April 1992

Personal Systems

OS/2 WORKPLACE SHELL

IBM Personal Systems Technical Solutions

IBM®

$12.00

IBM Personal Systems Technical Solutions is published quarterly by IBM United States technical support center, International Business Machines Corporation, Roanoke, Texas, U.S.A.

Editor	Libby Boyd
Communications	Elisa Davis
Cover Design	Corporate Graphics
Illustrator	Bill Carr
Publication Services	The TDA Group
Typesetting & Diagrams	Genesis Publications
Manager	Bill Hawkins

To correspond with *IBM Personal Systems Technical Solutions*, please write the editor at:

IBM Corporation
Internal Zip 40-A2-04
One East Kirkwood Blvd.
Roanoke, TX 76299-0015

To subscribe to this publication, call 1-800-551-2832. IBM employees should order through SLSS using order form number GBOF-1229.

Printed on recycled paper

Contents

Comparing Architectures: Micro Channel and EISA

Chet Heath
IBM Corporation
Boca Raton, Florida

This is part one of a two-part article that discusses and clarifies assertions and claims made when comparing IBM's Micro Channel® architecture with the Extended Industry Standard Architecture (EISA) specifications. The article is not intended to reflect on any specific product or manufacturer. Part two will discuss the restrictions on EISA's DMA and bus arbitration that are imposed by PC/XT/AT compatibility, and issues relating to the pitfalls of EISA's design and strategy compared to the design and strategy of Micro Channel architecture.

At every step in the evolution of computer systems, there have been periods in which the available technology or business pressures have brought about sudden advancement. We are now in such a generational state. Microcomputer systems are evolving from supporting single users to serving multiple concurrent users and tasks. Also, new technologies, such as multimedia, are appearing. These capabilities are the result of affordable, fast, reliable components that have brought about the migration of sophisticated computer system designs from the mainframe and minicomputer worlds to microcomputers.

In the design of a microcomputer system, the element that ties all the components together – the path that connects the computer to its Input/Output (I/O) devices – is called a *bus*. When the procedures that control the flow of information are also defined, that element is called a *channel*.

Because of its central position in the computer system, a bus or channel significantly influences the options of both system designers and users. A bus can control designers' latitudes for implementing processor and I/O technology alternatives in a system.

A lack of flexibility in this element can effectively limit the useful life of, or the range of applications that can run on, a family of system designs that use a single architecture. A system's interoperability with other equipment and its ability to be serviced are also affected.

Because of the above business consequences of the technical decision to choose one bus or channel over another, the microcomputer industry is studying the directions of several advanced bus architectures.

The first two serious candidates for advanced bus architectures appeared in 1987. Although they had been under development separately for several years, they were announced almost simultaneously. In both cases, their announcement was based on the availability of affordable component technology.

The first of these two bus architectures, proposed by Apple® Computer, was a version of the NuBus standard. Apple's NuBus was extended to include 32-bit addressability, automatic configuration, interrupt sharing, improved power distribution to the connector, and a higher speed data transfer mode that could accommodate the information traffic generated by advanced I/O adapters. The Apple NuBus could also support *bus master* adapters, which relieve the workload placed on the processor during multiple high-speed I/O operations.

Apple's NuBus was a radical departure from the bus in previous microcomputer products, in that NuBus did not support adapter cards of previous generations. The development of NuBus was apparently driven by Apple's ensuing announcements of multitasking, multimedia computer systems.

One month after the Apple NuBus announcement, IBM introduced the Micro Channel architecture. Micro Channel offered the same "multi" environment functions listed for NuBus. Micro Channel, however, was developed from an entirely different perspective, which resulted in a different design specification. The Micro Channel interface retained several features of earlier PCs – separate I/O and memory selection, third-party Direct Memory Access (DMA), and diagnostic error detection – which were not included in the multitasking menus followed by NuBus. These features were required to maintain compatibility with existing software that was developed for the IBM Personal Computer and its architectural derivatives. Like NuBus, Micro Channel was also a radical departure from the bus of the previous IBM PC/XT™/AT® computer generation. Micro Channel was defined for a much broader range of computer systems and operating systems. As the strategic parallel bus for IBM's microcomputer products, Micro Channel made mainframe architecture compatible with, and cost-effective for, personal computer operating systems and applications. Vendors other than IBM have also used Micro Channel architecture in computer systems that range from desktop microcomputers to small mainframe systems.

In October 1988, about 18 months after the announcement of Micro Channel, a consortium of microcomputer vendors, who are competitors to IBM, announced the Extended Industry Standard Architecture (EISA) bus. They also coined the term Industry Standard Architecture (ISA) to cover PC/XT/AT systems and cards derived from the original IBM Personal Computer. EISA defined the same list of fundamental functions as NuBus and Micro Channel, and incorporated the same definitions used in the Micro Channel specification for automatic configuration and level-sensitive interrupt sharing. The EISA announcement confirmed the need for an advanced bus architecture, while also retaining the capability of installing adapter cards from the previous generation of PC/XT/AT systems. The first EISA products appeared a year later, primarily as Intel®-based desktop systems and network servers.

While any comparison between Micro Channel and the Apple NuBus would quickly point out their vast differences in processor architecture, operating systems and applications, a debate did evolve that compared the capabilities of Micro Channel with those of EISA and ISA. Fueling that debate was the tremendous impact that the Micro Channel architecture would have on the large capital investment already made by computer manufacturers, as well as on related activities such as the investment in training and long-term component procurement contracts. The technical debate has divided into two camps: those who see the need for a clean break with the past, and those who want to evolve slowly or not at all. Much has been written and discussed about the ways – and schedules – for users to evolve to a new channel architecture platform. Some of the debate has been well-intended and well-informed; some has not.

This article is intended to examine some of the technical aspects of the EISA and Micro Channel architectures.

Evolution of Channel Architecture

The evolution of systems has followed a path of gradual evolution punctuated by sudden advancements in design. In a comparison of computer architectures, the choice that provides the best base for design is the one with the greatest evolutionary significance. Micro Channel can be seen as the most adaptively significant choice.

Here is a quick example. Advocates of both EISA and Micro Channel promote automatic configuration as a major usability feature. Automatic configuration requires the presence of logic on an adapter card. EISA cards have this built-in logic, so automatic configuration works for EISA cards. But it is impossible to add the required logic to the existing PC/XT/AT cards – whose designs may be several years old – that are now being installed in the EISA computer. So, while the EISA cards may automatically configure themselves, the remainder of the *whole* system cannot do the same.

The presence of a single PC/XT/AT adapter card in an EISA system requires:

1. Removing the covers
2. Identifying the card
3. Setting the switches on the card
4. Reinstalling the card to complete the configuration of the system
5. Replacing the covers

These are all manual operations. Steps 2, 3, and 4 must be done for every PC/XT/AT card in the system. No magic EISA hand sets the switches on PC/XT/AT cards; the user has that responsibility. In contrast, all Micro Channel adapter cards support automatic configuration of the system, with no requirement for human intervention, except perhaps to define an "override" through the keyboard.

When other advanced EISA functions, such as level-sensitive interrupts and high-speed data transfer, are implemented in EISA systems that contain PC, XT, or AT card designs, these other advanced EISA functions may also be either impaired or rendered inoperative. However, in true Micro Channel systems, the same advanced functions remain consistently operative, regardless of configuration.

In the same manner, there are differences between Micro Channel and EISA regarding interrupt sharing. The logic components often used in PC/XT/AT card designs will defeat the sharing of interrupt requests issued by either the EISA system or EISA adapter cards. There is even a potential that installation of PC/XT/AT card designs may lead to damage to the EISA system, its data, or the cards themselves. (This is explained in the box beginning on page 6.) In contrast, Micro Channel cards use a uniform design that does not interfere with other cards or adapters implemented on the system board.

Further, the higher speed transfer modes of EISA cards may be impeded by the capacitance that the older PC/XT/AT card designs add to the bus. No limit for capacitance (the electrical analog of elasticity) was specified for PC/XT/AT card designs. Many of these designs use older component technologies on their bus interface logic, which can limit the responsiveness of the system bus to rapidly changing signals. The older designs operated satisfactorily in older, slower systems, but can interfere with rapid transitions required for high-speed transfer. Short of a massive testing effort, there is no way to identify which PC/XT/AT card designs can import this problem into the EISA system. Micro Channel specifications set a limit on capacitance for every card and system design to forestall this problem.

In short, one can expect an entire EISA system to yield all the major EISA functions only if every card in the system has an EISA design. Even if an EISA system initially works with PC/XT/AT card designs installed, it may not work later if cards are added or the configuration is changed. The major EISA functions of automatic system configuration, interrupt sharing, and high-speed data transfer are impeded or excluded by PC/XT/AT card portability. Despite this, the portability of PC/XT/AT cards into EISA systems is a major selling point of EISA proponents.

Perhaps Micro Channel has a unique advantage in the above technical comparison. Because Micro Channel architects were able to start from a clean slate, they were not tied to the legacy – and the limitations – of PC/XT/AT designs. This is why Micro Channel cards and systems will typically work in any configuration. While a very limited set of exceptions to this general rule can be found[1], the exceptions are not exposures to sys-

[1] These exceptions were known to designers, and were considered acceptable for limited applications. For example, the IBM 3278 emulator board does not offer an alternate address, so it typically must fit into the first slot that Programmable Option Select (POS) configured in the system. This is considered acceptable, because it is rare to require two host attachments in the same desktop system. Later, the PS/2® configuration utility was modified to adjust the configuration for installation in any slot. As another example, some cards that are designed specifically for PS/2 machines carry ROMs that can be executed only by a specific microprocessor, so that when these cards are installed in RISC System/6000™ machines, they require special drivers. Also, RISC System/6000 machines have cards that are too big to be imported into all PS/2 systems. However, the dissimilar markets served by the RISC System/6000 and PS/2 systems often do not require the portability of these designs. It is acceptable for some card designs to be limited to a single market, as long as it is the designer's choice, and is not forced upon the design by a limitation in the architecture.

tem or data integrity; they are restrictions on configuration options or on portability between dissimilar system products.

Therefore, despite the lack of physical resemblance to its heritage, Micro Channel is the more evolutionary significant step, because it provides the most flexible and consistent base for design.

ISA is Not a True Standard

ISA has been described in three separate, inconsistent documents.

The first document that described the PC/XT/AT bus architecture was the *IBM RT PC® Technical Reference* manual. It discussed the timings of a few PC/XT/AT cards that could run in that machine. The RT PC supported only a subset of all the cards available, because most cards were designed for specific systems, or for specific models of systems, and were not portable to any other system design. For example, cards designed for the PC or XT computers might work in those systems, but not in the 6 or 8 MHz AT computers. The 6 MHz AT cards might not run on 8 MHz or faster systems. In fact, many so-called ISA systems that ran as fast as 16 MHz contained adapter cards that were never designed to work faster than 8 MHz. These systems were the least compatible of all so-called ISA solutions.

The second of the three documents came about when The Institute of Electrical and Electronics Engineers (IEEE®) attempted to bring order to chaos by publishing its P996 document. This document also attempted to specify the characteristics of adapter cards. However, it covered a disjoint subset of the 4,000 cards then available.

Finally, the EISA document described still another subset of what is expected from "industry-standard" architecture adapter cards.

The three documents do not agree with each other, because they were developed *after* the cards that they purport to specify. They also do not fully describe the general set of PC/XT/AT cards, because most of the cards were designed before the specification existed.

A "standard" that does not exactly and completely specify the elements that it is supposed to define is not a standard. It may be good marketing to repeatedly call ISA a standard, but that does not make it so.

True EISA cards, properly designed to the EISA specification, should fully comply with a standard by definition, but they comprise only a small percentage of all the EISA and ISA cards that are implied as portable into EISA systems. Given the scarcity of true EISA cards, the odds that all the adapter cards in a fully populated, eight-slot EISA system will fully comply with the EISA design standard are very small.

If the ISA "non-standard" is included in an EISA system, then the system becomes non-standard, even though the EISA specification is a standard. Micro Channel, on the other hand, does not accept any ISA cards. While logically evolutionary, Micro Channel is not physically evolutionary from ISA. The advantage, of course, in not being tied to ISA is that Micro Channel can truly be a consistent standard, whereas EISA cannot.

IBM is providing a laboratory to The Micro Channel Developers Association as a testing facility where developers of Micro Channel adapters can verify the interoperability of the advanced 32-bit streaming modes of the Micro Channel architecture specification. In addition, the National Software Testing Laboratory has evaluated many cards and systems for compatibility and interoperability (in single and multiple hardware and software configurations). Where is such a facility to verify that ISA cards meet *any* specification?

Software Compatibility Considerations

While many clone manufacturers once stated in their advertising that their products were "IBM-compatible," they now advertise "EISA-compatible" or "ISA-compatible" instead. This means that their products – both EISA and ISA – are compatible to an ISA non-standard (recall that EISA also incorporates this non-standard). However, many software applications state that they will run properly only on "IBM or 100% compatible" systems. This leaves software users in an unenviable position. When a software product does not run properly because a system does not meet the test of full IBM compatibility, the software vendor has no liability. On the other hand, the hardware manufacturer has no obligation to support the software. Who is left holding the bag? The user. The problem is easy to avoid: the user can simply ask "Is the system truly 100% IBM-compatible?"

The Bus is Only One Factor

A potential user's first question about a clone system that contains the Micro Channel interface is usually "How compatible is the clone with the IBM PS/2 family of computers?" IBM PS/2 computer compatibility requires more than just the addition of the Micro Channel interface to the design. The question of Micro Channel architecture compatibility involves issues such as bus timing and adherence to specification; these things, in turn, are often determined by which system chip set design is used in the clone system. (Many different system chip sets for the Micro Channel are now available from component vendors.)

Perhaps as important as the inclusion of the Micro Channel interface for PS/2 computer compatibility is the issue of Advanced BIOS (ABIOS) compatibility. EISA systems do not presently contain an ABIOS, which is required for full PS/2 compatibility. The issue, however, is disguised by all the attention to the presence of an alternate bus to the Micro Channel. Both full ABIOS compatibility and full compliance with the Micro Channel specification are required to build a satisfactory PS/2 Micro Channel-equivalent system that can run operating systems in an identical manner to an IBM PS/2 Micro Channel system.

ABIOS goes a step beyond the old Compatibility BIOS (CBIOS) that exists in PC/XT/AT systems. The old CBIOS defines character-by-character transfers that attend, sequentially, to only one device at a time. DOS and Windows® use the CBIOS, and they do their I/O sequentially. While CBIOS has been successfully cloned by other manufacturers, ABIOS is far more complex. ABIOS moves blocks, rather than characters, of information, and enables several I/O devices to be active simultaneously.

Satisfactory, compatible, legal copies of ABIOS are far more difficult to produce. This presents a significant challenge to production of a truly compatible clone of an IBM PS/2 system. The EISA world knows this. During the Bus Wars debate at the Fall 1990 COMDEX®, an EISA proponent stated that the difficulty in reproducing ABIOS was a major factor in the decision to adopt an alternative to Micro Channel architecture.

It is well known that the Micro Channel is not included in the design of an EISA system, but users are not alerted to the absence of ABIOS compatibility in EISA systems. This may be good marketing, but it is also bad science.

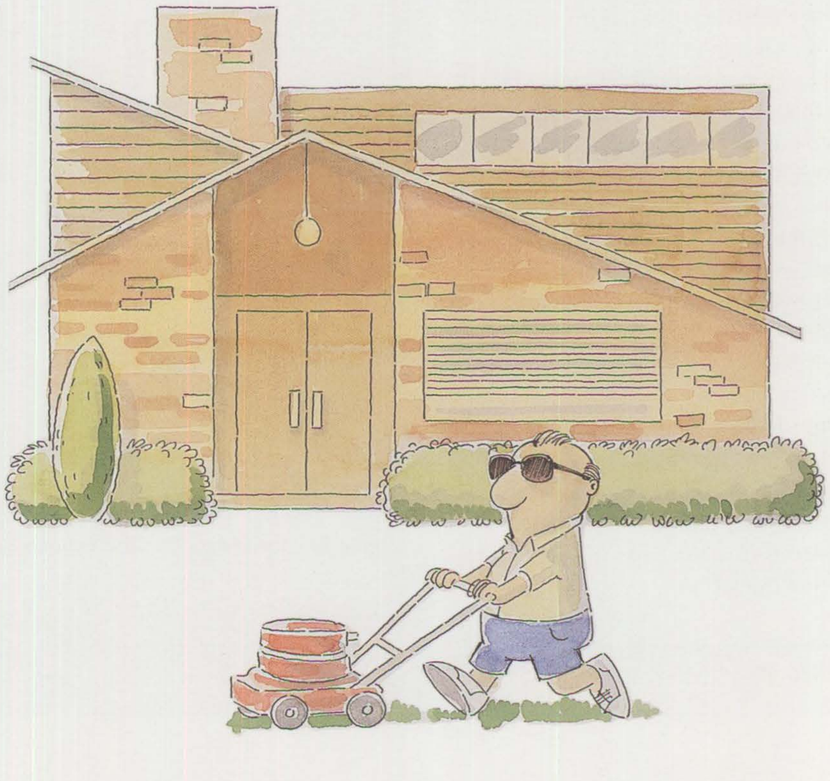
Is PC/XT/AT Card Portability an Advantage?

The argument that it is imperative to make PC/XT/AT cards portable to new EISA systems is based on the premise that users want to import their old cards into new systems. This premise has long since been refuted. According to the market research firm Dataquest®, only about 2.3 percent of all old adapter cards are ever ported to new systems. The reasons are simple and obvious: the old cards are as obsolete as the systems in which they were initially installed. The cards were depreciated as assets along with the systems they were in. Also, the old system unit itself would have no value without adapter cards for the floppy diskette drive, hard disk drive, display, and serial or parallel ports. In short, porting old adapter cards from one system to another is just not done.

Another favorite EISA argument is that PC/XT/AT card availability means that users can get the functional capabilities they need in EISA systems. With only a relatively few cards available for EISA, and with little incentive to develop cards for the handful of available EISA systems, EISA systems are highly dependent on the portability of the more abundant ISA cards to complete the desired configurations. So the reasoning is reduced to this: EISA is incomplete; EISA needs ISA portability to be complete; therefore, ISA portability into EISA is an advantage. This is circular reasoning that promotes a liability as an asset.

Based on this reasoning, users may have to spend money twice. Why? Users may first have to make new investments in old PC/XT/AT cards so that their EISA systems can initially operate, with AT function only. Then, later – if and when equivalent EISA function becomes available – users will have to discard their PC/XT/AT adapter cards and invest in equivalent EISA adapters to get the full EISA capability for which they had originally paid.

EISA promoters emphasize that users can retain their investment in existing



PC/XT/AT cards when converting to EISA. (As mentioned above, market research concludes that only 2.3% of existing adapter cards are ever ported to new computer systems; if they are, the value of the computer from which the adapter cards are removed is almost zero.) By emphasizing the portability of PC/XT/AT cards to EISA systems, EISA promoters ignore the larger picture: PC/XT/AT cards in EISA systems will become obsolete, and any investments made in them will be lost when users eventually invest in equivalent EISA cards to get full EISA functionality from their EISA computers. *Net:* Buying EISA does not preserve investment – it postpones the decision to discard the investment in PC/XT/AT cards while encouraging that investment to increase in the meantime.

When users buy the Micro Channel architecture, they invest only once in its function, and the function is available as soon as the Micro Channel system is installed. Approximately 1,200 cards are now available for Micro Channel systems, and well over seven million IBM PS/2 Micro Channel systems (see Reference for *Catalog*) are in place. The business case for implementing advanced-function cards, such as Small Computer Systems Interface (SCSI), eXtended Graphics Adapter (XGA™), and A Real-Time Interface Coprocessor (ARTIC), on the Micro Channel interface is very positive indeed. If one graduates to Micro Channel architecture, there is no need for PC/XT/AT card compatibility.

PC/XT/AT cards carry a host of design limitations: 10-bit addressing, hardwired DMA requests, non-shareable interrupts, fixed ROM/RAM assignments, excess bus capacitance, and more. The full legacy of limitations requires an entire chapter to describe in *The Micro Channel Architecture Handbook* (see Reference). Most functions of AT adapter cards have been reproduced and/or extended on the Micro Channel interface, within a consistent, fully functional architecture. *Net:* There is no need to inherit the impaired genes from PC/XT/AT cards. One can simply move to Micro Channel instead.

PC/XT/AT Card Compatibility and EISA Advanced Function are Mutually Exclusive

Recall that installation of a single PC/XT/AT card will defeat or impair much of the EISA function in a system. The user typically exchanges full EISA system function for PC/XT/AT card portability. The particular system resources used by PC/XT/AT cards in the EISA system, and the number of PC/XT/AT cards installed, govern which system functions are defeated, or the degree of injury to system capabilities.

Typically, if PC/XT/AT cards are installed in an EISA system, they impair:

How PC/XT/AT Cards Can Prohibit Interrupt Sharing

The drivers for interrupt request lines in PC/XT/AT systems must first pull the request line down to a 0 (zero), then rapidly up to a 1. The circuits that cause this transition come in two types. Type 1 is called a TTL, or totem-pole, driver. The name *totem pole* comes from the configuration of stacking the driver components serially above one another, as depicted in Figure A.

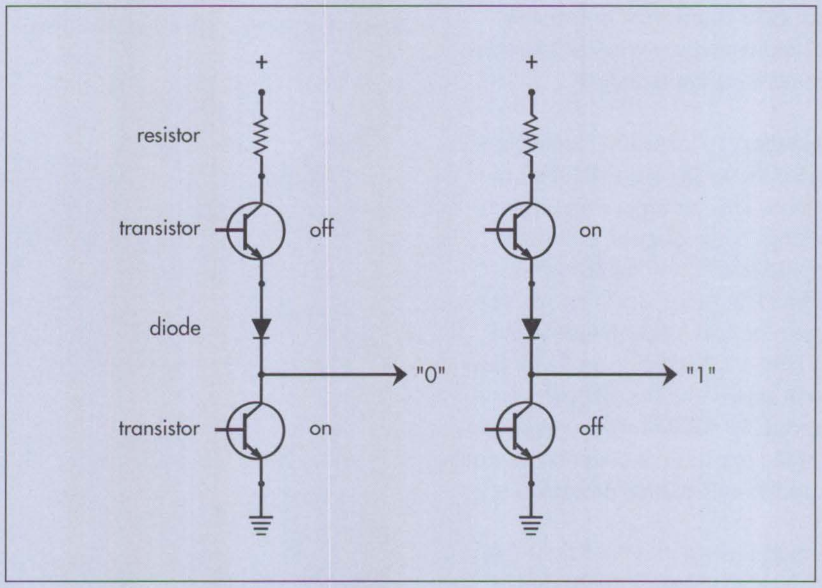


Figure A. Totem-Pole Circuits Operating at Logic 0 and Logic 1 Levels

In Figure A, in the circuit on the left, the top switch is off and the bottom switch is on, which pulls the output down to a 0. This is the situation *before* the edge-triggered interrupt transition.

In the circuit on the right, the top switch is on and the bottom switch is off, which pulls the output up to a 1. This is the situation *after* the edge-triggered interrupt transition.

In a system with concurrent I/O, interrupts cannot be shared between two PC/XT/AT cards, or between a

- Automatic configuration of the system
- Interrupt sharing between cards
- High-speed transfer above AT computer speeds

Each of these items is discussed in detail below.

Automatic Configuration

There is no value to automatic configuration of some EISA cards if the user must still remove the system's covers to change configuration, or to run diagnostics for the PC/XT/AT cards that have been imported to the system. To isolate a failing PC/XT/AT adapter card, users typically use a process of elimination: removing PC/XT/AT cards manually, one by one, and running diagnostics each time until the failing card has been identified.

Also, to identify which system resources are still available for the cards that have switches, the user must know how the automatic configuration has already configured any EISA cards installed in the system. Mixed automatic and manual configuration only makes setup more difficult. It makes remote diagnostics and network asset management nearly impossible without manual intervention to remove covers to inspect a client system.

Micro Channel systems can configure and diagnose systems without manual intervention, and can even do this remotely over a LAN when the LAN software supports this function. Programmable Option Select (POS) enables the computer itself to identify adapter cards, to set options, and to use diagnostics to isolate failing adapters, without human intervention and from a remote point. The computer can do these things for every card in the system, regardless of configuration, because PC/XT/AT cards are not portable into Micro Channel systems – a major advantage.

Interrupt Sharing

Interrupt sharing means that two or more cards share the same interrupt request line into the computer. All Micro Channel cards and EISA cards, by definition, can share an interrupt request line, but PC/XT/AT cards cannot. Consequently, in EISA systems, interrupt sharing is disabled on any interrupt request line used by a PC/XT/AT card. The problem is with the PC/XT/AT cards themselves. The full technical explanation follows.

To request interrupt service in PC/XT/AT systems, a card must first pull the request line down (low) to a 0 (zero), which means approximately zero volts of direct current, then rapidly pull it up (high) to a 1, which means approximately 2.4 to 5.0 volts DC. This rapid transition, or edge, tells the system that an adapter card needs attention to move data or to recover from an error, or that a supported device needs human intervention, such as loading paper in a

PC/XT/AT card and an EISA adapter that is on either the system board or another card. In Figure B, two PC/XT/AT adapters are connected on the EISA bus to share an interrupt. As before, the circuit on the left has not presented an edge-triggered interrupt, and the circuit on the right has just completed an edge-triggered interrupt transition from low to high. In this condition, a temporary short circuit exists as current flows through components R2, Q2, D2, through the connection on the interrupt request line, and finally through Q3. The length of time that the short circuit exists is a design variable that is not standardized in PC/XT/AT cards, but in any case, the short circuit rapidly heats the involved components. Eventually, either months or microseconds later, a component will fail due to the short circuit. The actual time of failure depends on

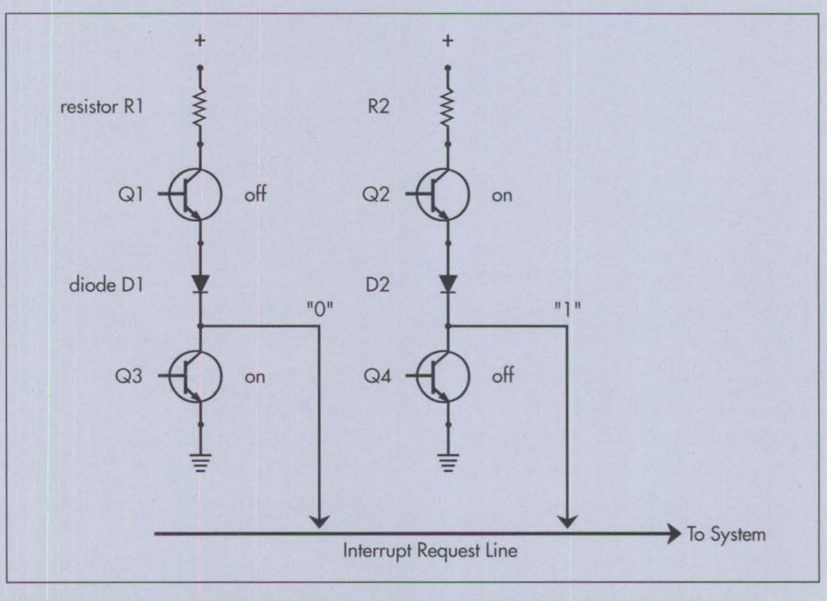


Figure B. Two PC/XT/AT Cards Connected to the Same Interrupt Request Line

how frequently the interrupts are requested, and how long the components heat each time.

Some PC/XT/AT cards have a second type of driver that is called *tri-state* or *3-state*. This design is a printer. If two cards are connected on the same interrupt request line, with each card concurrently operating and trying to set interrupts, then one card may be pulling high (after the edge), while the other is pulling low (before the edge). This will cause a momentary short-circuit across the power supply through the driver circuits. Depending on the contest of strength in the hardware, the interrupt request may or may not create the transition, and may be lost to the system.

It gets worse: this tug-of-war can radically shorten the life of the system. The competing components, which are on either adapter cards or the system board, have integrated circuits that will get overheated. After a while, the result will be a silicon heating – that is, a failure. To prove this, one need only set up two PC/XT/AT async cards on COM1, then drive them from two different sources of data using multitasking. The components may fail within microseconds or days; there is no way to predict the time of failure, or whether the failure will be on a system board driver or on a card.

EISA interrupt sharing is far from foolproof, because unlike in Micro Channel systems, the old PC/XT/AT cards are involved in the process. For a complete technical discussion, see the box beginning on page 6, "How PC/XT/AT Cards Can Prohibit Interrupt Sharing."

High-Speed Transfers Above AT Computer Speeds

High-speed transfers on the EISA bus depend on the ability of the lower 16-bit half of the bus – the PC/XT/AT half – to keep up with the upper half. But when the old PC/XT/AT cards are operated faster than 8 MHz with one wait state, they cannot recover within the time allowed before the next transfer. This is because there was no specification for the capacitance that PC/XT/AT cards place on the bus. This, in turn, is why so few card configurations work in fully populated systems that run the bus faster than the 375 nanosecond (ns) speed provided by the 8 MHz with one wait-state design point. Most systems whose processors run at faster speeds have to run PC/XT/AT cards on the bus no faster than 375 ns, or else they list a subset of cards that run in their systems. There is no way for the user to know which cards will and will not run faster than 375 ns without complex testing of each card design.

However, when EISA runs its proposed 33 MB per second DMA cycles, both the upper and lower halves of the data bus must respond within 240 ns, well before most PC/XT/AT cards will allow the lower half of the bus to respond. This problem is not likely to occur if just EISA cards are installed. The problem may not even appear until long after the initial PC/XT/AT cards are plugged in. It will eventually appear when a card is

variation of the totem-pole driver design. It allows installation of multiple adapters on the same interrupt request line, but for sequential operation only. The driver design is called tri-state because it can pull the output either high or low like totem pole, and it offers a third mode that electrically disconnects from its output line. The tri-state driver switches from low to high for only a short time before and after each edge-triggered interrupt transition. Otherwise, the tri-state design does not connect to the interrupt request line at all. As long as no two adapters operate at the same time, no temporary short circuit will occur. This concept, for example, enables multiple printer adapters to be installed in DOS systems.

But the tri-state concept has a hidden pitfall: it requires the operating system to use these devices sequentially rather than concurrently. This is not a limitation when single-tasking, because only one I/O operation at a time occurs. However, the microcomputer world is moving to multitasking to gain efficiency by using the time when progress is delayed in one operation to make useful progress in another. In this manner, several applications can run concurrently, faster than they can run sequentially.

But concurrent applications typically trigger concurrent I/O operations. Therefore, a multitasking operating system that is installed in a system with PC/XT/AT cards is configured with special software drivers that prevent any I/O device adapters installed on the same interrupt request line from operating concurrently. The alternative is to risk permanent damage to the system in the future (due to the temporary short circuiting) if I/O concurrency were later defined.

Consequently, although a computer may work acceptably when limited to sequential I/O operations, an EISA computer with ISA adapter cards supporting the I/O will also be *restricted* to sequential I/O where interrupts are shared in multitasking operating systems. This dilutes one of the major efficiencies of multitasking operating systems: concurrent I/O operation. *Net:* One task may run efficiently in memory, but then it may have to wait for another task to complete before its I/O operations can begin.

Both Micro Channel systems and pure EISA systems employ another method for signaling interrupts.

added later, because capacitance is cumulative – it grows as cards are added. To compound the situation, the problem will appear to be due to the last card installed, even though that card may not be the source of the excess capacitance.

Net: The user should expect that any AT card installed into an EISA system will degrade the system to an expensive AT system, and may potentially create spontaneous data errors. The user may never know about these errors until they are discovered in another way – perhaps after they have propagated throughout all files and data in the system. EISA functionality and AT card compatibility may be mutually exclusive.

Micro Channel Architecture Fixes Real Data Integrity Problems

Not long ago, I presented Micro Channel architecture to a large PC user group. At the point where I was discussing the advantages of electromagnetic compatibility and power distribution in the Micro Channel architecture, one attendee interrupted (because I had invited questions) by quoting a line from an EISA proponent's marketing brochure. He asked, "Aren't you fixing problems that don't exist? Have you ever seen a noise-induced error?"

I must agree that no error has ever identified itself as noise-induced, but I have had the rare pleasure (shared with many others) of spending a year of my life tracking down noise-induced errors in PC systems. So I responded, "Of course I have, and so have you; we all have. It is the way we all learned we had to save files often, and it was the most common complaint about hardware data and system integrity in PC/XT/AT systems."

I added that the common failure of locking up (some of us call this "beaming up") is typically caused by either a software design defect or a hardware error. If the source of the error is software, the failure typically recurs whenever the program reaches an identical state or situation. When hardware is the culprit, the failure most often is not repeatable by placing the system in an identical state. Instead, the telltale sign is a system lockup that occurs randomly.

One source of the hardware failure can be shown in a lab experiment involving a PC/XT/AT card, which is illustrated in Figure 1. If we place an analog scope probe on a ground pin of a system-board module, and another probe on a ground pin of a PC or XT combo card, we might observe a differential that often spikes to 0.5 volt or more. If we remove the screw that holds the card in place and also acts as a primary safety ground, the difference may creep higher. (For safety and data-integrity reasons, you

This method uses *open-collector* drivers, as shown in Figure C. Any number of open-collector drivers can operate concurrently on the same interrupt request line.

The open-collector circuit works fine unless a PC/XT/AT totem-pole driver is installed on the same interrupt request line. If the totem pole and open collector are installed on the same line, a short circuit will occur whenever the open-collector driver is active and the totem-pole driver has completed an edge-triggered interrupt. This situation is shown in Figure D.

The installation of multiple adapters of a similar type, such as communications adapters, may force the adapters to share the same interrupt request for compatibility with existing application software. However,

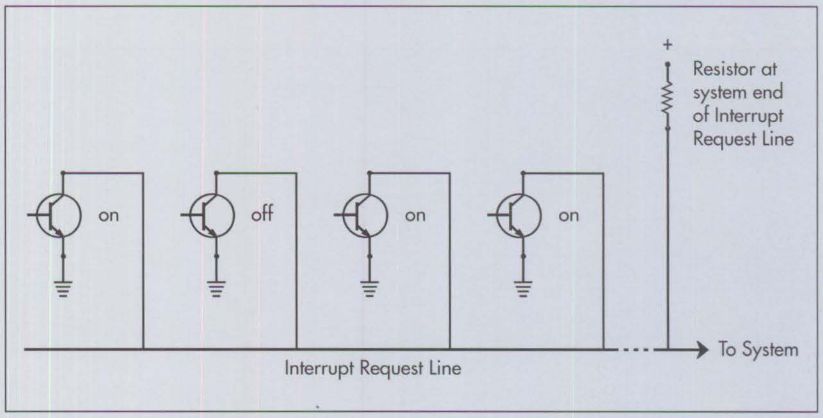


Figure C. Multiple Open-Collector Drivers on the Same Interrupt Request Line

if a PC/XT/AT card is installed in an EISA system, and it is configured to share an interrupt request line with EISA adapters, a transient short circuit through R1, Q1, D1, and Q4 and/or Q5 will occur. Q4 and Q5 will be active for extended periods of time for level-sensitive interrupts, allowing prolonged time for heating. If Q4 or Q5 is located on the system board, the expensive system board will be the failing unit.

