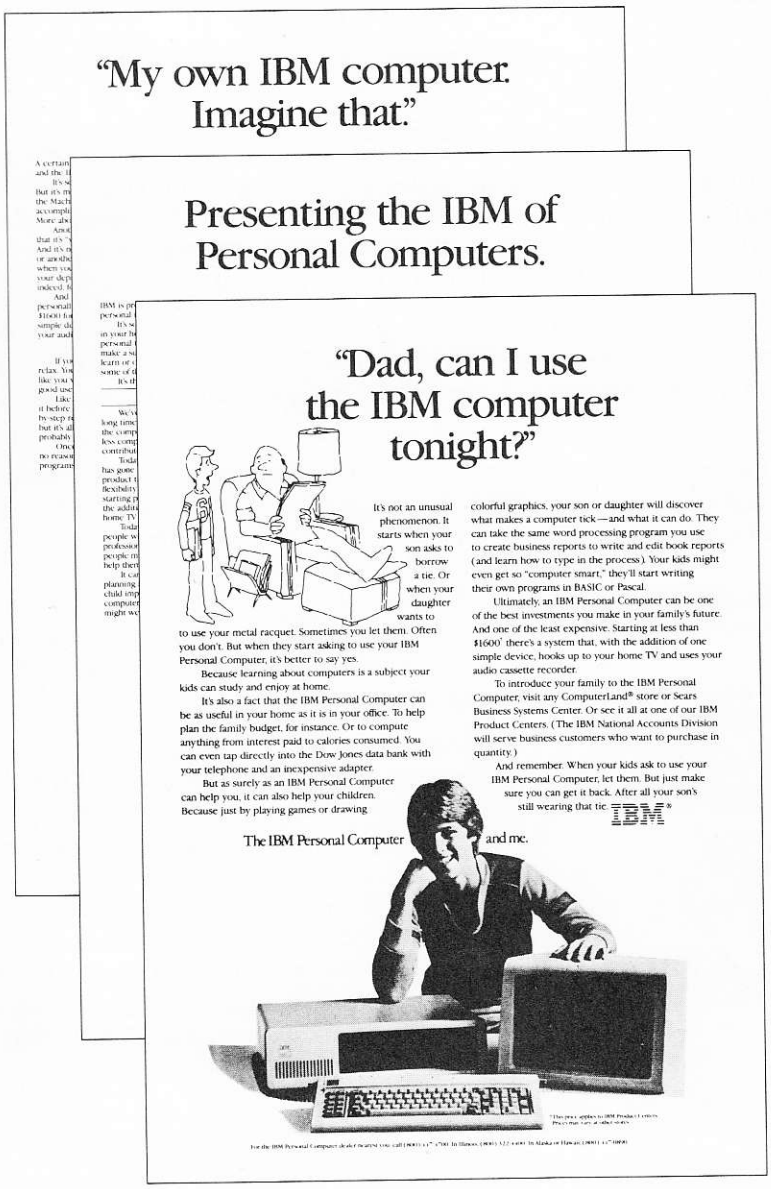
4/2 8. Check out SUBMIT command

4/2 9. Check out EDLIN edit of file larger than available memory. depends on # 21

4/2 Find out why F9 function key does not work correctly in the DOS

4/2 11. Indication from CHKDSK of available directory entries.

Part of Bob O'Rear's "laundry" list of operating-system changes and corrections for early April 1981. Around this time, interim beta copies were shipped to IBM for testing.



"My own IBM computer. Imagine that."

Presenting the IBM of Personal Computers.

"Dad, can I use the IBM computer tonight?"

It's not an unusual phenomenon. It starts when your son asks to borrow a tie. Or when your daughter wants to use your metal racquet. Sometimes you let them. Often you don't. But when they start asking to use your IBM Personal Computer, it's better to say yes.

Because learning about computers is a subject your kids can study and enjoy at home.

It's also a fact that the IBM Personal Computer can be as useful in your home as it is in your office. To help plan the family budget, for instance. Or to compute anything from interest paid to calories consumed. You can even tap directly into the Dow Jones data bank with your telephone and an inexpensive adapter.

But as surely as an IBM Personal Computer can help you, it can also help your children. Because just by playing games or drawing colorful graphics, your son or daughter will discover what makes a computer tick—and what it can do. They can take the same word processing program you use to create business reports to write and edit book reports (and learn how to type in the process). Your kids might even get so "computer smart," they'll start writing their own programs in BASIC or Pascal.

Ultimately, an IBM Personal Computer can be one of the best investments you make in your family's future. And one of the least expensive. Starting at less than $1600* there's a system that, with the addition of one simple device, hooks up to your home TV and uses your audio cassette recorder.

To introduce your family to the IBM Personal Computer, visit any ComputerLand® store or Sears Business Systems Center. Or see it all at one of our IBM Product Centers. (The IBM National Accounts Division will serve business customers who want to purchase in quantity.)

And remember. When your kids ask to use your IBM Personal Computer, let them. But just make sure you can get it back. After all your son's still wearing that tie.

IBM*

The IBM Personal Computer and me.

*The price applies to IBM Product Centers. Prices may vary at other stores.

For the IBM Personal Computer dealer nearest you, call [illegible].

Figure 1-1. The 1981 debut of the IBM Personal Computer.

In spite of such difficulties, however, the new operating system ran on the prototype for the first time in February 1981. In the six months that followed, the system was continually refined and expanded, and by the time of its debut in August 1981, MS-DOS, like the IBM Personal Computer on which it appeared, had become a functional product for home and office use.

Version 1

The first release of MS-DOS, version 1.0, was not the operating system Microsoft envisioned as a final model for 16-bit computer systems. According to Bill Gates, "Basically, what we wanted to do was one that was more like MS-DOS 2, with the hierarchical file system and everything...the key thing [in developing version 1.0] was my saying, 'Look, we can come out with a subset first and just go upward from that.'"

This first version — Gates's subset of MS-DOS — was actually a good compromise between the present and the future in two important respects: It enabled Microsoft to meet the development schedule for IBM and it maintained program-translation compatibility with CP/M.

Available only for the IBM Personal Computer, MS-DOS 1.0 consisted of 4000 lines of assembly-language source code and ran in 8 KB of memory. In addition to utilities such as DEBUG, EDLIN, and FORMAT, it was organized into three major files. One file, IBMBIO.COM, interfaced with the ROM BIOS for the IBM PC and contained the disk and character input/output system. A second file, IBMDOS.COM, contained the DOS kernel, including the application-program interface and the disk-file and memory managers. The third file, COMMAND.COM, was the external command processor — the part of MS-DOS most visible to the user.

To take advantage of the existing base of languages and such popular applications as WordStar and dBASE II, MS-DOS was designed to allow software developers to mechanically translate source code for the 8080 to run on the 8086. And because of this link, MS-DOS looked and acted like CP/M-80, at that time still the standard among operating systems for microcomputers. Like its 8-bit relative, MS-DOS used eight-character filenames and three-character extensions, and it had the same conventions for identifying disk drives in command prompts. For the most part, MS-DOS also used the same command language, offered the same file services, and had the same general structure as CP/M. The resemblance was even more striking at the programming level, with an almost one-to-one correspondence between CP/M and MS-DOS in the system calls available to application programs.

New Features

MS-DOS was not, however, a CP/M twin, nor had Microsoft designed it to be inextricably bonded to the IBM PC. Hoping to create a product that would be successful over the long term, Microsoft had taken steps to make MS-DOS flexible enough to accommodate changes and new directions in the hardware technology — disks, memory boards, even microprocessors — on which it depended. The first steps toward this independence from

BUSINESS
Digest

Big I.B.M.'s Little Computer

Its Desk-Top Model Brings A New Image

The U.S. Desktop Computer Market

UNIT SHIPMENTS | VALUE OF SHIPMENTS

TANDY
APPLE
COMMODORE
ATARI
HEWLETT-PACKARD
NORTHSTAR
TEXAS INSTRUMENTS
IBM
TEKTRONIX
All Others

Retail Sales In U.S. Up 1.3% in July

But Analysts Are Dubious of General Upturn

WASHINGTON

IBM's New Line Likely to Shake Up The Market for Personal Computers

By George Anders
Staff Reporter of The Wall Street Journal

NEW YORK—International Business Machines Corp. has made its bold entry into the personal-computer market, and experts believe the computer giant could capture the lead in the youthful industry within two years.

Yesterday the company introduced several versions of a small computer designed for use in homes, schools and offices. Prices

catch-up. The IBM machines operate on an Intel Corp. 8088 microprocessor, a faster and more powerful "chip" than those used in rivals' machines. IBM also has obtained for distribution such popular programs as VisiCalc, a financial forecasting model marketed by Personal Software Inc.

Other programs, or software, for the IBM equipment include the EasyWriter word-processing system, three accounting packages from Peachtree Software Inc. and

far greater, equivalent to more than 1,000 typewritten pages. The new IBM computers don't use all that capacity, but what they do use will enable them to work with longer programs and more data than competing machines and to display images on their video screens in greater detail.

But the added memory comes at a price. IBM acknowledges that a fully stocked computer will cost $6,000 or more. Its basic $1,565 machine comes with 16,000 characters

announcements we've seen in the industry," said Christopher Morgan, editor in chief of Byte, a personal computer magazine.

"People will now know that personal computers are not a fad or a flash in the pan," said Malcolm McConnell, executive vice president of Computerland, a chain of retail stores that will market the new I.B.M. products.

The price of the machines will range from $1,565 for a simple system that will require users to provide their own television-like display screens and

another $540.

ble extras runs
analyst at Mor
new system s
It's a nice
kage," he says.
little that you
here" already
ggests that IBM
rovided" price
equipped model
Apple III, the

InfoWorld
News For Microcomputer Users

IBM Announces New Microcomputer System

It's Official; One surprise

By Thom Hogan, IW Staff

NEW YORK, NY—Within a month you should be able to order an IBM Personal Computer. Whether or not that will have any effect on the microcomputing industry remains to be seen.

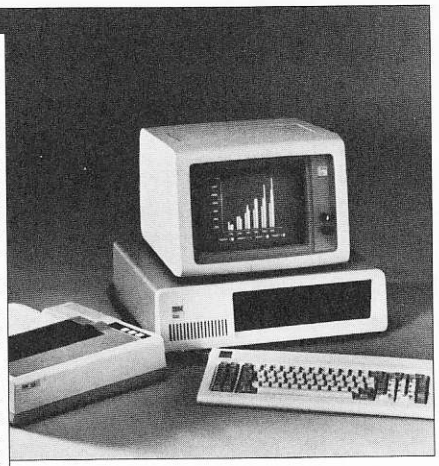
For those of you who have been reading *InfoWorld*, there were few surprises in the IBM announcement. Although the actual introduction took place a month later than anticipated, the features of the machine are virtually identical to the information we've already published.

tion of the unit taking place at the IBM facility at Boca Raton, Florida, as reported earlier in *InfoWorld*. Computerland, the newly created Sears Business Centers and IBM Product Centers will be selling the Personal Computer. IBM is also setting up a special division of the Data Processing Group to market the machine.

The price begins at $1565, slightly higher than we reported earlier. It includes the keyboard unit, an enhanced Microsoft BASIC in

OUTLOOK

IBM really gets personal.



ently unveiled its first offering in the personal computer market—the IBM Personal Computer. The unit, perhaps surprisingly, plays music and includes game software to say nothing of the standard features available.

The machine is impressive. It's starting price is a mere $1565. For that price the buyer gets the 83-key keyboard, the computer itself, based on an 8088 microprocessor, and 16k of main memory. This minimal configuration can use a tape cassette for mass storage and a television set (with an rf modulator) for a display. (The machine is fully FCC certified for home operation as a class B computing device.)

IBM is cognizant of the fact that this minimally configured machine probably won't last a serious computerist long before he wants to expand. The company offers upgraded versions of the machine, and will sell them in different configurations. For example, the firm lists a more typical configuration for home or school as 64k of main memory, one disk

continued on page 17

12 Personal Computing/October 1981

PERSONAL COMPUTERS

PERSONAL COMPUTER FROM IBM

The mainframer's long-awaited entry into the personal computing market aims for corporate as well as home users.

With uncharacteristic but resounding fanfare, IBM ended the summer's most popular guessing game for the industry by introducing its Personal Computer. Highly comparable to offerings from arch-contenders Apple and Radio Shack, the machine represents several new tacks for the leading computer manufacturer as it attempts to hitch its wagon to one of the fastest growing segments of the industry.

The computer, which is designed to appeal to home users as well as corporate professionals, ranges in price from $1,565 for a bare-bones configuration to $6,300 for the full-blown model. It will be sold through Sears and Computerland computer retail stores as well as directly to large corporate and educational users, IBM says, pointing out that it has set up a special national marketing team to handle such volume orders.

Donald Estridge, the articulate director of IBM's entry systems business who braved strobes and movie lights at the machine's Waldorf-Astoria introduction, declines to say how many personnel have been dedicated to the national marketing effort, but says it will be selling in volumes of 20 machines or more. Several weeks after the unveiling, he said response so far had been "very, very good," with orders being taken but no deliveries to be made before this month.

In addition to the game of Adventure, which Estridge said has been thoroughly exercised by his Boca Raton, Fla., staff, IBM has decked out the machine with an array of packaged applications programs that are expected to make it attractive to the corporate user.

Among these are the popular VisiCalc spreadsheet package from Personal Software, accounting packages from Management Science America's Peachtree Software operation, and Information Unlimited's EasyWriter word processing system. Although IBM wouldn't say, more independently developed packages are certain to be offered for the computer as well as packages

A sampling of the headlines and newspaper articles that abounded when IBM announced its Personal Computer.

MICROSOFT QUARTERLY

This policy is especially advantageous when a large number of programs is distributed using a single copy of the runtime module because only one royalty payment is paid.

(Microsoft still supports the runtime system used with previous versions. If application programmers link the old library to their applications, there is no royalty fee. This applies to versions 5.2 and earlier, too.)

This change in the BASCOM royalty policy reflects Microsoft's wish to increase the number of application packages on the market. This policy change, the addition of CHAIN with COMMON, and the implementation of the runtime module make BASCOM a much more flexible and powerful tool for the application programmer.

BASCOM 5.3 is available now for CP/M systems, including the Apple II with the Microsoft Softcard. Microsoft is committed to supporting BASCOM and the BASIC interpreter on many processors and operating systems, thus assuring that application programs created with BASCOM have, and will continue to have, the broadest possible market.

Paul Allen

IBM Breaks the 16-Bit Barrier

The most important feature of the new IBM Personal Computer is its 8088 CPU. IBM's choice of the 8088 opens up two areas of the industry that have been

Paul Allen, Vice President, Microsoft

on the verge of changing for the past 10 months: first, the industry's hesitancy over a serious 16-bit software commitment has finally been broken; and second, the capabilities of the 16-bit processors are finally being put to some really exciting uses.

A 16-bit processor gives software designers many advantages inherent in an enhanced instruction set. For example, we've taken advantage of the expanded addressing in our MS-LINK, a linker for Pascal or FORTRAN programs that are up to a megabyte in size. In 96K of memory, the Microsoft 8086 BASIC interpreter can execute a 64K program, almost double the size executable on an 8-bit runtime. Applications programs can be more sophisticated in their features, human engineering factors, and in solving problems that involve larger amounts of data.

The larger number of registers with the 8086/8088 processors also means that complex operations, such as floating point and graphics routines execute much faster. The speed of the graphics primitives in MBASIC-86 makes it very easy to construct a graphics application without machine language.

With the IBM announcement of the Personal Computer, it looks as though the industry is finally gearing up for serious 16-bit software support. In addition to the Microsoft software already provided for the IBM Personal Computer, we're planning a full line of 16-bit languages and end-user software tools. Application packages are rapidly being adapted to the 16-bit environment, especially those programs already written in Microsoft BASIC.

The "linch pin" of Microsoft's new 16-bit product line for the 8086/8088 is our compact, flexible operating system, MS-DOS. MS-DOS is the primary operating system on the IBM Personal Computer. We've maintained compatibility with existing CP/M 2.x operating system calls, so it's a straightforward process to convert 8080 and Z80 programs to run under MS-DOS. MS-DOS also provides a future upgrade path to the XENIX multi-user, multitasking environment. Other important features of MS-DOS include error recovery, device independent I/O, and built-in variable length disk reads and writes. What is now the standard operating system for the IBM Personal Computer will no doubt become an industry standard.

Now that the 16-bit software barrier has been crossed and the technical capabilities of the 16-bit processors are being appreciated, Microsoft expects to see many 16-bit personal computers. It's an industry move we've anticipated for quite some time and, given the momentum of IBM, it should soon be in full swing.

Microsoft COBOL Passes GSA Validation

Microsoft is always concerned about standards for all its products. The United States government, the largest user of computer equipment and software in the world, has developed tests for compliance with and implementation of standards for compilers. Testing of compilers, called validation, is performed by government inspectors, who are independent of software developers.

Microsoft submitted its COBOL compiler (under the CP/M operating system) for validation. The General Services Administration (GSA) performed the validation tests and validated Microsoft COBOL as a low-intermediate implementation of the 1974 ANSI standard for COBOL.

Why is Microsoft concerned about standards, and why did we submit Microsoft COBOL for validation? Mike Orr, COBOL product manager, offered the following reasons

(continued on back)

A page from Microsoft's third-quarter report for 1981.

specific hardware configurations appeared in MS-DOS version 1.0 in the form of device-independent input and output, variable record lengths, relocatable program files, and a replaceable command processor.

MS-DOS made input and output device independent by treating peripheral devices as if they were files. To do this, it assigned a reserved filename to each of the three devices it recognized: CON for the console (keyboard and display), PRN for the printer, and AUX for the auxiliary serial ports. Whenever one of these reserved names appeared in the file control block of a file named in a command, all operations were directed to the device, rather than to a disk file. (A file control block, or FCB, is a 37-byte housekeeping record located in an application's portion of the memory space. It includes, among other things, the filename, the extension, and information about the size and starting location of the file on disk.)

Such device independence benefited both application developers and computer users. On the development side, it meant that applications could use one set of read and write calls, rather than a number of different calls for different devices, and it meant that an application did not have to be modified if new devices were added to the system. From the

user's point of view, device independence meant greater flexibility. For example, even if a program had been designed for disk I/O only, the user could still use a file for input or direct output to the printer.

Variable record lengths provided another step toward logical independence. In CP/M, logical and physical record lengths were identical: 128 bytes. Files could be accessed only in units of 128 bytes and file sizes were always maintained in multiples of 128 bytes. With MS-DOS, however, physical sector sizes were of no concern to the user. The operating system maintained file lengths to the exact size in bytes and could be relied on to support logical records of any size desired.

Another new feature in MS-DOS was the relocatable program file. Unlike CP/M, MS-DOS had the ability to load two different types of program files, identified by the extensions .COM and .EXE. Program files ending with .COM mimicked the binary files in CP/M. They were more compact than .EXE files and loaded somewhat faster, but the combined program code, stack, and data could be no larger than 64 KB. A .EXE program, on the other hand, could be much larger, because the code, stack, and data segments could be loaded as modules in separate parts of memory determined by MS-DOS. Once the segments were in memory, MS-DOS then used part of the file header, or relocation table, to automatically set the correct addresses for each of the different program segments.

In addition to supporting .EXE files, MS-DOS made the external command processor, COMMAND.COM, more adaptable by making it a separate relocatable file just like any other program. It could therefore be replaced by a custom command processor, as long as the new file was also named COMMAND.COM.

Performance

Everyone familiar with the IBM PC knows that MS-DOS eventually became the dominant operating system on 8086-based microcomputers. There were several reasons for this, not least of which was acceptance of MS-DOS as the operating system for IBM's phenomenally successful line of personal computers. But even though MS-DOS was the only operating system available when the first IBM PCs were shipped, positioning alone would not necessarily have guaranteed its ability to outstrip CP/M-86, which appeared six months later. MS-DOS also offered significant advantages to the user in a number of areas, including the allocation and management of storage space on disk.

Like CP/M, MS-DOS shared out disk space in allocation units. Unlike CP/M, however, MS-DOS mapped the use of these allocation units in a central file allocation table — the FAT — that was always in memory. Both operating systems used a directory entry for recording information about each file, but whereas a CP/M directory entry included an allocation map — a list of sixteen 1 KB allocation units where successive parts of the file were stored — an MS-DOS directory entry pointed only to the first allocation unit in the FAT and each entry in the table then pointed to the next unit associated with the file. Thus, CP/M might require several directory entries (and more than one disk access) to load a file

larger than 16 KB, but MS-DOS retained a complete in-memory list of all file components and all available disk space without having to access the disk at all. As a result, MS-DOS's ability to find and load even very long files was extremely rapid compared with CP/M's.

Two other important features—the ability to read and write multiple records with one operating-system call and the transient use of memory by the MS-DOS command processor—provided further efficiency for both users and developers.