This fate is likely to occur in EISA systems because EISA systems claim to handle interrupt sharing

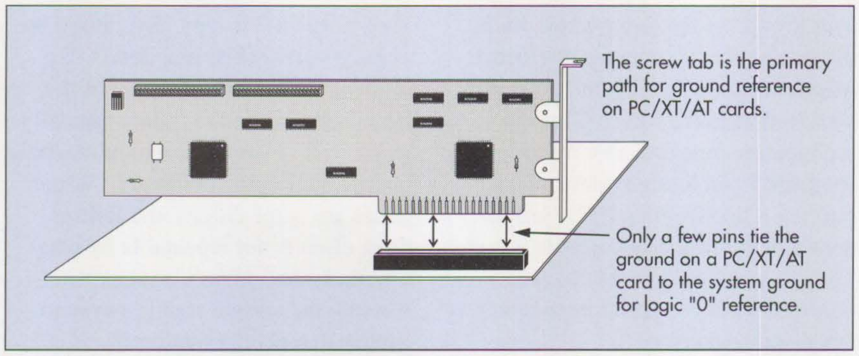

The screw tab is the primary path for ground reference on PC/XT/AT cards.
Only a few pins tie the ground on a PC/XT/AT card to the system ground for logic "0" reference.

Figure 1. How PC/XT/AT Cards Are Grounded

should *always* ensure that these screws are installed.)

Any conducted energy that now comes into the system (through either the ground shield of a cable connected to this card or a poorly grounded peripheral device) will push the voltage differential above 0.8 volts. This is the level at which common totem-pole (TTL) components are no longer guaranteed to recognize a zero logic level correctly. Between 0.8 and 2.4 volts (above 2.4 volts it is received as a 1 by TTL components), the value is a logical "I don't know," and it will be received as a data error.

So, *if* the card carries memory, BIOS or Power-On Self Test (POST) Read-Only Memory (ROM), or a memory-mapped interface to I/O, or even an I/O interface register; and *if* it is addressed at the moment when a noise spike is conducted from the outside world; and *if* the resulting data is used to construct an address for a conditional branch instruction; and *if* the condition is met and the address is used by the system; *then* the processor can be directed to the wrong place to pick up the next instruction in memory. *If* this happens, the system will try to execute the wrong instructions, data as instructions, or fragments of two instructions. The result is that the system loses its place in the logical flow of control; then the system will or will not clear the interrupt controller or associated pointers to interrupt-service routines. *If* the system clears the interrupt controller or pointers, the keyboard interrupt associated with a Ctrl-Alt-Del rebooting will not get through; *if* the interrupt controller is unaffected, the keyboard reset combination will get through. Either way, control of the system and the transient data stored in RAM are lost.

Notice that every *if* in the last paragraph is emphasized. Rarely are all the *if* conditions met, but when they are, the hardware locks up at random times. Would you like to know how much fun it was to find the cause of these hardware lockups?

and they claim PC/XT/AT card compatibility. It is not emphasized that these two functions might not be available at the same time. In contrast, this problem will never occur in Micro Channel systems because the use of PC/XT/AT adapter cards is prohibited by the mechanical design of the Micro Channel Connector.

Summary

PC/XT/AT card portability is actually a disadvantage that may impair the advanced functions of EISA. The Micro Channel's prohibition of PC/XT/AT cards is a major system integrity advantage for Micro Channel systems because it enables consistent functionality regardless of configuration.

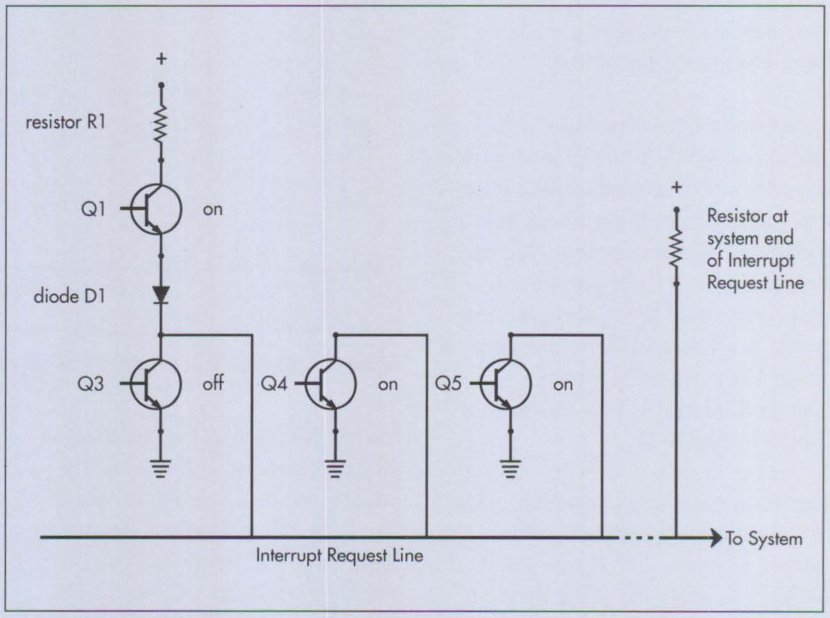


Figure D. Totem-Pole and Open-Collector Drivers on the Same Interrupt Request Line

By redesigning the card connector interface, thereby improving the ground connection to adapter cards, IBM has attacked the number one complaint about data integrity in PC/XT/AT systems. True, much progress has been made with PC/XT/AT systems by placing all system memory on the system board, by placing most BIOS ROM on the system board for standard functions (such as file adapters), and by installing ground and power planes on newer PC/XT/AT cards. Nevertheless, the problem can still occur. In contrast, this problem has been significantly reduced, if not altogether eliminated, in Micro Channel systems.

Turn the rhetorical question around: Have you ever seen a PS/2 system "beam up" from hardware? I posed this question to the 2,000 attendees at the user group meeting, and waited ten seconds for a response, but not one hand was raised.

What Makes a Channel Interface Proprietary?

EISA proponents often label Micro Channel systems as proprietary and EISA systems as open. Upon close inspection, a case can be made that the converse is true. The EISA architecture can be viewed as proprietary to system manufacturer, component manufacturer, and operating system. Here is why.

Two elements of the EISA architecture, which are billed as features, support the perspective that EISA is proprietary to processor technology and system manufacturer. These elements are:

- ISA and EISA produce synchronous bus cycles.
- EISA dictates a specific system structure that separates the processor, DMA controller, and memory onto a proprietary bus that is separate from the EISA I/O bus.

Synchronous Bus Cycles

When IBM defined the PC bus, it derived the timing of the bus transfer as it came from the processor and associated interface chips. Transfers were defined to be synchronous with the edge of the processor clock. This simplified the adapter logic as well. But it also created a problem as the bus cycle was accelerated: if the processor clock rate increased, the time to sample data from the bus was decreased. Many cards referenced the processor clock and further tuned their transfer to the processor's timing. So cards were tuned to the bus cycle, and as the number of cards for the PC and XT interface increased, the processor bus cycle became a factor that affected card compatibility when the IBM Personal Computer AT® was announced.

To limit the number of PC cards affected, AT systems inserted a waiting period of at least 125 billionths of a second (called a *wait state*) to enable 8-bit PC/XT cards to catch up. Still, many PC/XT cards were tuned to the 6 MHz AT clock, and did not work on the 8 MHz AT systems. Most attempts to standardize the AT interface after the fact have limited the I/O activity for PC/XT/AT cards to approximately the 8 MHz limit with at least one wait state.

Therefore, adapter cards have become tuned to the bus cycle of a single family of microprocessors, and the vast majority of PC/XT/AT system products are driven by this single family of microprocessors. (A large factor behind this is the circuit complexity and performance impact that other microprocessor designs have encountered when they have tried to reproduce the tuned bus cycle of PC/XT/AT cards.) The use of a single microprocessor family is okay when a system is starting out. However, as system designs mature and use alternate processors, or when different processors work in concert (such as in multimedia), it is an unreasonable restriction to have so much linkage to the design of a single processor bus cycle.

An architecture such as EISA, which tunes the system to a given processor family, also tunes the system to an exclusive set of operating systems and applications that are written using the primitive assembly-language instructions for that processor family. Assembly language is often used for performance reasons. A "tuned" architecture also cannot easily be used as a foundation for alternate processors, operating systems, and applications. (For example, a user may wish to migrate to a RISC system running UNIX®.) This makes a "tuned" architecture, such as EISA, a poor choice as a foundation for a broad range of systems. *Net:* The tuned architecture becomes married to the limited set of operating systems and applications supported by the processor family. (Note that, almost without exception, all EISA and ISA systems are supported by just one processor family.)

Micro Channel systems have been marketed with IBM and Intel 80X86 processors, as well as with IBM RISC and 370 processor designs. In addition, IBM has marketed bus master adapters for the Intel RISC processor family, the RT® RISC processor, and the Motorola™ 68000-series processors. The bus master card acts exactly like a processor, by directly controlling bus-attached cards and memory as "slaves." Motorola 88000-series RISC processors and custom bus-controller designs have been adapted to Micro Channel as well.

The Micro Channel interface is not tuned, because its interface produces both synchronous and asynchronous cycles, thereby providing a wide latitude toward matching the timings of a broad range of processor technologies. Also, Micro Channel does not

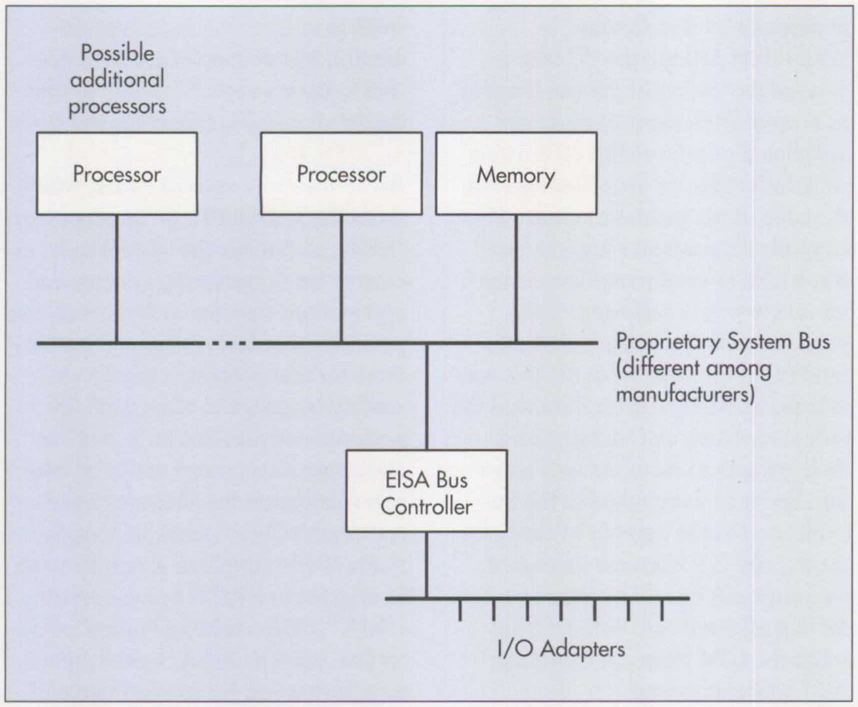

Possible additional processors
Processor
Processor
Memory
Proprietary System Bus (different among manufacturers)
EISA Bus Controller
I/O Adapters

Figure 2. EISA System Structure

define a processor clock. Synchronous cycles are produced as simple time extensions of a default specification. Asynchronous cycles are defined to move data within a wide range of timing variation, and all cards are defined to respond to a set of both types of transfer cycles. Micro Channel simply works better for the wide range of technologies that will be developed on this platform, and it is not even partially proprietary to one processor technology.

EISA Defines a Proprietary System Bus

A fundamental mandate of the EISA architecture is to separate the system structure into two parts: the EISA bus, and a proprietary "system bus" that contains the processor and memory. The DMA controller function is part of the logic that bridges the two buses; it is called an EISA Bus Controller. Figure 2 depicts the EISA system structure.

The two-bus system design has performance advantages, and it may assist when cache memory is defined. However, this design has two negatives:

- It is typically more expensive than a low-end system that has all elements on a common bus (because more logic equals more cost).
- It isolates the major elements of performance in a system – processor and memory – on a proprietary interface where they cannot be extended except by designs that incorporate the manufacturer's proprietary interface.

If the availability of these elements – which determine system performance – is constrained by the bus architecture by placing them on a proprietary interface *only*, then obsolescence is designed in.

The Micro Channel architecture does not mandate this particular structure; instead, the Micro Channel bus accommodates it as one of several possible system structures. Micro Channel enables the addition of processors and memory on the user-accessible interface for I/O adapters, even when a two-bus structure is implemented.

Far from being proprietary to IBM, Micro Channel systems and cards are now available from sources throughout the world, many of whom are members of the Micro Channel Developers Association.

Reference

Heath, Chet and Rosch, Winn. *The Micro Channel Architecture Handbook*. Brady Books, 1990. ISBN 0-13-583493-7 (IBM order number Z281-0300).

Sanderson, M. N. *Catalog of International Micro Channel Expansion Adapters*, 5th Edition (G360-2824-07).

***Chet Heath** is a Senior Technical Staff Member and engineer at IBM's Entry Systems Technology Laboratory, in his twenty-first year with IBM. He holds a BS in electrical engineering from the New Jersey Institute of Technology (Rutgers) and an MS in electrical engineering from the LSI Institute at the University of Vermont. Chet was the primary innovator of the PS/2's Micro Channel architecture and has led its definition since inception. He has received numerous awards for his work on Micro Channel architecture including IBM's eighth-level invention award, Outstanding Technical Achievement, Outstanding Innovation, and Corporate Technical Achievement awards. He has also received IBM Quality and Authors' Recognition Program awards. He was named one of the ten most influential people on PCs in the '80s by* PC User *magazine in the U.K.*

Portable Computer Trends and Directions

Leo L. Suarez and Carlos A. Alvarez
IBM Corporation
Boca Raton, Florida

Portable computers are redefining the traditional computing environment. This article covers the different classifications of portables, their characteristics, market segmentation, and technological trends.

Portable computers have evolved into small powerhouses that offer desktop system function and power in very small, lightweight packages. This rapidly expanding segment of the personal computer market will have a tremendous impact on how information is processed. Portable computers are increasingly prevalent in business, giving users the information and processing power they require, whether on the road, at a client's location, at home, or at the office.

The main reason for the high growth rate of portables is that they satisfy the needs of a large group of users who had previously been unable to take advantage of personal computers. This group of users is known as the *mobile work force*. It includes people such as traveling salespeople, public safety officers, public utility service personnel, home users with limited space, students, and professionals who have desktop personal computers in their offices but require comparable computing power away from their primary work location.

Types of Portable Systems

The capabilities and limitations of portables are bounded by their size and weight. In the portable arena, size plus weight equals function – the more function a machine has, the larger and heavier it is.

There are five general classes of portables:

- Transportables
- Laptops
- Notebooks
- Pen-based computers
- Palmtop PCs

Transportables

Transportables have the highest function and performance of all portables, and typically match the capabilities of their desktop counterparts. These machines have powerful processors, such as an Intel 80386 or 80486; several card slots for internal expansion; large hard disks (400 MB); full-sized, 101-key keyboards; and high-quality displays, such as plasma or active-matrix liquid crystal. Because of their function and expansion capabilities, these machines are usually powered by alternating current from a wall socket. Transportables tend to be significantly heavier than other portables, weighing 15 pounds or more. They are used primarily as desktop systems, but can be moved to other locations easily if necessary.

Examples of IBM transportables are Personal System/2® Models P70 and P75.

Laptops

Laptop computers have many ergonomic features found in transportables, such as full-function keyboards and large displays. Unlike the transportables, laptop systems are battery-powered and weigh less than 10 pounds. Their hard disk capacity ranges from 20 to 120 MB. Memory can be expanded well beyond 10 MB. Serial, parallel, and Video Graphics Adapter (VGA) ports, as well as a bus extension, are provided. Most laptop systems have provisions for internal modems. A key feature is the bus extension port, which enables the machine to be connected to a docking station (also called an expansion box). Docking stations provide the user with additional DASD bays, card slots, and ports. The combination of docking station and laptop computer gives the user the same capabilities as a desktop system.

This class of machine includes the IBM PS/2 Model L40 SX, with an 80386SX processor running at 20 MHz, a 60 MB hard disk, up to 18 MB of RAM, and an internal data/fax modem – all packaged in less than 8 pounds. These features make the L40 SX a very attractive portable system.

Notebooks

The size of a notebook computer has been defined by a *de facto* industry standard of 8.5 x 11 x 2 inches, because this size fits inside half of a typical briefcase. To get down to this size, some function is sacrificed, and ergonomic trade-offs are made. The most obvious are a smaller keyboard and a display with a smaller dot pitch. The keyboards in these systems usually have half-sized function keys, and the key layout is usually not consistent with those of larger desktop or laptop systems. Some keyboard functions may require multi-keystroke combinations.

Because portability is the main criterion for these machines, more emphasis is placed on battery life. The minimum battery life expectation for this class of machine is three hours. Weight is also a big factor: notebook computers typically weigh less than 7 pounds today, with ideal weights between 4 and 5 pounds. Although these machines are small, they still pack significant function: Intel 25 MHz 80386SX or similar processors; hard disks as large as 80 MB; and add-on features such as modems, co-processors, and VGA screens. These machines are sometimes capable of connecting to docking stations.

Pen-Based Systems

The keyboard is replaced by stylus input on pen-based systems. They are similar to electronic notepads, in which the operating system is designed to simulate a book with chapters and a table of contents. The size of a pen-based computer is based on an 8.5 x 11-inch notepad profile, with special emphasis on a thickness of 1 inch or less and very low weight, ideally below 4 pounds.

Despite their small size, tablet systems carry substantial computing power so that they can recognize and process a user's handwriting. Because they have to be small and lightweight, many functions typically found in other portables are removed. For example, a tablet computer has no internal floppy diskette drive and no hard disk: their storage function is performed by removable memory cards or solid-state disks. The emphasis of tablet systems is truly on ease of use and lightweight portability.

Palmtops

Palmtops, or hand-held computers, are the smallest of the portable systems, usually weighing less than a pound. Palmtop systems have limited functions and features, and their software interfaces are mostly proprietary or non-standard. The keyboards on these systems have calculator-type keys. Small liquid crystal displays, usually 5 inches (diagonal) or less, are used.

Factors Influencing the Design of Portables

Portable systems are developed in response to the requirements of the mobile work force. These users require systems that are:

- Usable
- Portable
- Compatible
- Dependable
- Expandable
- Durable

Despite their small size, tablet systems carry substantial computing power so that they can recognize and process a user's handwriting.

Mobile workers put special emphasis on size, expansion, and operating environment.

Size and Weight

Portability and usability are competing requirements. Naturally, users would like all the convenience of a desktop system without the size and weight. The keyboard and screen are the critical components in this area – they are necessary for user interaction with the computer, but they add weight and bulk. Therefore, the right trade-offs of function versus size and weight must be made.

Expansion

Connectivity is a major force in the explosion of this market. Cellular communication, Radio Frequency (RF) networks, and nationwide information networks will all play larger roles in computing as the portable market expands. Users will expect the ability to retrieve information from other systems, whether using a modem at home, a cellular unit on a train, RF at a client's site, or a token ring port at the office.

Environment

Dependability and durability in a non-traditional computer environment are major challenges. Whereas office computers are rarely outside air-conditioned environments, portable computers are expected to work and travel through a wide range of environments, from hot, humid summers to cold, dry winters. Portable computer designers are starting to account for these factors. For example, the PS/2 Model L40 SX has both temperature and humidity sensors that will not let the system operate beyond its environmental limits.

Who Uses Portable Computers?

People of varied occupations and pursuits are using portable computers. The examples below show the versatility and different uses of portable systems in today's environment.

Route Salesperson

Route salespeople normally carry clipboards for filling out invoices and other documentation. A clipboard can be replaced by a pen-based system with which they can capture all this data, keep track of their inventory, and later upload the data to the company's host system.

Public Safety Squads

Police and fire departments require information quickly, usually in remote locations. Police officers use portables, for example, to check car registrations and drivers' licenses directly from their squad cars. Fire departments are computerizing the layouts and contents of buildings.

These systems, installed in fire trucks, can show firefighters the locations of dangerous chemicals before they reach the site of a fire.

Home Users
The use of portables at home is evolving. Today, your home computer and your office computer are probably separate machines. Portable computers, however, allow the same machine to be used in both places, enabling users to have the same applications and data at home and at work.

Students
To prepare students for today's computer-intensive environment, many schools are requiring students to have computers. Students use their computers to prepare reports, analyze data, and take examinations. The portability of the systems enables students to use them in classrooms, dormitories, and even at home during school breaks. This is also the most severe operating environment for portable systems.

Business People
Portability is the key element that is changing the way a business meets its computing needs. Because computers can now enter work environments that were previously inaccessible, businesses benefit from additional productivity. For example, insurance salespeople who are showing the advantages of life insurance plans in customers' homes can dramatically improve their sales potential if they can instantaneously customize their offerings to suit those clients. Executives who are out of town and need to retrieve last-minute information from their office computers can maximize their effectiveness, even while traveling. They can then record the outcomes of their meetings, analyze the data, and send action items back to their offices on a timely basis, again increasing productivity.

Trends in Portable Computers
Environmental trends and changing work habits will continue to expand the need for portable computers. As seen in the practical uses cited above, connectivity to larger computer systems is a major requirement. Significant challenges must be undertaken to include connectivity capabilities, and to enhance existing subsystems so that they match or surpass their desktop system counterparts. Equally important is that all of this must be offered in a lightweight package and be able to run for extended periods (eight hours or more) on a single battery charge. In fact, technology trends can be summed up in one word: miniaturization, which means more function in a smaller package.

Technology Challenges
Designers of portable computers must meet the ever-present demands for smaller, lighter, yet more powerful systems. Several computer subsystems are key to overcoming the weight, power, and size limitations. These particular subsystems have significant technology challenges ahead. Technological advancement in these areas will determine which manufacturers of portables will have the most competitive, desirable products. These key subsystems are:

- Hard disk drives and floppy diskette drives
- Display screens
- Rechargeable batteries
- Mechanical packaging
- Communications

Hard Disk Drives
The key challenge in hard disk drives is to continually miniaturize their packaging while increasing their storage capacities. Most users want at least 40 MB of storage. A typical OS/2 user finds that 40 MB is the minimum capacity that will effectively use the full features of the operating system. Also, because of space limitations, a 2.5-inch hard disk is the maximum width that permits the computer to remain within the 10 x 12 x 2-inch package size.

Currently, the highest capacity of 2.5-inch hard disks that are available in volume is 80 MB, with 120 MB disks close behind. These disks typically consume only about 2.8 watts of power when reading from or writing to the disk, compared to 10 watts for a typical 3.5-inch hard disk in a desktop system. In addition, these hard disks have unique power management features that allow the disks to enter a power-saving state, during which the disk's power drain is reduced to 0.3 watts or less. The data access time of these disks is typically 17 milliseconds or less, with data transfer rates reaching more than 1.5 MB per second. This performance is usually better than that of most hard disks in desktop systems.

Within the next 12 to 18 months, capacities of 2.5-inch hard disks will increase beyond 210 MB, and their access-time performance will decrease below 15 milliseconds.

As exciting as the 2.5-inch disk technology is, even 2.5 inches is too big, because users want very slim notebook computers that are 1 inch thick and weigh 4 pounds or less. Therefore, hard disk technology is being driven on two major fronts: sub-2-inch hard disk drives, and silicon hard disks.

Sub-2-inch form factors have already been announced in the industry. These hard disks are typically called 1.8-inch hard disks. Their capacities are around 20 MB now; however, we will see 40 MB and higher capacity hard disks within the next 12 months.

Silicon hard disks are a totally different approach to data storage. The entire electromechanical assembly and interface of the typical hard disk drive has been replaced by electrically erasable, readable read-only memory. This greatly increases performance and reliability while decreasing power consumption and size. The biggest challenge facing silicon hard disks is their cost, which is typically an order of magnitude higher than conventional electromechanical disks. Therefore, "affordable" silicon disk capacities currently are low – 10 MB or less. Used together with compression chips, capacities can be doubled cheaply. Affordable larger (40 MB or more) silicon files, however, are still several years away.

Floppy Diskette Drives

Floppy diskette drives have a unique challenge: standardization of media size. Many readers remember the time and labor it took to convert from 5.25-inch to the current 3.5-inch diskettes. Because of this experience, the microcomputer industry has been very reluctant to move to 2-inch media, even though it is available today.

As a result, the emphasis for floppy disk drives has been on reducing their thickness, weight, and power drain while increasing their capacity. Today, "super-slim" floppy diskette drives have a thickness of 15 mm, versus 25 mm for a floppy drive found on a desktop computer. These super-slim drives weigh less than 250 grams, and are available in 2 MB (unformatted) capacities.

Increasing the capacity while decreasing thickness and weight presents a special technical challenge. Suppliers are announcing 4 MB super-slim capability with weight dropping below 170 grams. These very light-weight drives, however, will require much engineering work to ensure that they operate reliably in the hazardous operating environment of portables, especially in high shock and vibration conditions.

As with hard disks, the ultimate solution probably will be to eliminate the electromechanical variable of the floppy drive, and migrate to other solutions such as smart cards – read/write cards that adhere to standards such as the Personal Computer Memory Card International Association (PCMCIA) interface.

Another opportunity for floppy diskette drives will come with the introduction of small, lightweight read/write optical drives. This solution offers removable media with extremely high capacities of 120 MB or more. Significant challenges remain to reduce the power drain, which today is almost six times that of floppy diskette drives.

The clear choice for leading-edge portables is active-matrix LCD.

Display Screens

Users want their portable's display to be comparable in image quality and responsiveness to the display on their desks. This means that the image must be viewable without distortion at extreme angles, and image updates must be very fast (for example, seeing a mouse cursor move across the screen without losing sight of it). Another key feature is color.

Because of the unique packaging requirements of portables, flat-panel displays must be used. There are three types of flat-panel technology: plasma, electroluminescent (EL), and liquid crystal. Both plasma and EL have severe limitations: they consume very high power (more than 20 watts for plasma, and over 12 watts for EL, for a 640 x 480 pixel display). They have limited ability to produce color, due to plasma manufacturing difficulties, and low brightness in blue phosphor in the case of EL.

Therefore, portable systems designers are left with Liquid Crystal Display (LCD) technology. There are many types of liquid crystal technologies, but two are dominant in the portable computer market: Super-Twisted Nematic (STN) and active matrix. (See the article titled "LCD Panel Technology" in this issue.)

There are major differences between STN and active-matrix LCD. STN has severe limitations with viewing angle and response time, but it is still the only LCD technology that is available in volume and at low cost.

The clear choice for leading-edge portables is active-matrix LCD. The most popular type of active-matrix LCD is Thin-Film-Transistor (TFT) LCD. Images on a TFT LCD can be viewed at extreme angles with minimal distortion. The images themselves have superb clarity, and TFT has very fast response time, so that fast-moving images can be seen. Equally important, TFT LCD is the only real choice for a viewable, full-color, flat-panel screen in the foreseeable future. Most TFT LCDs today are produced by pilot and low-volume manufacturing facilities. Manufacturing yields are very low today because of very strict quality specifications. In a TFT display, each transistor corresponds to an individual pixel, and every transistor must function correctly, or else the entire display panel has to be scrapped. As a result, volumes are low and unit costs are high. As TFT LCD manufacturers overcome these challenges, their products should enjoy tremendous growth rates.

Batteries

Until recently, advancements in rechargeable-battery technology have not progressed as quickly as in other areas of computer technology. In recent years, most rechargeable battery development has focused around consumer electronics and electric automobiles. With the demand for portables growing quickly, more emphasis is being placed on rechargeable batteries for computers. Today, computer users have two principal choices of battery technology: Nickel Cadmium (NiCd) and Metal Hydride (NiMH).

Although NiCd has existed longer, it has two major drawbacks: lower energy density and environmental waste hazards. NiCd densities today are around 160 watt hours per liter (WH/L), with slight increases to around 185 WH/L projected for 1992. The recently introduced NiMH batteries offer the user up to 25% more energy density in the same amount of volume, coupled with fewer hazardous materials. The availability of NiMH batteries will increase during 1992.

Lithium-ion (Li-ion) technology holds promise within the next few years. Li-ion can potentially double the energy density of NiCd. Densities of 280 WH/L could be achieved before 1997, with first introduction in portable systems in 1993 or 1994. Li-ion also appears to have a simple charge and discharge capability, which will make designing circuits for battery charging much simpler. Also, true analog fuel gauges can be easily displayed.

Mechanical Packaging

The main challenge in mechanical packaging technology is to make computers lighter and smaller while increasing their ruggedness. These two opposing forces make this area of portable design one of the most difficult. Reducing size requires developing new, lighter plastic materials for the outside case. The motherboard, which has traditionally been reduced in size through component integration, must turn to flexible circuit boards, and/or multi-chip carriers in which a ceramic "carrier" forms miniature circuit boards for components.

New ways of connecting to the outside world will include eliminating connectors and relying instead on other forms of data transmission, such as infrared or radio frequencies. Meanwhile, miniaturized connectors and other unique mechanical interconnect approaches will bridge the time gap.

Communications

Portable computers will reach their maximum usefulness when their users can connect the systems to any communications network with minimal effort and few attachments. In this scenario, the portable computer becomes an extension of a company's host computer. Connectivity is a crucial requirement of portable systems.

Connectivity is achieved today by an external adapter, which can be a card that plugs into a docking station, a small box that plugs into one machine's I/O ports, or (for a modem) an adapter that is installed inside the machine.

In the future, connectivity features will be a standard part of the system, or will be available as easily upgradeable features. An example is a complete modem/fax function on a "smart card" form factor, which would plug into a standard PCMCIA interface connector. In this way, a modem, additional memory, and token ring cards can plug into the same PCMCIA connector.

In the near future, portables will operate in a wireless communication environment. Connectivity in this environment will then be easy and transparent to the portable user, because many heavy computing and conversion tasks can be delegated to desktop or other host systems. The ability to do this is not far away.

Summary

Microcomputer buyers and their changing working environments are driving system makers toward portability. Miniaturization, which has always been a cornerstone of the electronics industry, is enabling computer makers to meet customers' wants and needs. The growing mobile work force represents a significant market segment that IBM intends to address with a full line of products from laptops to pen-based systems.

***Leo L. Suarez** is a product manager for portable system strategy and requirements in IBM's Personal Systems Line of Business. He joined IBM in 1978 as a component test engineer on the Series/1™ and workstations. He has since held positions ranging from technology staff at division headquarters to numerous management assignments in both development and manufacturing. His most recent assignment was development of the IBM L40 SX laptop computer. Leo holds BS and MS degrees in electrical engineering from the University of Miami.*

***Carlos A. Alvarez** is an advisory engineer and program manager at IBM's Entry Systems Technology Laboratory, currently responsible for portable systems technical strategy. He joined IBM in 1981 as a design engineer for Series/1 special-bid development. He has been involved in the development of several industrial computer products, such as the 7175. He was the engineering manager for the 7541, 7542, 7561, and 7662. Carlos holds a BS in electrical engineering, MS in industrial engineering, and an MBA from the University of Miami.*

LCD Panel Technology

Jim Paolantonio
IBM Corporation
Boca Raton, Florida

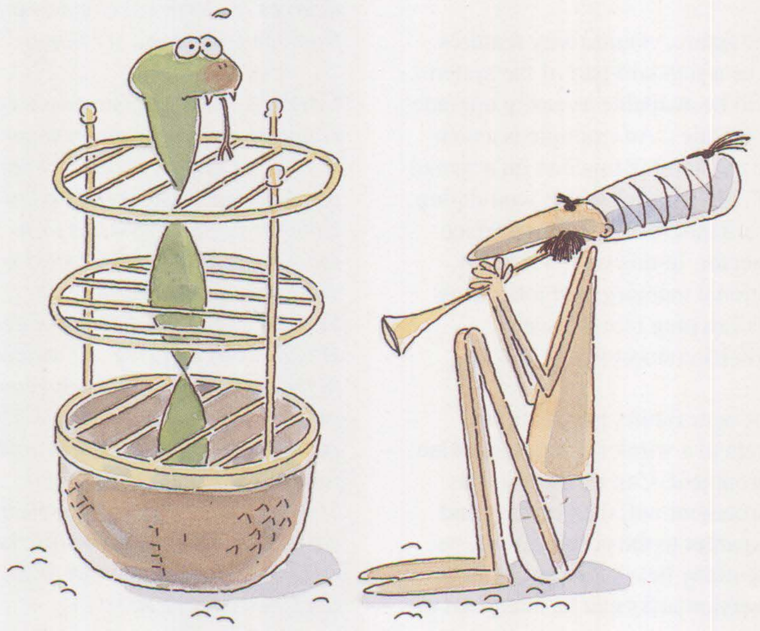
By 1995, portable PCs will dominate the personal computer market, replacing desktop PCs. One of the most important components in a portable PC is the Liquid Crystal Display (LCD). This article discusses the operation and key characteristics of LCD technologies.

LCDs were first introduced in watches and calculators. Early portable PCs used plasma and electroluminescent display panels. Today, Super-Twisted Nematic (STN) monochrome and active-matrix color Thin-Film Transistor (TFT) LCD panels are the most prevalent types used.

Twisted Nematic LCDs

Twisted Nematic (TN) LCD operation is quite simple in principle. When a nematic-composition liquid crystal is sandwiched between two grooved surfaces, the crystals align themselves with the orientation of the grooved surface that is in proximity. When these grooved surfaces are at 90° angles to each other, the crystals create a 90° twist. Light waves traveling through the liquid crystal layer are also twisted 90°. When an electric current is applied to the liquid crystals, they realign themselves in a vertical orientation. Light passing through this configuration is not twisted, thus exiting the liquid crystal layer in the same orientation as it entered.

LCD panels are created by sandwiching a liquid crystal layer between two light-polarizing filters oriented 90° to each other. Light passing through the bottom polarizer is oriented in one direction. As the light then passes through the liquid crystal layer, it is twisted 90°, thus exiting with the same orientation as the upper filter. The light passes through the upper polarizer unimpeded. When an electric charge is applied to the liquid crystal, the crystals form a vertical orientation. The light aligns itself with the orientation of the lower polarizer, and is not twisted as it passes through the liquid crystal layer. The upper polarizer blocks the light, since the light is oriented 90° with respect to the upper polarizer. This creates the effect of an on/off light switch.



Super-Twisted Nematic LCDs

Twisted nematic technology has been expanded to create STN liquid crystal layers. These layers twist light between 180° and 260°, which results in a better contrast ratio. One undesirable artifact of STN crystals, however, is that they create yellow-green or bright blue displays. To counter this effect, two liquid crystal layers with compensating twists are sandwiched together. This is called Double Super-Twisted Nematic (DSTN) and creates a black-and-white LCD panel. The same result is now obtained by sandwiching a super-twisted liquid crystal layer between two compensating plastic films. This newer technique has enabled the creation of multicolor LCD displays.