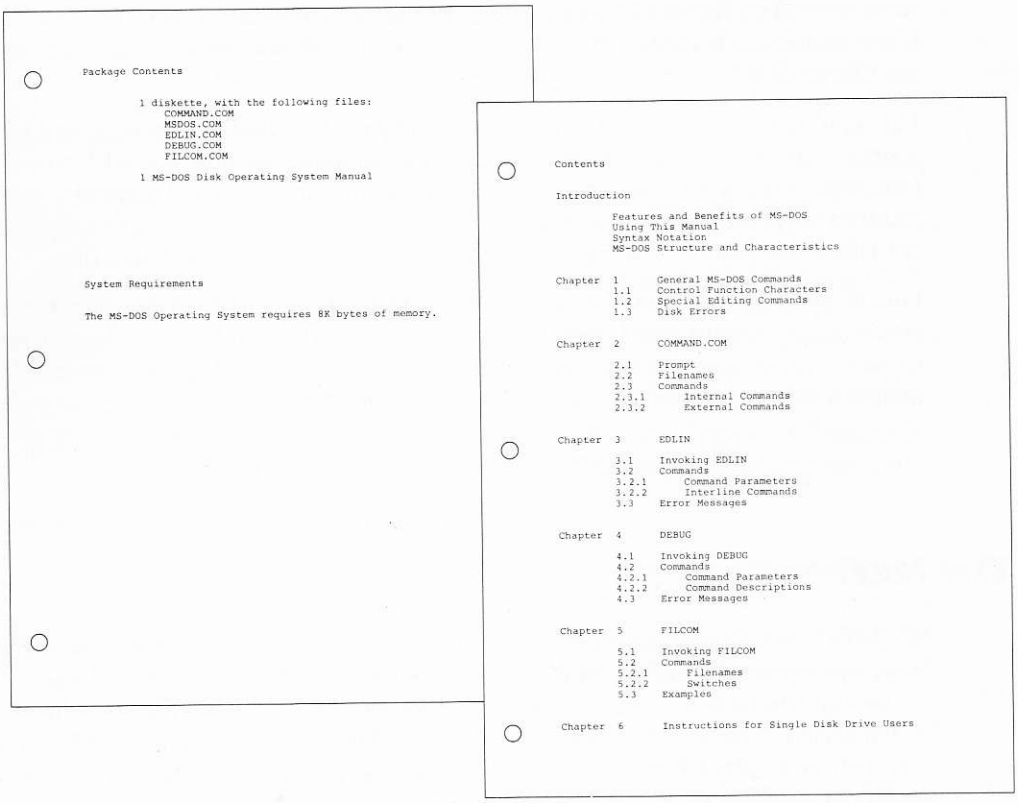
The independence of the logical record from the physical sector laid the foundation for the ability to read and write multiple sectors. When reading multiple records in CP/M, an application had to issue a read function call for each sector, one at a time. With MS-DOS, the application could issue one read function call, giving the operating system the beginning record and the number of records to read, and MS-DOS would then load all of the corresponding sectors automatically.

Another innovative feature of MS-DOS version 1.0 was the division of the command processor, COMMAND.COM, into a resident portion and a transient portion. (There is also a third part, an initialization portion, which carries out the commands in an AUTOEXEC batch file at startup. This part of COMMAND.COM is discarded from memory when its work is finished.) The reason for creating resident and transient portions of the command processor had to do with maximizing the efficiency of MS-DOS for the user: On the one hand, the programmers wanted COMMAND.COM to include commonly requested functions, such as DIR and COPY, for speed and ease of use; on the other hand, adding these commands meant increasing the size of the command processor, with a resulting decrease in the memory available to application programs. The solution to this trade-off of speed versus utility was to include the extra functions in a transient portion of COMMAND.COM that could be overwritten by any application requiring more memory. To maintain the integrity of the functions for the user, the resident part of COMMAND.COM was given the job of checking the transient portion for damage when an application terminated. If necessary, this resident portion would then load a new copy of its transient partner into memory.

Ease of Use

In addition to its moves toward hardware independence and efficiency, MS-DOS included several services and utilities designed to make life easier for users and application developers. Among these services were improved error handling, automatic logging of disks, date and time stamping of files, and batch processing.

MS-DOS and the IBM PC were targeted at a nontechnical group of users, and from the beginning IBM had stressed the importance of data integrity. Because data is most likely to be lost when a user responds incorrectly to an error message, an effort was made to include concise yet unambiguous messages in MS-DOS. To further reduce the risks of misinterpretation, Microsoft used these messages consistently across all MS-DOS functions and utilities and encouraged developers to use the same messages, where appropriate, in their applications.



Package Contents

1 diskette, with the following files:
COMMAND.COM
MSDOS.COM
EDLIN.COM
DEBUG.COM
FILCOM.COM

1 MS-DOS Disk Operating System Manual

System Requirements

The MS-DOS Operating System requires 8K bytes of memory.

Contents

Two pages from Microsoft's MS-DOS version 1.0 manual. On the left, the system's requirements — 8 KB of memory; on the right, the 118-page manual's complete table of contents.

In a further attempt to safeguard data, MS-DOS also trapped hard errors — such as critical hardware errors — that had previously been left to the hardware-dependent logic. Now the hardware logic could simply report the nature of the error and the operating system would handle the problem in a consistent and systematic way. MS-DOS could also trap the Control-C break sequence so that an application could either protect against accidental termination by the user or provide a graceful exit when appropriate.

To reduce errors and simplify use of the system, MS-DOS also automatically updated memory information about the disk when it was changed. In CP/M, users had to log new disks as they changed them — a cumbersome procedure on single-disk systems or when data was stored on multiple disks. In MS-DOS, new disks were automatically logged as long as no file was currently open.

Another new feature — one visible with the DIR command — was date and time stamping of disk files. Even in its earliest forms, MS-DOS tracked the system date and displayed it at every startup, and now, when it turned out that only the first 16 bytes of a directory entry

were needed for file-header information, the MS-DOS programmers decided to use some of the remaining 16 bytes to record the date and time of creation or update (and the size of the file) as well.

Batch processing was originally added to MS-DOS to help IBM. IBM wanted to run scripts — sequences of commands or other operations — one after the other to test various functions of the system. To do this, the testers needed an automated method of calling routines sequentially. The result was the batch processor, which later also provided users with the convenience of saving and running MS-DOS commands as batch files.

Finally, MS-DOS increased the options available to a program when it terminated. For example, in less sophisticated operating systems, applications and other programs remained in memory only as long as they were active; when terminated, they were removed from memory. MS-DOS, however, added a terminate-and-stay-resident function that enabled a program to be locked into memory and, in effect, become part of the operating-system environment until the computer system itself was shut down or restarted.

The Marketplace

When IBM announced the Personal Computer, it said that the new machine would run three operating systems: MS-DOS, CP/M-86, and SofTech Microsystem's p-System. Of the three, only MS-DOS was available when the IBM PC shipped. Nevertheless, when MS-DOS was released, nine out of ten programs on the *InfoWorld* bestseller list for 1981 ran under CP/M-80, and CP/M-86, which became available about six months later, was the operating system of choice to most writers and reviewers in the trade press.

Understandably, MS-DOS was compared with CP/M-80 and, later, CP/M-86. The main concern was compatibility: To what extent was Microsoft's new operating system compatible with the existing standard? No one could have foreseen that MS-DOS would not only catch up with but supersede CP/M. Even Bill Gates now recalls that "our most optimistic view of the number of machines using MS-DOS wouldn't have matched what really ended up happening."

To begin with, the success of the IBM PC itself surprised many industry watchers. Within a year, IBM was selling 30,000 PCs per month, thanks in large part to a business community that was already comfortable with IBM's name and reputation and, at least in retrospect, was ready for the leap to personal computing. MS-DOS, of course, benefited enormously from the success of the IBM PC — in large part because IBM supplied all its languages and applications in MS-DOS format.

But, at first, writers in the trade press still believed in CP/M and questioned the viability of a new operating system in a world dominated by CP/M-80. Many assumed, incorrectly, that a CP/M-86 machine could run CP/M-80 applications. Even before CP/M-86 was available, *Future Computing* referred to the IBM PC as the "CP/M Record Player" — presumably in anticipation of a vast inventory of CP/M applications for the new computer — and led its readers to assume that the PC was actually a CP/M machine.

Microsoft, meanwhile, held to the belief that the success of IBM's machine or any other 16-bit microcomputer depended ultimately on the emergence of an industry standard for a 16-bit operating system. Software developers could not afford to develop software for even two or three different operating systems, and users could (or would) not pay the prices the developers would have to charge if they did. Furthermore, users would almost certainly rebel against the inconvenience of sharing data stored under different operating-system formats. There had to be one operating system, and Microsoft wanted MS-DOS to be the one.

The company had already taken the first step toward a standard by choosing hardware independent designs wherever possible. Machine independence meant portability, and portability meant that Microsoft could sell one version of MS-DOS to different hardware manufacturers who, in turn, could adapt it to their own equipment. Portability alone, however, was no guarantee of industry-wide acceptance. To make MS-DOS the standard, Microsoft needed to convince software developers to write programs for MS-DOS. And in 1981, these developers were a little confused about IBM's new operating system.

An operating system by any other name...

A tangle of names gave rise to one point of confusion about MS-DOS. Tim Paterson's "Quick and Dirty Operating System" for the 8086 was originally shipped by Seattle Computer Products as 86-DOS. After Microsoft purchased 86-DOS, the name remained for a while, but by the time the PC was ready for release, the new system was known as MS-DOS. Then, after the IBM PC reached the market, IBM began to refer to the operating system as the IBM Personal Computer DOS, which the trade press soon shortened to PC-DOS. IBM's version contained some utilities, such as DISKCOPY and DISKCOMP, that were not included in MS-DOS, the generic version available for license by other manufacturers. By calling attention to these differences, publications added to the confusion about the distinction between the Microsoft and IBM releases of MS-DOS.

Further complications arose when Lifeboat Associates agreed to help promote MS-DOS but decided to call the operating system Software Bus 86. MS-DOS thus became one of a line of trademarked Software Bus products, another of which was a product called SB-80, Lifeboat's version of CP/M-80.

Finally, some of the first hardware companies to license MS-DOS also wanted to use their own names for the operating system. Out of this situation came such additional names as COMPAQ-DOS and Zenith's Z-DOS.

Given this confusing host of names for a product it believed could become the industry standard, Microsoft finally took the lead and, as developer, insisted that the operating system was to be called MS-DOS. Eventually, everyone but IBM complied.

Developers and MS-DOS

Early in its career, MS-DOS represented just a small fraction of Microsoft's business—much larger revenues were generated by BASIC and other languages. In addition, in the first two years after the introduction of the IBM PC, the growth of CP/M-86 and other

environments nearly paralleled that of MS-DOS. So Microsoft found itself in the unenviable position of giving its support to MS-DOS while also selling languages to run on CP/M-86, thereby contributing to the growth of software for MS-DOS's biggest competitor.

Given the uncertain outcome of this two-horse race, some other software developers chose to wait and see which way the hardware manufacturers would jump. For their part, the hardware manufacturers were confronting the issue of compatibility between operating systems. Specifically, they needed to be convinced that MS-DOS was not a maverick — that it could perform as well as CP/M-86 as a base for applications that had been ported from the CP/M-80 environment for use on 16-bit computers.

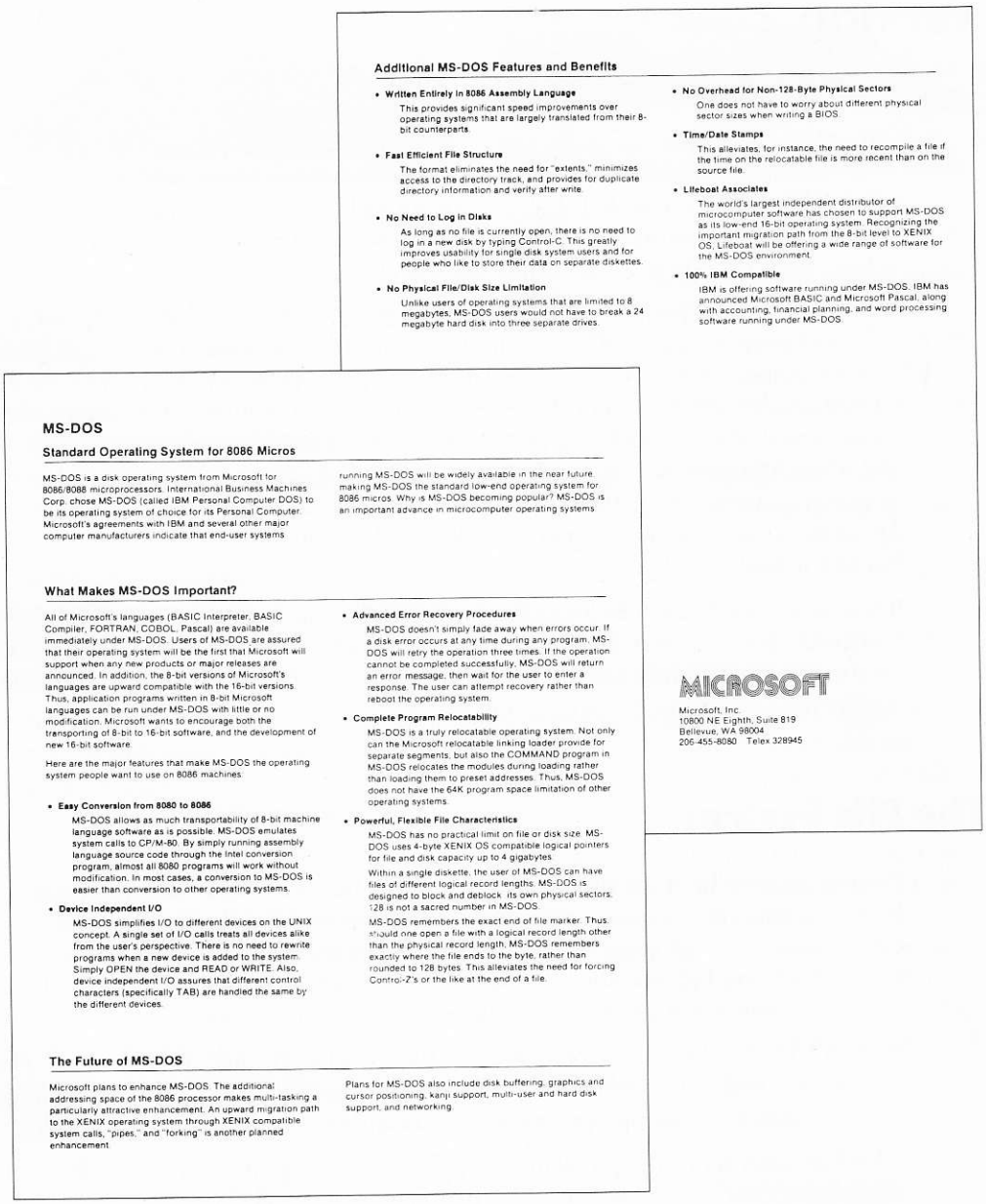
Microsoft approached the problem by emphasizing four related points in its discussions with hardware manufacturers:

- First, one of Microsoft's goals in developing the first version of MS-DOS had always been translation compatibility from CP/M-80 to MS-DOS software.
- Second, translation was possible only for software written in 8080 or Z80 assembly language; thus, neither MS-DOS nor CP/M-86 could run programs written for other 8-bit processors, such as the 6800 or the 6502.
- Third, many applications were written in a high-level language, rather than in assembly language.
- Fourth, most of those high-level languages were Microsoft products and ran on MS-DOS.

Thus, even though some people had originally believed that only CP/M-86 would automatically make the installed base of CP/M-80 software available to the IBM PC and other 16-bit computers, Microsoft convinced the hardware manufacturers that MS-DOS was, in actuality, as flexible as CP/M-86 in its compatibility with existing — and appropriate — CP/M-80 software.

MS-DOS was put at a disadvantage in one area, however, when Digital Research convinced several manufacturers to include both 8080 and 8086 chips in their machines. With 8-bit and 16-bit software used on the same machine, the user could rely on the same disk format for both types of software. Because MS-DOS used a different disk format, CP/M had the edge in these dual-processor machines — although, in fact, it did not seem to have much effect on the survival of CP/M-86 after the first year or so.

Although making MS-DOS the operating system of obvious preference was not as easy as simply convincing hardware manufacturers to offer it, Microsoft's list of MS-DOS customers grew steadily from the time the operating system was introduced. Many manufacturers continued to offer CP/M-86 along with MS-DOS, but by the end of 1983 the technical superiority of MS-DOS (bolstered by the introduction of such products as Lotus 1-2-3) carried the market. For example, when DEC, a longtime holdout, decided to make MS-DOS the primary operating system for its Rainbow computer, the company mentioned the richer set of commands and "dramatically" better disk performance of MS-DOS as reasons for its choice over CP/M-86.



Additional MS-DOS Features and Benefits

- **Written Entirely in 8086 Assembly Language**
  This provides significant speed improvements over operating systems that are largely translated from their 8-bit counterparts.
- **Fast Efficient File Structure**
  The format eliminates the need for "extents," minimizes access to the directory track, and provides for duplicate directory information and verify after write.
- **No Need to Log in Disks**
  As long as no file is currently open, there is no need to log in a new disk by typing Control-C. This greatly improves usability for single disk system users and for people who like to store their data on separate diskettes.
- **No Physical File/Disk Size Limitation**
  Unlike users of operating systems that are limited to 8 megabytes, MS-DOS users would not have to break a 24 megabyte hard disk into three separate drives.
- **No Overhead for Non-128-Byte Physical Sectors**
  One does not have to worry about different physical sector sizes when writing a BIOS.
- **Time/Date Stamps**
  This alleviates, for instance, the need to recompile a file if the time on the relocatable file is more recent than on the source file.
- **Lifeboat Associates**
  The world's largest independent distributor of microcomputer software has chosen to support MS-DOS as its low-end 16-bit operating system. Recognizing the important migration path from the 8-bit level to XENIX OS, Lifeboat will be offering a wide range of software for the MS-DOS environment.
- **100% IBM Compatible**
  IBM is offering software running under MS-DOS. IBM has announced Microsoft BASIC and Microsoft Pascal, along with accounting, financial planning, and word processing software running under MS-DOS.

MS-DOS

Standard Operating System for 8086 Micros

MS-DOS is a disk operating system from Microsoft for 8086/8088 microprocessors. International Business Machines Corp. chose MS-DOS (called IBM Personal Computer DOS) to be its operating system of choice for its Personal Computer. Microsoft's agreements with IBM and several other major computer manufacturers indicate that end-user systems running MS-DOS will be widely available in the near future, making MS-DOS the standard low-end operating system for 8086 micros. Why is MS-DOS becoming popular? MS-DOS is an important advance in microcomputer operating systems.

What Makes MS-DOS Important?

All of Microsoft's languages (BASIC Interpreter, BASIC Compiler, FORTRAN, COBOL, Pascal) are available immediately under MS-DOS. Users of MS-DOS are assured that their operating system will be the first that Microsoft will support when any new products or major releases are announced. In addition, the 8-bit versions of Microsoft's languages are upward compatible with the 16-bit versions. Thus, application programs written in 8-bit Microsoft languages can be run under MS-DOS with little or no modification. Microsoft wants to encourage both the transporting of 8-bit to 16-bit software, and the development of new 16-bit software.

Here are the major features that make MS-DOS the operating system people want to use on 8086 machines:

- **Easy Conversion from 8080 to 8086**
  MS-DOS allows as much transportability of 8-bit machine language software as is possible. MS-DOS emulates system calls to CP/M-80. By simply running assembly language source code through the Intel conversion program, almost all 8080 programs will work without modification. In most cases, a conversion to MS-DOS is easier than conversion to other operating systems.
- **Device Independent I/O**
  MS-DOS simplifies I/O to different devices on the UNIX concept. A single set of I/O calls treats all devices alike from the user's perspective. There is no need to rewrite programs when a new device is added to the system. Simply OPEN the device and READ or WRITE. Also, device independent I/O assures that different control characters (specifically TAB) are handled the same by the different devices.
- **Advanced Error Recovery Procedures**
  MS-DOS doesn't simply fade away when errors occur. If a disk error occurs at any time during any program, MS-DOS will retry the operation three times. If the operation cannot be completed successfully, MS-DOS will return an error message, then wait for the user to enter a response. The user can attempt recovery rather than reboot the operating system.
- **Complete Program Relocatability**
  MS-DOS is a truly relocatable operating system. Not only can the Microsoft relocatable linking loader provide for separate segments, but also the COMMAND program in MS-DOS relocates the modules during loading rather than loading them to preset addresses. Thus, MS-DOS does not have the 64K program space limitation of other operating systems.
- **Powerful, Flexible File Characteristics**
  MS-DOS has no practical limit on file or disk size. MS-DOS uses 4-byte XENIX OS compatible logical pointers for file and disk capacity up to 4 gigabytes.
  Within a single diskette, the user of MS-DOS can have files of different logical record lengths. MS-DOS is designed to block and deblock its own physical sectors; 128 is not a sacred number in MS-DOS.
  MS-DOS remembers the exact end of file marker. Thus, should one open a file with a logical record length other than the physical record length, MS-DOS remembers exactly where the file ends to the byte, rather than rounded to 128 bytes. This alleviates the need for forcing Control-Z's or the like at the end of a file.