Active-Matrix Thin-Film Transistor LCDs

Active-matrix color TFT LCD panel technology has recently been introduced. With this technology, each subpixel (red, green, or blue) on the display is controlled by a thin-film transistor. A driver circuit puts out discrete levels of voltage to the transistor/liquid crystal layer as shown in Figure 1. The nematic liquid crystal structure is twisted a varying amount, depending on the voltage level. Light passing through the first polarizer is then twisted by the corresponding amount as it passes through the liquid crystal layer. The amount of light

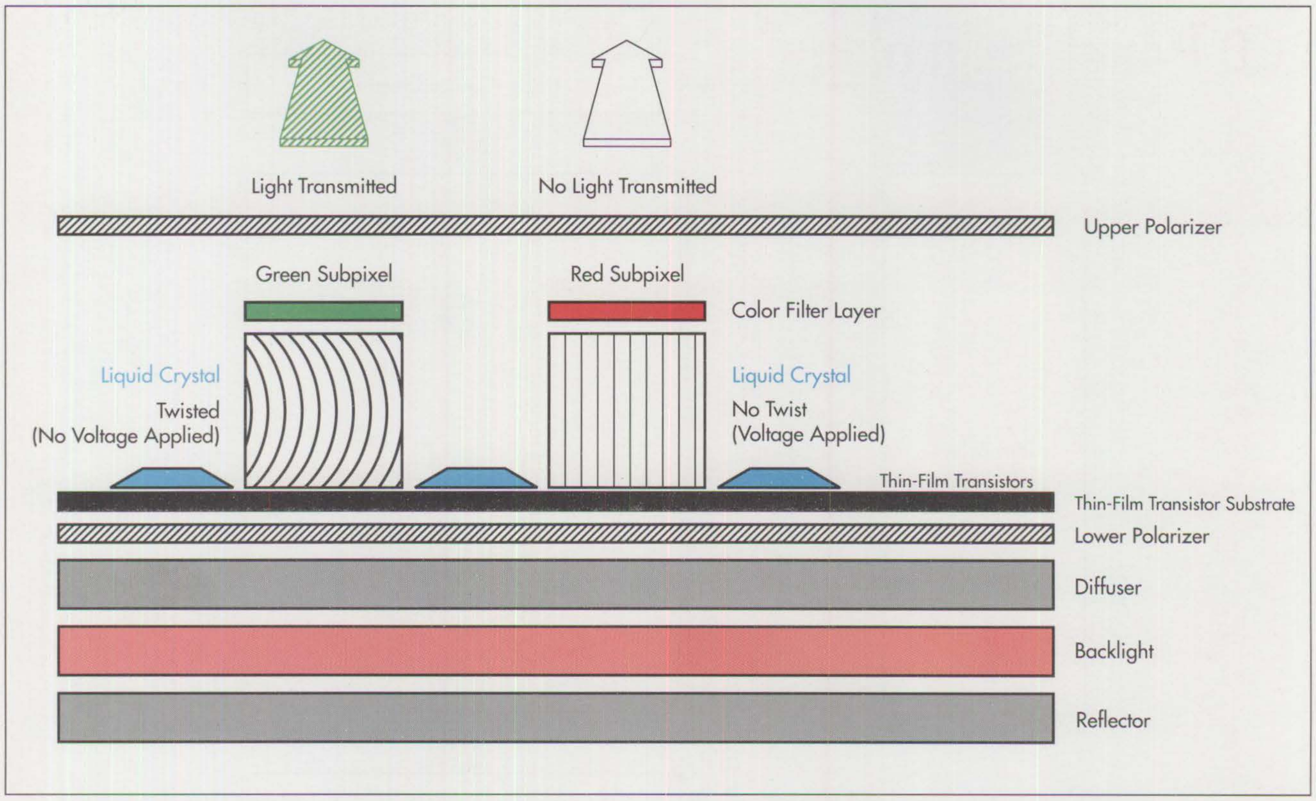


Figure 1. TFT LCD Panel Array

transmitted through the second polarizer depends on the angle of the light, analogous to a Venetian blind. The result is that the intensity of light transmitted through the panel can be controlled by the discrete voltage applied to the liquid crystal layer. With the addition of color filters over the liquid crystal layer, red, green, and blue subpixels are created.

Technology Trade-offs

As with most technologies, there are many trade-offs to consider. Lighting techniques affect battery life. Also, there are some key functional differences between the STN and TFT LCD panels. Among these differences are power, brightness, contrast ratio, viewing angle, and response time.

Early monochrome reflective STN LCD panels used ambient light for display. The ambient light that was incident upon the panel was reflected back to the viewer through a reflective back-panel layer. Brightness was inadequate in many environments. To increase the brightness, back or edge lighting was added to the LCD panel using cold or hot cathode fluorescent lamps. The fluorescent lamp increases the overall power consumption of the LCD panel, negatively impacting battery life, which is critical to users of portable PCs.

A compromise approach is achieved using *transflective* LCD panels, particularly in tablet-sized PCs. The transflective panel provides a backlight that can be turned on in low indoor ambient light. When the panel is used outside in bright ambient environments, the reflected light provides sufficient brightness, and the backlight can be turned off.

Power is at a Premium

Power in a portable PC is a scarce commodity. Battery life in today's portable computers averages between two and four hours – and users need more. Every attempt is made to conserve power by using low-power components and employing power-management techniques.

The LCD panel is one of the major consumers of power. The color TFT LCD panel is more power-hungry than the monochrome STN panel. Typical power consumption of color TFT LCD panels hovers between 13 and 15 watts, while monochrome STN panels average about 3 to 4 watts.

Contrast Ratios

Major considerations in LCD panels are their brightness and contrast ratios. Sufficient brightness averages between 60 and 80 (candelas/m^2). The contrast ratios of the STN and TFT panels differ significantly. STN panels have a contrast ratio between 10:1 and 15:1. In comparison, TFT panels offer significantly better contrast ratios between 30:1 and 50:1.

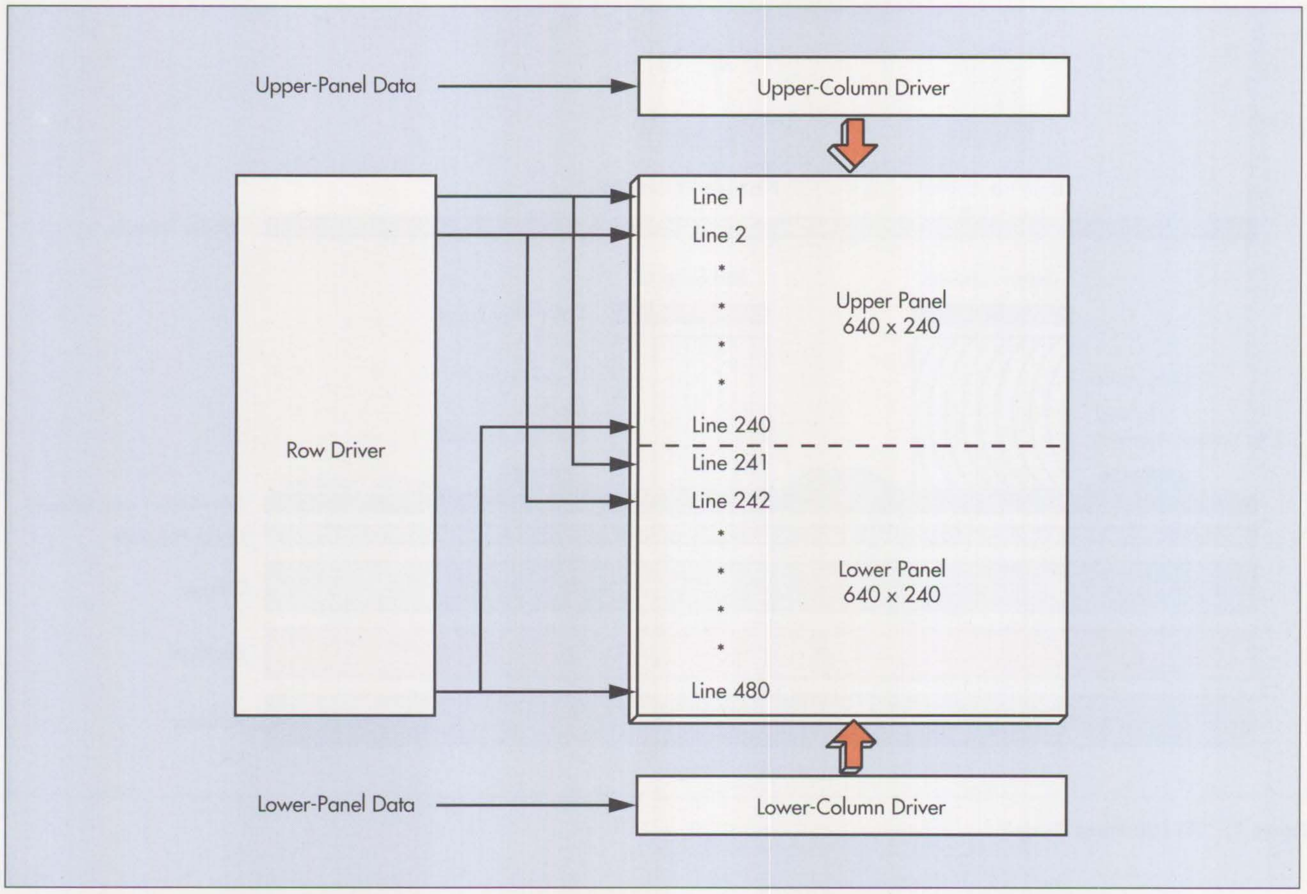


Figure 2. STN LCD Split Panel Operation

Contrast ratios, however, cannot be compared independently without considering the related factor of viewing angle. Viewing angle is the maximum angle from which the viewer can observe the panel while still perceiving a sufficient contrast ratio. Again, TFT panels offer superior quality. STN panels exhibit viewing angles of 40° horizontal and 35° vertical (with respect to the plane of the panel), while TFT panels offer viewing angles up to 80° horizontal and 50° vertical. There is some color washout at extreme viewing angles, particularly with the color black.

Response Time

With CRTs, an electron beam scans each individual line on the screen sequentially. There are 480 lines, and each line has 640 pixels, yielding a 640 x 480 screen resolution. As the electron beam scans across a particular line, each of the 640 pixels is individually illuminated for approximately 40 nanoseconds.

The response time of individual LCD pixels is several orders of magnitude slower than that of CRT pixels. TFT LCD pixels (which are actively driven by transistors) have response times in the 35 millisecond range. Because the LCD pixel response time is significantly slower than the pixel response time of CRTs, a different scanning technique must be employed. Instead of illuminating individual pixels on each line, a whole line (row) of 640 pixels is turned on simultaneously, and held on for the whole duration of the scan line (approximately 30 microseconds). Individual LCD pixels are not as bright as their CRT counterparts. However, because the LCD pixels are turned on for a much longer time, the perceived brightness of these pixels by the human eye is comparable to that of a CRT. The whole panel of 480 lines is scanned in approximately 16 milliseconds, which equates to a vertical refresh rate of 60 Hz.

STN LCD panel pixels have a response time from 150 to 400 milliseconds, which is significantly slower than the response time of TFT pixels. For STN LCD pixels, another scanning technique must be used. The 640-pixel by 480-line STN panel is split into upper and lower panels, each having 240 lines, as shown in Figure 2. Two rows of 640 pixels – one line from the upper panel and one line from the lower – are turned on in parallel, and are held on for the duration of one scan line. This increases the refresh rate for the STN panel, and helps reduce flicker artifacts.

Grayscales on STN Panels

The pixels on the STN LCD panel are intrinsically capable of only two states: ON (light shining through) or OFF (no light). To emulate color pictures requires more than two states. This is where the LCD controller comes in. LCD controllers borrow an idea from printers called *halftoning*. Printers, like STN LCD panels, can print only black or white. However, if a small square of characters (or pixels in the case of the LCD panel) are grouped together, and various combinations of the individual characters (pixels) are turned ON or OFF, this creates *grayscales*. The human eye integrates the various patterns of ON and OFF characters (pixels) and perceives them as an intermediate grayscale. LCD controllers use the same technique to create up to 64 grayscale levels. This technique, called *dithering*, can be used to emulate color images.

Creating More Colors on TFT Panels

With the current level of TFT LCD technology, up to 8 discrete light-intensity levels can be created for each red, green, or blue subpixel. This equates to a total of 512 colors that can be displayed per pixel (8 red x 8 green x 8 blue = 512 color combinations). By 1993, there should be 4,096 colors available (16 intensity levels for each subpixel). The intrinsic color capability (either 512 or 4,096 colors) of the TFT panel can be expanded to 185,000 colors through the LCD controller module that interfaces with the LCD panel.

For TFT LCD panels, the dithering technique has been taken one step further. TFT LCD panels use *spatial dithering* and a new technique called *frame-rate modulation.* The spatial-dithering technique is similar to that used on STN LCD panels; that is, small groups of red/green/blue subpixels (referred to as *logical* pixels) are turned on in various combinations of the 8 different intensity levels.

TFT LCD panels offer several advantages over STN panels for color, contrast ratio, viewing angle, and response time.

Again, the human eye integrates these various color combinations and perceives them as new colors. At the same time, the technique of frame-rate modulation is employed. Frame-rate modulation is really just dithering in the time domain. LCD panels display one frame every 60th of a second. The various subpixels making up the logical pixel are turned OFF or ON over time in alternate frames that the LCD panel displays. Again, the human eye perceives this as a new color. The dithering and frame-rate modulation techniques can create more than 185,000 perceived colors on current LCD panels. One drawback of these techniques is that they increase the perception of flicker. The algorithms for dithering and frame-rate modulation are adjusted to reduce the amount of flicker to a minimum.

Cost Trade-offs

TFT LCD panels offer several advantages over STN panels for color, contrast ratio, viewing angle, and response time. However, there are cost trade-offs. TFT panel manufacturing technology is still not mature, and manufacturing yields are very low. Consequently, there is a significant cost differential between TFT and STN LCD panels. Color TFT panels cost about three times as much as monochrome STN panels. As the yields on TFT panels increase, this cost differential will narrow, but there will still be a market for STN panels into the foreseeable future.

***Jim Paolantonio** is an advisory electrical engineer for Visual Subsystems within IBM Entry Systems Technology in Boca Raton. His current responsibilities include VLSI® LCD controller-chip development for portable PCs. Jim's previous assignments included XGA video hardware development and systems development engineering for the IBM PC AT, XT 286, and PS/2 Model 80. Jim holds BS and MS degrees in electrical engineering from Purdue University.*

The OS/2 Workplace Shell

Rob Talley
IBM Corporation
Dallas, Texas

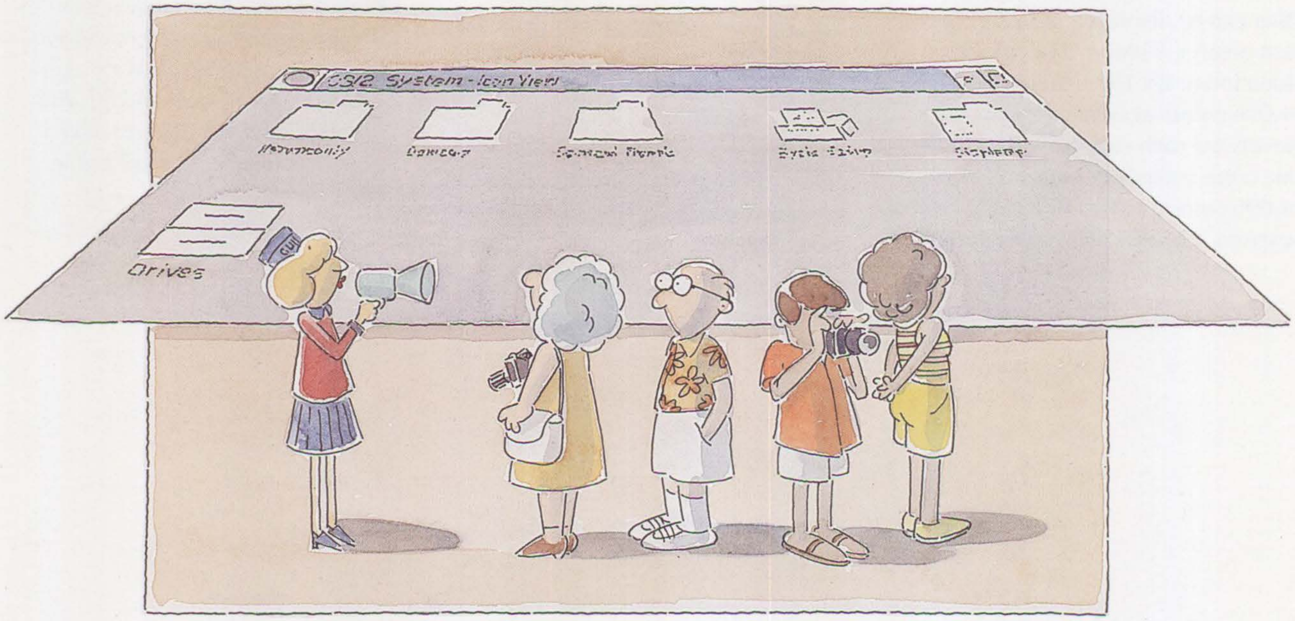
This article gives a tour of the Workplace Shell in IBM Operating System/2® 2.0, as seen from the perspective of a user who is familiar with previous versions of OS/2 and Microsoft Windows® 3.0.

The OS/2® Workplace Shell is a fourth-generation user interface. With its object-oriented approach, the OS/2 Workplace Shell adapts the way the computer works to the way people work. Third-generation interfaces, such as Microsoft Windows 3.X and OS/2 1.3, put a colorful, pictorial face on top of second-generation command menus. Users of third-generation interfaces, however, still must understand how the operating system is structured, and they have to structure their work that way. In contrast, even novice computer users will find the OS/2 Workplace Shell easy to learn because it is more intuitive.

Since OS/2 1.0 was introduced in 1987, IBM has been systematically collecting, analyzing, and validating user feedback about usability. The successively improved interfaces – from the menus of OS/2 1.1, to the windowing graphical user interface of OS/2 1.3, to the object-oriented Workplace Shell of OS/2 2.0 – are in direct response to user feedback.

IBM OS/2 usability researchers and interface designers in Boca Raton, Florida work closely with their counterparts in IBM's Common User Access™ (CUA™) architecture group in Cary, North Carolina. The CUA group combines user requirements with the available body of industry usability research to develop the CUA interface specifications. The Boca Raton group performs detailed interface design, validation, and refinement. The OS/2 Workplace Shell is based on IBM's CUA '91 specifications, with refinements from the research done in Boca Raton. The underpinnings of the Workplace Shell are the System Object Model (SOM) constructs. SOM provides a standard method of interaction between functions that are written using conventional languages and functions written using object-oriented languages. This gives conventional language applications the power and features found in most object-oriented applications.

Thanks to the unprecedented scope of OS/2's beta test program, several thousand users have contributed to



the design of the OS/2 Workplace Shell. Many suggestions received from early users of the Workplace Shell have been implemented in OS/2 2.0.

Getting Started

By understanding a few basic concepts of the OS/2 Workplace Shell, you will quickly see the powerful capabilities of this new user interface. The following sections will help you get started.

The Tutorial

When OS/2 2.0 is booting for the first time after installation, the Tutorial appears in an open window on the screen. You can view the Tutorial while the system completes the initialization process. The Tutorial introduces the main concepts of OS/2 2.0 and the OS/2 Workplace Shell. Topics in the Tutorial include Using the Mouse, Using Objects, Using Windows, OS/2 System Overview, and How to Get Help. The Tutorial, recommended for even the most seasoned PC user, takes 30 to 45 minutes and is well worth the time.

Objects

When you are working at your desk (not at your computer), you typically have several things on your desk: letters, documents, file folders, pens, a stapler, and a notepad. Each of these items can be thought of as an object on your desktop. The OS/2 Workplace Shell was designed to recognize the importance of everyday work. The Workplace Shell has its own *OS/2 Desktop*. The attributes of the computer that you work with – data files, programs, and physical components – are *objects* on your OS/2 Desktop. These objects appear on your computer screen as pictures, or

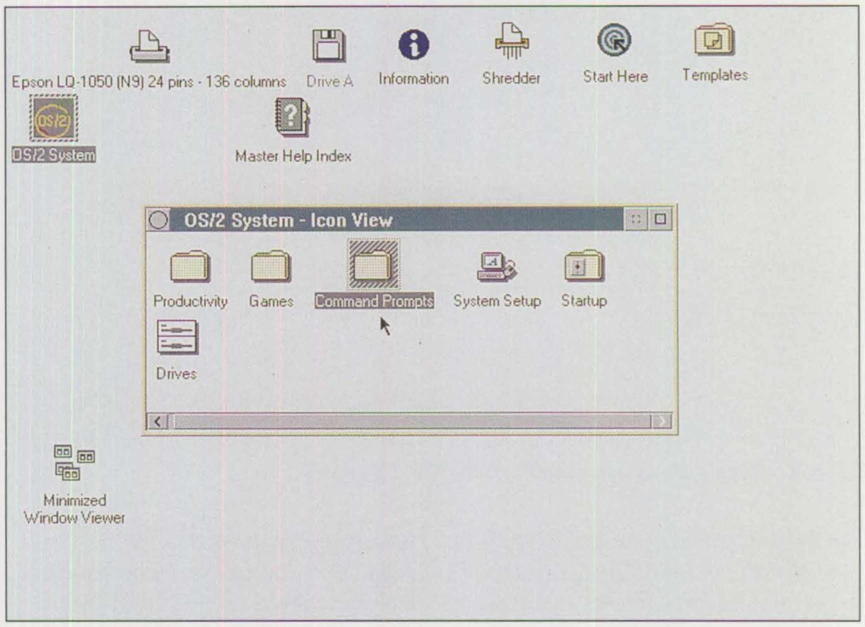


Figure 1. OS/2 Desktop

icons. The OS/2 Workplace Shell lets you create objects that reflect the way you work. It also lets you change the appearance of your OS/2 Desktop to suit your needs. Object orientation is one of the keys that makes using the OS/2 Workplace Shell intuitive.

OS/2 classifies objects into four types:

- A *data-file* object contains information. Text files, spreadsheet data files, memos, letters, documents, image files (drawings), video, and sound are examples of data-file objects.
- A *program* object represents an executable program file. Word processors, text editors, terminal emulators, spreadsheets, and games are examples of program objects.
- A *device* object represents a physical component of your system. Examples include printers, drives, and CD-ROMs.
- A *folder* object is a container for other objects. When a folder contains another folder, the second folder is called a *subfolder*. Experienced OS/2 and Microsoft Windows users may think of a folder as similar to a group, although folders have more function than their predecessors. A folder may also be thought of as similar to a directory, and a subfolder as similar to a subdirectory.

OS/2 Desktop

Previous versions of OS/2 and Microsoft Windows were organized differently from OS/2 2.0. They required that you work within windows that had titles such as Desktop Manager, File Manager, Print Manager, Program Manager, and Control Panel. You had to switch from one manager to another to perform related tasks. The OS/2 Workplace Shell has none of these managers. The user no longer has to know how the system is struc-

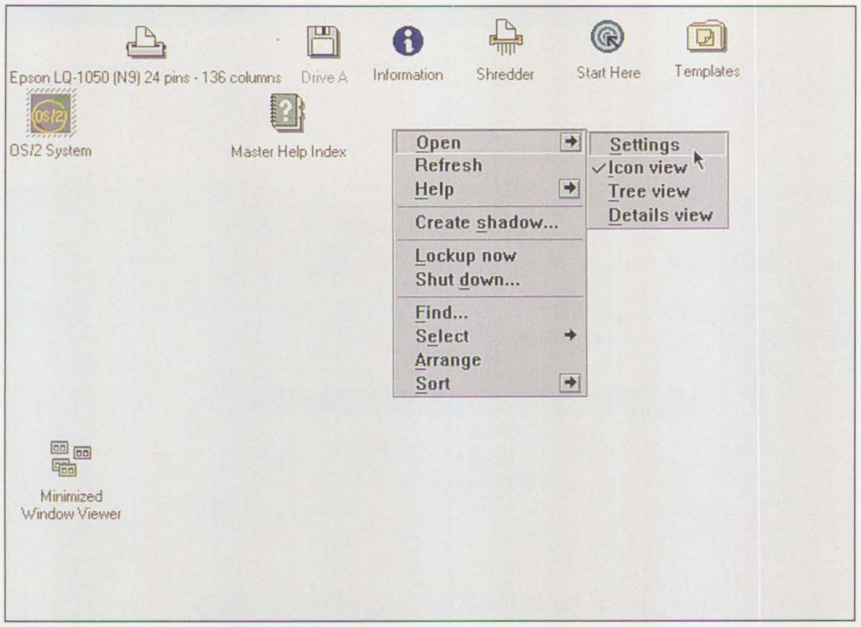


Figure 2. Pop-up Menu Associated with the OS/2 Desktop

tured to be able to do work. All functions that the various managers used to perform are now readily available while the user is working with objects.

The OS/2 Desktop, shown in Figure 1, is a special folder that fills the entire screen and contains all other folders and objects. Experienced OS/2 and DOS users may think of the OS/2 Desktop folder as similar to the root directory.

Menus

A unique feature of the OS/2 Workplace Shell is the way you access the menus associated with objects. Instead of having a menu bar for each object in the OS/2 Desktop, as in OS/2 1.3 and Windows 3.0, OS/2 2.0 introduces pop-up menus for each folder and object, including the OS/2 Desktop. A pop-up menu is a context menu. A context menu contains the actions and choices available for a type of object, a specific object, or a specific folder.

Context menus provide a better interface. You can access an object's actions through context menus more directly than through other types of menus, because you do not have to select a choice from a menu bar first. Context Menus allow you to see all the actions that can be performed on an object in its current state and only those actions that can be performed. If you select multiple objects, the context menu will indicate all the actions that can be performed on all the selected objects in their current state.

To access an object's pop-up menu, place the mouse pointer over the object, then click mouse button 2. Remember mouse button 2? In previous versions of OS/2, it was used only to drag an object from one group to another. Version 2.0 makes much more extensive use of this button. (*Note:* For the remainder of this article, mouse button actions that do not specify a button are referring to mouse button 1; references to button 2 are explicit. Whether the mouse is configured for the right or left hand, button 1 is the button pressed by the index finger.)

To display the pop-up menu associated with the OS/2 Desktop, place the mouse pointer on the *background* area of the OS/2 Desktop. (Be sure the mouse pointer is *not* on any of the other objects.) Then click mouse button 2; the pop-up menu appears, as shown in Figure 2.

The OS/2 Desktop menu contains a choice called Shutdown. Shutdown will flush all buffers and close all open windows on your system. This allows applications to notify you if you have made changes to data files that have not been "saved." The Workplace Shell uses "Lazy Write" technology to save all changes to the objects as you make them. This preserves the look of your desktop and allows you to bring up your system in the same state as when you shutdown. Even with a power failure, most of your changes will be preserved.

If additional options are available for a menu choice, an arrow appears to the right of the pop-up menu choice. In the OS/2 Desktop pop-up menu, the options Open, Help, Select, and Sort have additional choices. Placing the pointer on the arrow next to Open and clicking once reveals a cascade menu with four additional options: Settings, Icon View, Tree View, and Details View. The check mark to the left of Icon View shows that Icon View is the default view of the system. The default view of the OS/2 Desktop is the view shown in Figure 1.

Contents of pop-up menus vary from object to object. However, some choices appear in most pop-up menus: Open, Move, Copy, Print, Delete, Create Another, Create Shadow, and Help.

Basic Functions of the OS/2 Workplace Shell

Working with objects on your OS/2 Desktop is easy. While the OS/2 Workplace Shell lets you use the keyboard instead of a mouse, this article focuses on using the mouse to manipulate objects. Here are typical actions you can perform.

Opening Objects

Each type of object can be opened by double-clicking on that object, just as in previous versions of OS/2 and Windows.

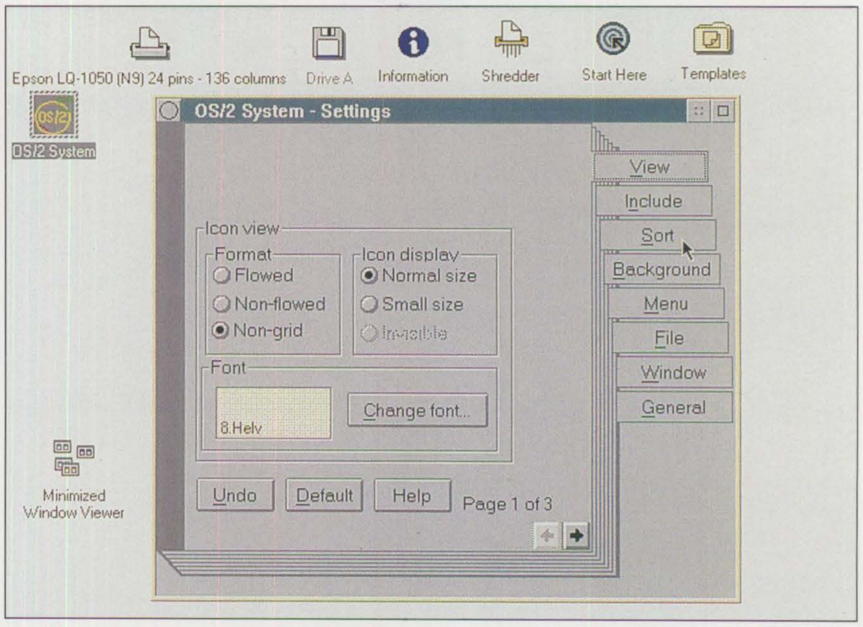
When you select Open from a pop-up menu, the resulting default action depends on the type of object that you are opening. If the object is a program object, selecting Open starts the program. If it is a device object, the current view or the settings view (if there is no current view) is shown. Opening a folder displays the contents of that folder. Opening a data-file object starts the OS/2 System Editor and loads the data file by default. In addition, if the data-file object being opened is associated with a program object, then two events occur: the executable program is loaded and started, and the data file is opened.

Moving Objects

You can easily arrange your OS/2 Desktop by moving an object from one location to another. To move an object, use mouse button 2 to drag the object to the new location. An object can be moved to a different location within the same folder or to another folder.

Copying Objects

The OS/2 Desktop can be customized by creating multiple instances of an object in various locations. To create an exact duplicate of an object (and contents of folders), including all the settings, make a copy of the object. Suppose you want to have an OS/2 Window command prompt located on the OS/2 Desktop as well as in the Command Prompts folder. Using the mouse, place the pointer over the OS/2 Window object. Then hold down the Ctrl key button and drag the object, using mouse button 2, to the OS/2 Desktop. (During the copying process, the object being copied is displayed as a faded object.) When you have positioned the object where you



OS/2 System - Settings

Icon view
Format: Flowed / Non-flowed / [✓ Non-grid]
Icon display: [✓ Normal size] / Small size / Invisible
Font: 8.Helv

Figure 3. OS/2 System Settings Notebook

want to place it on the OS/2 Desktop, release mouse button 2.

Creating Another Object

To create a second object of the same type, use Create Another from the object's pop-up menu. The new object contains the default settings for the original object type. You can also create another object type by using a template of the object type from the Templates folder, as discussed later in this article.

Creating Shadows of Objects

A *shadow* of an object contains all the contents and settings of the original object; that is, a shadow appears to be identical to the original. In reality, the shadow object contains pointers back to the original object. If you later make a change to one of the properties of the shadow object, the change is actually made to the original object. Likewise, a change in the original object appears in the shadow.

If you want your object to appear in another OS/2 Desktop container, you should create a shadow rather than a copy. There are two reasons for this: 1) you will never have to be concerned about keeping the shadow and the original in sync; and 2) it saves disk space, because the shadow uses a pointer to the original object, rather than duplicating the original object and all its contents.

To create a shadow of an object, first place the pointer over the object, then press and hold mouse button 2. Next, press and hold both the Ctrl and Shift keys while dragging the object to the location where you want the shadow to be placed. Notice the line connecting the original object to the shadow object that you are dragging. When you have the shadow object where you want it, release mouse button 2 first, then release the Ctrl and Shift keys. The shadow's icon text is a different color from the text of the original object to identify the object as a shadow.

Using Find

You can locate objects on the OS/2 Desktop by selecting Find from any contextual pop-up menu. Suppose you want to locate the EPM editor. Place the mouse pointer on the background of the OS/2 Desktop, click mouse button 2 to bring up the

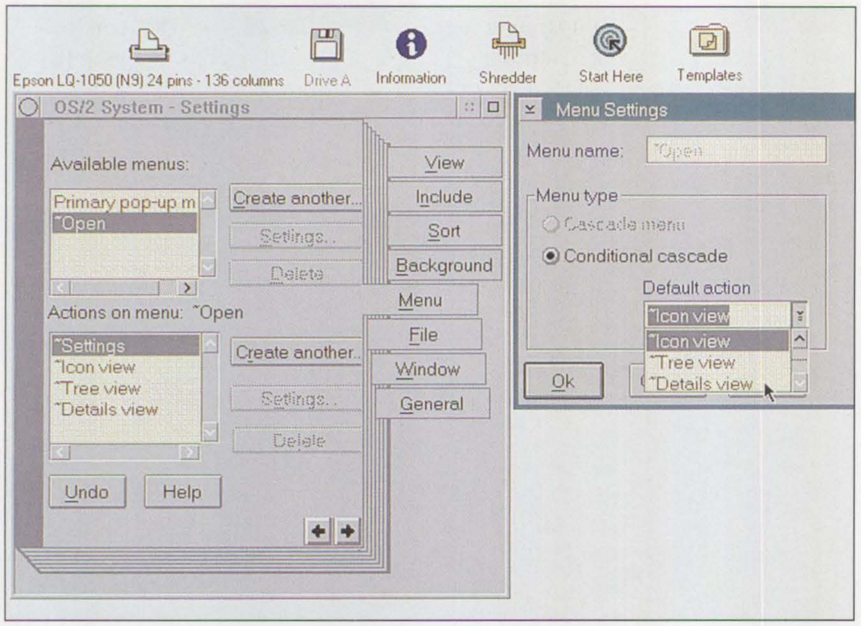


Figure 4. Changing the "Open" Default Action

contextual menu, then select Find. This brings up a pop-up window with an entry field where you type "EPM.EXE" and select the Program file from the list box. The Locate pushbutton gives you an opportunity to change the folder or drive from which the search starts. Radio buttons allow you to select whether to search all subfolders or just the folder from which the search begins. After you press the Find button, a Find Results window appears, displaying the EPM icon. You can now start EPM from the Find Results window by double-clicking on its icon.

Using Settings Notebooks

Every object (except the shredder) has *settings*. Settings are properties or characteristics of an object that tell that object how to behave. For example, you can view the settings of the OS/2 System object by bringing up its contextual pop-up menu. Then, place the pointer over the arrow next to the Open choice and click on it. Finally, choose the Settings option by again clicking the mouse button. This brings up the OS/2 System Settings notebook, which contains all the settings for the OS/2 System folder, as shown in Figure 3.

The same technique is used to open the Settings notebooks for all the objects in your system.

The Settings notebook has several pages with tabs, which show the types of information associated with the object. For the OS/2 System folder, View contains settings for the way icons are displayed within the folder. You can see the settings for the Details and Tree views by pressing on the arrow in the lower right corner of the visible View page.

The Include tab in the Settings notebook allows you to limit the types of objects that are visible in the folder. For example, you may only want to have data files in your folder. You would then highlight the Data Files selection in the Type list box.

The Background tab lets you select either an image bitmap or a color for the background of your folder.

The Menu tab provides a list of available menus, and sets the default behavior for the context menu associated with the folder. To change the default action to Details view, select the Open choice in the Available Menus list box. Select the Settings push button to the right of the Available Menus list box. This brings up a Menu Settings window with a Default Action data field. Selecting the down arrow beside the data field shows the choices available on the cascade menu, as seen in Figure 4. Now, select the Details View choice to make it the default when you next open the OS/2 System folder and select OK.

The File tab provides a place to enter a subject for the object, as well as the path and file name. Subsequent pages of the File setting will give the date and time that the file was created and last changed; the file size; its extended attributes; and file flags, such as Hidden and Read-only. The third page provides a chronological history of events relating to the object, a place for comments you want to include, and a place for key phrases you want to use for a search criterion using the Find choice from any pop-up menu.