The Future of MS-DOS

Microsoft plans to enhance MS-DOS. The additional addressing space of the 8086 processor makes multi-tasking a particularly attractive enhancement. An upward migration path to the XENIX operating system through XENIX compatible system calls, "pipes," and "forking" is another planned enhancement.

Plans for MS-DOS also include disk buffering, graphics and cursor positioning, kanji support, multi-user and hard disk support, and networking.

MICROSOFT

Microsoft, Inc.
10800 NE Eighth, Suite 819
Bellevue, WA 98004
206-455-8080 Telex 328945

A Microsoft original equipment manufacturer (OEM) marketing brochure describing the strengths of MS-DOS.

Version 2

After the release of PC-specific version 1.0 of MS-DOS, Microsoft worked on an update that contained some bug fixes. Version 1.1 was provided to IBM to run on the upgraded PC released in 1982 and enabled MS-DOS to work with double-sided, 320 KB floppy disks. This version, referred to as 1.25 by all but IBM, was the first version of MS-DOS shipped by other OEMs, including COMPAQ and Zenith.

Even before these intermediate releases were available, however, Microsoft began planning for future versions of MS-DOS. In developing the first version, the programmers had had two primary goals: running translated CP/M-80 software and keeping MS-DOS small. They had neither the time nor the room to include more sophisticated features, such as those typical of Microsoft's UNIX-based multiuser, multitasking operating system, XENIX. But when IBM informed Microsoft that the next major edition of the PC would be the Personal Computer XT with a 10-megabyte fixed disk, a larger, more powerful version of MS-DOS — one closer to the operating system Microsoft had envisioned from the start — became feasible.

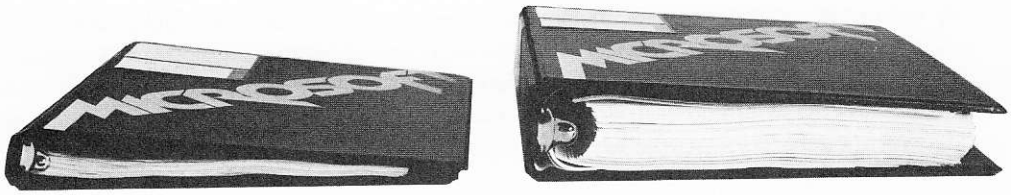
There were three particular areas that interested Microsoft: a new, hierarchical file system, installable device drivers, and some type of multitasking. Each of these features contributed to version 2.0, and together they represented a major change in MS-DOS while still maintaining compatibility with version 1.0.

The File System

Primary responsibility for version 2.0 fell to Paul Allen, Mark Zbikowski, and Aaron Reynolds, who wrote (and rewrote) most of the version 2.0 code. The major design issue confronting the developers, as well as the most visible example of its difference from versions 1.0, 1.1, and 1.25, was the introduction of a hierarchical file system to handle the file-management needs of the XT's fixed disk.

Version 1.0 had a single directory for all the files on a floppy disk. That system worked well enough on a disk of limited capacity, but on a 10-megabyte fixed disk a single directory could easily become unmanageably large and cumbersome.

CP/M had approached the problem of high-capacity storage media by using a partitioning scheme that divided the fixed disk into 10 user areas equivalent to 10 separate floppy-disk drives. On the other hand, UNIX, which had traditionally dealt with larger systems, used a branching, hierarchical file structure in which the user could create directories and subdirectories to organize files and make them readily accessible. This was the file-management system implemented in XENIX, and it was the MS-DOS team's choice for handling files on the XT's fixed disk.



The MS-DOS version 1.0 manual next to the version 2.0 manual.

Partitioning, IBM's initial choice, had the advantages of familiarity, size, and ease of implementation. Many small-system users — particularly software developers — were already familiar with partitioning, if not overly fond of it, from their experience with CP/M. Development time was also a major concern, and the code needed to develop a partitioning scheme would be minimal compared with the code required to manage a hierarchical file system. Such a scheme would also take less time to implement.

However, partitioning had two inherent disadvantages. First, its functionality would decrease as storage capacity increased, and even in 1982, Microsoft was anticipating substantial growth in the storage capacity of disk-based media. Second, partitioning depended on the physical device. If the size of the disk changed, either the number or the size of the partitions must also be changed in the code for both the operating system and the application programs. For Microsoft, with its commitment to hardware independence, partitioning would have represented a step in the wrong direction.

A hierarchical file structure, on the other hand, could be independent of the physical device. A disk could be partitioned logically, rather than physically. And because these partitions (directories) were controlled by the user, they were open-ended and enabled the individual to determine the best way of organizing a disk.

Ultimately, it was a hierarchical file system that found its way into MS-DOS 2.0 and eventually convinced everyone that it was, indeed, the better and more flexible solution to the problem of supporting a fixed disk. The file system was logically consistent with the XENIX file structure, yet physically consistent with the file access incorporated in versions 1.x, and was based on a root, or main, directory under which the user could create a system of subdirectories and sub-subdirectories to hold files. Each file in the system was identified by the directory path leading to it, and the number of subdirectories was limited only by the length of the pathname, which could not exceed 64 characters.

In this file structure, all the subdirectories and the filename in a path were separated from one another by backslash characters, which represented the only anomaly in the XENIX/MS-DOS system of hierarchical files. XENIX used a forward slash as a separator, but versions 1.x of MS-DOS, borrowing from the tradition of DEC operating systems, already used the forward slash for switches in the command line, so Microsoft, at IBM's request, decided to use the backslash as the separator instead. Although the backslash

character created no practical problems, except on keyboards that lacked a backslash, this decision did introduce inconsistency between MS-DOS and existing UNIX-like operating systems. And although Microsoft solved the keyboard problem by enabling the user to change the switch character from a slash to a hyphen, the solution itself created compatibility problems for people who wished to exchange batch files.

Another major change in the file-management system was related to the new directory structure: In order to fully exploit a hierarchical file system, Microsoft had to add a new way of calling file services.

Versions 1.x of MS-DOS used CP/M-like structures called file control blocks, or FCBs, to maintain compatibility with older CP/M-80 programs. The FCBs contained all pertinent information about the size and location of a file but did not allow the user to specify a file in a different directory. Therefore, version 2.0 of MS-DOS needed the added ability to access files by means of handles, or descriptors, that could operate across directory lines.

In this added step toward logical device independence, MS-DOS returned a handle whenever an MS-DOS program opened a file. All further interaction with the file involved only this handle. MS-DOS made all necessary adjustments to an internal structure — different from an FCB — so that the program never had to deal directly with information about the file's location in memory. Furthermore, even if future versions of MS-DOS were to change the structure of the internal control units, program code would not need to be rewritten — the file handle would be the only referent needed, and this would not change.

Putting the internal control units under the supervision of MS-DOS and substituting handles for FCBs also made it possible for MS-DOS to redirect a program's input and output. A system function was provided that enabled MS-DOS to divert the reads or writes directed to one handle to the file or device assigned to another handle. This capability was used by COMMAND.COM to allow output from a file to be redirected to a device, such as a printer, or to be piped to another program. It also allowed system cleanup on program terminations.

Installable Device Drivers

At the time Microsoft began developing version 2.0 of MS-DOS, the company also realized that many third-party peripheral devices were not working well with one another. Each manufacturer had its own way of hooking its hardware into MS-DOS and if two third-party devices were plugged into a computer at the same time, they would often conflict or fail.

One of the hallmarks of IBM's approach to the PC was open architecture, meaning that users could simply slide new cards into the computer whenever new input/output devices, such as fixed disks or printers, were added to the system. Unfortunately, version 1.0 of MS-DOS did not have a corresponding open architecture built into it — the BIOS

contained all the code that permitted the operating system to run the hardware. If independent hardware manufacturers wanted to develop equipment for use with a computer manufacturer's operating system, they would have to either completely rewrite the device drivers or write a complicated utility to read the existing drivers, alter them, add the code to support the new device, and produce a working set of drivers. If the user installed more than one device, these patches would often conflict with one another. Furthermore, they would have to be revised each time the computer manufacturer updated its version of MS-DOS.

By the time work began on version 2.0, the MS-DOS team knew that the ability to install any device driver at run time was vital. They implemented installable device drivers by making the drivers more modular. Like the FAT, IO.SYS (IBMBIO.COM in PC-DOS) became, in effect, a linked list—this time, of device drivers—that could be expanded through commands in the CONFIG.SYS file on the system boot disk. Manufacturers could now write a device driver that the user could install at run time by including it in the CONFIG.SYS file. MS-DOS could then add the device driver to the linked list.

By extension, this ability to install device drivers also added the ability to supersede a previously installed driver—for example, the ANSI.SYS console driver that supports the ANSI standard escape codes for cursor positioning and screen control.

Print Spooling

At IBM's request, version 2.0 of MS-DOS also possessed the undocumented ability to perform rudimentary background processing—an interim solution to a growing awareness of the potentials of multitasking.

Background print spooling was sufficient to meet the needs of most people in most situations, so the print spooler, PRINT.COM, was designed to run whenever MS-DOS had nothing else to do. When the parent application became active, PRINT.COM would be interrupted until the next lull. This type of background processing, though both limited and extremely complex, was exploited by a number of applications, such as SideKick.

Loose Ends and a New MS-DOS

Hierarchical files, installable device drivers, and print spooling were the major design decisions in version 2.0. But there were dozens of smaller changes, too.

For example, with the fixed disk it was necessary to modify the code for automatic logging of disks. This modification meant that MS-DOS had to access the disk more often, and file access became much slower as a result. In trying to find a solution to this problem, Chris Peters reasoned that, if MS-DOS had just checked the disk, there was some minimum time

Two members of the IBM line of personal computers for which versions 1 and 2 of MS-DOS were developed. On the left, the original IBM PC (version 1.0 of MS-DOS); on the right, the IBM PC/XT (version 2.0).

a user would need to physically change disks. If that minimum time had not elapsed, the current disk information in RAM — whether for a fixed disk or a floppy — was probably still good.

Peters found that the fastest anyone could physically change disks, even if the disks were damaged in the process, was about two seconds. Reasoning from this observation, he had MS-DOS check to see how much time had gone by since the last disk access. If less than two seconds had elapsed, he had MS-DOS assume that a new disk had not been inserted and that the disk information in RAM was still valid. With this little trick, the speed of file handling in MS-DOS version 2.0 increased considerably.

Version 2.0 was released in March 1983, the product of a surprisingly small team of six developers, including Peters, Mani Ulloa, and Nancy Panners in addition to Allen, Zbikowski, and Reynolds. Despite its complex new features, version 2.0 was only 24 KB of code. Though it maintained its compatibility with versions 1.x, it was in reality a vastly different operating system. Within six months of its release, version 2.0 gained widespread public acceptance. In addition, popular application programs such as Lotus 1-2-3 took advantage of the features of this new version of MS-DOS and thus helped secure its future as the industry standard for 8086 processors.

Versions 2.1 and 2.25

The world into which version 2.0 of MS-DOS emerged was considerably different from the one in which version 1.0 made its debut. When IBM released its original PC, the business market for microcomputers was as yet undefined — if not in scope, at least in terms of who and what would dominate the field. A year and a half later, when the PC/XT came on the scene, the market was much better known. It had, in fact, been heavily influenced by IBM itself. There were still many MS-DOS machines, such as the Tandy 2000 and the Hewlett Packard HP150, that were hardware incompatible with the IBM, but manufacturers of new computers knew that IBM was a force to consider and many chose to compete with the IBM PC by emulating it. Software developers, too, had gained an understanding of business computing and were confident they could position their software accurately in the enormous MS-DOS market.

In such an environment, concerns about the existing base of CP/M software faded as developers focused their attention on the fast-growing business market and MS-DOS quickly secured its position as an industry standard. Now, with the obstacles to MS-DOS diminished, Microsoft found itself with a new concern: maintaining the standard it had created. Henceforth, MS-DOS had to be many things to many people. IBM had requirements; other OEMs had requirements. And sometimes these requirements conflicted.

Hardware Developers

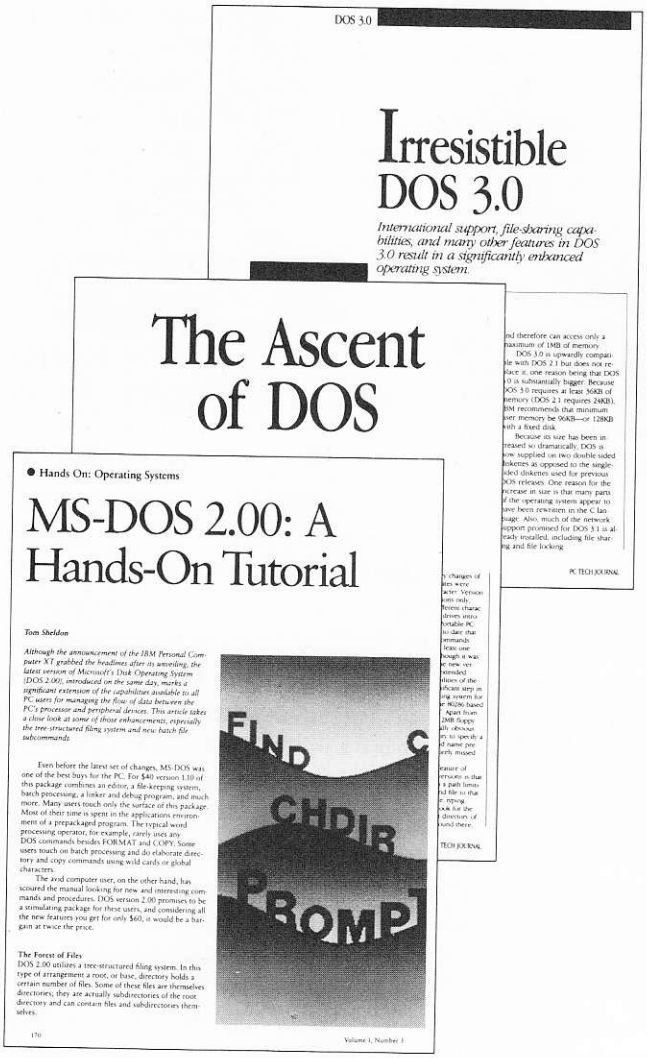
When version 2.0 was released, IBM was already planning to introduce its PCjr. The PCjr would have the ability to run programs from ROM cartridges and, in addition to using half-height 5¼-inch drives, would employ a slightly different disk-controller architecture. Because of these differences from the standard PC line, IBM's immediate concern was for a version 2.1 of MS-DOS modified for the new machine.

For the longer term, IBM was also planning a faster, more powerful PC with a 20-megabyte fixed disk. This prospect meant Microsoft needed to look again at its file-management system, because the larger storage capacity of the 20-megabyte disk stretched the size limitations for the file allocation table as it worked in version 2.0.

However, IBM's primary interest for the next major release of MS-DOS was networking. Microsoft would have preferred to pursue multitasking as the next stage in the development of MS-DOS, but IBM was already developing its IBM PC Network Adapter, a plug-in card with an 80188 chip to handle communications. So as soon as version 2.0 was released, the MS-DOS team, again headed by Zbikowski and Reynolds, began work on a networking version (3.0) of the operating system.

Meanwhile . . .

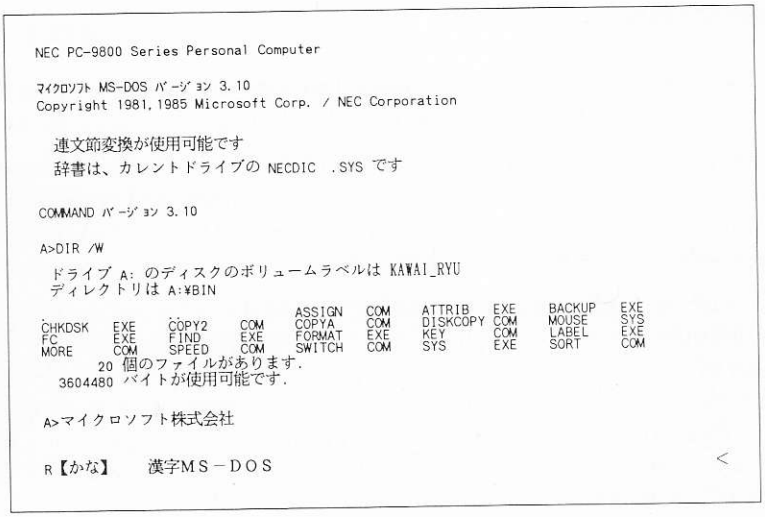
The international market for MS-DOS was not significant in the first few years after the release of the IBM PC and version 1.0 of MS-DOS. IBM did not, at first, ship its Personal Computer to Europe, so Microsoft was on its own there in promoting MS-DOS. In 1982, the company gained a significant advantage over CP/M-86 in Europe by concluding an agreement with Victor, a software company that was very successful in Europe and had already licensed CP/M-86. Working closely with Victor, Microsoft provided special development support for its graphics adaptors and eventually convinced the company to offer its products only on MS-DOS. In Japan, the most popular computers were Z80 machines, and given the country's huge installed base of 8-bit machines, 16-bit computers were not taking hold. Mitsubishi, however, offered a 16-bit computer. Although CP/M-86 was Mitsubishi's original choice for an operating system, Microsoft helped get Multiplan and FORTRAN running on the CP/M-86 system, and eventually won the manufacturer's support for MS-DOS.



A sample of the reviews that appeared with each new version of MS-DOS.

In the software arena, by the time development was underway on the 2.x releases of MS-DOS, Microsoft's other customers were becoming more vocal about their own needs. Several wanted a networking capability, adding weight to IBM's request, but a more urgent need for many — a need *not* shared by IBM at the time — was support for international products. Specifically, these manufacturers needed a version of MS-DOS that could be sold in other countries — a version of MS-DOS that could display messages in other languages and adapt to country-specific conventions, such as date and time formats.

Microsoft, too, wanted to internationalize MS-DOS, so the MS-DOS team, while modifying the operating system to support the PCjr, also added functions and a COUNTRY command that allowed users to set the date and time formats and other country-dependent variables in the CONFIG.SYS file.



```
NEC PC-9800 Series Personal Computer

ﾏｲｸﾛｿﾌﾄ MS-DOS ﾊﾞｰｼﾞｮﾝ 3.10
Copyright 1981,1985 Microsoft Corp. / NEC Corporation

  連文節変換が使用可能です
  辞書は、カレントドライブの NECDIC  .SYS です

COMMAND ﾊﾞｰｼﾞｮﾝ 3.10

A>DIR /W
 ドライブ A: のディスクのボリュームラベルは KAWAI_RYU
 ディレクトリは A:¥BIN

                        ASSIGN  COM   ATTRIB  EXE   BACKUP  EXE
CHKDSK  EXE   COPY2  COM   COPYA   COM   DISKCOPY COM   MOUSE   SYS
FC      EXE   FIND   EXE   FORMAT  EXE   KEY     COM   LABEL   EXE
MORE    COM   SPEED  COM   SWITCH  COM   SYS     EXE   SORT    COM
      20 個のファイルがあります.
  3604480 バイトが使用可能です.

A>マイクロソフト株式会社

R【かな】    漢字ＭＳ－ＤＯＳ                                    <
```

A Kanji screen with the MS-DOS copyright message.

At about the same time, another international requirement appeared. The Japanese market for MS-DOS was growing, and the question of supporting 7000 Kanji characters (ideograms) arose. The difficulty with Kanji is that it requires dual-byte characters. For English and most European character sets, one byte corresponds to one character. Japanese characters, however, sometimes use one byte, sometimes two. This variability creates problems in parsing, and as a result MS-DOS had to be modified to parse a string from the beginning, rather than back up one character at a time.

This support for individual country formats and Kanji appeared in version 2.01 of MS-DOS. IBM did not want this version, so support for the PCjr, developed by Zbikowski, Reynolds, Ulloa, and Eric Evans, appeared separately in version 2.1, which went only to IBM and did not include the modifications for international MS-DOS.

Different customers, different versions

As early as version 1.25, Microsoft faced the problem of trying to satisfy those OEM customers that wanted to have the same version of MS-DOS as IBM. Some, such as COMPAQ, were in the business of selling 100-percent compatibility with IBM. For them, any difference between their version of the operating system and IBM's introduced the possibility of incompatibility. Satisfying these requests was difficult, however, and it was not until version 3.1 that Microsoft was able to supply a system that other OEMs agreed was identical with IBM's.