The General tab contains the title of the object, a view of the icon, and a check box titled Template. Change the title of the object by typing the new name in the Title window on this page. Change the look of the Icon by invoking the Icon editor from the Edit Icon pushbutton. Finally, you can create a template of your customized object by checking the Template check box.

Associating Objects

You can associate data-file objects with program objects, so that the data file is displayed in a selected program when the program is opened. In previous versions of OS/2 and in Windows 3.0, this is done in the File Manager by first selecting the data file, then selecting Associate from the File menu. In the OS/2 Workplace Shell, you can associate an individual data-file object with a program object. To make this type of association, open the Settings notebook for your data file, and select the Menu tab. Select Open from the Available Menus list, then select the Create Another pushbutton next to the Actions on Menus list box. Next, enter the program name in the Menu Item field as you want it to appear on the object pop-up menu. Now, enter the full path and program name in the Program Name field. If you do not know the full path, use the Find pushbutton to obtain it. When you have entered the proper path and program name, select the OK pushbutton.

Program objects can be associated with data files in a similar fashion. From the Program object's Settings notebook, select the Association tab. To associate by file type, select one or more types from the Available Types list, then select Add. To associate by file name, type a file name in the New Name field, then select Add. Global file name characters can be used. For example, using the program object settings for your favorite editor, type *.DOC to associate all data files that have an extension of .DOC with the desired program object. Double-clicking on one of these data files will invoke your program and load the data file.

When you associate a program object with a type of data-file object, the program is listed as an additional choice on the pop-up menu for all data-file objects of that type. This means that once you associate a program object with a type of data-file object, you will open both the program and the associated data-file by double-clicking on the data file object.

Guided Tour of the OS/2 Desktop

OS/2 1.3 users will quickly notice significant differences in version 2.0. For example, where is Main? For that matter, where are any groups? Where is the Print Manager icon? Gone! In OS/2 2.0, you see a screen (Figure 1) that represents your OS/2 Desktop. It contains several small graphical images (icons), labeled Start Here, OS/2 System, Shredder, Master Index, Drive A, Information, and Templates. If you chose a printer during the installation of OS/2, you will also see an icon containing the name of the printer you installed.

Start Here

The Start Here object has information to help you begin learning about OS/2 2.0, adding and using programs, customizing your OS/2 Desktop, finding information, using the Help facility, printing, and sharing data. Selecting a topic from the Start Here object will lead you to in-depth information for complete customization of your system and for performing your everyday tasks.

OS/2 System Folder

Unless the Tutorial has appeared (because it is the first time this copy of OS/2 2.0 is being used), the first thing you see when you start OS/2 2.0 is an open window titled OS/2 System – Icon View. That window contains several icons that look like manila folders in Figure 1. The icons are titled Productivity, Games, Command Prompts, and Startup. Another icon, labeled Drives, looks like a picture of two disk drives with one stacked on top of the other. Yet another icon, titled System Setup, looks like a PC.

To explore the Command Prompts icon, double-click your mouse on it. You will see the familiar system command prompts: OS/2 Window, OS/2 Full Screen, and DOS Full Screen. You will also see three command prompts that are new in OS/2 2.0: DOS Window, WIN-OS2 Full Screen, and WIN-OS2 Window.

Try the WIN-OS2 Full Screen object by double-clicking on it. OS/2 switches to full-screen mode and, after a few seconds, the WIN-OS/2 Program Manager (shown in Figure 5) appears on the screen. The WIN-OS/2 Program Manager screen looks like the Windows 3.0 Program Manager screen under DOS, except that the Games window is missing. There are also two new icons on the screen,

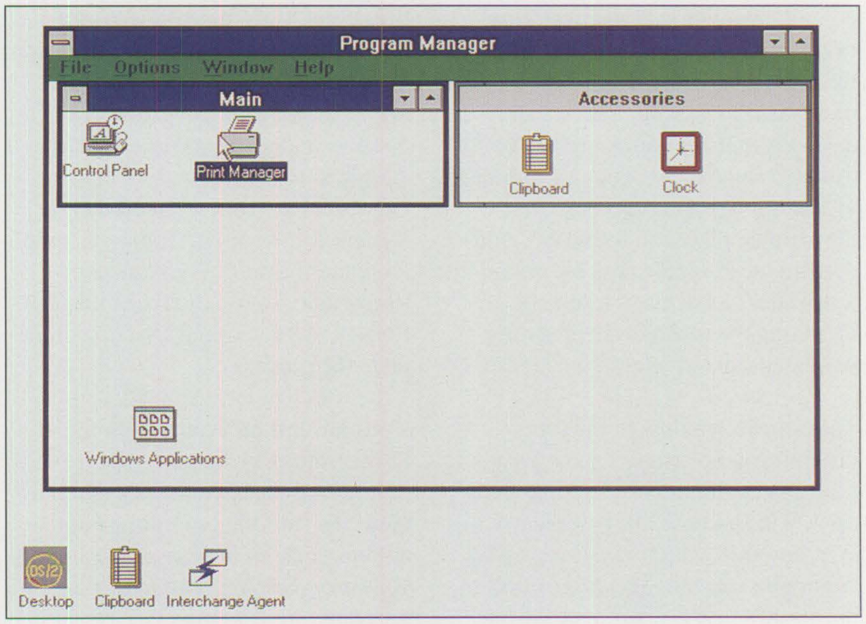


Figure 5. The WIN-OS/2 Program Manager in OS/2 2.0

below the WIN-OS/2 Program Manager: a Dynamic Data Exchange (DDE) Interchange Agent and an OS/2 Desktop object. The DDE Interchange Agent enables OS/2 applications and Windows applications to share data dynamically. Double-clicking on the OS/2 Desktop object will return to the OS/2 Desktop, leaving the WIN-OS/2 session running in the background.

Now, close the WIN-OS2 Full Screen session by double-clicking on the title bar icon at the top left corner of the WIN-OS2 Presentation Manager. This returns you to the OS/2 Desktop.

You can similarly explore the other command prompts. They behave the same way they did under OS/2 1.3 with one exception: whenever you click on a window's Minimize icon, the icon representing the object is placed in the Minimized Window Viewer (MWV). All non-system objects will, by default, be placed in the MWV. System objects will have a Hide button instead of a Minimize button, and will, by default, be hidden in their original container. Hash marks on the object indicates a view of the object is open. This is the default behavior for OS/2 2.0. However, the default can be changed (in the Systems folder) to display an icon on the OS/2 Desktop, which indicates a minimized open object. You can also change the behavior of individual non-system objects so that they can be placed in the MWV, hidden or placed on the desktop regardless of the global system setting. Changing the global system setting will be discussed later in the article.

To restore a window to the foreground, you no longer have a Task List, as you did in OS/2 1.3 and Windows 3.0. In OS/2 2.0, you use a Window List, which you bring up by either pressing Ctrl-Esc or positioning your mouse pointer over the background of the OS/2 Desktop and simultaneously pressing mouse buttons 1 and 2. The Window List shows all the applications and window views that are currently open in the system. In the Window List, objects are listed in the order in which they were opened on the OS/2 Desktop.

Once you select (highlight) items in the Window List, you can display a pop-up menu to show, hide, cascade, or close open windows and applications by pressing mouse button 2.

Now, let us explore the contents of other folders in the OS/2 System window.

Drives Object: Opening the Drives folder reveals individual drive objects for each logical drive on your system. You can view your directories and files in a tree structure similar to that used in the Windows 3.0 and OS/2 1.3 File Managers. To view the details of the files on the drive, bring up the context pop-up menu for the drive, then select Details.

In the OS/2 1.3 and Windows 3.0 File Managers, the Disk menu includes commands that enable you to format diskettes and copy files or entire diskettes. In the OS/2 Workplace Shell, these commands are located on the pop-up menu for the diskette drive or hard disk drive. To format a diskette in drive A:, select the Drive A object on the Desktop or in the Drives folder from within the OS/2 System folder. Now, display the pop-up menu for the Drive A object. (Remember to use mouse button 2.) Finally, select Format Disk to complete the process.

You can start programs from the Drives object by locating your program using the Tree, Icon, or Details view. To start the program, simply double-click on the icon representing your program. You can also drag the program object to the OS/2 Desktop or to a folder of your choice, or make a shadow of the object anywhere you want one to be.

Productivity Folder: The Productivity folder contains several small applications that are designed to get the new user started and productive. The programs include: Seek and Find for locating files on the system; PM Chart for creating presentation graphics; PM Terminal for ASCII communications; Calculator; Calendar; Daily Planner; Sticky Pads; the OS/2 System Editor; and a new Enhanced Editor. Also in the Productivity folder is Picture Viewer, which encompasses the three Picture Utilities in previous versions of OS/2.

Games Folder: The Games folder contains several games that not only offer entertainment, but also help users become proficient with mouse movement and object manipulation. The games are Jigsaw, OS/2 Chess, Cat and Mouse, Scramble, Reversi, and Klondike Solitaire. Jigsaw and Klondike Solitaire are excellent for learning drag-and-drop functions (also see "New Applications in OS/2 2.0" in this issue).

System Setup Folder: The System Setup folder contains several system-wide controls and customization functions just like the Control Panel available in OS/2 1.3 and Windows 3.0. The Print Spooler and the Scheme, Color, and Font palettes are available in the System Setup folder. Customization settings for country, system, sound, system clock, keyboard, and mouse are also available in the System Setup folder.

For example, to customize screen colors, open the Color Palette object in the System Setup folder. To change the default settings for your mouse, open the Mouse object. To change the default settings for some keys, or to activate Special Needs (a new feature), open the Keyboard object.

The system object allows you to change behavior of your system. You can determine whether confirmation pop-ups will appear for delete functions, whether minimized objects appear on your OS/2 Desktop as they do in OS/2 1.3, whether Print Screen is enabled, and whether opening a currently opened object will yield the originally opened object or a second copy of that object.

To easily create program objects of applications that were previously installed on your system, use the Migrate Applications utility. Opening this utility brings up a Find Programs window, which has a list box containing all the drives in your computer. Highlight the drive(s) containing the applications to migrate. There is also a data field that designates the default database that contains all the predefined settings for selected applications. There are also three check boxes designated as DOS, Windows, and OS/2 programs. Select the types of programs to migrate, and press the Find button. A Migrate Programs window appears with a listing of all the selected types of applications on your system that were identified by the database.

To migrate applications not listed in the default database, press the Add Programs button. This action causes all other .EXE, .BAT, .CMD, and .COM files to be listed in an Add Programs window. You can then select or deselect these files for migration. Pressing the Migrate button will create the program objects identified in the default database and place them into a folder entitled "DOS Applications" or "WIN-OS2 Applications." All the program object settings will be preset for each of the applications migrated to these folders. All other program objects are placed into a folder called "Additional DOS Applications," "Additional WIN-OS2 Applications," or "Additional OS/2 Applications." You may need to alter some of the settings for these program objects to allow them to run optimally on your system.

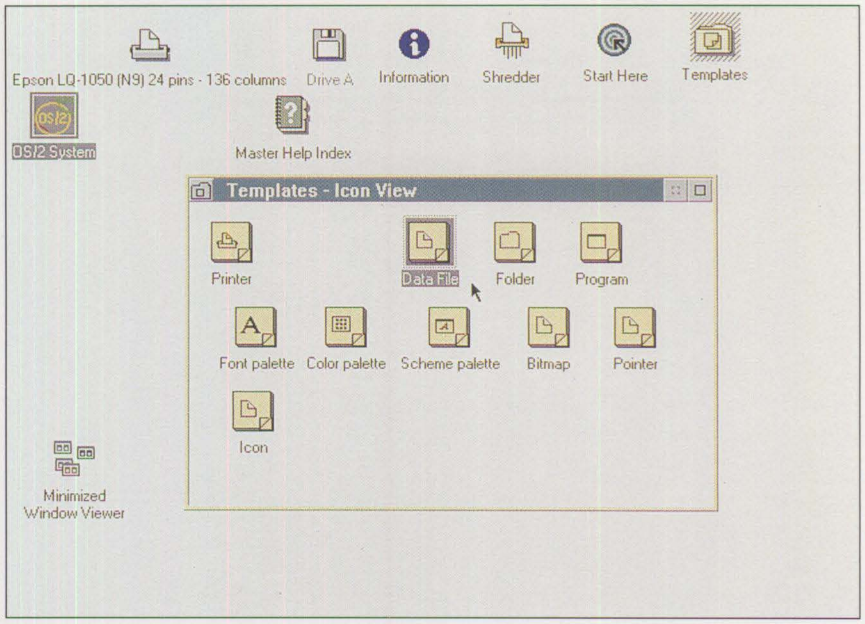


Figure 6. Templates Folder

If DOS and Windows 3.0 were on the computer before OS/2 2.0 is installed, the migration program also picks up the Windows 3.0 .INI files and the Windows 3.0 groups to use in OS/2 2.0. You can verify this by clicking on the Win-OS2 Full Screen icon in the Command Prompts folder. You will see the entire Windows 3.0 Presentation Manager that you had previously installed under DOS.

Another icon in the System Setup folder is the Selective Install folder. If you did not initially install an OS/2 feature in your OS/2 environment, but you later decide to install it, you can double-click on the Selective Install icon. You will see the selection menu that you would have seen had you chosen the Select Features and Install option during your initial installation of OS/2 2.0.

Startup Folder: The Startup folder is intended to contain program objects that are to be started during system initialization, but are not running when you shut down your system. For example, a batch file that starts a network server or requester would be placed in the Startup folder. To place a copy of your program object in the Startup folder, locate the program's icon using either the Find function or the Drives folder. Then, drag the program icon (using mouse button 2 while simultaneously holding down the Ctrl key) to the Startup folder, and release the mouse button and the Ctrl key.

Every time your system starts, the program objects in the Startup folder will automatically start.

Note: You should *not* place a copy of objects in the Startup folder that you normally start from within another folder. If a program is left open on your OS/2 Desktop when you shut down your system, the next time the system is booted, you will have two copies of that program on your OS/2 Desktop: one from the Startup folder and another that was opened from the program object folder.

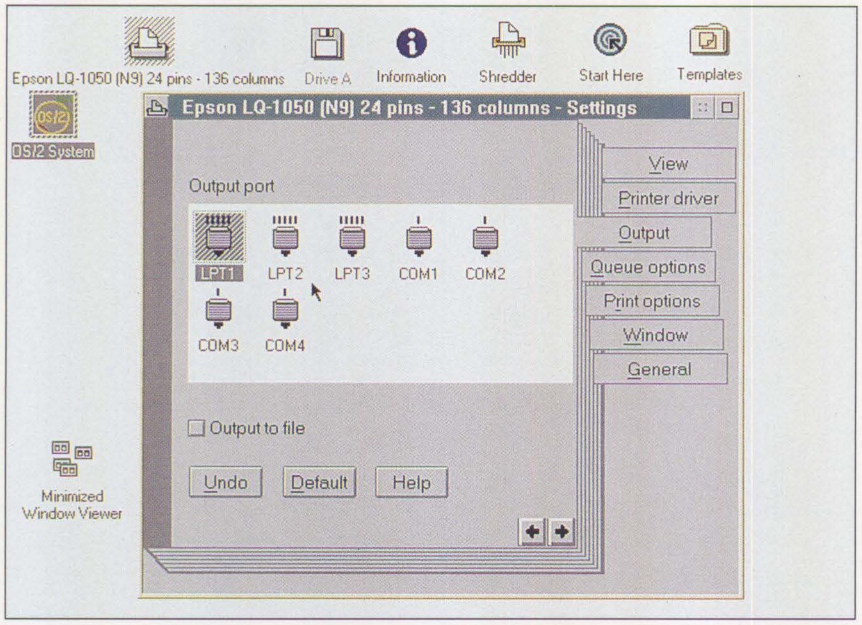


Figure 7. Printer Settings Notebook

Templates Folder

Double-clicking on the Templates folder brings up the Icon View of the folder, shown in Figure 6. Inside the folder are templates for Printer, Program, Font Palette, Color Palette, Scheme Palette, Data File, Folder, Bitmap, Metafiles, PIF files, Pointer, and Icon. A template has properties that distinguish it from other objects. Think of a template as a tear-off pad of blank object types.

To create a new folder, simply drag the folder template (using mouse button 2) to your OS/2 Desktop, and release the mouse button. This action creates a new folder on your OS/2 Desktop.

Name the folder using the direct name-editing features. To do this, place the mouse pointer on the name of the folder. Since you have just created it, its name is Folder. Press and hold the Alt key, then click the mouse button. This changes the name field to a field that you can edit. Now, type the new name for the folder. When you have finished with the name, move the mouse pointer away from the name field, and click the mouse button.

Master Help Index

The Master Help Index provides a wealth of information about OS/2 2.0. It is also the best place to get acquainted with the notebook metaphor. The Master Help Index notebook contains information ranging from accessing diskettes, to changing keyboard settings and object names, to write-protecting diskettes.

Information Folder

Double-clicking on the Information folder icon shows several icons. If you installed the Tutorial, its icon will be in the Information folder. During the installation of OS/2 2.0, other icons, such as the Command Reference, REXX Information, and Glossary, can be selected to be included in the Information folder.

The Command Reference icon contains all OS/2 and DOS commands that OS/2 2.0 supports. The explanation for each command includes a short narrative about the purpose of the command and all the parameters available to the command. The REXX Information icon has details about REXX, from how to get started in REXX to advanced REXX techniques. The Glossary defines many terms used in an OS/2 environment.

Print Object

The OS/2 Workplace Shell sets up print objects that determine how and where files will be printed. If a printer was selected during system installation, the print object that represents that printer is located on the OS/2 Desktop. To print the contents of an object, drag the file to the print object using mouse button 2. OS/2 then invokes a data handler that enables the printer to print the data-file object.

You open the Printer context menu by clicking on the Printer icon with mouse button 2. The context menu shown in Figure 7 includes options for setting the default printer and changing the status of the printer. The status of any individual print job can be obtained by selecting Details from the Open choice on the context menu. Selecting the Settings icon opens the Printer Settings notebook. The notebook allows you to choose either the Icon View or Details as the default view of the printer. The printer driver can be selected, as well as the output port it uses. Separator pages, print queue drivers, and timing considerations can all be set from the Printer Settings notebook.

If you need to install a printer after you have installed the operating system, use the Print template located in the Templates folder.

Shredder

Objects can be deleted by dragging them (using mouse button 2) to the shredder, then releasing the mouse button. If the object you delete is a data file, both the object and your data are deleted. If the object is a program object, it is deleted, but its associated executable (that is, program) remains on your hard drive. If the object is a folder, the folder and its contents are deleted from your system.

When you drag a shadow of an object to the shredder, a pop-up window informs you that you are about to delete a shadow of an object that is located elsewhere on the OS/2 Desktop. Deleting the shadow has no effect on the original object. If you want to delete the original object, which will also delete all of its shadows, you must locate the original object and drag it to the shredder. When you choose to delete an original object, a pop-up window asks you to confirm that you want to delete the original object and all its shadows. You can either confirm your decision or change your mind.

Additional System Functions

Several additional functions are available to assist you in working with OS/2 2.0.

Using the System Editor

The System Editor is the default editor in OS/2 2.0, and is similar to the System Editor in previous versions of OS/2. In Windows 3.0, the Notepad program is used to create and edit data files.

If you open a data-file object that is not associated with any other program object, it will, by default, be associated with the System Editor.

To edit a new file, you create a data-file object. To create the object, open the Templates folder, and select the Data-file template. (A template has properties that distinguish it from other objects. In this example, the data-file template has properties that tell the operating system "this a blank data-file object.") Drag the data-file template to any blank location on the OS/2 Desktop folder. This action creates a new, blank data-file object. To edit the data-file object, double-click on the object. This action starts the System Editor.

Lockup

The Lockup program that locks the computer system is located in the OS/2 2.0 Settings notebook for the OS/2 Desktop folder. To access the Lockup feature, display the pop-up menu for the OS/2 Desktop folder. To do this, move the mouse pointer to any blank area of the OS/2 Desktop folder. Then press mouse button 2. Next, move the mouse pointer to the arrow to the right of the Open choice, and press mouse button 1.

The Lockup feature provides automatic dimming of your display screen.

Select the Settings choice to bring up the Settings notebook. Finally, select the Lockup tab.

The Lockup feature provides automatic dimming of your display screen, which helps prevent image burn-in. Also, if you do not enter any keystrokes within a certain interval of time (which you can set), the Lockup program is invoked.

Shutting Down Your System

In Windows 3.0, you exit by selecting the Exit Windows choice from the File pull-down menu in the Program Manager. You could save the current layout of your Program Manager window by selecting the Save Settings choice on a pop-up window that appears every time you exit.

In previous versions of OS/2, you use the Shutdown choice from the Desktop pull-down on the Desktop Manager window. You have the choice of setting applications or groups to open at startup if so desired.

In the OS/2 Workplace Shell, your layout is saved each time you make changes. To shut down the system, use mouse button 2 to bring up the OS/2 Desktop context menu, then select Shutdown. The next time the system is booted, all open objects will be returned to an open state. Program objects started from the Desktop will have the same data file loaded that was loaded when the system was shut down.

Summary

OS/2 2.0, a powerful operating system, can run OS/2, DOS, and windowed applications simultaneously in individual sessions. The OS/2 Workplace Shell, the fourth-generation user interface, is more powerful than any other IBM PC user interface. Its capabilities, working as you want your system to work, are nearly boundless. Customization of your OS/2 Desktop objects is no longer a science – in OS/2 2.0, it is easy to learn and easy to do.

Once you learn the basic features and interfaces of the OS/2 Workplace Shell, you will never want to return to the old ways of working with your computer.

Note: This product was under development at the time this article went to press. Additional features and changes may be forthcoming.

Rob Talley *is a senior market support representative in IBM's Personal Systems Services and Support organization in Roanoke, Texas. Rob has supported OS/2 since 1987; his key responsibilities include the OS/2 Workplace Shell, and DOS and Microsoft Windows emulation. He received a Bachelor's degree in business administration from the University of Texas at Arlington.*

New Applications in OS/2 2.0

Jean N. Shortley
IBM Corporation
Boca Raton, Florida

The 29 productivity and entertainment packages supplied with OS/2 2.0 will save time and money in selecting and installing useful applications. Users will find that they are instantly productive with OS/2, and that its applications are valuable in their professional and personal pursuits.

Operating System/2 2.0 comes with 29 new productivity and entertainment applications. These applications provide value immediately upon installation. All productivity applications and games are installed during the default installation of OS/2 2.0. If not enough disk space is available, you can use the selective installation process to choose which applications to install.

All 29 applications are Presentation Manager® (PM) graphics applications that run in protected mode. All can run simultaneously in separate windows that are either in the background or minimized.

To begin working with these new applications, double-click (select) the OS/2 System object from the Workplace Shell, then select either the Productivity or Games folder. Figure 1 shows the contents of the Productivity and Games folders under the Workplace Shell.

Games Folder

The six games that come with OS/2 2.0 include OS/2 Chess, Reversi, Klondike Solitaire, Jigsaw, Cat and Mouse, and Scramble. These games not only provide entertainment, but also help users get accustomed to the OS/2 environment.

OS/2 Chess is challenging and educational. Its online help gives a brief history of chess, its rules, and key strategies. OS/2 Chess tracks valid moves and elapsed time, and you can save a game and retrieve it later. Reversi is a game of logic in which you try to outmaneuver the computer while you fill the screen with your colored game pieces. You can increase the level of difficulty in Reversi as you progress. Cat and Mouse is a mouse skill-building game. When the mouse is moved, the cat follows it; when the mouse is idle, the cat naps. When the Hide option is enabled, the OS/2 Desktop is cleared until the cat is clicked on by the mouse or the Alt key is pressed.

All the games offer online help about the objects of the games, rules of play, ways to earn points, and even helpful hints. Users benefit from these games by becoming better skilled at interacting with the graphical user interface, manipulating a mouse, and interacting with the online help facility.

Productivity Folder

Inside the Productivity Folder are new productivity tools such as an enhanced text editor, an asynchronous communications tool, a small database, a file/text search utility, and a daily/monthly/yearly planner.

Text Editing

The Enhanced Editor supports editing multiple files at once. Other features include support for the OS/2 Desktop's Cut/Copy/Paste functions between open documents, printing from within a document, and fonts that can be displayed and printed. Fonts can be set for an entire document or for selected text within a document.

Multiple system fonts, attributes, and point sizes can be used in a single document. Additionally, the Enhanced Editor prints the document as it appears on the screen. You can preview the document by selecting WYSIWYG (What-You-See-Is-What-You-Get) mode, and you can print the viewed

OS/2 2.0, the 32-bit version of OS/2, is designed to take advantage of the advanced processing power of Intel 80386/80486 and compatible microprocessors. A few of the many new OS/2 2.0 features include an object-oriented (icon-oriented) graphical user interface called the Workplace Shell; support for multiple DOS sessions; support for Windows applications; and the Super FAT file system.

OS/2 2.0 runs on 80386-, 80386SX-, and 80486-based computers. OS/2 2.0 requires a minimum of 4 MB of RAM, although 6 MB is recommended. The base OS/2 2.0 system requires 18 MB of storage for installation; full installation of all features requires 30 MB. It also requires one fixed disk drive, one 3.5- or 5.25-inch high-density diskette drive, and a supported graphics display adapter. A mouse is highly recommended for efficient interaction with the new Workplace Shell. Additional hard disk space of 10 to 15 MB is also recommended for a swapper file (a file used by OS/2 to place inactive data onto the hard disk and to better utilize system memory for active data).

page or the entire document at that point. Other key features include a command line for users who prefer to enter commands from the keyboard; ASCII file import; an enhanced text search-and-replace; programmable Enter and Pad Enter keys for specific line functions; and a timed autosave feature.

The Seek and Scan Files utility is handy for searching one or more disks quickly for files or text. You can do a lengthy search in the background while you are busy doing another task on the OS/2 Desktop.

When a match is found (whether it is text or a file), it is displayed in a list box that can be browsed or edited. The file can be copied, erased, or renamed within the utility. If the file is executable, it can be started from Seek and Scan Files. If the file is a text file and the Enhanced Editor has been started, the desired file can be highlighted with mouse button 1. It is then grabbed by pressing and holding mouse button 2. Then it can be dropped on the Enhanced Editor icon (drag-and-drop). The file is automatically loaded as an Enhanced Editor document.

The OS/2 System Editor, included in earlier versions of OS/2, is also provided with OS/2 2.0.

Database

The database application included with OS/2 2.0 has many uses. One is to maintain a list of personal contacts that can include up to 5,000 records. (Data files can be created easily using the PM interface and pull-down menus.) Each record, displayed as it would appear on an index card, can contain eight fields (lines) of 30 characters each. Specific records can be found by performing a search on any field (name, phone, and so on). Data records that meet the search criteria are instantly displayed.

If your system is configured for a modem, the database can also be used as an automatic dialer. From the database, select Customize from the menu bar, then select the port corresponding to your modem. The following steps are used to configure a COM port under the Workplace Shell so that the database can access the modem:

1. Select any print icon.
2. Click on mouse button 2.
3. Select Open.
4. Select Settings.
5. Select Output from the Settings notebook.
6. Select the port you want to use.
7. Configure the communication parameters necessary for your modem.
8. Select OK.
9. Close the Settings notebook.

Also included in the database is a reporting feature. Customized reports can be created from the database.

OS/2 Desktop Diary

The Activities List, Daily Planner, Monthly Planner, Calendar, and

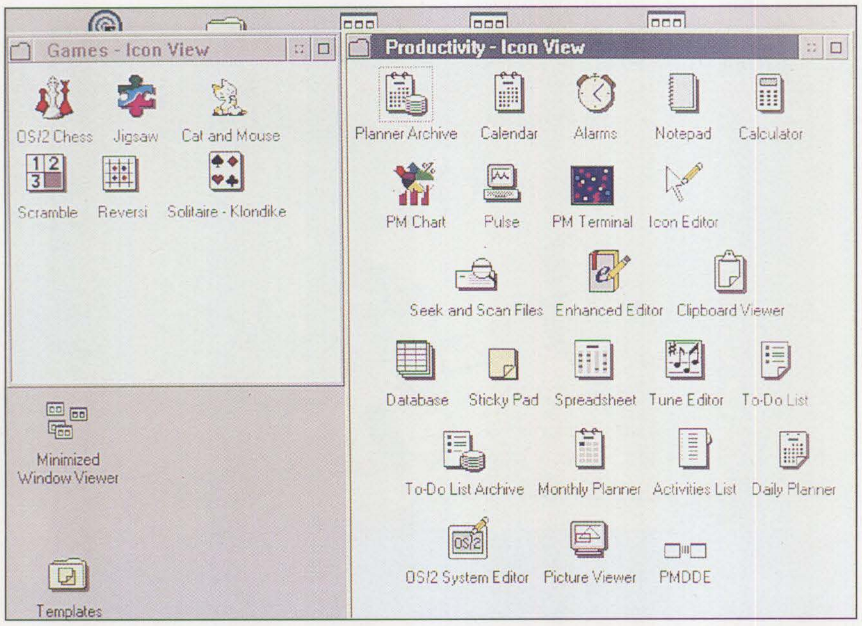


Figure 1. OS/2 2.0 Productivity and Games Folders

Alarm packages are interrelated to allow for efficient time planning on the OS/2 Desktop. Appointments and activities can be recorded in the Daily Planner by selecting the day from the Calendar, Monthly Planner, or Daily Planner objects. Daily activities can be seen in an abbreviated form in the monthly view. When a file has been opened, double-clicking on a particular day of the month in the Calendar object or Monthly Planner object automatically brings up the detailed Daily Planner. The Activities List package can generate a report, sorted either by description or by date, of a day's activities entered into the Daily Planner. An alarm with its own programmable tune can be set to go off before an appointment, and to repeat itself at preselected intervals. Additionally, a log of all activities is maintained.

The OS/2 Desktop's To-Do List is handy for prioritizing daily or weekly tasks that are separate from the appointments or activities entered into the Daily Planner. Tasks can be sorted by their assigned priority, date, or description. Both the To-Do List and planner objects offer support for archiving data on disk and retrieving it for status reports at a later time.

Communications

OS/2 2.0 comes with an asynchronous communications package, PM Terminal, by Softronics Corporation.

PM Terminal comes with entries in its Session Phonebook for most major bulletin boards. Using the PM interface, it is easy to add or modify any entries for a specific terminal emulator, connection path, screen color, or file transfer protocol. The package shipping with OS/2 offers nine major file transfer protocols, including XModem, YModem, and Kermit. One exception is the IND$FILE protocol used by IBM mainframe systems.

Any session can be customized to be started automatically when PM Terminal is opened, or the session can be started as minimized, full-screen, or windowed. Only one host session is supported from any connection.

Other features in PM Terminal include keyboard remapping, on-screen data capturing, and macro support.

Other Helpful Tools

In addition to the database, diary, communications package, and enhanced editor, OS/2 2.0 has other helpful tools to make you more productive. A mouse-supported calculator features a memory function and a tally tape that can be printed. The five-page notepad and the Sticky Pad utilities can be used to jot down ideas and thoughts as you would on a Post-It™ note. The Pulse program displays the system's usage in a continuously updated graph.

PM Chart, from Micrografx®, Inc., uses the OS/2 Desktop to create charts. Six different chart types – line, bar, horizontal bar, area, table, and pie – are offered to display spreadsheet data. Text with customizable alignment, style, and size can be applied to chart headings or titles. PM Chart can be used to create simple spreadsheets, and support is available to import already-created spreadsheets or data from Micrografx applications, Lotus 1-2-3®, Microsoft Excel, and others – nine data formats in all.

Note: This product was under development at the time this article went to press. Additional features and changes may be forthcoming.

***Jean N. Shortley**, a senior associate programmer in IBM's Entry Systems Technology Laboratory in Boca Raton, Florida, was formerly a member of the printer device-driver development and test teams and the Multiple Virtual DOS Machines (MVDM) test team. Jean holds a BS in education and an MEd in educational microcomputer research, both from Florida Atlantic University in Boca Raton, Florida.*

Unattended Installation of OS/2 2.0

Paul Brightly
IBM Corporation
Boca Raton, Florida

OS/2 is now faster and easier to install than ever. As a result, the cost of installing OS/2 and its applications can be drastically reduced, if not eliminated, by unattended installation. This article describes the features of the new OS/2 Install program, which allows flexible methods of installing OS/2 2.0.

How often does someone consider installation procedures when purchasing software? The typical answer to this question – hardly ever – has changed in recent years. Not long ago, software installations usually came with one or more diskettes, simple procedures, and somewhat bland interfaces. Today's software, particularly operating systems, requires more complex installation procedures using many diskettes. Understandably, computer users are demanding easier installation processes.

Consider the implications of installing software in a large corporate environment with many users. A large operating system is not easy for first-time users to install. Most large groups of users find it very expensive to install software. If the users do not understand how to install the software, they may have to hire consultants. The cost of installation can be as much as, or more than, the cost of the software!

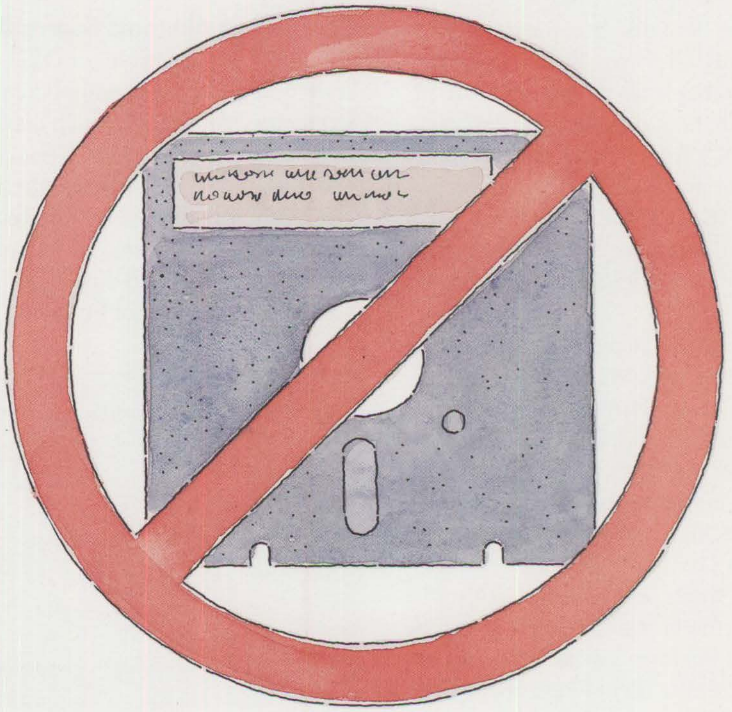
The OS/2 2.0 development team attacked this problem from all directions. OS/2 2.0 installation is more flexible about where it gets and installs source files. A new user interface makes the installation much easier to understand. It is even possible to have OS/2 install itself – unattended.

Flexible Source and Target Drives

OS/2 2.0 can be installed from almost anywhere – diskettes, CD-ROM, LAN, another hard disk, or any virtual device that looks like a disk – to anywhere with any interface. The installation can get its files from any disk connected locally, over a LAN, or from a host connection. It can be controlled by a user (via the keyboard) or it can be unattended (via a text response file). The target system can be any local hard disk partition. With the new OS/2 Boot Manager, the operating system can be installed on any primary or logical partition. Figure 1 shows the overall installation process for OS/2 2.0.