Before then, to satisfy the OEM customers, Microsoft combined versions 2.1 and 2.01 to create version 2.11. Although IBM did not accept this because of the internationalization code, version 2.11 became the standard version for all non-IBM customers running any form of MS-DOS in the 2.x series. Version 2.11 was sold worldwide and translated into about 10 different languages. Two other intermediate versions provided support for Hangeul (the Korean character set) and Chinese Kanji.

Software Concerns

After the release of version 2.0, Microsoft also gained an appreciation of the importance — and difficulty — of supporting the people who were developing software for MS-DOS.

Software developers worried about downward compatibility. They also worried about upward compatibility. But despite these concerns, they sometimes used programming practices that could guarantee neither. When this happened and the resulting programs were successful, it was up to Microsoft to ensure compatibility.

For example, because the information about the internals of the BIOS and the ROM interface had been published, software developers could, and often did, work directly with the hardware in order to get more speed. This meant sidestepping the operating system for some operations. However, by choosing to work at the lower levels, these developers lost the protection provided by the operating system against hardware changes. Thus, when low-level changes were made in the hardware, their programs either did not work or did not run cooperatively with other applications.

Another software problem was the continuing need for compatibility with CP/M. For example, in CP/M, programmers would call a fixed address in low memory in order to request a function; in MS-DOS, they would request operating-system services by executing a software interrupt. To support older software, the first version of MS-DOS allowed a program to request functions by either method. One of the CP/M-based programs supported in this fashion was the very popular WordStar. Since Microsoft could not make changes in MS-DOS that would make it impossible to run such a widely used program, each new version of MS-DOS had to continue supporting CP/M-style calls.

A more pervasive CP/M-related issue was the use of FCB-style calls for file and record management. The version 1.x releases of MS-DOS had used FCB-style calls exclusively, as had CP/M. Version 2.0 introduced the more efficient and flexible handle calls, but Microsoft could not simply abolish the old FCB-style calls, because so many popular programs used them. In fact, some of Microsoft's own languages used them. So, MS-DOS had to support both types of calls in the version 2.x series. To encourage the use of the new handle calls, however, Microsoft made it easy for MS-DOS users to upgrade to version 2.0. In addition, the company convinced IBM to require version 2.0 for the PC/XT and also encouraged software developers to require 2.0 for their applications.

At first, both software developers and OEM customers were reluctant to require 2.0 because they were concerned about problems with the installed user base of 1.0 systems — requiring version 2.0 meant supporting both sets of calls. Applications also needed to be able to detect which version of the operating system the user was running. For versions 1.x, the programs would have to use FCB calls; for versions 2.x, they would use the file handles to exploit the flexibility of MS-DOS more fully.

All told, it was an awkward period of transition, but by the time Microsoft began work on version 3.0 and the support for IBM's upcoming 20-megabyte fixed disk, it had become apparent that the change had been in everyone's best interest.

Version 3

The types of issues that began to emerge as Microsoft worked toward version 3.0, MS-DOS for networks, exaggerated the problems of compatibility that had been encountered before.

First, networking, with or without a multitasking capability, requires a level of cooperation and compatibility among programs that had never been an issue in earlier versions of MS-DOS. As described by Mark Zbikowski, one of the principals involved in the project, "there was a very long period of time between 2.1 and 3.0—almost a year and a half. During that time, we believed we understood all the problems involved in making DOS a networking product. [But] as time progressed, we realized that we didn't fully understand it, either from a compatibility standpoint or from an operating-system standpoint. We knew very well how it [DOS] ran in a single-tasking environment, but we started going to this new environment and found places where it came up short."

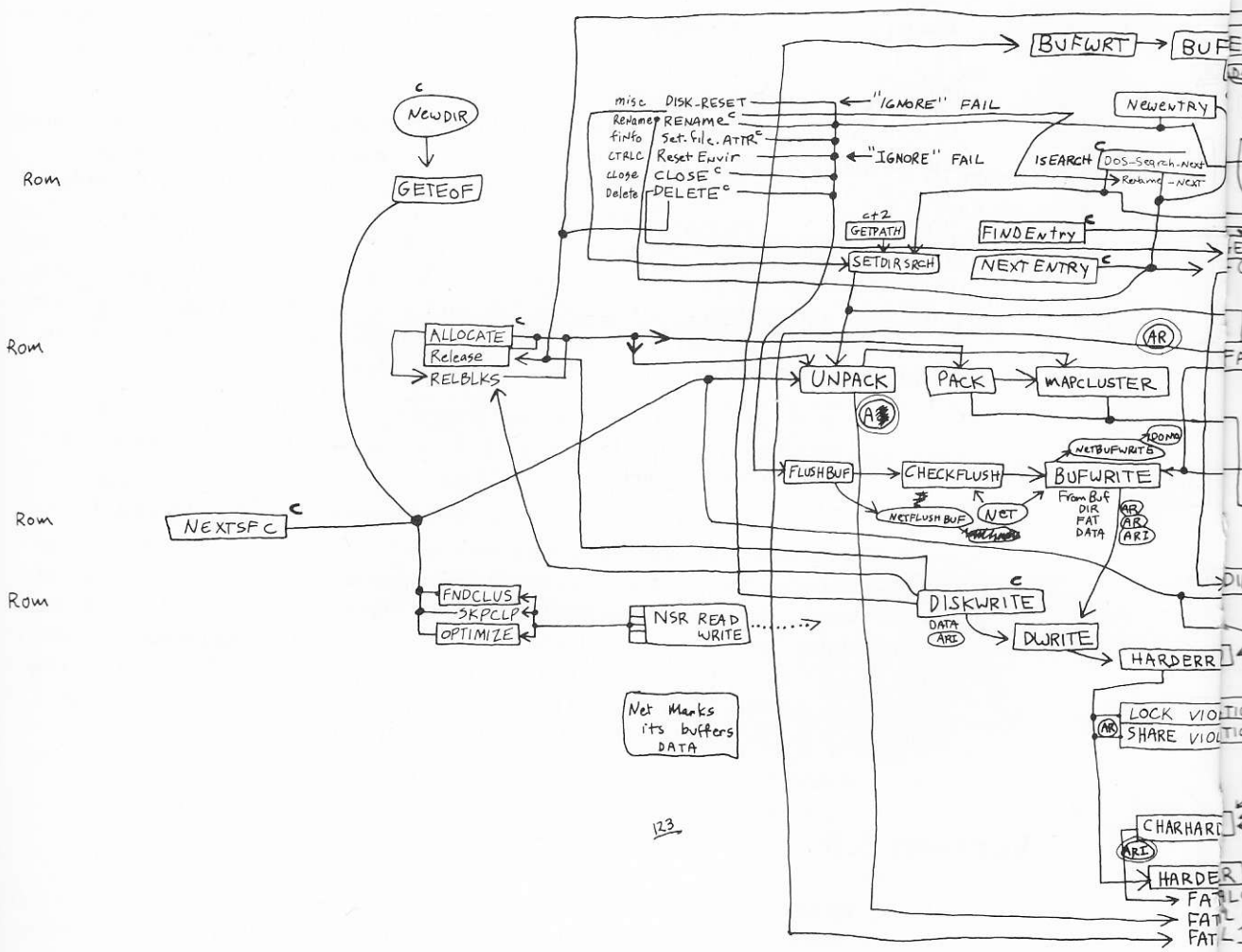
In fact, the great variability in programs and programming approaches that MS-DOS supported eventually proved to be one of the biggest obstacles to the development of a sophisticated networking system and, in the longer term, to the addition of true multitasking.

Further, by the time Microsoft began work on version 3.0, the programming style of the MS-DOS team had changed considerably. The team was still small, with a core group of just five people: Zbikowski, Reynolds, Peters, Evans, and Mark Bebic. But the concerns for maintainability that had dominated programming in larger systems had percolated down to the MS-DOS environment. Now, the desire to use tricks to optimize for speed had to be tempered by the need for clarity and maintainability, and the small package of tightly written code that was the early MS-DOS had to be sacrificed for the same reasons.