Alternate Source Drive

OS/2 can be installed from any hard disk drive connected to the computer. The OS/2 installation program simply detects which hard disk has the source files, and bypasses the diskette drive. The installation program does not care what or where the disk is. It can use any device that can be made to look like a disk with a supported file system. With an ever-growing number of diskettes required for installation, this feature becomes very useful in a mass-installation environment.



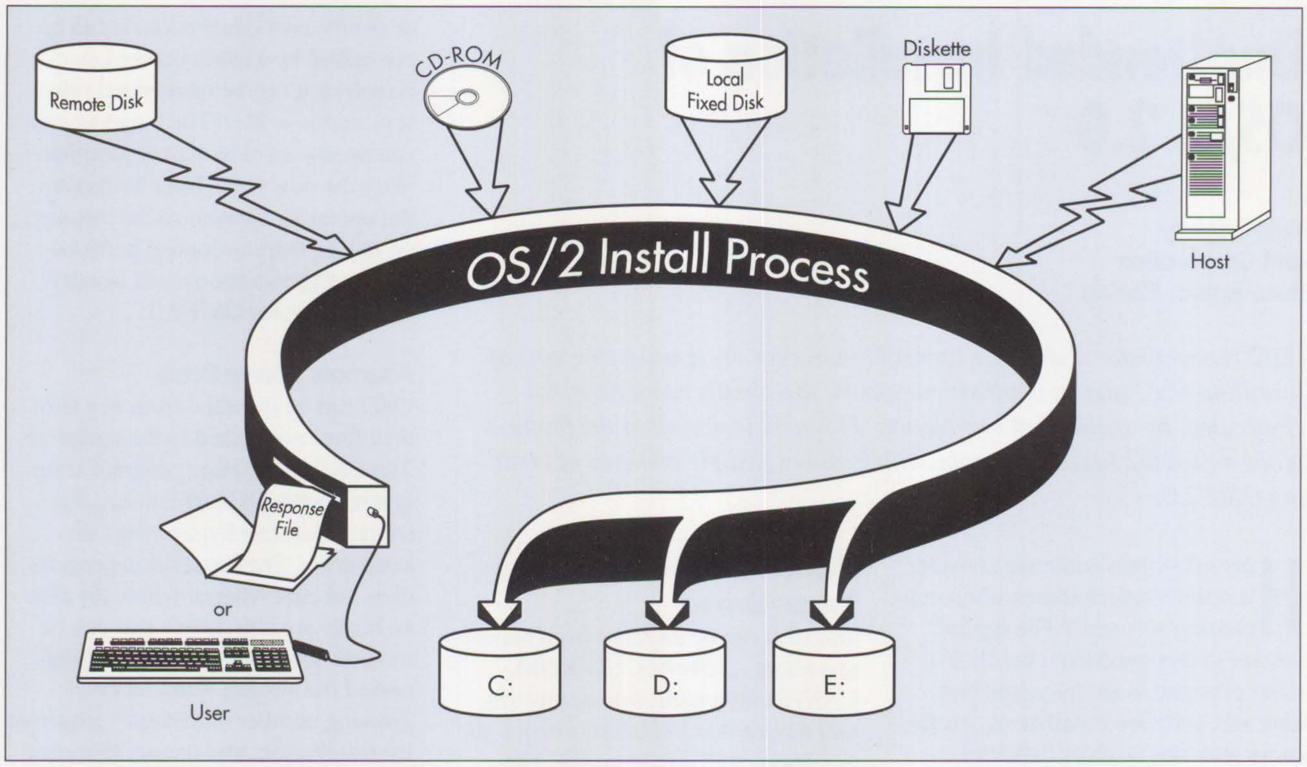


Figure 1. OS/2 Install Process

A boot diskette is necessary for installation. However, you can set up a CD-ROM or fixed disk, together with the boot diskette, for installing OS/2 2.0. Or, if you have a LAN requester that is small enough to fit on the OS/2 boot diskette, you can install OS/2 from a redirected drive. This has already been prototyped within IBM OS/2 development – we use a mini-requester that works wonderfully on the OS/2 2.0 boot diskettes.

Installing from a redirected drive may not seem impressive at first glance – after all, other software packages install and run from a LAN disk. But consider the OS/2 environment. A powerful multitasking operating system is being installed. We must be able to install new, unpartitioned systems from a single source drive. We support different installable file systems, plus all the functions and features of OS/2 2.0. We do not have the luxury of a preinstalled operating system and communications software.

Creating an Alternate Source Drive

Creating a source drive for OS/2 installation is easy. You simply create a directory for each 3.5-inch diskette that is shipped with OS/2. Then, copy the files from each diskette into the diskette's corresponding subdirectory. Assuming the source disk is drive D:, the required directory names are as shown in Figure 2.

Once the installation program knows where these directories are, it can access the files there rather than prompting the user for diskettes. The installation directories are not required to be in any specific subdirectory, but all directories must be created within the same subdirectory. This allows you to set up a source drive with many software products and data contained in separate subdirectories. For instance, I set up my source drives

```
d:\disk_0     Installation Diskette
d:\disk_1     Diskette 1
   .   .
   .   .
   .   .
d:\disk_n     Last numbered diskette
d:\disk_P1    Printer device-driver diskette 1
   .   .
   .   .
   .   .
d:\disk_Pn    Last printer device-driver diskette
```

Figure 2. Directory Names for Creating Alternate Source Drive

using an OS/2 subdirectory named OS2SE20 on drive D:, like this:

```
d:\os2se20\disk_0  Installation
                   Diskette
d:\os2se20\disk_1  Diskette 1
      .                 .
      .                 .
      .                 .
```

You can create additional subdirectories for the applications that you install with the operating system.

Modifying the Boot Diskette

OS/2 2.0 requires two diskettes to boot. The following discussion pertains to the first of the two boot diskettes. If the source drive requires software to make it work, that software must be accessible from the OS/2 boot diskette or a local drive. For instance, if you use a CD-ROM drive, you must copy its device drivers to the OS/2 Installation Diskette 1. You then must edit the CONFIG.SYS file on Diskette 1 to add the required statements for the CD-ROM.

Although there is not much space available on Diskette 1, you should be able to delete some files to make room. (Remember that by the time the installation program is active, your disk device drivers are already loaded, and the installation program can access the files it needs from the new source disk.) You cannot delete any files that the boot process needs. In general, this means you cannot delete Dynamic Link Libraries (DLLs) and certain device drivers. You can safely delete FDISK.EXE and CMD.EXE, because they are not used at boot time and they now exist on the new source drive. Also, if you do not use certain device drivers, you can delete them. If you are installing workstations that already have the device drivers needed for your alternate source drive, you can simply reference the fixed disk from the CONFIG.SYS file on Diskette 1.

Triggering Installation from an Alternate Source Drive

Although OS/2's installation program does not care which drive it uses or how it is connected, you do need to tell it where to find the installation source files. There are several ways to accomplish this.

It is quite easy to plug a fixed disk into certain PS/2 models temporarily for installing OS/2.

The first two methods involve modifying the CONFIG.SYS file on OS/2 Installation Diskette 1. The first method modifies the OS2_SHELL statement. The installation program gains control of the system via the OS2_SHELL statement in CONFIG.SYS when the Install Disk and Diskette 1 are booted. (In previous releases, this information was contained in the PROTSHELL statement.) The normal OS2_SHELL statement on Diskette 1 is:

```
OS2_SHELL=SYSINST2.EXE
```

You can pass a parameter to SYSINST2.EXE that specifies the disk drive and directory where the source files are located. For example, if you are installing from the OS2SE20 directory on drive D:, you can specify this as a parameter to the installation program[1]:

```
OS2_SHELL=SYSINST2.EXE
   D:\OS2SE20
```

The second method is more flexible. The OS/2 installation process looks in its environment segment for the SourcePath variable. This environment variable could be programmatically set if another program is active before the installation program, but you also can set it in CONFIG.SYS. For example:

```
SET SOURCEPATH=D:\OS2SE20
```

Your PATH and DPATH statements should contain the source path used. You can also use the SourcePath variable in a response file.

The most flexible method is the *trigger file* method. A trigger file is an ASCII text file that contains the source drive and directory name to be used for installation. This method is necessary for installing from CD-ROM. Previous methods required that you know from which drive letter you are installing. The drive letter of a CD-ROM varies, depending on how many fixed-disk partitions are in the system. After querying which drives are attached to the system, the installation program searches the root directory of every disk for a trigger file named OS2SE20.SRC. The source drive can thus be determined dynamically at installation time. In this case, the PATH and DPATH are modified for you.

The trigger file method is also the preferred method if you plan to carry a fixed disk from machine to machine as a source drive. It is quite easy to plug a fixed disk into certain PS/2® models temporarily for installing OS/2.

Alternate Target Drive

The new OS/2 2.0 Boot Manager lets you install and boot OS/2 on any local partition. You can install Boot Manager either before or during the

[1] The code lines that appear within the text for the remainder of this article should be entered as a single line.

Keyword	Specifies ...
AlternateAdapter	Secondary adapter for two display systems
BaseFileSystem	File system to format with
CDROM	CD-ROM IFS files to install
ConfigSysLine	Text line to be appended to CONFIG.SYS
Copy	Additional file(s) to be copied
CountryCode	Country to install
CountryKeyboard	Keyboard to install
DefaultPrinter	Default printer to install
DiagnosticAids	Reliability/Availability/Serviceability utilities to install
DisplayAdapter	Primary display adapter
Documentation	Documentation to install
DOSEnvironment	Whether to install DOS environment
DPMI	DOS Protected-Mode Interface (DPMI) options to install
EarlyUserExit	Program or command file to run early during the installation
ExitOnError	Whether the install program should return control when an error occurs or when it completes
ExtendedInstall	Another installation program to be run after OS/2 Install
Fonts	Fonts to install
FormatPartition	Whether install partition should be formatted
ID	Installation identification string
Include	Additional response file to process
MigrateConfigFiles	Whether configuration files should be migrated
MoreBitmaps	Whether extra bitmaps should be installed
Mouse	Mouse device driver to install
MousePort	Mouse port that the mouse should be attached to
OptionalFileSystem	Whether optional file systems are needed
OptionalSystemUtilities	System utilities to install
OS2IniData	Data to be written to OS2.INI file
PrimaryCodePage	Primary code page to be used
PrinterPort	Default printer port
ProcessEnvironment	Whether response data is written to environment segment
ProgressIndication	Whether progress indicators are displayed
RebootRequired	Whether the machine is automatically booted after installation
REXX	Whether REXX is installed
SerialDeviceSupport	Whether the serial device driver is installed
SourcePath	Disk and directory path to install from
TargetDrive	Drive that OS/2 should be installed on
ToolsAndGames	Tools and games to install
UserExit	A program or command file to run at the end of OS/2 Install
Version	Version of the operating system

Figure 3. Supported Keywords for Response Files

installation process by using the new version of the FDISK program. For more information, consult the online *OS/2 Command Reference*.

FDISK has a new full-screen interface as well as a command-line interface. The command-line interface may be useful for setting up multiple bootable partitions from another program or command file. Typical uses for the Boot Manager allow multiple operating systems to coexist on the same machine or to provide fault tolerance. That is, if problems occur on one operating system, another bootable partition serves as a fallback system.

New Interfaces

Graphical Interface

The OS/2 2.0 Install program has an easy, intuitive graphical interface. This interface allows the user to see only the questions required to install the system. The user is provided with visual feedback, indicating the progress of the installation. Icons show the user different features that can be installed, including sample fonts and country-specific information. A CONFIG.SYS editor is provided to allow easy migration from previous CONFIG.SYS files to the OS/2 2.0 CONFIG.SYS file.

As the operating system grew in complexity, its installation process also became complex. Therefore the graphical interface became essential for normal diskette-based installation. This reduces the complexity of installing the operating system.

Response-File Interface

An alternate new interface enables a simple text file to drive the installation. This is a wonderful mechanism for duplicating configurations on a mass scale, for remote installation, or for simply eliminating the need to help users who cannot install OS/2 by themselves.

This text file, called a *response file*, essentially replaces the user interface of the installation program. If diskettes are used, the only intervention required is to insert diskettes when prompted. Messages show the progress of the installation process. If the installation is initiated remotely, no local assistance is required. Because the user interface is not used, Presentation Manager (PM) is not available during response-file installation. The remainder of the installation occurs in a full-screen session.

Response-file installation can be initiated in two ways. First the boot diskette, then the DPATH, are searched for a file named OS2SE20.RSP. If it is found, the installation program, SYSINST2.EXE, passes control to the response-file installation program, RSPINST.EXE. This program is essentially the same program, but with the ability to process response files. After taking control, RSPINST.EXE processes the file and continues the installation without prompting for information. (An environment variable named RESPONSEFILE can override the name of the response file.)

A second method requires modifications to CONFIG.SYS. Simply replace SYSINST2.EXE on the OS2_SHELL statement with RSPINST.EXE and pass the name of the response file. (This is what SYSINST2.EXE does if it passes control to RSPINST.EXE.) For example:

```
SET OS2_SHELL=rspinst.exe
   a:\os2se20.rsp
```

Response files have a simple, free format. A default value is provided for required responses that do not appear in the response file. Blank lines and any lines beginning with an asterisk (*) are treated as comments. All other lines have the format:

```
KeyWord = KeyValue
```

```
***************************************************
*                                                 *
* Fonts                                           *
*                                                 *
*    Specifies which fonts should be installed    *
*                                                 *
*    Valid Parms:                                 *
*                                                 *
*       0 = None                                  *
*       1 = All (DEFAULT)                         *
*       2 = Courier              (bitmap)         *
*       3 = Helvetica            (bitmap)         *
*       4 = System Mono-spaced   (bitmap)         *
*       5 = Times Roman          (bitmap)         *
*       6 = Courier              (outline)        *
*       7 = Helvetica            (outline)        *
*       8 = Times New Roman      (outline)        *
*                                                 *
***************************************************

Fonts = 1
```

Figure 4. The Fonts Response Keyword in SAMPLE.RSP

Here, KeyWord is a response variable name, and KeyValue is either a text string or an ASCII numeric value. Some keywords, such as Fonts, can contain multiple ASCII numeric values separated by commas.

Most keywords that require ASCII numeric values can occur more than once in a response file. Yet if they do, only the last instance of the keyword is used. This can be useful if you are using a default response file, and you want to override some default values by including your own response file at the end of the file. For instance, a default response file may contain a Fonts keyword, which selects all possible fonts. If you include another Fonts response at the end of the file, the second instance of the Fonts keyword changes the selections.

Conversely, most keywords that accept a text string as input can occur multiple times. The ConfigSysLine keyword appends a text line at the end of the installed CONFIG.SYS file. Several additional lines may be needed in CONFIG.SYS to support whatever applications are used. Therefore, multiple ConfigSysLine keywords are supported.

Figure 3 lists the supported keywords. Note that the keywords in Figure 3 and elsewhere in this article are shown in mixed case. This is strictly for readability. You can use upper-, lower-, or mixed-case when you type the information.

OS/2 2.0 comes with a sample response file named SAMPLE.RSP. With it, you can create your own response file easily. SAMPLE.RSP is installed in the \OS2\INSTALL directory of the target disk; it also can be found on one of the installation diskettes. The file is self-documented, with descriptions and examples for every supported keyword. For example, the Fonts section is shown in Figure 4.

Every line is a comment except for the last line, Fonts=1. This statement causes all fonts to be installed. If you want only Courier and Times Roman fonts, you can change this line to:

```
Fonts = 2, 5, 6, 8
```

```
************************************************************
*                                                          *
* Copy                                                     *
*                                                          *
*    Specifies a source file and destination directory     *
*    of a file to be copied during install. Errors are     *
*    ignored, though they will be logged. Packed files     *
*    are acceptable since UNPACK will do the copy.         *
*    There may be multiple instances of this keyword.      *
*    No validity checking is done.                         *
*                                                          *
*    Valid Parms:                                          *
*                                                          *
*       KEYWORD= parameters source destination             *
*                                                          *
*         where source file = valid filename               *
*          parameters = valid UNPACK parameters            *
*           and destination = valid directory name         *
*                                                          *
*         ex: Copy = readme.dat  c:\os2                    *
*                                                          *
************************************************************
```

Figure 5. The Copy Response Keyword in SAMPLE.RSP

Parameters must be separated using commas; spaces are optional.

Figure 5 displays the Copy keyword and contains only comments; there is no default action. Unlike the Fonts keyword in Figure 4, the value of this response keyword is a text string. All spaces in the text string are preserved.

For details about all response-file keywords, see the comments in SAMPLE.RSP.

OS/2 Installation Extensions

The purpose of most response-file keywords is to provide the information normally gathered by the graphical interface. Alone, these keywords allow OS/2 to be installed using a response file. Because the goal is to enable remote, unattended installation, it would be useful if additional software could be installed at the same time. Then, instead of running a separate installation process for every software package being distributed, everything can be installed along with OS/2 by using the response file. The following are some keywords created for this purpose.

ConfigSysLine

The ConfigSysLine keyword appends a text line to the installed CONFIG.SYS file. In OS/2 2.0, it does not replace an existing CONFIG.SYS line, but it is quite useful for installing applications. All spaces, and any other characters after the first, are preserved. The statement:

```
ConfigSysLine =
DEVICE=DEVICE.SYS /parm1 /parm2
```

appends the following line at the end of the installed CONFIG.SYS (spaces before DEVICE= are preserved):

```
DEVICE=DEVICE.SYS /parm1 /parm2
```

Copy

The Copy keyword copies extra files during the installation of OS/2. It is typically used for installing additional software or data to customize a workstation. Its parameters are very similar to those in the OS/2 COPY command. The following response-file line copies the file README.DAT into the C:\OS2 subdirectory:

```
Copy = readme.dat c:\os2
```

You can specify any parameters that are acceptable to the OS/2 UNPACK program. (Although UNPACK is shipped with OS/2, the IBM Programmer's Toolkit 2.0 is needed to use the corresponding PACK program.)

PACK and UNPACK were recently enhanced with new function and extensive performance tuning. One new feature is *bundle files*, which packs many files into one file. The new installation procedure uses this bundling feature to copy the features selected for installation.

The following response-file line extracts all files contained in the bundle file VGA and copies them to the fixed disk C:.

```
Copy = vga c:
```

Similarly, the following response extracts the file INI.RC from the VGA bundle file and copies it to the fixed disk C:.

```
Copy = vga c: /n:ini.rc
```

The bundle file may contain the destination directory.

From an OS/2 2.0 command prompt, the equivalent command for the previous example is:

```
UNPACK VGA C: /N:INI.RC
```

EarlyUserExit

It may be useful to run a program or command (batch) file before OS/2 is installed. For example, you may want to back up files, delete files, or run another installation program before OS/2 installation. The EarlyUserExit keyword lets you do this at the beginning of the installation process. Although the OS/2 Install program is active, at this point it has not done any work. As many EarlyUserExit response-file lines may be specified as necessary. Each user-exit process is executed synchronously, one at a time, in the order in which the EarlyUserExit keywords appear in the response file. After processing all EarlyUserExits, the installation continues. If no path name is given, the PATH is searched for the specified program. Any program used must run in full-screen mode, because PM is not available during response-file installation. For example:

```
EarlyUserExit = cleanup.cmd
```

Note: If you want to run a program or command file before OS/2 Install is invoked, you can specify it in the OS2_SHELL statement in CONFIG.SYS on OS/2 Installation Diskette 1. If you do this, it is then the responsibility of your program or command file to invoke OS/2 Install. Insert the name of the program immediately before the installation program. For example:

```
OS2_SHELL=<your program here>
   RSPINST.EXE A:\OS2SE20.RSP
```

ExitOnError

If OS/2 Install is invoked by another process, that process may expect to regain control when the installation completes. By default, the installation program prompts the user to reboot the system after successful completion. If there is an error, by default OS/2 Install displays an error panel and prompts the user to take action. The ExitOnError keyword informs the installation program that it must return control to the calling process with a return value upon either successful or unsuccessful completion.

ExtendedInstall

The ExtendedInstall keyword allows the OS/2 Install program to be chained with another installation program. If this keyword is used, then when the OS/2 installation completes, the specified program and parameters execute asynchronously. Then OS/2 Install exits memory to free up space. For example:

```
ExtendedInstall = PROGRAM.EXE
   parm1 parm2 parm3
```

Use this method when installing another product that uses the response-file design, or one that can complete its installation in full-screen mode without user intervention.

The ExtendedInstall keyword allows the OS/2 Install program to be chained with another installation program.

In the future, the ExtendedInstall keyword is the method that will be used to install, in a single installation run, an OS/2 workstation with IBM's LAN Server, Communications Manager, Database Manager, and so on.

Include

Typical large installation environments have some common response values and a few unique values for most workstations. An easy way to implement this is to use the Include keyword. The Include response imbeds another response file at the line where the Include is located. You can use any number of Include responses. If the included response-file name does not contain a drive and path, the DPATH is searched. For example:

```
Include = custom.rsp
```

OS2IniData

OS2IniData writes profile strings to the OS2.INI configuration file. This is useful now that more and more applications store data there. This keyword has three parameters delineated by slashes. All spaces and characters between slashes are preserved. For example:

```
OS2IniData = / App Name /
   KeyName / KeyValue /
```

ProcessEnvironment

Because the environment segment is easy to access from programs and command files, the installation process stores all response-file keywords and values as environment variables. If memory is constrained, the ProcessEnvironment response can be used to prevent environment segment processing. For example:

```
ProcessEnvironment=0
```

ProgressIndication

This keyword determines whether the display provides the user with information about the progress of the installation. If another program is used to display something else during installation, this keyword prevents OS/2 Install from interfering with the display. For example:

```
ProgressIndication=0
```

RebootRequired

This keyword automatically reboots the system when installation completes. This is useful for unattended installation (see Remote Installation discussion that follows). For example:

```
RebootRequired=1
```

SourcePath

In the response file, the installation source drive and directory can be specified. This is one way to specify the source path; other ways were mentioned earlier. For example:

```
SourcePath=Z:\OS2SE20
```

TargetDrive

The target drive may be specified with this keyword. However, this will not cause the Boot Manager to be installed on the workstation. If TargetDrive specifies anything other than the default drive, then the Boot Manager must have been previously installed and the target partition must be large enough to accept the entire operating system. This is one of the few keywords that causes an unrecoverable installation error if used improperly. Refer to the *OS/2 2.0 Command Reference* for more information about the Boot Manager.

UserExit

UserExit works exactly like an EarlyUserExit, except that it executes synchronously at the end of installation rather than at the beginning. Any program specified must run in full-screen mode, because PM is not available during response-file installation.

Remote Installation

Unattended Installation

The most exciting use of response files is for installing unattended systems. (The unattended installation feature is not provided with OS/2 Install.) Response files provide a fundamental tool required to enable unattended installation (except for the communication link, of course). Using response files, a creative mind could come up with some interesting implementations. This section mentions some interesting things we have done in the OS/2 development organization.

To demonstrate the possibilities, assume we have a mini-requester, as previously described. If a communication link exists (or can be established), then a remote process, communicating with a local program, could initiate the installation.

Once we have that all-important link, the rest is easy. We simply use the methods described in this article to install OS/2. The difference is that we will not be booting from diskettes. If a LAN connection exists, we can create a minimal OS/2 2.0 system in a subdirectory and boot from it, not from the installation diskettes.

Sounds too easy? The OS/2 Installation Diskette and Diskette 1 contain everything you need to boot OS/2 2.0. A simple command file could copy all these files from a LAN disk to a subdirectory on the target system. Because the system files no longer need to be contiguous, they can now be copied to the root directory of a locked drive (although the SYSINSTX program will do the job nicely). An installation CONFIG.SYS has to be set up to point to the boot directory and the mini-requester. A programmatic reboot starts the installation process as if it were booted from diskette.

The most exciting use of response files is for installing unattended systems.

My favorite feature of this scenario is that it does not matter which system is installed. A command file can initiate the process. Since a minimal OS/2 2.0 system is copied before beginning the installation, DOS or any version of OS/2 can be run. Once the system is rebooted, it runs in the OS/2 2.0 environment. What have we accomplished? We have just eliminated the diskettes that were previously required to upgrade from DOS or OS/2 version 1.X to OS/2 version 2.0!

Want to get creative? Now that we know how to boot a subdirectory, why not add some fault tolerance? Suppose our current system is OS/2 version 1.3. Also, suppose our command file renames the \OS2 directory and the system files. We can back up the rest of the files in the root directory. Now the OS/2 2.0 Install program will not tamper with any OS/2 1.3 files. If OS/2 2.0 Install has an unrecoverable error, it can return the error to a frontend program that apologizes, cleans up, restores the OS/2 1.3 directory names, and boots the OS/2 1.3 system. The possibilities are unlimited!

Why Use OS/2 Install?

Many of you have already written or used your own version of a LAN-based OS/2 installation. If so, you may wonder why you should use the OS/2 Install program.

Until now, most LAN OS/2 installations have meant installing one full-function workstation, then cloning (copying) it onto many other systems. This causes problems that must be overcome with each release by writing your own programs and command files. The obvious problem is to determine what the supported (shipped) installation program does other than copying files. Installing OS/2 is a non-trivial task. It migrates user and system information from previous releases of OS/2 and DOS. Seemingly minor differences in hardware can cause entirely different device drivers to be installed. A lot of code logic hides behind the pretty, new interface of installation. Only the OS/2 2.0 Install program knows exactly what to do, and its logic changes with every release. If you have your own LAN installation process, it would inevitably need rework with each new release. Why reinvent the wheel?

Another reason to use the OS/2 Install program is that, if you use any other installation program to clone OS/2, your installed systems may not be eligible for support. Any number of errors might occur if the system is improperly installed. The OS/2 2.0 Install program, however, is part of a supported IBM product. It knows how to install each new release of OS/2.

In addition, for the first time, the installation program can be updated with bug fixes through the normal corrective service process. IBM's corrective service facility also will be enhanced to run in the redirected environment.

System administrators should concentrate on integrating existing installation procedures with IBM's installation process. Their goal should be to use IBM's new installation function to install entire workstations in a single OS/2 installation process. IBM does not provide the full workstation – IBM provides the tools with which to accomplish the installation.

New Design Principles for Installation Programs

Using the design of the OS/2 Install process, development shops will not have to package an entire remote installation. They should enable it as we have done in OS/2 development. This requires an unattended interface that can be called from, or chained together with, an OS/2 response file. References to diskette drive A: or boot partition C: can no longer be hard coded. Ideally, source and target drives should be referenced as variable strings, and file names should be appended before passing them on to whatever Application Programming Interfaces (APIs) are needed. The APIs will validate file names. This approach allows flexibility for unusual naming conventions, such as long file names and Universal Naming Convention (UNC) names.

Many OS/2 applications use a graphical interface for installation. It can be cumbersome to provide both graphical and response-file interfaces, but both are necessary. When designing installation programs, it is wise to write a worker engine that is decoupled from the interfaces. This minimizes duplicate code while allowing users to adapt easily to new environments.

Conclusion

There is a new breed of installation program for the 1990s. The OS/2 development team is striving to excel in this area by providing flexible installation interfaces that can be used to install the operating system and related software easily, in a remote, unattended fashion.

Note: This product was under development at the time this article went to press. Additional features and changes may be forthcoming.

***Paul Brightly** is a senior associate programmer in the OS/2 Development organization in Boca Raton, Florida. Paul has worked on OS/2 since he started with IBM in 1987. He worked on OS/2 utilities, then the system editor in OS/2 1.1. For the last four years, Paul has concentrated on OS/2 installation. He is presently working in PM development. He earned a Bachelor's of Applied Science degree in computer information systems from Florida Atlantic University.*

OS/2 Communications Manager Trace Events

Alice Turlington
IBM Corporation
Dallas, Texas

The OS/2 Communications Manager provides users and administrators with several tools to determine which components may be failing or causing problems. Among these are the message log, the error log, and the trace facility. This article describes the use of the trace facility.

There are several reasons to usethe trace facility in OS/2 Communications Manager. The simplest scenario is when all the configured functions are working correctly. In this case, the user should make a trace of the scenario for the records. If problems are encountered later, the good trace data can be compared to the problem trace data.

Another scenario is when two devices are having a problem communicating. If the error log and message log do not provide enough information to fix the problem, then a trace can get the detailed information to solve the problem.

A user can also invoke the trace facility to find problems with user-written applications for specific communications Application Programming Interfaces (APIs). This will help the user determine which calls were issued and which return codes resulted.

Using the Trace Facility

There are two ways to use the trace facility. The first is to start tracing a specific event after Communications Manager has been running. For example, a user needs to log on to a host and a database server across the LAN. The logon to the host is successful; however, when starting the connection to the database server, communication fails. In this case, to identify the source of the problem, it is not necessary to trace the information connecting to the host. Instead, start the connection to the host, then turn on the trace to capture the data that is going to the database server.

The second choice is to start the trace automatically when Communications Manager is started. This allows the user to trace the functions that Communications Manager is automatically starting. For example, Communications Manager allows the user to automatically start the 3270 sessions. If this option is set, then when Communications Manager is brought up, Communications Manager keeps control until the 3270 sessions are brought up as well. If there is a problem with the host connection, and the auto-start 3270 session is selected, then the user will not see the connection traces if the auto-trace is not turned on.

What the Trace Facility Traces

Many functions and components can be traced by the trace facility, including the following:

- **APIs**: APPC, SRPI, ACDI, SERVICES, X.25, EHLLAPI, LUA_RUI, LUA_SLI, and SUBSYSM
- **DLCs**: DFT, IBMPCNET, SDLC, IBMTRNET, X.25, TWINAXIAL, X.25 FRAM, and ETHERAND
- **Events**: Figure 1 lists all 30 Communications Manager trace events that can be used in problem determination and contains detailed information on each of the trace events. Trace events are not documented anywhere, but the information captured by trace events may help identify the source of the problem. An example of this is event 5, which captures data about Exchange Identifiers (XIDs). XIDs are used to establish logical links. Therefore, if there is a problem establishing a session with another device, event 5 trace data can help determine whether or not the XID exchange is successful, and if not, what data is being passed.

If the trace data becomes so large that it will not fit in the trace file, the file wraps; that is, new entries will be written over previous ones. Therefore, be careful when tracing and interpreting system events.

The traced data is written to a file in binary format. To make this data readable, use the trace formatting tool provided by Networking Services/2 or OS/2 Extended Services.

Figure 2 shows an example of the XID trace entry in Hex format. Figure 3 shows the formatted output of the Hex XID entry in Figure 2.

Turning On Trace Events

Trace events can be turned on in two ways. One is to use the menu interface provided by Communications Manager; the other is to use the command-line interface. OS/2 Extended Services comes with a command-line interface. OS/2 Extended Edition 1.3 and earlier releases do not provide the command-line interface. However, if Systems Application Architecture® (SAA™) Networking Services/2 is installed, use the command-line interface. For more information about the command-line interface, refer to the

System Event Number	Event and Function
01	APPC CCB This trace event displays command control block information. These trace entries have little significance outside the development organization.
02	APPC INT Except for the fact that the verb IDs are different, these trace events look exactly like the regular APPC API trace entries. Likewise, these entries are copies of control blocks that are passed from the application to the APPC subsystem and the control blocks that are returned from the APPC subsystem to the application.
03	APPC PROCESS APPC is comprised of several processes and functions. These processes are based on the SNA LU 6.2 architectural model. These processes can be created, destroyed, dispatched, and suspended. The process event entries trace the progress of these processes.
04	APPC SEND/RCV This trace event traces the exchange of internal communications signals.
05	APPC XID This system event traces the XID frames that are exchanged.
06	ASYNC API This trace event collects data from internal events within the ACDI subsystem. It should be selected for all ACDI and Async Emulator traces.
07	ASYNC EMULATOR EVENT01 This trace event traces the internal operations of the Async Emulator. It should be selected for all async emulation problems.
08	ASYNC EMULATOR EVENT02 This trace event logs Async File Transfer information. Select this trace when tracing a file transfer problem.
09	DFT The DLC trace events capture information that is passed within the DLC layer. This is the DLC-level trace event for DFT.
10	IBMPCNET Same as event 9, except that it traces the PCNET data.
11	SDLC Same as event 9, except that it traces the SDLC data.
12	COMMON SERVICES Trace event for the Services API. These APIs are Communications Manager services verbs (Convert, Copy_Trace_to_File, Define_Dump, Define_Trace, and so on.
13	SRPI EVENT01 Trace event for the Server-Requester Programming Interface (SRPI).
14	IBMTRNET Same as event 9, except that it traces the Token-Ring data.
15	3270 EVENT01 This trace selection invokes the tracing of internal 3270 function calls.
16	3270 EVENT02 This trace selection traces a set of very low-level internal calls.
17	5250 EMULATOR EVENT01 Trace event for the 5250 workstation feature.
18	5250 EMULATOR EVENT02 Same as event 17.

Figure 1. OS/2 Communications Manager Trace Events (continued)

System Event Number	Event and Function
19	X.25 API EVENT01 (XACI) This trace selection causes tracing of the X.25 Adapter Code Interface (XACI) to occur. This data is not readily interpreted by the customer or support personnel.
20	X.25 DLC This trace selection should be used for SNA problems in the X.25 environment. This data is not readily interpreted by the customer or support personnel.
21	TWINAXIAL DLC Same as event 9, except that it traces the TWINAXIAL data.
22	SRPI EVENT02 Same as event 13.
23	X.25 API EVENT02 (INTERNAL VERBS) This trace selection causes the tracing of internal X.25 API operations.
24	X.25 API EVENT03 (Q-THREAD) This selection is important for determining internal problems within X.25 API operations.
25	LUA Trace event for LU Application (LUA).
26	Not currently used.
27	3270 HOST GRAPHICS I Trace event for Host Graphics.
28	3270 HOST GRAPHICS II Same as event 27.
29	ETHERAND (ETHERNET) Same as event 9, except that it traces the ETHERAND data.
30	SUBSYSTEM MANAGEMENT API Trace event for the Subsystem Management APIs. These APIs allow applications to call the Subsystem Management functions, such as CNOS, DEACTIVATE_LOGICAL_LINK, DEACTIVATE_SESSION, and so on.

Figure 1. OS/2 Communications Manager Trace Events

Networking Services/2 Problem Determination Guide (SC52-1113).

Follow these steps to use Communications Manager's menu interface to turn on the trace events:

1. From Communications Manager's main screen, press F10 to go to the action bar.
2. Select *Advanced.*
3. From the pull-down menu, select option 3, *Problem Determination Aids.*
4. Select option 2, *Trace Services.*
5. Go to step 8 if you are not using the auto-trace facility. Continue with step 6 if you are using the auto-trace facility.
6. Select option 6, *Auto-Trace Services.*
7. Choose option 1, *Select and Store Auto-Traces.* Skip step 8.
8. Select option 1, *Select Traces.*
9. At the Trace Type Selection menu, move the cursor to *Advanced Trace Selections.* Press the spacebar to select it, then press Enter.
10. At the Advanced Trace Selections menu, there are different events. Move the cursor to the event(s) you want to turn on, and press the spacebar to turn on an event. To turn off the selection, press the spacebar again.
11. After selecting the desired events, press Enter to save the information.
12. At the Auto-Trace Services panel, select *Trace Storage Size.*
13. Set the size to 16 x 64K.
14. At the Auto-Trace Services panel, select *Enable Auto Trace.*
15. Press the Esc key to leave the Auto-Trace Services menu.
16. Press the Esc key to leave the Problem Determination Aids menu. You are now at the

```
<==SEND=====  IBMTRNET  #00 40000155101104 42B817C2926f2520 XID    10:07:48:81
    325F05D1 53110000 8034C000 00000080      <2_.JS....4......>
    00010B71 00078900 01000001 000E0BF4      <...q..i........4>
    C1D7D7D5 4BC1D3C9 C3C50E09 F77CC1C1      <APPNKALICE..7|AA>
    C1C1C1C1 C1102800 1111040E 02F5F6F2      <AAAAA.(......562>
    F1F2F1F3 F0F1F1F0 F0161103 130011F8      <121301100......8>
    F5F7F0F0 F0F0F0F0 F0F0F0F0 F0F0F0        <570000000000000 >
```

Figure 2. OS/2 Communications Manager XID Trace – Hex

```
FMTTRACE
(C) Copyright IBM Corporation 1990, 1991
Line:      1   Send XID
Time stamp: 10:07:48:81
DLC type: IBMTRNET
Adapter number: 00
Destination address: 40000155101104
ALS ID: 42B817C2926F2520
XID:
   Format = 3
   Node type = 2
   Total length = 95
   Node ID = 0x05d15311
   Init-self = Cannot receive
   Stand-alone BIND = Can receive
   BIND segment generation = Can generate
   BIND segment receipt = ACTPU requested
   APPN network node = No
   CP-CP sessions requested = Yes
   CP-CP sessions supported = Yes
   Exchange states indicator = Negotiation proceeding
   Second init non-activation = Not supported
   Adaptive BIND pacing sender = Supported
   Adaptive BIND pacing receiver = Supported
   Parallel TGs = Supported
   TG number = 0
   DLC type = SDLC
   DLC data length = 11
   ABM = Supported
   Link station role = Negotiable
   Short hold mode status = Not reconnection
   Short hold mode capability = Not supported
   Transmit/receive capability = Primary
   Maximum receive BTU length = 1929
   SDLC CR profile = SNA
   SIM/RIM options = Not supported
   Maximum I-frames before ACK = 1
   Network name control vector:
      Network name type = CP
      Name = APPN.ALICE
   Network name control vector:
      Network name type = LS
      Name = @AAAAAAA
   Product set ID control vector:
      Hex dump:
         10280011 11040e02 f5f6f2f1 f2f1f3f0  <.(......56212130>
         f1f1f0f0 16110313 0011f8f5 f7f0f0f0  <1100......857000>
         f0f0f0f0 f0f0f0f0 f0f0               <0000000000      >
```

Figure 3. OS/2 Communications Manager XID Trace – Formatted

Communications Manager main menu, and the trace events are turned on.

17. Exit Communications Manager.
18. Restart Communications Manager and re-create the problem.
19. At the point of the failure, go back into Communications Manager's Problem Determination Aids and Trace Services.
20. From Trace Services, select *Stop Traces.*
21. Select *Copy Storage Trace to File.*

Summary

There are many different tools to help you perform problem determination in Communications Manager. This article showed one of the tools: trace events. Trace events should be used as the final problem determination tool. Check the message log, error log, and other traces before using the trace events. Understanding the use of these various tools will help find the source of the problem.

***Alice Turlington** has supported OS/2 since its inception. She specializes in Communications Manager, and has often presented to customers on IBM's satellite broadcasts and at technical update meetings. She holds BS and MS degrees in computer science from the University of Texas at Dallas.*

IBM and Novell LAN Software Coexistence

Doug Spelce and Steve French
IBM Corporation
Austin, Texas

This article discusses how IBM and Novell® communications software can share LAN hardware. The article describes the latest software from both companies: IBM's OS/2 LAN Requester Version 2.0 and Novell's NetWare Requester for OS/2 2.0.

Supporting multiple network file systems on the same system is nothing new to OS/2. With the release of OS/2 Extended Edition (EE) 1.30.1 (CSD 5015), IBM supported the simultaneous operation of both IBM OS/2 LAN Requester and Novell NetWare® Requester for OS/2 in the same machine, sharing the same LAN adapter – a configuration called LAN coexistence. Now, coexistence has been significantly improved with the release of IBM OS/2 LAN Server (LS) 2.0 and IBM Extended Services (ES) 1.0.

History

In early 1991, Novell and IBM launched into agreements stating that both companies would march toward mutually defined goals for product compatibility, availability, and support. The agreements have been well received by IBM and Novell customers, who often have mixed IBM LAN Server and Novell NetWare environments.

With the release of OS/2 1.2 in 1989, IBM provided support for multiple Installable File Systems (IFSs) in the same machine. This support was significantly enhanced in the spring of 1991 with the release of OS/2 EE 1.30.1, which allowed multiple LAN IFSs to register to receive Universal Naming Convention (UNC) requests. Previous versions of the OS/2 kernel had made it difficult for two LAN IFSs to be installed at once, essentially prohibiting the simultaneous operation, or coexistence, of IBM's OS/2 LAN Requester and Novell's NetWare Requester for OS/2. OS/2 2.0 and LS 2.0 make it easier for multiple network file systems to communicate smoothly through shared adapters.

Initially, the ability to share LAN adapters, introduced by OS/2 EE 1.30.1, applied only to Token-Ring configurations. Novell's TOKENEE.SYS driver worked with IBM's Communications Manager drivers to complete the task. Later in 1991, when the CMGRLAN.SYS driver replaced TOKENEE.SYS, Novell added support for PC Network and Ethernet™ configurations. However, CMGRLAN.SYS proved to have significant limitations for adapter sharing, particularly for Ethernet – problems that were not resolved until the release of ODINSUP.SYS.

ODINSUP.SYS is Novell's latest device driver that supports the Network Driver Interface Specification (NDIS) protocol stack on top of the Open Data-Link Interface (ODI). ODINSUP.SYS is shipped as part of NetWare Requester for OS/2 2.0. ODINSUP.SYS is also compatible with OS/2 EE 1.30.1 for Ethernet environments, in conjunction with the appropriate Communications Manager network and protocol drivers. It is also compatible with OS/2 1.30.2.

Coexistence with NetWare is not simply a matter of multiple networking packages sharing the same network hardware. It means that IBM and Novell communications software share a LAN adapter with the many different protocols (IEEE 802.2, NetBIOS, TCP/IP, IPX, and so on). Therefore, the discussion and examples below are not exclusive to LANs, but are also appropriate for configurations where ES 1.0 applications, such as Communications Manager and Database Manager, are involved.

Obstacles to Coexistence

Many customers wonder why coexistence of network products from different vendors (or even the same vendor) is an issue. In this world of increasing standardization, why are network vendors unable to agree on the basics? Some reasons that networks have continued to be so different are as follows:

- **Historical**. A primary responsibility is to protect the investment of existing customers by giving them smooth migration to future releases. For example, IBM would like to provide a way for customers who use the older IBM PC LAN Program to change, in stages, to the IBM OS/2 LAN Server.
- **Technical differences**. Features that give distinct advantages to the typical user of one product may not be important for the typical user of another. For example, companies that target their products at small, isolated LANs are not as concerned about wide-area network and routing issues as IBM and Novell. Therefore, protocols that are flexible enough to easily handle routings across large networks may not be used.

- **Multiple standards**. There are many standards to choose from, at almost every layer of the protocol stack. For example, at low levels (the MAC layer), Ethernet and Token-Ring are just two of many popular protocols. At higher levels, should you use SPX/IPX, or NetBIOS, or TCP/IP, or should a network software developer focus on Open Systems Interconnection (OSI) standards?

Some specific challenges for the coexistence of LAN Server and NetWare are the following:

- **File system**. How can the system send all requests for one drive letter (such as L:), which may be mapped by Novell software, to a Novell server, while requests using other drive letters (such as G: and H:) are sent to different IBM servers such as LS 2.0 High-Performance Servers and PC LAN Program servers? See Figure 1. How will the system determine which network to use to send UNC requests, such as for opening named pipes? Will OS/2 DOS box or Virtual DOS Machine (VDM) requests work similarly for both networks?
- **Session establishment and user accounts**. How do you handle IBM's User Profile Management (UPM) LOGON versus Novell's LOGIN, and how do you manage two different sets of user accounts?
- **Network protocols**. From NDIS to ODI, can IPX/SPX coexist smoothly with IBM NetBIOS? Which drivers should you use?

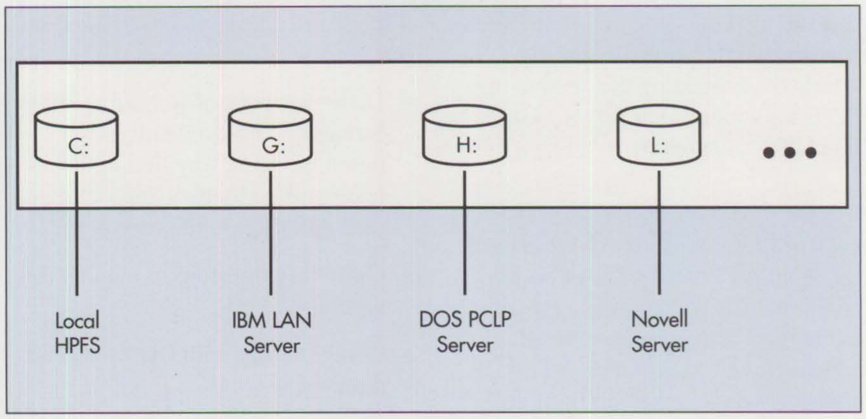


Figure 1. Assigning Drive Letters to Different Media

Logon and Session Establishment

Perhaps the first place that users will notice a difference between IBM and Novell is in how one first accesses the network: LOGON versus LOGIN. In OS/2, UPM handles LOGON and provides a facility for both local and network logon to IBM's OS/2 LAN Server. Novell's LOGIN program is not directly related, so it is possible to be logged on, with different user IDs and passwords, to both Novell and IBM LAN servers. For convenience, it may be more useful to use the same user IDs and passwords on both networks.

The primary purpose of IBM's LOGON program is to register a user ID and password into memory located in the requester machine. Since the user ID and password will be stored locally, if the session between a requester and the domain controller is broken, other server machines can still be reached because they validate the user ID and password separately. IBM's LOGON checks that the user ID and password are correct at the domain controller, where the primary copy of the user database is kept. Each server must check for proper authority for network access to its own resources. To do this, each server uses the user ID and password that are passed across the network when access to the new server is first attempted.

The number of active sessions connected to a server, and the amount of network file and print activity that the server must support, limit the number of requesters that can use an IBM LAN server at one time. Each active connection from an IBM LAN requester to a server represents one NetBIOS session, and the maximum number of NetBIOS sessions that each adapter can support is 254. It is possible to have more than 254 simultaneous NetBIOS sessions from an IBM domain controller by using multiple adapters, depending on the speed of the domain controller and the workload required.

LOGON establishes a session from the requester to the domain controller. (This session will be disconnected in approximately two hours if nothing is sent between the two.) When a user executes the NET WHO command to list who is logged on at the domain controller, frequently more users are listed than when the NET SESSION command is typed from the domain controller. This happens because NET SESSION represents only the active connections from that server. Users are removed from the NET WHO list when they LOGOFF, whereas they are removed from the NET SESSION list when they either LOGOFF or turn off their computers.

IBM's LOGON and Novell's LOGIN have slightly different purposes. The primary functions of Novell's LOGIN are to validate the user ID and password, and to run the login script.

LOGIN also logs you out of any file servers to which you are already attached, eliminating the need to logout from your current server before logging in to another.

Another NetWare utility, closely related to LOGIN, is the ATTACH utility. With ATTACH you can access additional file servers while remaining logged in to your current file server. However, although ATTACH connects you to a file server, it does not create a drive mapping to that file server, like the LOGIN utility.

The LOGIN and ATTACH utilities, like all other NetWare utilities, are installed on the NetWare file server and are initially accessed via a drive letter (usually drive L:) redirected to the NetWare file server at system boot time. After LOGIN successfully completes, the LOGIN drive is remapped to the directory on the NetWare file server where other OS/2 utilities are installed.

For IBM, LOGON validates the user ID and password, establishes a session to the domain controller, adds public and private applications to the desktop, stores the user ID for use by UPM, and stores the user ID in the redirector in case reconnection to a server is ever needed. LOGON also automatically connects the logon assignments and home directories, and runs the logon profile or script. These extra functions account for much of the difference between the time it takes to LOGON with IBM versus LOGIN with Novell.

Role of the File System

If all media were the same, and if floppy disks, hard disks, network drives, and CD-ROMs were all handled in the same way by the operating system, no one would need IFSs. They provide a mechanism for managing different types of media using a single operating system. IFSs allow for:

- Different types of storage media to be handled by applications in much the same way, hiding needless complexity by using standardized API calls to the file system
- Optional extensions to the naming scheme
- Compatibility with DOS files and directories
- Flexible and transparent access to resources by the user
- Redirection of requests for data to remote systems
- Diversity in storage devices to optimize for specific purposes
- Coexistence of very different media in one system

The key to coexistence is to provide a fast, clean file system mechanism that transparently handles requests from an application and redirects them to the right network.

As shown in Figure 2, the key to coexistence is to provide a fast, clean file system mechanism that transparently handles requests from an application and redirects them to the right network. A typical flow might be:

1. A user working with IBM DisplayWrite®/2 opens a document.
2. The program does an OS/2 DosOpen call of that file.
3. From the drive letter, the kernel IFS router determines which IFS should handle that request. For example, drive F: may be registered as a Novell drive by a previous MAP command, and drives G: and H: may be NET USEd to different IBM servers.
4. The IFS router passes the request in a standardized form to the appropriate file system, such as NETWKSTA.SYS.
5. NETWKSTA.SYS formats the request as a network command using Server Message Blocks (SMBs) and sends it across the network using NetBIOS.
6. When the response comes back from the server, NETWKSTA.SYS passes the response back in a standardized form to the OS/2 kernel, which then passes it back to the application program waiting for completion of the DosOpen call.

The application program may not notice whether this file is on an IBM or a Novell drive. To "register" a drive, the user types the IBM NET USE or the Novell MAP command. Programmatically, the interface is very similar, because the OS/2 API provides a standardized DosFSAttach call. IBM also provides a similar NetUseAdd( ) call. Having a clean, standardized programming interface that was designed for networks, yet permits many diverse user interfaces, is an advantage of OS/2.

Named pipes and UNC names are special cases that must be handled by the kernel IFS router. Named pipes are a powerful programming interface that are easy to use. They provide a two-way communication path between processes for sending and receiving control information and data across the network. Both named pipes and files can be referred to by UNC names. For example:

```
\\servername\PIPE\pipename
```

and

```
\\servername\sharename\filename
```

The IFS router has a two-pass method for resolving UNC names, starting with the network IFS that is listed first in CONFIG.SYS. For this reason, in your CONFIG.SYS, you should consider placing the statements relating to the network that you use more often for UNC access *above* the statements relating to the network that you use less often.

Named pipes pose a special problem for the IFS router, because IBM puts both the server and client parts of the network named-pipe code into the file system to get maximum performance. When an IBM requester sends a named pipe request to an IBM server, the request is handled in the file system, NETWKSTA.SYS, rather than being sent to the server code. This results in very fast named pipe access, but it requires that the IFS router and server send pipe requests for IBM network named pipes to NETWKSTA.SYS rather than to the kernel.

Other types of requests that must be handled transparently by the IFS router include requests that come in from the OS/2 1.3 DOS box or an OS/2 2.0 Virtual DOS Machine, and requests for printing. OS/2 2.0 provides easy-to-use, transparent access to multiple networks from the OS/2 Desktop through the LAN-aware Workplace Shell (see the article titled "The OS/2 Workplace Shell" in this issue for an overview of its features.)

The IFS interface makes remote drives appear much like local hard disks. Therefore, the IBM LAN Server can share remote drives that are connected to other types of networks, such as TCP/IP, and make those remote drives available to users of IBM OS/2 LAN Requester. This two-step access to other networks functions as a primitive form of gateway. For example, by going through a machine with IBM LAN Server and TCP/IP with Network File System (NFS) installed, data can be transferred from AIX machines running NFS to IBM DOS or OS/2 requesters.

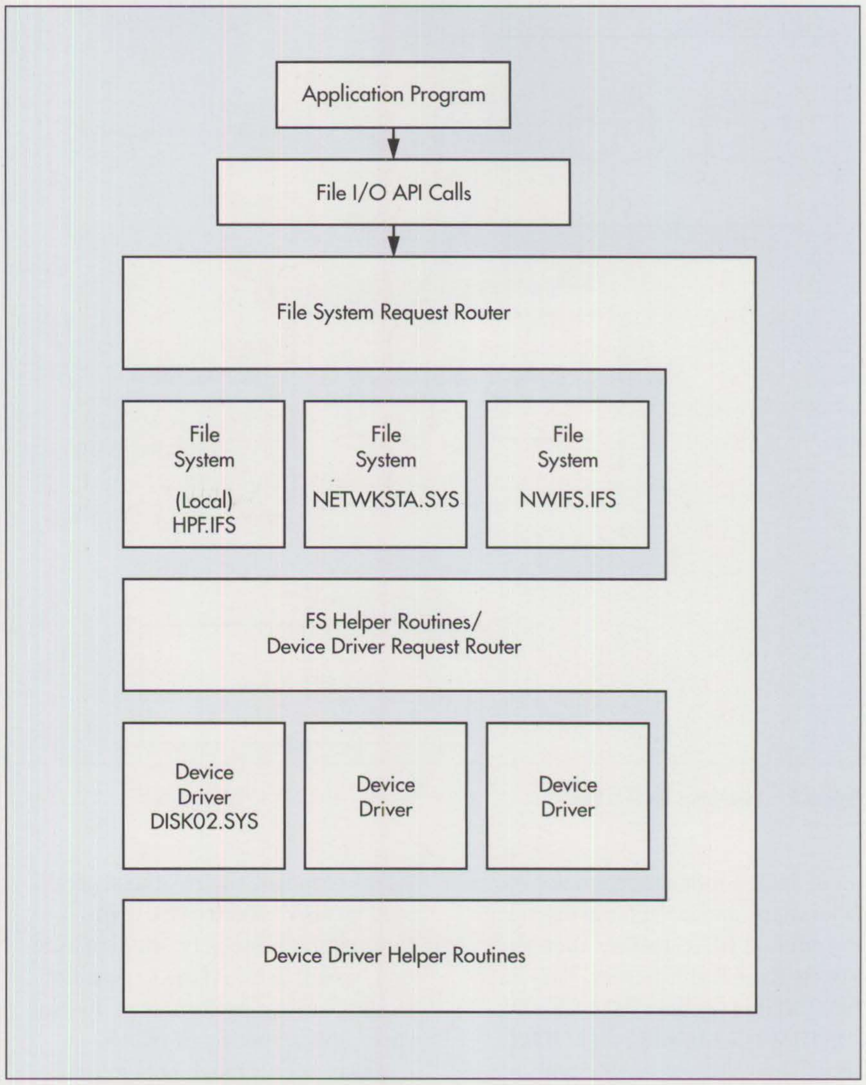


Figure 2. Relationships Between Installable File Systems and Other Key OS/2 Subsystems

This two-step access can be convenient, since NFS does not have to be installed on every requester, and DOS requesters can be hooked in transparently. Some operations, such as defining domain-wide aliases, are not possible for two-step redirections. Two-step redirection works for many file systems, but the version of Novell's LAN Requester for OS/2 1.3 that we tested did not support sharing with IBM LAN Server. OS/2 2.0-compatible file systems should be able to be shared by IBM LAN Server and used by IBM LAN Requester across the network. Some inconsistencies may arise because of file naming differences in DOS, OS/2, and AIX.

To illustrate this concept, in Figure 3, drive F: on the requester is connected to drive C: on the server, while drive G: is connected to an AIX machine via drive N: on the server.

ODI and NDIS Support

3Com®, IBM, and other major networking manufacturers joined together to create an industry stan-

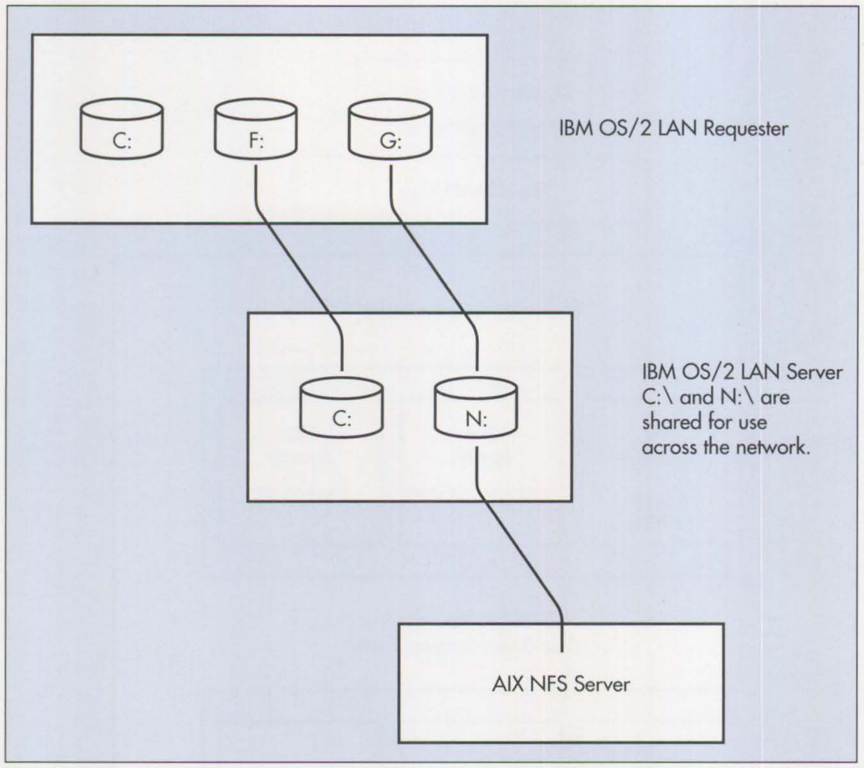


Figure 3. Two-Step Redirection

dard, device-independent networking layer that standardized the interface to common MAC drivers such as Token-Ring and Ethernet. This interface (NDIS) enables IBM LS 2.0 and IBM ES 1.0, and other NDIS-compliant software, to efficiently share a common interface to the adapter cards, which greatly simplifies configuration.

IBM LS 2.0 and ES 1.0 have full support for NDIS-compatible network and protocol drivers. (This is a major enhancement over 1.30.1 versions of LAN Server and Extended Edition, which provided NDIS-compatible communications only for Ethernet.) Novell NetWare Requester for OS/2 uses this enhanced support of the NDIS architecture in an interesting way. The Novell protocol stack emulates an NDIS-compatible MAC driver (as shown in Figure 4), actually causing standard NDIS requests to be routed through a large part of the Novell protocol stack. This enables NDIS requests, such as those generated by IBM's NetBIOS, to go through the same network hardware that Novell uses. At the expense of slightly slower performance in some cases, this rerouting of NDIS requests allows much better coexistence than in previous releases.

NDIS requests are actually routed through Novell's MAC drivers, such as TOKEN.SYS. Because of this, only the network adapters whose drivers are compatible with the ODI can be used, not the full selection of NDIS-compatible network hardware. Changes (described in the next section) must be made to CONFIG.SYS and PROTOCOL.INI for coexistence. These changes may need to be removed temporarily when installing software (such as IBM ES 1.0) that expects the NDIS rather than the ODI MAC drivers to be installed in CONFIG.SYS.

NetWare Requester for OS/2 has always been ODI-based. Now, with the addition of full NDIS network driver and protocol driver support to IBM LAN Server 2.0 and Extended Services 1.0, NetWare's Requester for OS/2 can be fully exploited in the coexistence environment.

The ODI interface was developed during the same time frame as Novell NetWare 386 and the original NetWare Requester for OS/2 (Version 1.2). It is similar to the NDIS interface in that it supports multiple LAN adapters and multiple protocols in a single workstation. The ODI interface creates a logical network board, allowing multiple frame formats to be sent across the same network hardware. Drivers written for ODI (such as for Novell's versions of IPX, NetBIOS, and TCP/IP) do not have to be aware of the particular type of LAN hardware. ODI programmers do not need to include code in their protocol device drivers to support specific LAN topologies. Even though Novell uses ODI, not NDIS, as the interface to its adapter software, Novell's support of ODI "underneath" NDIS now enables both Novell and IBM software to share adapters.

Figure 4 shows how ODINSUP enables coexistence of OS/2 LAN Server 2.0 and Extended Services 1.0 with NetWare Requester for OS/2.

Changes to CONFIG.SYS

Novell's ODI MAC driver, ODINSUP.SYS, enables coexistence with OS/2 LAN Requester 2.0 and with other communications software compatible with IBM's NDIS communications configuration. Installation of the ODINSUP driver is simple, but it requires modifying the CONFIG.SYS, NET.CFG, and PROTOCOL.INI files. This section illustrates the steps to install and configure an IBM/Novell "dual" LAN Requester for OS/2 1.30.2 and 2.0.

When OS/2 LAN Requester 2.0 or Extended Services 1.0 is installed with NetBIOS support, an NDIS LAN driver for a particular LAN adapter is installed, and is loaded by the CONFIG.SYS. For example, the IBMTOK.OS2 NDIS MAC driver is installed if the network is IBM Token-Ring. For communications coexistence between OS/2 LAN Requester 2.0 and NetWare Requester for OS/2, the NDIS LAN driver is replaced with the ODI LAN driver for IBM Token-Ring (TOKEN.SYS), and the Link Support Layer (LSL) driver (LSL.SYS) must be added to support building the ODI stack. Furthermore, ODINSUP.SYS must be substituted into the NDIS stack to hook into the LSL.

Figure 5 shows a portion of a CONFIG.SYS file configured for OS/2 LAN coexistence under OS/2 1.30.2 and OS/2 2.0. This example assumes that an IBM Token-Ring is the primary adapter and a 3Com 3C523 Ethernet adapter is the secondary adapter.

In Figure 5, the load order is important: LSL.SYS and the ODI LAN drivers must be loaded before ODINSUP.SYS. The NDIS drivers, PROTMAN.OS2 and NETBIND.EXE, are still required. PROTMAN.OS2 must be loaded before ODINSUP.SYS. Figure 5 shows that the NDIS MAC drivers IBMTOK.OS2 and ELNKMC.OS2 are commented out, because their functionality has been replaced by the combination of LSL.SYS, TOKEN.SYS, 3C523.SYS, and ODINSUP.SYS. Also, it is recommended that Novell's NetBIOS drivers (NETBIOS.SYS and NBDAEMON.EXE) be used only in non-LAN Server/Requester environments.

Note: All the NetWare Requester drivers, except ODINSUP.SYS, are added automatically by the program that installs NetWare Requester for OS/2.

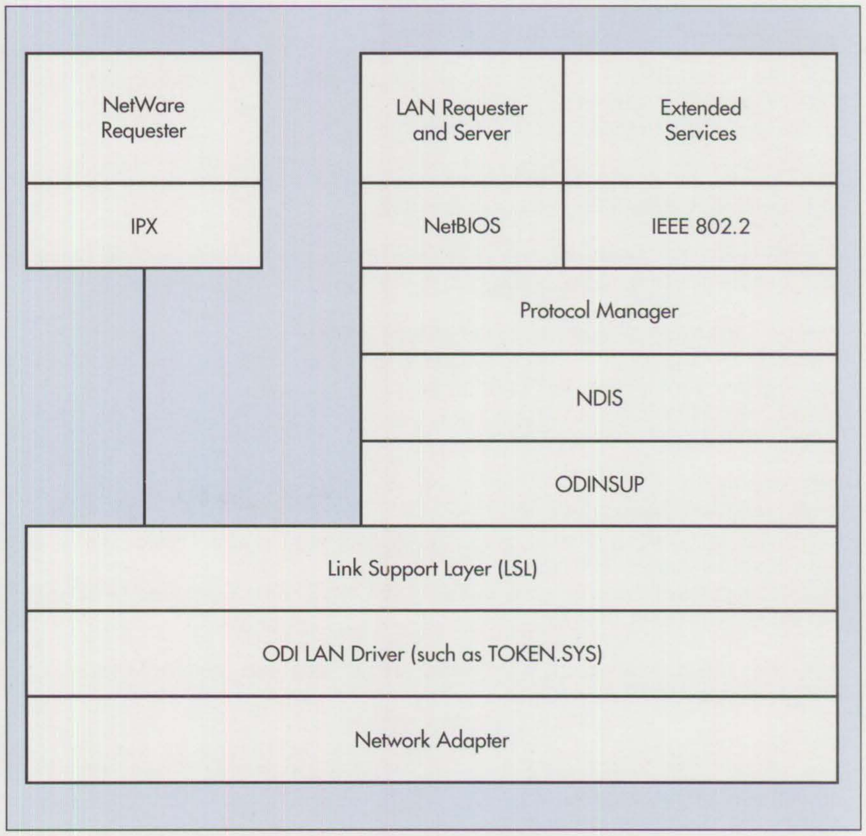


Figure 4. Coexistence of OS/2 LAN Server and NetWare Requester

Figure 6 shows how the CONFIG.SYS file would look under OS/2 EE 1.30.1, using the NDIS-compatible Ethernet MAC driver for the 3Com 3C523 adapter.

Configuration Using NET.CFG

There are several NetWare configuration issues to be resolved when installing the ODI MAC drivers. Most of this installation involves modifying the NET.CFG file. This file is like the IBM OS/2 LAN Requester's IBMLAN.INI file and the NDIS parameter file PROTOCOL.INI, which contain fields for configurable parameters. The NET.CFG file must be modified for the following purposes:

- To increase the size of the LSL buffers
- To specify binding information for the ODI MAC protocol driver, ODINSUP.SYS (required if you are using multiple boards, or if you do not want the default for a particular LAN board)
- To enable the supported frame types for the ODI LAN driver

When using ODINSUP.SYS, it may be necessary to increase the size of the LSL buffers if packet sizes larger than the default of 1,514 are desired. This default value is sufficient for Ethernet and for use with Token-Ring cards that support a maximum 2 KB frame size. A larger LSL buffer size may be desired for increased performance on those LAN adapters that support it. For example, a packet size of 4,210 may be good for configuring for IBM's 16 Mbps Token-Ring network board. See your network board's documentation for more information about the optimum packet size.

```
REM Modify the LIBPATH to include the path to the .DLL files. This path must be specified after
REM the \MUGLIB\DLL directory entry to prevent possible conflicts with the NETAPI.DLL in the
REM C:\NETWARE directory.
LIBPATH=C:\MUGLIB\DLL;...C:\NETWARE;...

REM Append to the PATH statement the path to the NetWare OS/2 utilities (usually L:\OS2).
SET PATH=C:\IBMLAN\NETPROG;...L:\OS2;...

REM Modify the DPATH to include the directory where the NetWare message (.MSG) files were copied.
SET DPATH=C:\IBMLAN\NETPROG;...;C:\NETWARE;...
.
rem --- NetWare Requester statements BEGIN ---
REM The Link Support Layer driver, LSL.SYS, begins the NetWare ODI driver stack and must be loaded
REM before any other NetWare driver.
DEVICE=C:\NETWARE\LSL.SYS
RUN=C:\NETWARE\DDAEMON.EXE

REM The ODI LAN driver must match the adapter installed. This example is for IBM Token-Ring.
DEVICE=C:\NETWARE\TOKEN.SYS
REM Used only with TOKEN.SYS, ROUTE.SYS must load immediately after, for source routing.
DEVICE=C:\NETWARE\ROUTE.SYS
REM This driver is for the 3Com 3C523 Ethernet (secondary adapter).
DEVICE=C:\NETWARE\3C523.SYS

REM The IPX driver must be loaded after the ODI LAN driver.
DEVICE=C:\NETWARE\IPX.SYS

REM The SPX driver is optional, but must be loaded to use the printing utilities and named pipes.
REM It is also required to run Transport Level Interface (TLI) applications.
DEVICE=C:\NETWARE\SPX.SYS
RUN=C:\NETWARE\SPDAEMON.EXE

REM Named-pipe support is optional, and must be loaded after the SPX driver. Two configurations
REM are possible: “client only” and “server.”

REM Load the NMPIPE.SYS driver for named-pipe, client-only support.
DEVICE=C:\NETWARE\NMPIPE.SYS

REM Load the NPSERVER.SYS driver, along with NMPIPE.SYS, for named pipes server support.
DEVICE=C:\NETWARE\NPSERVER.SYS

REM NPDAEMON.EXE must be executed for both named-pipe configurations. If configured as a
REM “server,” a server name must be specified.
RUN=C:\NETWARE\NPDAEMON.EXE np_servername

REM The NetWare OS/2 requester driver must be loaded after IPX, SPX, and the named-pipes drivers.
DEVICE=C:\NETWARE\NWREQ.SYS

REM The NetWare OS/2 Installable File System must be loaded after the requester driver.
IFS=C:\NETWARE\NWIFS.IFS

REM The NetWare daemon must be loaded after the NetWare OS/2 requester driver.
RUN=C:\NETWARE\NWDAEMON.EXE

REM The NetBIOS driver is optional and should only be considered in a non-LS environment.
rem DEVICE=C:\NETWARE\NETBIOS.SYS
rem RUN=C:\NETWARE\NBDAEMON.EXE

REM The NetWare support for MVDMs is optional, but it is required to run the DOS NetWare
REM utilities, that is, SYSCON.EXE.
DEVICE=C:\NETWARE\VIPX.SYS
```

Figure 5. CONFIG.SYS File for LAN Coexistence Under OS/2 1.30.2 and OS/2 2.0 (continued)

```
REM The NetWare Print Spooler driver is optional. It is required only if you want to print to a
REM NetWare network printer queue.
RUN=C:\NETWARE\NWSPOOL.EXE
rem --- NetWare Requester statements END ---

DEVICE=C:\IBMCOM\LANMSGDD.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTMAN.OS2 /I:C:\IBMCOM
DEVICE=C:\IBMCOM\PROTOCOL\LANDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\LANDLLDD.OS2
DEVICE=C:\IBMCOM\PROTOCOL\NETBEUI.OS2
DEVICE=C:\IBMCOM\PROTOCOL\NETBIOS.OS2

REM The ODINSUP.SYS driver must be loaded before NETWKSTA.SYS (.200).
DEVICE=C:\NETWARE\ODINSUP.SYS
REM The NDIS MAC driver(s) must be commented out (IBMTOK.OS2 and ELNKMC.OS2 in this example).
rem DEVICE=C:\IBMCOM\MACS\IBMTOK.OS2
rem DEVICE=C:\IBMCOM\MACS\ELNKMC.OS2
RUN=C:\IBMCOM\PROTOCOL\LANDLL.EXE
RUN=C:\IBMCOM\PROTOCOL\NETBIND.EXE
RUN=C:\IBMCOM\LANMSGEX.EXE
DEVICE=C:\IBMLAN\NETPROG\RDRHELP.SYS
IFS=C:\IBMLAN\NETPROG\NETWKSTA.SYS /I:C:\IBMLAN
```

Note: For OS/2 2.0, RDRHELP.SYS would be named RDRHELP.200 and NETWKSTA.SYS would be named NETWKSTA.200.

Figure 5. CONFIG.SYS File for LAN Coexistence Under OS/2 1.30.2 and OS/2 2.0

The following shows how to modify NET.CFG for adapters that support a maximum frame size of 4 KB.

```
link support
    buffers 15 4210
```

When the NetWare Requester is initialized, it tries to establish a connection with the primary network board. Using multiple network boards can increase communication speed if the workstation is connected to multiple physical networks.

The ODI MAC driver must bind to an ODI LAN driver for network communication to take place. The ODI MAC driver, by default, will attempt to locate a supported ODI LAN driver if no binding information is specified in the NET.CFG file. If an ODI LAN driver is found, ODINSUP.SYS will attempt to bind to it.