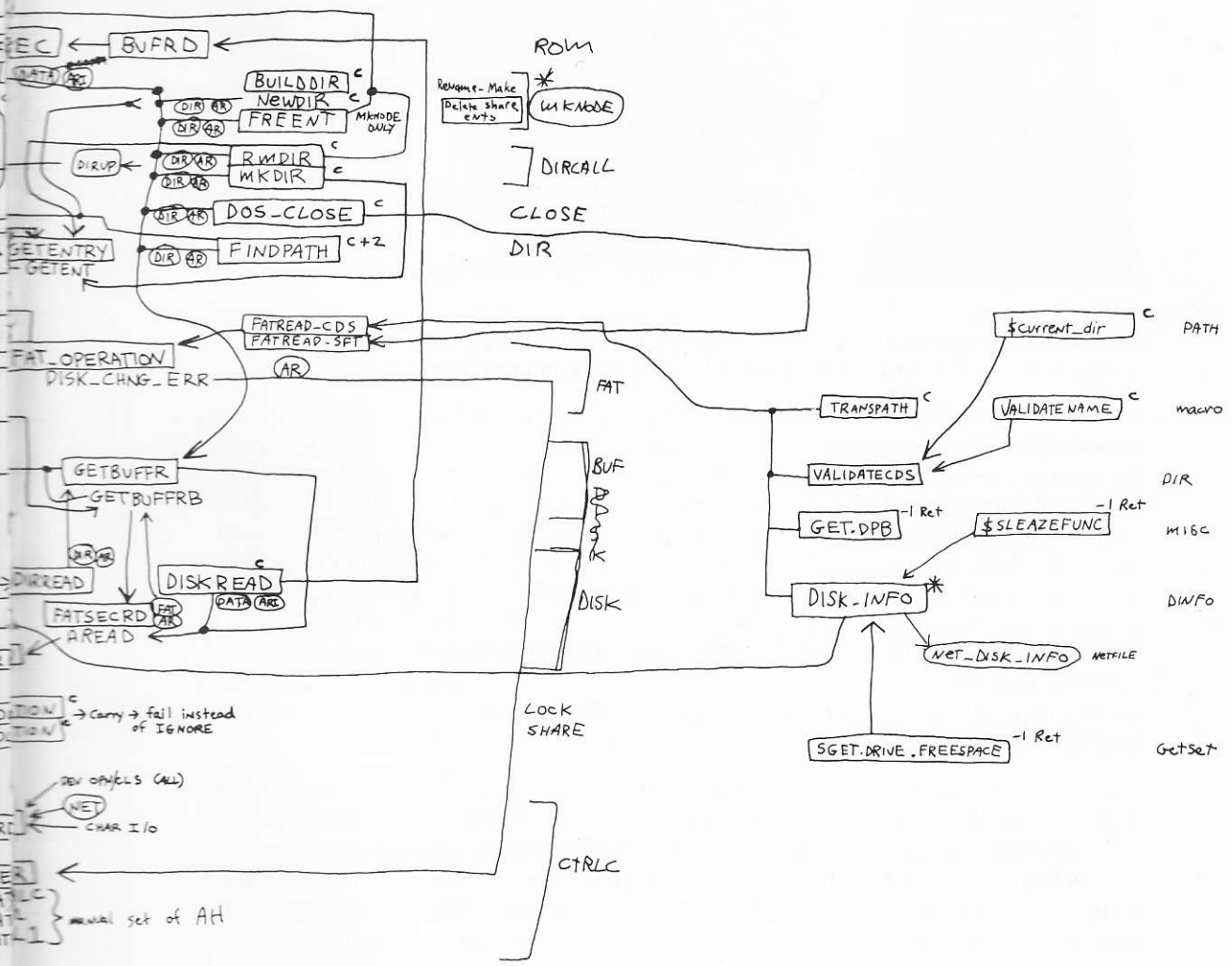
Version 3.0

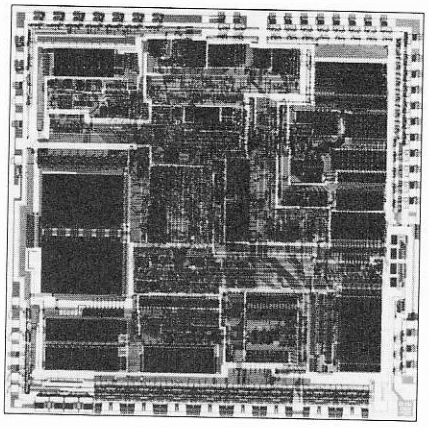
All told, the work on version 3.0 of MS-DOS proved to be long and difficult. For a year and a half, Microsoft grappled with problems of software incompatibility, remote file management, and logical device independence at the network level. Even so, when IBM was ready to announce its new Personal Computer AT, the network software for MS-DOS was not quite ready, so in August 1984, Microsoft released version 3.0 to IBM without network software.

Version 3.0 supported the AT's larger fixed disk, its new CMOS clock, and its high-capacity 1.2-megabyte floppy disks. It also provided the same international support included earlier in versions 2.01 and 2.11. These features were made available to Microsoft's other OEM customers as version 3.05.



Aaron Reynolds's diagram of version 3.0's network support, sketched out to enable him to add the fail option to Interrupt 24 and find all places where existing parts of MS-DOS were affected. Even after networking had become a reality, Reynolds kept this diagram pinned to his office wall simply because "it was so much work to put together."


BUFRD
BUILDDIR
NEWDIR
FREENT
MKNODE ONLY
RMDIR
MKDIR
DOS_CLOSE
FINDPATH
C+2
DIRUP
GETENTRY
GETENT
ROM
Rename-Make
Delete share ents
UNMODE
DIRCALL
CLOSE
DIR
FATREAD_CDS
FATREAD_SFT
FAT_OPERATION
DISK_CHNG_ERR
AR
FAT
GETBUFFR
GETBUFFRB
BUF
DISK
DISKREAD
FATSECRD
AREAD
DIRREAD
Carry → fail instead of IGNORE
LOCK
SHARE
NEW OPEN/CLS CALL
NET
CHAR I/O
CTRLC
manual set of AH
$current_dir
PATH
TRANSPATH
VALIDATENAME
macro
VALIDATECDS
DIR
GET_DPB
-1 Ret
$SLEAZEFUNC
MISC
DISK_INFO
DINFO
NET_DISK_INFO
NETFILE
SGET_DRIVE_FREESPACE
GetSet



The Intel 80286 microprocessor, the chip at the heart of the IBM PC/AT, which is shown beside it. Version 3.0 of MS-DOS, developed for this machine, offered support for networks and the PC/AT's 1.2-megabyte floppy disk drive and built-in CMOS clock.

But version 3.0 was not a simple extension of version 2.0. In laying the foundation for networking, the MS-DOS team had completely redesigned and rewritten the DOS kernel.

Different as it was from version 1.0, version 2.0 had been built on top of the same structure. For example, whereas file requests in MS-DOS 1.0 used FCBs, requests in version 2.0 used file handles. However, the version 2.0 handle calls would simply parse the pathname and then use the underlying FCB calls in the same way as version 1.0. The redirected input and output in version 2.0 further complicated the file-system requests. When a program used one of the CP/M-compatible calls for character input or output, MS-DOS 2.0 first opened a handle and then turned it back into an FCB call at a lower level. Version 3.0 eliminated this redundancy by eliminating the old FCB input/output code of versions 1 and 2, replacing it with a standard set of I/O calls that could be called directly by both FCB calls and handle calls. The look-alike calls for CP/M-compatible character I/O were included as part of the set of handle calls. As a result of this restructuring, these calls were distinctly faster in version 3.0 than in version 2.0.

More important than the elimination of inefficiencies, however, was the fact that this new structure made it easier to handle network requests under the ISO Open System Interconnect model Microsoft was using for networking. The ISO model describes a number of protocol layers, ranging from the application-to-application interface at the top level down to the physical link — plugging into the network — at the lowest level. In the middle is the transport layer, which manages the actual transfer of data. The layers above the transport layer belong to the realm of the operating system; the layers below the transport layer are traditionally the domain of the network software or hardware.

On the IBM PC network, the transport layer and the server functions were handled by IBM's Network Adapter card and the task of MS-DOS was to support this hardware. For its other OEM customers, however, Microsoft needed to supply both the transport and the server functions as software. Although version 3.0 did not provide this general-purpose networking software, it did provide the basic support for IBM's networking hardware.

The support for IBM consisted of redirector and sharer software. MS-DOS used an approach to networking in which remote requests were routed by a redirector that was able

to interact with the transport layer of the network. The transport layer was composed of the device drivers that could reliably transfer data from one part of the network to another. Just before a call was sent to the newly designed low-level file I/O code, the operating system determined whether the call was local or remote. A local call would be allowed to fall through to the local file I/O code; a remote call would be passed to the redirector which, working with the operating system, would make the resources on a remote machine appear as if they were local.

Version 3.1

Both the redirector and the sharer interfaces for IBM's Network Adapter card were in place in version 3.0 when it was delivered to IBM, but the redirector itself wasn't ready. Version 3.1, completed by Zbikowski and Reynolds and released three months later, completed this network support and made it available in the form of Microsoft Networks for use on non-IBM network cards.

Microsoft Networks was built on the concept of "services" and "consumers." Services were provided by a file server, which was part of the Networks application and ran on a computer dedicated to the task. Consumers were programs on various network machines. Requests for information were passed at a high level to the file server; it was then the responsibility of the file server to determine where to find the information on the disk. The requesting programs—the consumers—did not need any knowledge of the remote machine, not even what type of file system it had.

This ability to pass a high-level request to a remote server without having to know the details of the server's file structure allowed another level of generalization of the system. In MS-DOS 3.1, different types of file systems could be accessed on the same network. It was possible, for example, to access a XENIX machine across the network from an MS-DOS machine and to read data from XENIX files.

Microsoft Networks was designed to be hardware independent. Yet the variability of the classes of programs that would be using its structures was a major problem in developing a networking system that would be transparent to the user. In evaluating this variability, Microsoft identified three types of programs:

- First were the MS-DOS-compatible programs. These used only the documented software-interrupt method of requesting services from the operating system and would run on any MS-DOS machine without problems.
- Second were the MS-DOS-based programs. These would run on IBM-compatible computers but not necessarily on all MS-DOS machines.
- Third were the programs that used undocumented features of MS-DOS or that addressed the hardware directly. These programs tended to have the best performance but were also the most difficult to support.

Of these, Microsoft officially encouraged the writing of MS-DOS-compatible programs for use on the network.

Network concerns

The file-access module was changed in version 3.0 to simplify file management on the network, but this did not solve all the problems. For instance, MS-DOS still needed to handle FCB requests from programs that used them, but many programs would open an FCB and never close it. One of the functions of the server was to keep track of all open files on the network, and it ran into difficulties when an FCB was opened 50 or 100 times and never closed. To solve this problem, Microsoft introduced an FCB cache in version 3.1 that allowed only four FCBs to be open at any one time. If a fifth FCB was opened, the least recently used one was closed automatically and released. In addition, an FCBS command was added in the CONFIG.SYS file to allow the user or network manager to change the maximum number of FCBs that could be open at any one time and to protect some of the FCBs from automatic closure.

In general, the logical device independence that had been a goal of MS-DOS acquired new meaning—and generated new problems—with networking. One problem concerned printers on the network. Commonly, networks are used to allow several people to share a printer. The network could easily accommodate a program that would open the printer, write to it, and close it again. Some programs, however, would try to use the direct IBM BIOS interface to access the printer. To handle this situation, Microsoft's designers had to develop a way for MS-DOS to intercept these BIOS requests and filter out the ones the server could not handle. Once this was accomplished, version 3.1 was able to handle most types of printer output on the network in a transparent manner.

Version 3.2

In January 1986, Microsoft released another revision of MS-DOS, version 3.2, which supported 3½-inch floppy disks. Version 3.2 also moved the formatting function for a device out of the FORMAT utility routine and into the device driver, eliminating the need for a special hardware-dependent program in addition to the device driver. It included a sample installable-block-device driver and, finally, benefited the users and manufacturers of IBM-compatible computers by including major rewrites of the MS-DOS utilities to increase compatibility with those of IBM.

The Future

Since its appearance in 1981, MS-DOS has taken and held an enviable position in the microcomputer environment. Not only has it "taught" millions of personal computers "how to think," it has taught equal millions of people how to use computers. Many highly sophisticated computer users can trace their first encounter with these machines to the original IBM PC and version 1.0 of MS-DOS. The MS-DOS command interface is the one with which they are comfortable and it is the MS-DOS file structure that, in one way or another, they wander through with familiarity.

Microsoft has stated its commitment to ensuring that, for the foreseeable future, MS-DOS will continue to evolve and grow, changing as it has done in the past to satisfy the needs of its millions of users. In the long term, MS-DOS, the product of a surprisingly small group of gifted people, will undoubtedly remain the industry standard for as long as 8086-based (and to some extent, 80286-based) microcomputers exist in the business world. The story of MS-DOS will, of course, remain even longer. For this operating system has earned its place in microcomputing history.

JoAnne Woodcock