To support more than one ODI LAN driver, or to specify which ODI LAN driver to bind to, requires modifying the NET.CFG file. ODINSUP.SYS can be bound to as many as four ODI LAN drivers.

Bind entries specify the name of the ODI LAN driver and, optionally, the instance number of the adapter. The name of the ODI LAN driver is normally the name of the ODI LAN driver's file (such as TOKEN for TOKEN.SYS). If more than one LAN adapter is installed in the workstation, the instance number may be required. If an instance number is not specified, ODINSUP.SYS will bind to the first ODI LAN driver found. For example, in Figure 7, two 3Com Ethernet 3C523 adapters are installed. Here, ODINSUP.SYS will bind to the first loaded instance of the ODI LAN driver. If not specified, the default value for the instance number is 1.

Currently, ODINSUP.SYS supports only ODI LAN drivers that are compatible with Token-Ring and Ethernet LANs. The NET.CFG file must be modified to enable the minimum number of frame types for Token-Ring and Ethernet. For Token-Ring ODI LAN drivers, the TOKEN-RING and TOKEN-RING_SNAP frame types must be specified. For the Ethernet ODI LAN driver, the ETHERNET_802.2, ETHERNET_SNAP, and ETHERNET_II frame types must be specified. Figure 8 contains a sample NET.CFG file that shows how these frame types are specified.

The NET.CFG file may be used for configuration requirements other than those addressed above. Refer to the Novell booklet *NetWare Requester for OS/2* (furnished with the documentation for the NetWare operating system) for more information about configuring NetWare drivers by using the NET.CFG file.

Changes to PROTOCOL.INI

The NDIS PROTOCOL.INI file is still required to tell the NDIS protocols which ODI MAC driver to bind to. Normally, all PROTOCOL.INI information for NDIS MAC drivers can be removed, since no ODI MAC-

```
REM ---------BEGINNING OF CONFIG.SYS---------------------
.
.
.
LIBPATH=C:\MUGLIB\DLL;...C:\NETWARE;...
SET PATH=C:\IBMLAN\NETPROG;...L:\OS2;...
SET DPATH=C:\IBMLAN\NETPROG;...;C:\NETWARE;...
.
.
.
REM --- NetWare Requester statements BEGIN ---
DEVICE=C:\NETWARE\LSL.SYS
RUN=C:\NETWARE\DDAEMON.EXE
DEVICE=C:\NETWARE\3C523.SYS
DEVICE=C:\NETWARE\IPX.SYS
DEVICE=C:\NETWARE\SPX.SYS
RUN=C:\NETWARE\SPDAEMON.EXE
DEVICE=C:\NETWARE\NMPIPE.SYS
DEVICE=C:\NETWARE\NPSERVER.SYS
RUN=C:\NETWARE\NPDAEMON.EXE   np_servername
DEVICE=C:\NETWARE\NWREQ.SYS
IFS=C:\NETWARE\NWIFS.IFS
RUN=C:\NETWARE\NWDAEMON.EXE
rem DEVICE=C:\NETWARE\NETBIOS.SYS
rem RUN=C:\NETWARE\NBDAEMON.EXE
DEVICE=C:\NETWARE\VIPX.SYS
RUN=C:\NETWARE\NWSPOOL.EXE
REM --- NetWare Requester statements END ---

REM --- Drivers for Comm. Mgr. and IBM LAN Requester v1.30.1 ---
DEVICE=C:\CMLIB\LANDD.SYS
DEVICE=C:\CMLIB\PROTMAN.OS2 /I:C:\CMLIB

REM ODINSUP replaces the NDIS MAC driver ELNKMC.OS2.
DEVICE=C:\NETWARE\ODINSUP.SYS
REM DEVICE=C:\CMLIB\ELNKMC.OS2

DEVICE=C:\CMLIB\ETHERDD.SYS CFG=C:\CMLIB\LANREQ.CFG
RUN=C:\CMLIB\ACSEPSYS.EXE
RUN=C:\CMLIB\NETBIND.EXE

DEVICE=C:\CMLIB\NETBDD.SYS CFG=C:\CMLIB\LANREQ.CFG
DEVICE=C:\IBMLAN\NETPROG\RDRHELP.SYS
IFS=C:\IBMLAN\NETPROG\NETWKSTA.SYS /I:C:\IBMLAN

REM --------END OF CONFIG.SYS-------------------------
```

Figure 6. CONFIG.SYS File for OS/2 EE 1.30.1 and OS/2 2.0

specific information is necessary in the PROTOCOL.INI file.

The PROTOCOL.INI file must also specify sections for each NDIS protocol used. Part of PROTOCOL.INI is the BINDINGS= statement, which specifies the NDIS MACs to which the protocol should bind. The name specified should be the name of the ODI LAN driver (such as 3C523, TOKEN, 3C503) that is installed. If the ODI LAN driver's name starts with a number (such as 3C523), the MAC name for the BINDINGS= statement must begin with the letter *x*. Therefore, the BINDINGS= statement for the 3C523 LAN driver would be BINDINGS = x3C523. If binding to multiple ODI LAN drivers, the entries on the BINDINGS= statement should be separated by a comma (such as BINDINGS=TOKEN,x3C503).

When an adapter instance other than the first is used (such as when ODINSUP.SYS is bound to the second TOKEN ODI LAN driver), the MAC name must have the instance number appended to the end of the

```
link support
    buffers 15 4210

protocol odinsup
    bind token     ;Bind to the first  instance of TOKEN ODI LAN driver
    bind 3c523 1   ;Bind to the first  instance of 3C523 ODI LAN driver
    bind 3c523 2   ;Bind to the second instance of 3C523 ODI LAN driver
```

Figure 7. NET.CFG File for One Token-Ring and Two 3C523 LAN Adapters

```
link support
      buffers 15 4210

protocol odinsup
      bind token       ;Bind to the first  instance of TOKEN ODI LAN driver
      bind 3c523 1     ;Bind to the first  instance of 3C523 ODI LAN driver
      bind 3c523 2     ;Bind to the second instance of 3C523 ODI LAN driver

link driver token
      frame token-ring          ;Required
      frame token-ring_snap     ;Required

link driver 3c523
      frame ethernet_802.3      ;Optional
      frame ethernet_802.2      ;Required
      frame ethernet_ii         ;Required
      frame ethernet_snap       ;Required
```

Figure 8. NET.CFG File showing Token-Ring and Ethernet Frame Types

ODI LAN driver's name (such as TOKEN2 for the second instance, x3C5234 for the fourth instance). Appropriate MAC names are displayed when ODINSUP.SYS is loaded.

Figure 9 shows a sample PROTOCOL.INI file for OS/2 1.30.2 and OS/2 2.0 running OS/2 LAN Requester 2.0 and NetWare Requester for OS/2. It shows the binding instruction changes required for the ODI MAC driver.

For OS/2 EE 1.30.1, using the NDIS-compatible Ethernet MAC driver for the 3Com 3C523 adapter, the PROTOCOL.INI would look like the one in Figure 10.

It is important to note that when configuring the network resource parameters for your LAN hardware, you must use NetWare's NET.CFG file, not the PROTOCOL.INI file, which is installed as part of the IBM NDIS-compatible communications software. In fact, with ODINSUP.SYS installed, the functionality of the PROTOCOL.INI file is limited to just the BINDINGS = statement – all other configurable parameters are ignored. Therefore, if you had customized PROTOCOL.INI before installing ODINSUP.SYS, you must translate these configurable parameters to NetWare's NET.CFG file. For example, a "NETADDRESS = T40003800FCB9" statement in PROTOCOL.INI is translated by inserting a "NODE ADDRESS 40003800FCB9" statement in the LINK DRIVER section of NET.CFG.

For more information about files like PROTOCOL.INI used for tuning communications parameters, refer to *IBM OS/2 LAN Server 2.0 Network Administrator Reference, Volume 2: Performance Tuning* (S04G-1033).

Performance

Upon installation, ODINSUP.SYS consumes approximately 3,950 bytes of memory. Each additional adapter that ODINSUP.SYS binds will increase memory usage by approximately 1,050 bytes.

Novell did a limited performance test using the 3Com EtherLink II® network adapter, comparing the native NDIS ELNKII.OS2 MAC driver and the ODI 3C503.SYS LAN driver with ODINSUP.SYS. The configuration using the ODINSUP.SYS driver was 8% slower than the native NDIS MAC driver. Novell claims that the performance degradation is the result

```
[PROT_MAN]
DriverName = PROTMAN$

[IBMLXCFG]
LANDD_nif = LANDD.nif
NETBEUI_nif = NETBEUI.nif

;*-----------------------------------------------*
;*------------- PROTOCOL SECTION -------------*
;*-----------------------------------------------*

[LANDD_nif]
DriverName = LANDD$
; Bindings = IBMTOK_nif,ELNKMC_nif
; For first instance of TOKEN board and first instance of 3C523 board.
Bindings = TOKEN,x3C523

[NETBEUI_nif]
DriverName = netbeui$
; Bindings = IBMTOK_nif,ELNKMC_nif
; For first instance of TOKEN board and first instance of 3C523 board.
Bindings = TOKEN,x3C523
```

Note: The above PROTOCOL.INI example is a complete file. All portions of the PROTOCOL.INI that are NDIS-related can be deleted or REMarked out.

Figure 9. PROTOCOL.INI File with Binding Instructions for ODI.MAC Driver

```
;-------------- Protocol Manager Definition -------------

[PROTOCOL_MANAGER]
      DriverName = PROTMAN$

;----------- IBM ETHERAND Protocol Definition -----------

[ETHERAND]
      DriverName = OS2EE$
      Bindings = x3C523          ;for first instance of 3C523 board

      ; Bindings = x3C5232       ;for second instance of 3C523 board
```

Note: Communications Manager cannot handle an OS2EE entry such as Bindings = x3C523,x3C5232.

Figure 10. PROTOCOL.INI File for Ethernet Adapter Under OS/2 EE 1.30.1

of Novell's receive buffering implementation in OS/2, as compared to the use of ReceiveLookAhead buffers, which both DOS ODI and OS/2 NDIS exploit.

The order in which the two network installable file systems – NWIFS.IFS for NetWare and NETWKSTA.SYS (NETWKSTA.200 in OS/2 2.0) for IBM LAN Requester – are loaded in CONFIG.SYS involves trade-offs. If any IBM OS/2 LAN Server names duplicate NetWare server names on the same network, then accessing a resource by its UNC name results in connection to the first one mentioned in CONFIG.SYS.

For example, if NWIFS.IFS comes before NETWKSTA.SYS (.200), then NetWare servers will be searched first for UNC names, resulting in quicker initial connection for Novell resources, but slightly slower connection to IBM server resources. Usually, it is preferable to install NWIFS.IFS before NETWKSTA.SYS in the CONFIG.SYS file. The order of path elements that refer to network drives or UNC names can also affect performance. If references to NetWare path elements come after references to drives redirected to IBM LAN Server, some users may encounter problems when the system automatically searches for files on that NetWare path.

The difference in performance when NWIFS.IFS and NETWKSTA.SYS (NETWKSTA.200) are loaded in reverse order is significantly smaller in OS/2 2.0, because of improvements in the mechanism used for searching UNC names. OS/2 2.0 uses a two-pass mechanism. It allows the OS/2 kernel to first check whether either IFS already has a connection to the desired server. After that, the kernel asks each IFS, in turn, to locate that server name on its network. This reduces the delay that the second IFS incurs when the first IFS tries futilely to contact a server that is on the second IFS's network. In OS/2 2.0, therefore, the order of the two LAN IFSs in CONFIG.SYS is less important.

Limitations

The ODINSUP.SYS driver for OS/2 requires an LSL.SYS file dated on or after May 19, 1991, and a TOKEN.SYS file dated on or after September 17, 1991.

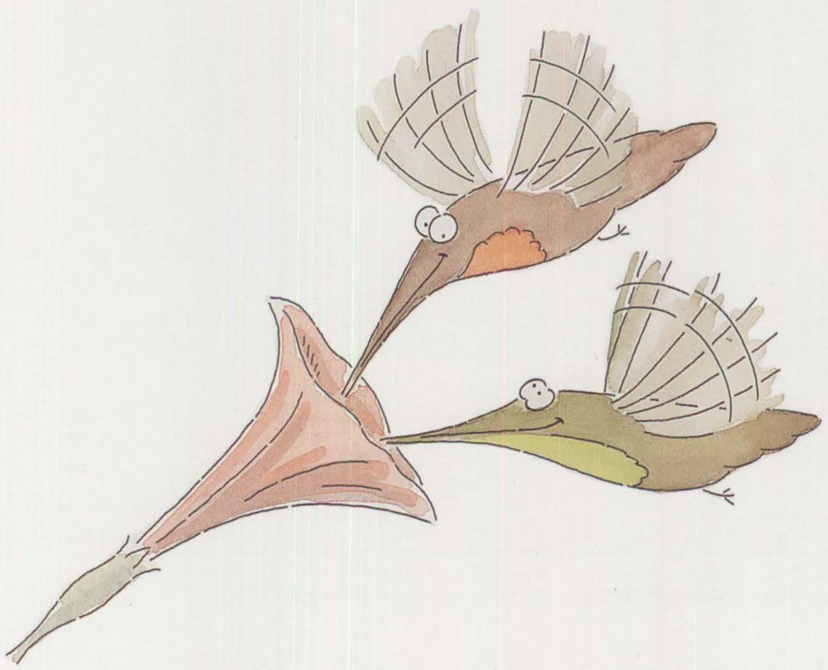
The ODINSUP driver supports only those configurations for which an ODI-compatible LAN driver is available for the network adapter. This may limit the use of ODINSUP, especially in the Ethernet environment, since Novell is shipping a limited number of ODI LAN drivers with its latest release of the NetWare Requester for OS/2. As of this writing, other vendors are developing several ODI LAN adapter drivers.

Instead of using ODINSUP.SYS, the CMGRLAN.SYS driver can be used in combination with the native NDIS MAC drivers (provided with LAN Server 2.0 and Extended Services 1.0) for Token-Ring and PC Network coexistence configurations. However, CMGRLAN.SYS is not compatible with Ethernet in OS/2 1.30.2 or OS/2 2.0, using the native NDIS network drivers. For Token-Ring and PC Network, NETWKSTA.SYS (.200) must be loaded before CMGRLAN.SYS for binding to take place, and to allow the NetWare Requester (NWREQ.SYS) to locate a file server at IPL time and to perform server-client I/O.

Currently, starting the IBM OS/2 LAN Server on a machine with the NetWare Requester drivers installed is not officially supported. However, most environments in which this configuration is attempted should execute successfully.

Doug Spelce *is a senior associate programmer in IBM Austin's Personal Systems Programming Center. He joined IBM in 1987 and presently works in the OS/2 LAN Server System Test organization, specializing in LAN coexistence testing. His system test assignments have included PC LAN Program, DOS LAN Requester, OS/2 LAN Server/Requester, Lotus Notes®, and Novell LAN coexistence. Doug received a BS in computer science from Texas A&M University.*

Steve French *is a staff programmer in IBM Austin's Personal Systems Programming Center. He joined IBM in 1989 and is currently the lead programmer for the OS/2 LAN Requester. Steve received a BA in computer science and an MS in electrical and computer engineering from Rice University.*



IBM 8209 LAN Bridge Connects Ethernet Clients to Novell and IBM Servers

Dana Beatty
IBM Corporation
Boca Raton, Florida

This article shows how to connect Ethernet clients through the IBM 8209 LAN Bridge to either a DOS Novell Server or an IBM OS/2 LAN Server.

Both Ethernet and Novell are recent additions to IBM Local Area Networks (LANs). Although there are manuals, software, and hardware to help you include Ethernet and Novell in an IBM LAN, the task still looms large. Fortunately, the task is not as involved as it first appears.

Figure 1 shows a sample configuration for this environment. Note that both the IBM LAN Server and Novell NetWare Server are on the IBM Token-Ring LAN, but the only connection to these servers for the Ethernet clients is via the IBM 8209 LAN Bridge. Furthermore, the IBM Token-Ring LAN is operating at 16 Mbps.

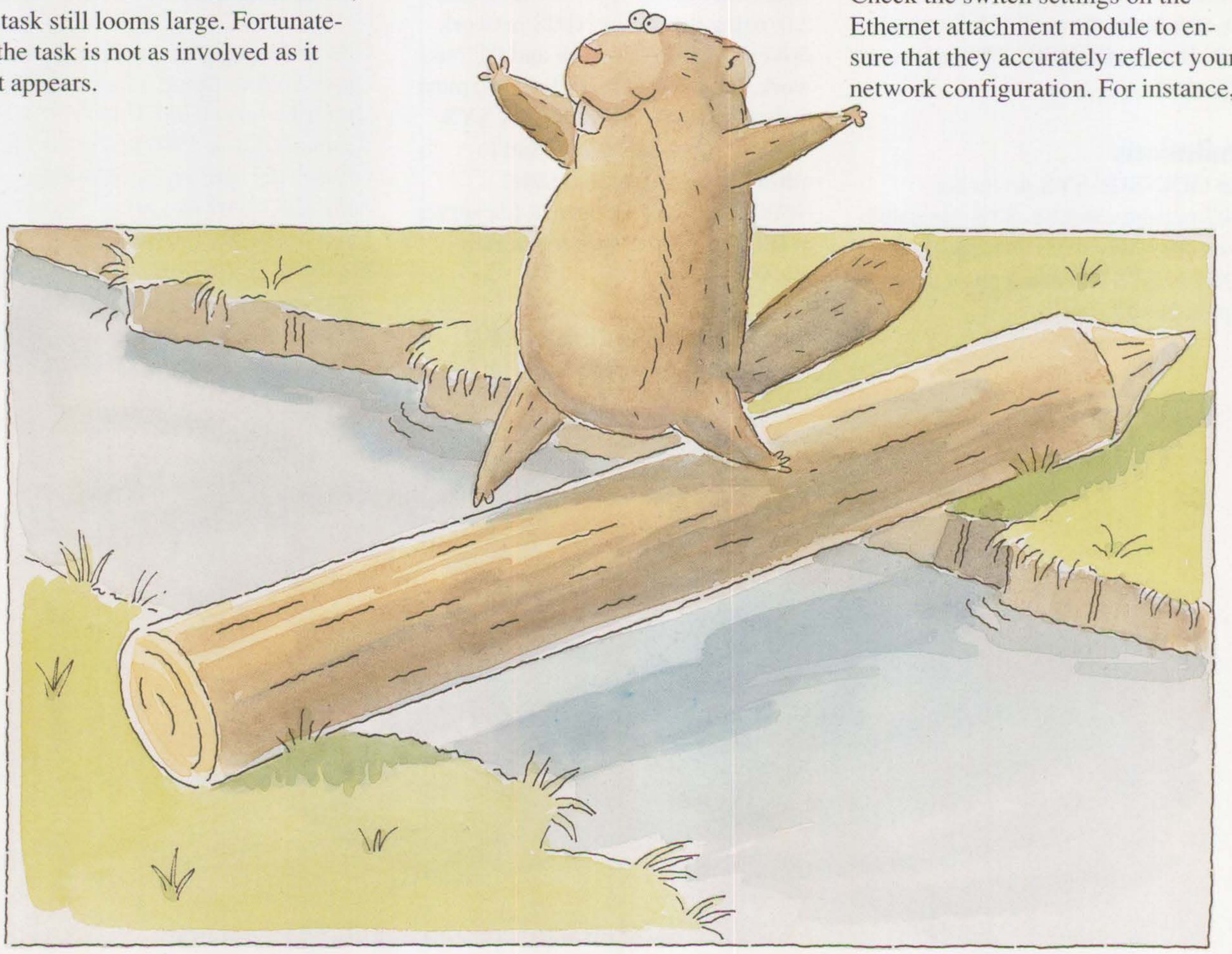
Configuration Steps

Cabling these individual networks is beyond the scope of this article. Once the cabling is done, some crucial steps must be followed so that these heterogeneous networks can communicate.

IBM 8209 LAN Bridge

The original Ethernet attachment module will not work with Novell's IPX. Therefore, you should ensure that the enhanced Ethernet attachment module for the IBM 8209 Bridge is the latest level[1].

Check the switch settings on the Ethernet attachment module to ensure that they accurately reflect your network configuration. For instance,



[1]As of January 1992, the latest part number for the IBM 8209 Bridge is 74F8629 with an EC level of C25206.

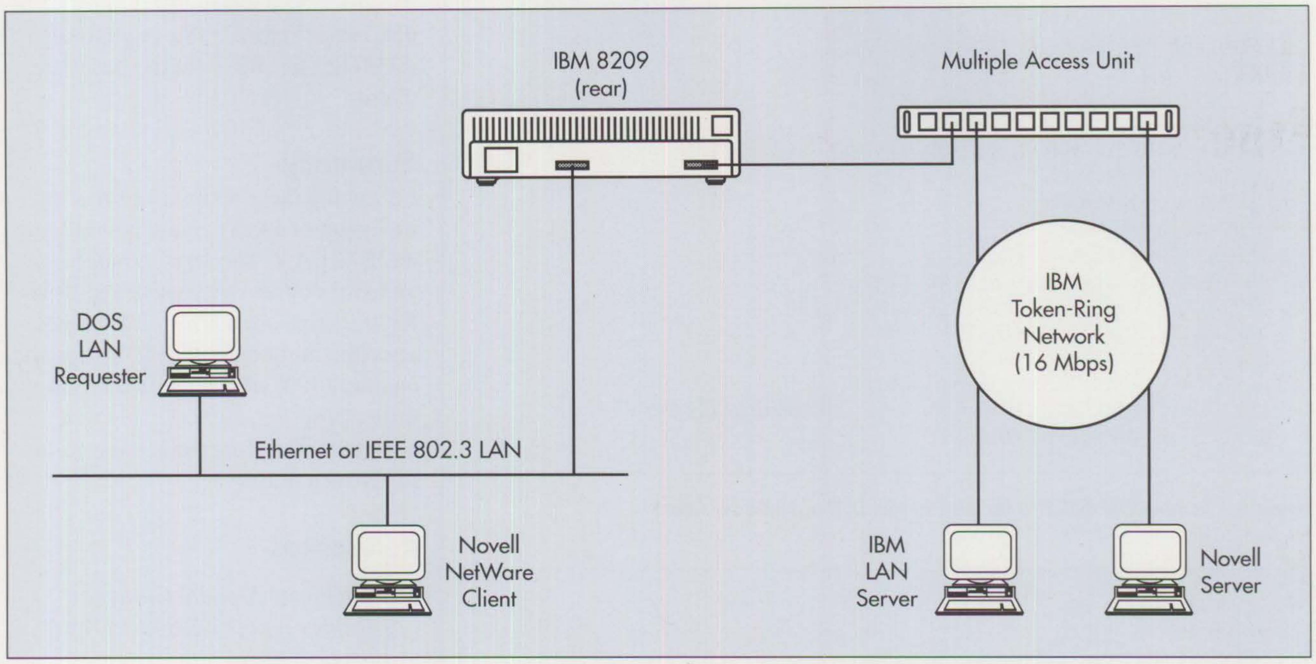
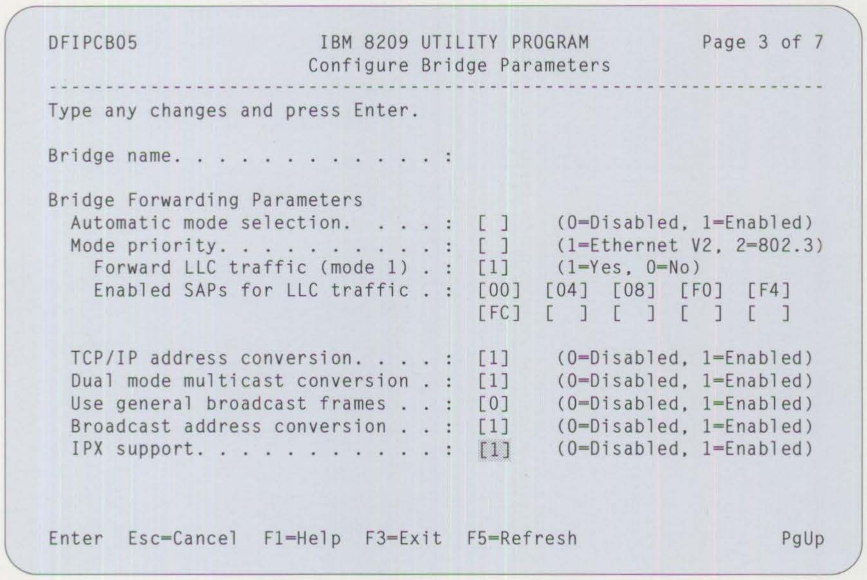


Figure 1. Network Configuration



```
DFIPCB05                IBM 8209 UTILITY PROGRAM            Page 3 of 7
                       Configure Bridge Parameters
---------------------------------------------------------------------
Type any changes and press Enter.

Bridge name. . . . . . . . . . . . :

Bridge Forwarding Parameters
  Automatic mode selection. . . . :  [ ]    (0=Disabled, 1=Enabled)
  Mode priority. . . . . . . . . . :  [ ]    (1=Ethernet V2, 2=802.3)
    Forward LLC traffic (mode 1) . :  [1]    (1=Yes, 0=No)
    Enabled SAPs for LLC traffic . :  [00]  [04]  [08]  [F0]  [F4]
                                      [FC]  [  ]  [  ]  [  ]  [  ]

  TCP/IP address conversion. . . . :  [1]    (0=Disabled, 1=Enabled)
  Dual mode multicast conversion . :  [1]    (0=Disabled, 1=Enabled)
  Use general broadcast frames . . :  [0]    (0=Disabled, 1=Enabled)
  Broadcast address conversion . . :  [1]    (0=Disabled, 1=Enabled)
  IPX support. . . . . . . . . . . :  [1]    (0=Disabled, 1=Enabled)


Enter  Esc=Cancel  F1=Help  F3=Exit  F5=Refresh                   PgUp
```

Figure 2. Third Screen of Bridge Parameters in IBM 8209 Bridge Utility Program

switch number 5 is set to ▶ to denote that the IBM Token-Ring side of the network is operating at 16 Mbps. The remaining switches are left at their default settings.

Next, enable IPX support using the IBM 8209 Bridge Utility program, version 3.0 or later. Select the Configure Bridge Parameters option from the Main menu. There are seven screens of bridge parameters. Page forward to the third screen, which is shown in Figure 2. Change the IPX Support field to "1" to enable Novell to communicate across the bridge. Be sure to press the F3 key to save these changes.

Novell Server

Because this server is attached to the IBM Token-Ring network, source-routing support must be enabled. Novell provides an IBM Token-Ring source-routing driver for communication across IBM Token-Ring source-routing bridges. *All* Novell clients must enable this support to communicate across this bridge. To do so, the AUTOEXEC.NCF file must include the statement LOAD ROUTE, which loads the source-routing driver, as shown in Figure 3.

```
file server name NWARE311A
ipx internal net 386311A
load TOKEN
bind IPX to TOKEN net=A
load route
load install
load monitor
```

Figure 3. Sample Novell Server AUTOEXEC.NCF File

Client Logon

Client logon can be easy and transparent. As an example, one Ethernet client is configured as a DOS LAN

```
BREAK=ON
BUFFERS=20
FILES=100
LASTDRIVE=Z
SHELL=C:\DOS\COMMAND.COM /P /E:256
DEVICE=C:\HIMEM.SYS
DEVICE=C:\DOS\ANSI.SYS
FCBS=16,8
IFS=C:\PCLP13BS\REDIR.SYS
INSTALL=C:\PCLP13BS\IFSFUNC.EXE
INSTALL=C:\DOS\FASTOPEN.EXE C:=(50,25)
DEVICE=\LANMAN\PROTMAN.EXE
DEVICE=\LANMAN\MACETH.DOS
DEVICE=\LANMAN\DXMAOMOD.SYS 001 ┐
DEVICE=\LANMAN\DXMEOMOD.SYS     │ Required for DLR only
DEVICE=\LANMAN\DXMTOMOD.SYS     ┘
```

Figure 4. Sample CONFIG.SYS for an Ethernet DLR or NetWare Client

```
\NETBIND
@ECHO OFF
SET COMPSPEC=C:\DOS\COMMAND.COM
PATH=C:DOSLAN;C:\DOS
PROMPT $P$G
YNPROMPT Y N 30 START DOS LAN REQUESTER (Y/N) ?
IF ERRORLEVEL 1 GOTO NODLR
NET START
IF ERRORLEVEL 1 GOTO NODLR
CALL INITFSI.BAT
NET
:NODLR
```

Figure 5. Sample AUTOEXEC.BAT for DLR Client

```
@ECHO OFF
SET COMPSPEC=C:\DOS\COMMAND.COM
PATH=C:DOSLAN;C:\DOS;C:\NetWare
PROMPT $P$G
YNPROMPT Y N 30 START NetWare REQUESTER (Y/N) ?
IF ERRORLEVEL 1 GOTO NONETW
IPX
NET5
PROMPT $P$G
S:
LOGIN WS1
PATH Z:.;Y:.;C:\DOS;C:\
:NONETW
```

Figure 6. Sample AUTOEXEC.BAT for IPX/DOS 5.0 NetWare Client

Requester (DLR), and the other Ethernet client is configured as a Novell NetWare client. Both clients log on to their respective servers, transparently, through the IBM 8209 Bridge. Two files are important to this task: CONFIG.SYS and AUTOEXEC.BAT. Figure 4 shows a sample CONFIG.SYS that can be used for either the NetWare or DLR client. Figure 5 outlines a sample AUTOEXEC.BAT for the DLR client, while Figure 6 shows a sample AUTOEXEC.BAT for the NetWare client.

Summary

Following these steps simplifies the inclusion of Ethernet and Novell into an IBM LAN. The keys to heterogeneous connectivity using the IBM 8209 Bridge are having an Ethernet attachment module with IPX support, enabling IPX support with the bridge utility program, and ensuring that source-routing support is enabled on the Novell nodes.

References

- *Attachment Module Guide for Ethernet and IEEE 802.3 LANs* (GA27-3891)
- *Novell IPX Protocol Support Supplement to the Attachment Module Guide for Ethernet and IEEE 802.3 LANs* (GD21-0047)
- *DOS ODI Workstation* (Novell part number 100-000871-001)
- *8209 Local Area Network Bridge* (SA21-9994)

***Dana L. Beatty** is a staff programmer in the Communications Subsystem Development I department in IBM Entry Systems Technology, Boca Raton, Florida. She joined IBM in 1985 in Austin, Texas as a systems analyst, working on the design and development of applications written to the IEEE 802.2 API. Ms. Beatty has written several articles for the* IBM Personal Systems Developer, *has instructed systems engineers at the Arthur K. Watson International Education Centre, and has presented at SHARE and GUIDE conferences. Ms. Beatty has a BA in computer science from the University of Texas in Austin.*

Backup and Restore in an IBM NetWare Environment

David G. Weingard
Cheyenne Software, Inc.
Roslyn, New York

ARCserve® is a software program that resides on a LAN server and backs up all the data on the entire network. It recognizes a variety of network hardware and data-file formats. ARCserve can also distribute data between the network and mainframe environments, thereby enabling enterprise-wide data management.

A LAN administrator, who uses IBM PS/2 hardware and the Novell NetWare operating system, wants a powerful LAN backup and restore solution...

A key client loses a PS/2 Model 95 NetWare server in a fire, and needs the data that was stored on it – right now...

A data processing manager wants to manage corporate data on an enterprise-wide basis, integrating information that resides on IBM mainframe systems, NetWare servers, and user workstations...

The solution to these types of needs is Cheyenne Software's® ARCserve, a server-based backup and restore software product for all NetWare LANs. ARCserve, whose architecture is shown in Figure 1, includes support for all IBM SCSI tape drives, optical drives, and Micro Channel host-adapter hardware. ARCserve backs up or restores an entire LAN, including local servers, remote servers, workstations, program/data/NetWare bindery files, and NetWare file attributes and trustee assignments. The average transfer rate from an IBM NetWare Server to the IBM 2.3 GB tape unit is 15 MB per minute.

Server-based backup and restore of network data permits the maximum performance of IBM PS/2 hardware. With server-based backup and restore, data passes directly from the server, through a Micro Channel SCSI adapter, to an IBM tape drive or optical device as fast as the hardware can handle it. In contrast, workstation-based backup and restore solutions are limited to the speed of the LAN. They impact network performance because data must travel from the server to the workstation.

ARCserve's Auto Pilot feature provides automated data management based on the target (file server, volume, directory, or workstation) and options specified by the user. Auto Pilot acts as an invisible operator, guiding the user through intelligent tape management, tape rotation, offsite recommendations, and more. Auto Pilot also has disaster-recovery functions that restore the server to its status at an earlier date.

Application Examples

The following real-life scenarios illustrate how data can be managed optimally in a corporate environment using ARCserve.

Scenario 1: Hierarchical Storage Management

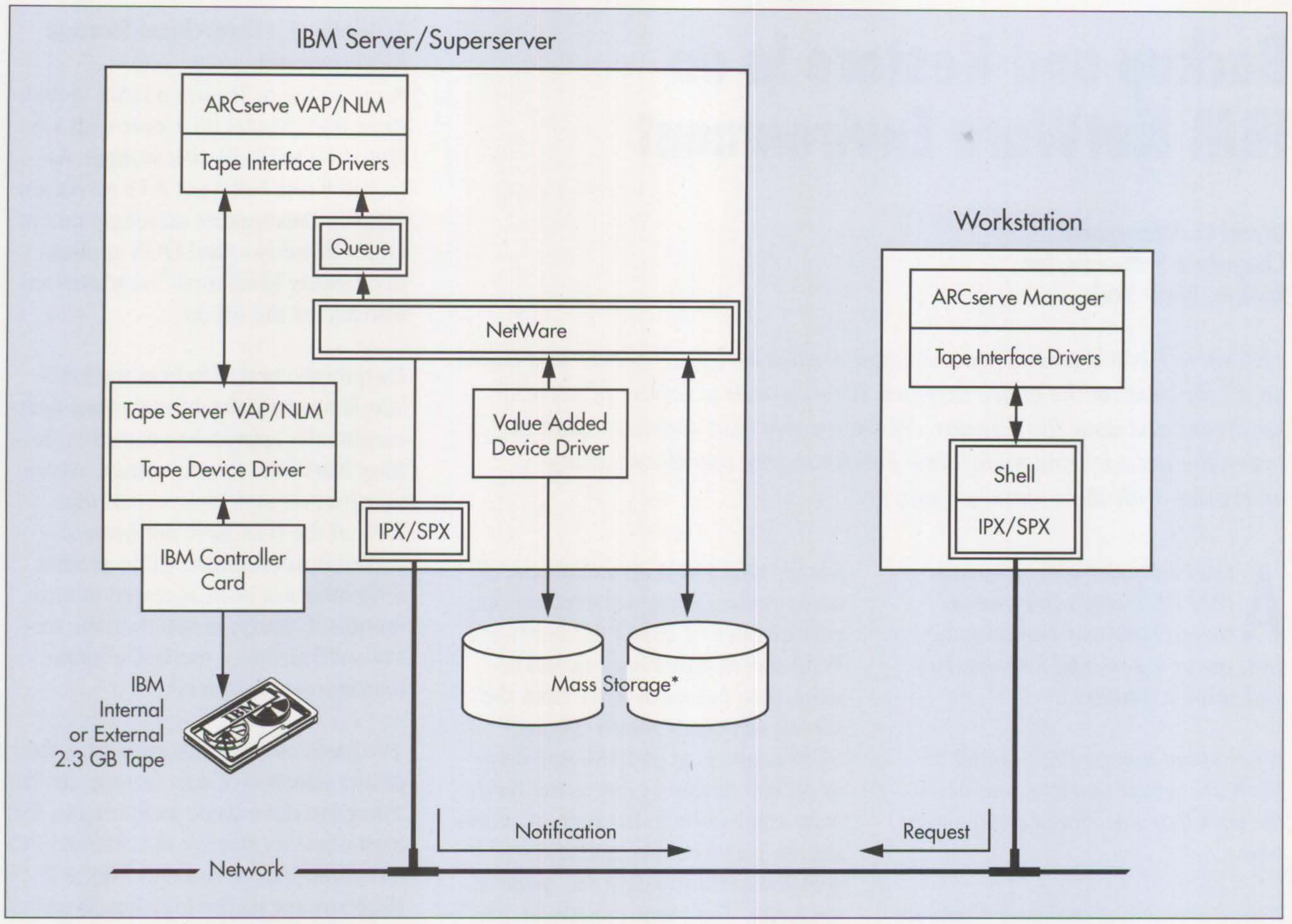
A corporate office has a LAN including a PS/2 Model 95 server with 1.6 GB of online hard disk storage. Attached to the NetWare 3.11 server are 150 PS/2 computers running a mix of OS/2, Windows, and DOS applications. Sixty Macintosh® workstations also access the server.

Despite the best efforts of the LAN administrators, the server's hard disk continually approaches capacity, limiting NetWare's performance. Analysis of server statistics reveals that 25% of the files have not been accessed in six months, while another 35% have not been accessed in three months. Clearly, to maintain the server's efficiency, a method of data management is crucial.

The basis of data management is the proper selection of data storage media. There are three basic technologies for high-capacity storage in LAN and PC environments, as listed in Figure 2. Each storage technology has its advantages and disadvantages.

The advantage of hard disks is their extraordinary speed, offset by their high cost per megabyte. Optical storage devices are now much faster, and the cost per megabyte is quite attractive because of their higher capacities. Tape is by far the most cost-effective medium for storage of voluminous data, but its relatively slow access time makes its use as an online storage device impractical in time-sensitive situations. All three are available as removable media; however, only optical devices and tapes are robust enough to survive a drop on the floor.

Each technology provides significant benefits for specific applications, but no single approach can cost-effectively address all needs. The most effective strategy is to use each form of



* Mass storage devices include IBM Optical Drive, IBM Mainframe DASD, Hard Disk, Optical Jukebox, and CD-ROM.

Figure 1. ARCserve Architecture

storage concurrently. Hierarchical Data Management (HDM) is the process that makes this task possible.

Hierarchical storage, simply defined, is the ranking of data by importance and speed requirements, and its storage on the appropriate media. Only data that must be accessed instantly should consume expensive hard disk space. Such data is known as *hot* data. It should remain on expensive, fast media as long as it is frequently accessed. *Warm* data should be readily available in large quantities, but does not need to be accessible instantaneously. Optical storage is often used to store warm data. *Cold* data is both backup data and data that is accessed infrequently. Because most data in a network will become cold sooner or later, it is best to store it on tape at that point.

Storage Device	Access Time	Cost Per Megabyte	Capacity
Hard Disk	10 Milliseconds (hot)	$2.54	2 Gigabytes
Optical Disk	60 Milliseconds (warm)	$0.40	1 Gigabyte (2 sides)
Tape (DAT)	30 Seconds (cold)	$0.01	5 Gigabytes

Figure 2. High-Capacity Storage Technologies

Applying these concepts, the corporate office in this scenario turned to the use of an optical device for storage of warm data and the IBM 2.3 GB tape drive for cold data. A combination of ARCserve's unattended job and Auto Pilot features was used to manage this data effectively. Network administrators were able to remove more than half of the data on the server, after which the network ran smoothly. Should more tape storage be required, up to seven tape drives can be daisy-chained to one SCSI adapter, with ARCserve's Auto Pilot function managing the data.

Scenario 2: Quick Recovery after a Disaster

The corporate office in Scenario 1 continues to enjoy months of successful LAN operations using IBM hardware, Novell NetWare 3.11, and Cheyenne Software's ARCserve product. Then, at 6:00 A.M. on Friday the 13th, a fire rages through the lab and destroys the file server. To make matters worse, the company's president has scheduled a press conference at 1:00 P.M. to disclose the financial status of the firm to its investors. Although his notes are on his workstation, the accompanying financial data resides on the file server.

IBM's customer support is able to obtain a replacement PS/2 Model 95 and IBM 2.3 GB tape unit, and IBM personnel reconfigure the PS/2 as a NetWare 3.11 file server by 11:00 A.M. Now, the only remaining issue is the data – but how does one restore almost one gigabyte of data in less than two hours?

IBM's network support people enter the company's tape vault, and pick up the tape containing last night's ARCserve backup. They insert the tape into the IBM 2.3 GB drive, which then pulls up ARCserve's Quick Recover menu. Quick Recover is designed to help the user recover from just such a disaster. This feature is designed for complete system crashes, or for virus infection, when the user wants to return the target to a previous date. ARCserve prompts the user for the date to restore to, then informs the user which tapes (by tape name) to restore from.

The IBM support person gives ARCserve yesterday's date, and a job is submitted to the queue, restoring the server to yesterday's image. In just over an hour – averaging 15 MB per minute – ARCserve restores nearly 1 GB of data.

At 12:15 P.M., the LAN is once again operational. The president can access the financial data and successfully make the presentation to investors.

Scenario 3: Integration of Mainframe Data

Corporations use mainframes for their customized software applications, centralized storage, and high-speed computing needs. The proliferation of Novell networks within these companies has highlighted the need to transfer data easily between the Novell LANs and the host mainframes, and to access that data easily. Products from IBM and Phaser Systems enable VM or VMS mainframe DASDs to appear as a NetWare volume. Legent®, incorporating Spectrum Concepts, also offers a LAN/host connectivity solution with its XCOM®/SDS product. Using these products with ARCserve provides complete LAN/host data management services.

For example, IBM's LANRES product is available on both VM and VMS. LANRES has two components: mainframe-resident software and a NetWare Loadable Module (NLM) running on the NetWare server. The two platforms are connected by a PS/2 Channel Adapter (PCA) card that resides inside the PS/2 NetWare server and by a direct channel connection to the mainframe. A disk-serving function, one of the four major components in LANRES, enables mainframe DASDs to appear as a PC disk, thereby allowing access by NetWare clients. NetWare clients can then benefit from the ease of maintenance, reliability, and capacities of mainframe storage.

Consider the mixed workstation environment outlined in the first two scenarios, in which the LAN supports OS/2, DOS, Windows, and Macintosh clients. If the corporation has a mainframe tape-backup system, its LAN administrators can pull data from the NetWare LAN to the backup system on the mainframe. This "pull" is done using LANRES for the required disk-sharing and ARCserve to move the data between the mapped logical volumes. Either attended or unattended jobs can be scheduled using ARCserve's scripts. Workstations are accessed securely using the ARCserve resident agents.

Similarly, LAN administrators can "push" data from the shared DASD to the NetWare server. All systems in an enterprise can receive regular updates from a mainframe, and old releases can be automatically destroyed. This concept can be applied to application programs that require periodic updating or to corporate data that is shared and modified on both the mainframe and LAN. Updates can occur on the mainframe, LAN server, remote servers, or network workstation.

A corporation that wants to integrate its entire corporate data, to standardize its software applications on an enterprise-wide basis, and to manage all this with one software package, can accomplish these things easily using the configurations and software products discussed in this article.

David G. Weingard *is product manager at Cheyenne Software, a developer of sophisticated LAN and business management software products. His background includes a blend of sales and marketing expertise, coupled with hands-on software design and systems engineering experience. David received an MBA from Adelphi University and a BA in computer science from the City University of New York. Cheyenne Software, an IBM Business Partner, is at 55 Bryant Avenue, Roslyn, NY 11576. Phone: (516) 484-5110.*

The DOS Protected-Mode Environment

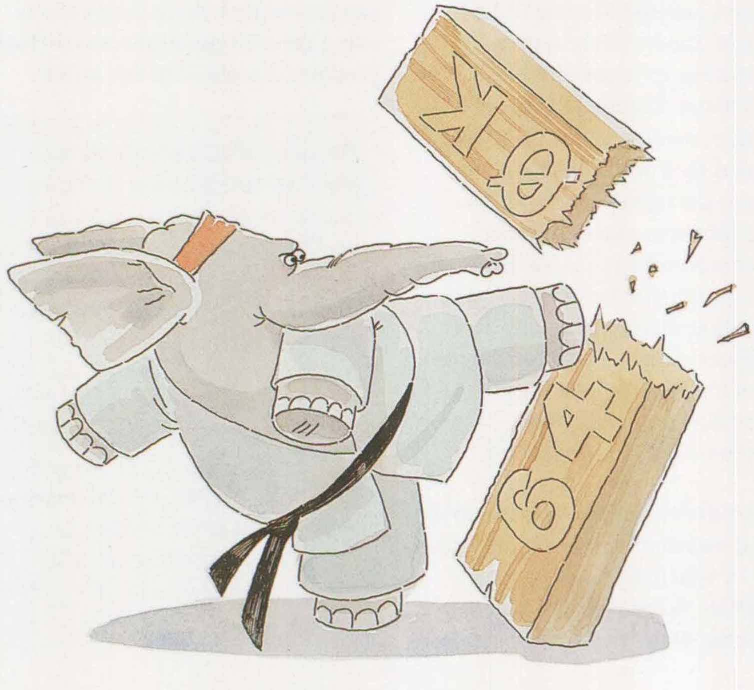
Pylee Lennil
IBM Corporation
Boca Raton, Florida

This article gives an in-depth look into the DOS protected-mode environment. It also discusses two protected-mode interfaces – Virtual Control Program Interface (VCPI) and DOS Protected-Mode Interface (DPMI) – and how DOS extenders run DOS applications in the protected-mode environment.

DOS is a real-mode operating system; that is, it runs in the real mode of Intel 80X86 processors. The real mode has two restrictions: DOS and applications are limited to the first 1 MB of memory space, and multitasking – running multiple DOS applications simultaneously – is not possible. These limitations can be overcome by running the DOS applications in the protected-mode environment.

The DOS protected-mode environment is a DOS environment created by control programs that run in the protected mode of the processor. A control program can be any of the following:

- An expanded memory emulator, such as EMM386, that emulates expanded memory blocks in extended memory (memory above 1 MB) by bank-switching expanded memory pages above 640 KB
- A DOS extender, such as Phar Lap's® RUN386, that runs DOS applications in extended memory and provides large memory space
- A multitasking program, such as Microsoft Windows 386, that runs multiple DOS programs concurrently in extended memory



Operating Modes of 80X86 Processors

Before examining the way control programs work, let us look at the three operating modes provided by the Intel 80X86 family of processors.

Real Mode

Real mode is supported by all processors in the Intel 80X86 family. DOS and DOS applications normally run in this mode. Real mode has three restrictions:

- It does not allow multiple programs to run simultaneously.
- It offers almost no memory protection.
- Its address space is restricted to the first 1 MB of memory.

Protected Mode

Protected mode is supported by Intel 80286 and later processors. In this mode, the operating system and the applications run in the protected mode above 1 MB. This mode is used by operating systems such as OS/2 and AIX, and by protected-mode control programs such as expanded memory emulators and DOS extenders. Protected mode provides a separate address space for each application and allows multiple applications to run simultaneously. It also supports virtual memory, which provides large address space for applications.

Virtual 86 Mode

Virtual 86 mode is supported by Intel 80386 and later processors. In this mode, a control program running in protected mode can create one or more 1 MB segments called *virtual machines*, and can execute a DOS application in each virtual machine. The program that executes in this address space behaves exactly as it does in real mode. Each virtual machine has its own address space, I/O port space, and interrupt vector table. Because the virtual machines operate in protected mode, these programs can execute simultaneously without interfering with the control program or with programs running in other virtual machines. Control programs that use virtual 86 mode include Microsoft's expanded memory emulator EMM386, Phar Lap's DOS extender RUN386, and Windows 386.

DOS Extenders

A DOS extender is a control program that provides a DOS protected-mode environment on top of the DOS real-mode environment. It allows a DOS application to run in the protected-mode environment. A DOS extender provides two major benefits:

- It provides large memory space to applications.
- Unlike expanded or extended memory managers, a DOS extender permits an application to access the entire physical memory without any special interface.

An 80386 DOS extender uses virtual 86 mode to create a DOS protected-mode environment in a virtual machine. It runs an application in the virtual machine, and allows the application to access real-mode DOS and ROM BIOS functions under its control. It also gives the application a large memory space, up to 16 MB.

How does an application access services from DOS and BIOS, which are running in real mode? When the application issues a DOS or BIOS service call, such as read a file or write a sector, the DOS extender intercepts the request and switches the processor to real mode. If the function call requires a data transfer, it copies the data from the protected-mode buffer to the real-mode buffer. It then calls DOS or BIOS to execute the function. When the function is returned, the extender switches the processor back to protected mode. If necessary, it copies the data to the protected-mode buffer and transfers the result to the application.

As shown in Figure 1, a DOS extender has two sections: an initialization section that runs in real mode, and a control section that runs in protected mode. When the DOS extender is loaded, the initialization section executes first. The initialization section performs the following actions:

- Allocates memory buffers below 640 KB that will be used for transferring data to real mode when calling DOS and BIOS functions
- Detects the presence of other memory managers, such as an expanded memory emulator or the DOS Protected-Mode Interface (DPMI)

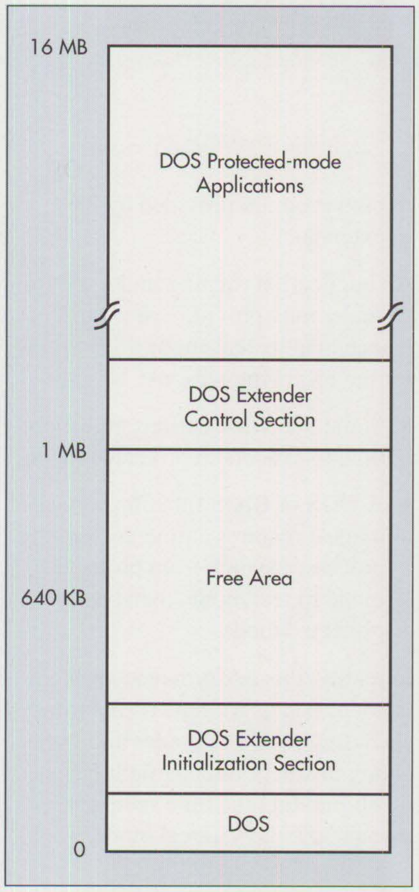

16 MB
DOS Protected-mode Applications
DOS Extender Control Section
1 MB
Free Area
640 KB
DOS Extender Initialization Section
DOS
0

Figure 1. DOS Extender Memory Map

- Builds global and local descriptor tables to support addressing modes
- Loads an application and makes necessary relocations and fixes
- Switches from real mode to protected mode for handling interrupts

The control section of the DOS extender consists primarily of interrupt handlers for the following types of interrupts:

- Interrupts generated by processor faults and hardware error
- Interrupts generated by external devices for device services
- Software interrupts, issued by DOS applications running in the DOS protected-mode environment to access DOS and BIOS services

The control section must provide functions for handling system service requests that are issued by a DOS application running in the virtual machine. These requests can be:

- Service calls for file operations – Open, Read, Write, and so on
- Service calls that require little or no processing – character I/O functions for console, printer, or serial ports, and so on
- Function calls supported only by the DOS extender

A DOS application that runs in protected mode under the control of an 80386 DOS extender must obey certain restrictions. The following restrictions apply when writing to a protected-mode application or when converting an application from real mode to protected mode:

- Unsupported and undocumented software interrupts may not work directly.
- Do not access memory that is not owned by the application.
- Do not move data into code segments.

Initialization	
00H	Detect presence of VCPI
01H	Get protected-mode interface
02H	Get maximum physical memory address
Memory Allocation	
03H	Get number of free 4 KB pages
04H	Allocate a 4 KB page
05H	Free a 4 KB page
06H	Get physical address of a 4 KB page
System Register Access	
07H	Get value of CR0 register
08H	Read debug registers
09H	Load debug registers
Interrupt Controller Management	
0AH	Get 8259 interrupt vector mappings
0BH	Set 8259 interrupt vector mappings
Mode Switching	
0CH	Switch from virtual 86 mode to protected mode
0DH	Switch from protected mode to virtual 86 mode

Figure 2. VCPI Functions

- Do not use segment registers as temporary storage registers.
- Do not perform address arithmetic on pointers.
- All 16-bit instructions must be converted to 32-bit instructions.
- DOS and ROM BIOS calls that refer to memory management must be converted into DOS extender memory management calls.

The following are general guidelines for building a DOS protected-mode application that will run under a DOS extender:

1. Compile the application using a compiler.
2. Link the application using the linker provided by the extender.
3. Bind the extender loader and kernel to the application by using the bind program provided by the extender.

An application running under a DOS extender may provide less performance than its real-mode counterpart for the following reasons:

- Some 8086 instructions take more time to execute in protected mode.
- A DOS or BIOS function service request requires a time-consuming processor switch from protected mode to real mode, then back to protected mode.

In reality, this performance problem is not as bad as it sounds – the time spent for switching modes and executing instructions is partially negated by eliminating the time spent to manage overlays in real mode.

A DOS extender is available in two forms: a stand-alone version that can be used to load and execute a DOS application in protected mode, and a bundled version in which the extender and the application are bound as a single executable file (as in the case of Paradox® 386 and Interleaf® Publisher). In the latter, when the application is executed, the DOS extender is loaded first, then the DOS extender loads and executes the application.

Protected-Mode Interfaces

Running multiple DOS protected-mode control programs can cause resource contention. The contention occurs when control programs access two resources: the extended memory area (above 1 MB) and protected-mode features such as mode switching, interrupt handling, and memory allocation in 80386 and later processors. For instance, an expanded memory emulator and a DOS extender may try to allocate the same extended memory area. In addition, when they access protected-mode features, protected-mode conflicts occur. Control programs can coexist only if they adhere to a predefined interface that accesses resources in a well-behaved, hardware-independent fashion.

A protected-mode interface is just such an interface. It resolves the memory contention problem and protected-mode conflicts. There are two protected-mode interfaces: Virtual Control Program Interface (VCPI), and DOS Protected-Mode Interface (DPMI).

Virtual Control Program Interface

VCPI was the first interface defined to allow control programs such as EMS emulators and DOS extenders to coexist in 80386 and later processors. VCPI functions are implemented in control programs called VCPI servers. These servers provide VCPI functions to client control programs. Clients access system resources by calling the VCPI functions provided by the servers. The list of VCPI functions is given in Figure 2.

Even though VCPI resolves the two major resource contentions, it was inadequate for supporting multitasking control programs. This is because a control program that supports VCPI cannot do the following:

- Provide virtual hardware devices
- Provide full virtual memory management support
- Prevent applications running in a multitasking environment from interfering with each other's memory space

These problems occur primarily because VCPI functions are designed to run in a lower privilege level (ring 3) of the 80386 processor, whereas VCPI clients run in a higher privilege level (ring 0). Therefore, a VCPI server that executes VCPI functions in a lower privilege level loses control to the hardware.

Another deficiency in VCPI is that its functions are based on the hardware paging done by 80386 processors, so they cannot be implemented in 80286-based computers.

DOS Protected-Mode Interface

To rectify the limitations of VCPI, DPMI was defined. Unlike VCPI, DPMI functions are designed to run in a higher privilege level, while DPMI clients run in a lower privilege level. In addition, DPMI functions can be implemented in 80286-based computers.

Figure 3 lists the DPMI functions. A detailed description of DPMI can be found in the *DOS Protected-Mode Interface (DPMI) Specification, Version 1.0*, published by Intel Corporation (Intel order number 240977-001).

Protected-Mode Interface Hosts and Clients

Protected-mode control programs operate in an environment that consists of host and clients. For example, an expanded memory emulator, such

DOS Memory Management Services provide a protected-mode interface to real-mode DOS memory management functions (INT 21H functions 48, 49, and 4A).	
0100H	Allocate DOS memory block
0101H	Free DOS memory block
0102H	Resize DOS memory block
LDT Management Services allocate, free, and modify the Local Descriptor Table (LDT).	
0000H	Allocate LDT descriptor
0001H	Free LDT descriptor
0002H	Map real-mode segment to descriptor
0003H	Get selector increment value
0006H	Get segment base address
0007H	Set segment base address
0008H	Set segment limit
0009H	Set descriptor access rights
000AH	Create alias descriptor
000BH	Get descriptor
000CH	Set descriptor
000DH	Allocate specific LDT descriptor
000EH	Get multiple descriptor
000FH	Set multiple descriptor
Extended Memory Management Services allocate, resize, and release blocks of physical memory above the 1 MB boundary.	
0501H	Allocate extended memory block
0502H	Free extended memory block
0503H	Resize extended memory block
0504H	Allocate linear memory block
0505H	Resize linear memory block
0506H	Get extended memory page attributes
0507H	Set extended memory page attributes
050AH	Get memory block size and base
Page Management Functions lock and unlock memory pages.	
0600H	Lock linear region
0601H	Unlock linear region
0602H	Mark real-mode region as pageable
0603H	Relock real-mode region
0702H	Mark page as demand paging candidate
0703H	Discard page contents

Figure 3. DPMI Functions (continued)

Interrupt Management Services set the interrupt mechanism to handle the hardware and software interrupts generated by the DOS protected-mode application.	
0200H	Get real-mode interrupt vector
0201H	Set real-mode interrupt vector
0202H	Get processor exception handler vector
0203H	Set processor exception handler vector
0204H	Get protected-mode interrupt vector
0205H	Set protected-mode interrupt vector
0210H	Get extended processor exception handler vector for protected mode
0211H	Get extended processor exception handler vector for real mode
0212H	Set extended processor exception handler vector for protected mode
0213H	Set extended processor exception handler vector for real mode
0900H	Get and disable virtual interrupt state
0901H	Get and enable virtual interrupt state
0902H	Get virtual interrupt state
Translation Services provide a mechanism to make procedure calls by the control program to transfer control from protected mode to real mode.	
0300H	Simulate real-mode interrupt
0301H	Call real-mode procedure with far return frame
0302H	Call real-mode procedure with interrupt return frame
0303H	Allocate real-mode callback address
0304H	Free real-mode callback address
0305H	Get state save/restore address
0306H	Get raw processor mode switch addresses
Miscellaneous Services provide information about host services.	
0400H	Get DPMI version
0401H	Get DPMI capabilities
0A00H	Get vendor-specific Application Programming Interface (API) entry point

Figure 3. DPMI Functions

as EMM386, is a VCPI host, while a DOS extender is a VCPI client. Interface functions are implemented in the host, which serves the interface functions to clients. Normally, a DOS extender uses the BIOS INT 15H function to allocate extended memory. However, if a VCPI server is already running, the DOS extender requests the memory from the VCPI server by using VCPI calls. Similarly, a DPMI client can invoke DPMI functions to request protected-mode resources from a DPMI server.

Using Protected-Mode Interfaces

The protected-mode interfaces VCPI and DPMI are not normally used by DOS applications running in the protected-mode environment. Instead, they are used by client control programs such as DOS extenders. The client program checks to see if a VCPI or DPMI server is active. If the server is present, the client invokes the interface functions to access protected-mode resources. If the server is not present, the client program can access protected-mode resources directly.

VCPI functions are invoked using INT 67H interrupt calls. DPMI functions are invoked using INT 2FH and INT 31H interrupt calls. INT 2FH is used to detect the presence of a DPMI server, and INT 31H is used to invoke DPMI functions.

Summary

The DOS protected-mode environment is provided by protected-mode control programs such as DOS extenders, expanded memory emulators, and multitasking programs. The DOS protected-mode environment provides large memory space to DOS applications and enables multitasking. Control programs must use DPMI so that they can coexist without resource contention. DOS extenders are beneficial to applications that require large memory space. Unlike expanded and extended memory managers, DOS extenders require no special interface for an application to access large extended memory space.

__Pylee Lennil__ is a staff programmer in the IBM Entry Systems Technology Laboratory in Boca Raton, Florida. He is presently involved in the development of AIX. Previously, he spent several years in PC DOS development. Pylee joined IBM in 1983. He received a BS in physics from Kerala University in India, and an MS in computer engineering from the University of Massachusetts at Lowell.

DOS Disk Management

Pylee Lennil
IBM Corporation
Boca Raton, Florida

This article describes the data structures and commands that DOS uses to manage data stored on disks.

DOS maintains a file system for storing and accessing data files on disks and diskettes. The DOS file system consists of a root directory, subdirectories, and the files themselves. DOS performs the following functions to manage its file system on a hard disk:

- Divides a hard disk into partitions using the FDISK command
- Formats the disk partitions using the FORMAT command
- Checks disk data inconsistencies using the CHKDSK command
- Creates, updates, or deletes files and directories

Before looking at these DOS functions in detail, let us review the structure of a hard disk.

Structure of a Hard Disk

A hard disk contains one or more revolving platters that are stacked vertically, as shown in Figure 1. Each platter has both a top and a bottom surface. Information is stored on each surface and is accessed by a read/write head.

Each surface is divided into thin magnetic *tracks* that are laid out as concentric circles. Tracks on each surface are numbered starting with track zero. The total number of tracks depends on the size of the platter.

Each surface of every platter looks identical; that is, each surface has the same number of tracks, and a track with a particular number (for example, track 9) occupies the same position on every surface. Because the platters are stacked vertically, all the tracks that have identical numbers are stacked as well. These identical tracks on all surfaces of all platters form a vertical *cylinder*. Reading an entire cylinder means reading all of its identical tracks. Each cylinder has the same number as its identical tracks; for example, cylinder 9 consists of all tracks 9.

Every track is further divided into sectors. A *sector* is the most basic addressable unit of storage on a hard disk. A sector normally holds 512 bytes of data. Sectors are numbered, and identical sectors are stacked vertically.

A specific sector on the disk is identified by three numbers: the number of the cylinder that contains the sector, a head number that identifies a specific platter, and the sector number within that track.

The FDISK command in DOS divides a hard disk into areas called partitions, as shown in Figure 2. A

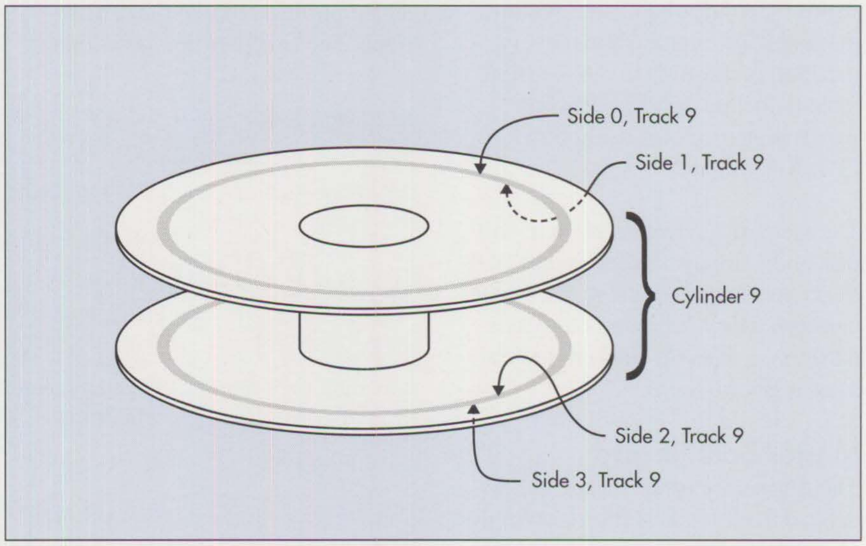


Figure 1. Hard Disk Tracks and Cylinders

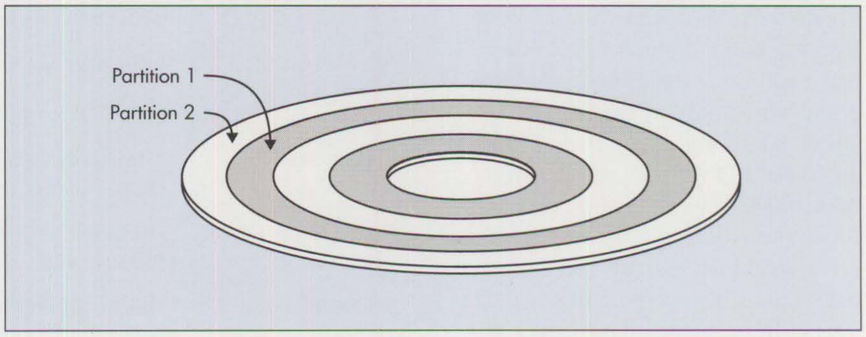


Figure 2. Hard Disk Partitions

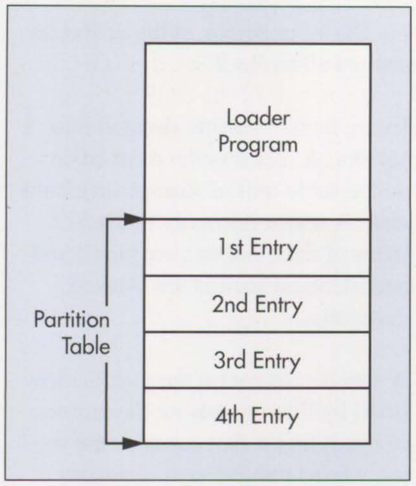


Figure 3. Layout of Master Boot Record and Partition Table

partition is a group of sectors in one or more cylinders. The starting and ending sectors are each identified by specific cylinder, head, and sector numbers. Information about every partition is recorded in the partition table in the Master Boot Record, which is also created using the FDISK command.

The main difference between a hard disk and a floppy diskette is that DOS treats the entire diskette space as one partition. Because there is only one partition, a diskette does not have a Master Boot Record.

Master Boot Record

The first sector of the disk (cylinder 0, head 0, sector 1) is reserved for the Master Boot Record. The first cylinder and first head are both numbered 0, but the first sector is called sector 1. Figure 3 shows that the Master Boot Record contains a loader program and a partition table. The loader program is used only by the ROM BIOS. At the end of the Power-On Self Test, ROM BIOS searches for a bootable diskette in the first diskette drive. If no diskette is in the first drive, ROM BIOS loads and executes the loader program that is in the Master Boot Record. This loader program uses information in the Master Boot Record's partition table to search for a bootable DOS disk partition.

The partition table contains four 16-byte structures, called *partition entries*, that describe the partitions on the disk. Figure 4 defines the fields within a partition entry.

How does DOS divide the disk into partitions? As shown in Figure 5, there are four entries in the partition table. DOS uses the first two entries. The other two entries can be used by other operating systems, such as OS/2 or UNIX, to create their own partitions on the same disk.

Although DOS can leave space on the disk for the partitions of other operating systems, DOS normally allocates the entire disk as either a single partition or two major partitions. The first partition is called the DOS *primary partition*, and the second one is the DOS *extended partition*. The extended partition can be subdivided into subpartitions called *logical drives*. The logical drives form a chained list by connecting each logical drive to the next one. DOS assigns the drive letter C: to the primary partition, and D:, E:, F:, and so on, to the logical drives.

DOS Disk Partition Structure

Each DOS disk partition is divided into the five areas shown in Figure 6. The FORMAT command creates and initializes these areas when the partition is formatted. The first area is the Boot Sector, which contains a loader program and a BIOS Parameter Block (BPB). The next two areas are the two copies of the File Allocation Table (FAT). The FAT describes the locations of file blocks on the disk. DOS maintains two copies of the

Field	Description
Partition Status	00H Non-bootable partition 80H Bootable partition If the partition is bootable, then the first two files in the partition must be the DOS system files (IBMBIO.COM and IBMDOS.COM).
Starting Sector	The starting sector of the partition, indicated by a starting cylinder, head, and sector number
Partition Type	00 = Not used 01 = DOS primary partition with 12-bit FAT 02 = XENIX® file system partition 03 = XENIX file system partition 04 = DOS primary partition with 16-bit FAT 05 = DOS extended partition 06 = DOS partition with more than 32 MB
Ending Sector	The ending sector of the partition, indicated by an ending cylinder, head, and sector number
Boot Sector Offset	Offset of the first sector (boot sector) from partition sector (measured in sectors)
Size of Partition	Number of sectors in the partition

Figure 4. Fields within a Partition Entry

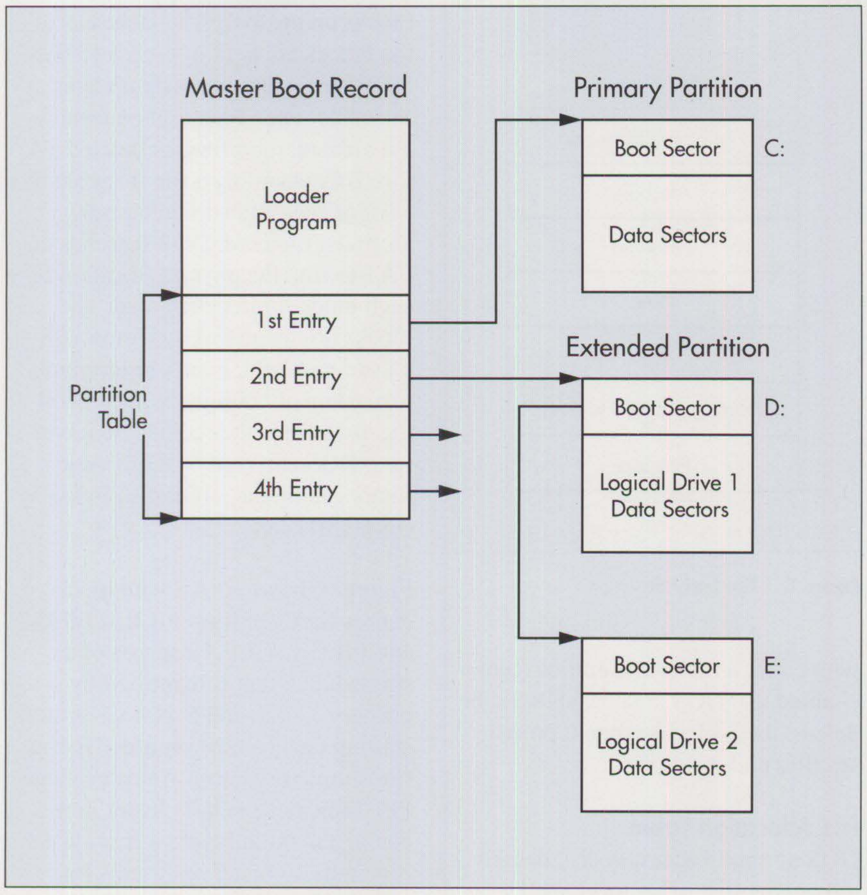


Figure 5. Partition Table and Partitions

FAT in case one gets damaged. The next area is the root directory. It is a table that contains a list of file entry structures describing files and directories listed under the file system root directory. The remaining area in the partition is reserved for file entries that are associated with files and subdirectories that are not in the root directory. This last area also contains the file data blocks – the actual data.

Boot Sector

The boot sector is created in the first sector of the disk partition when the partition is formatted using the FORMAT command. The boot sector consists of four fields, as shown in Figure 7. The first field contains a 1-byte jump instruction (0E9H or 0EBH) followed by an 8-bit or 16-bit displacement. The jump instruction is used to transfer control to the loader program located in the fourth field. The second field is an 8-byte vendor identification string that identifies the PC manufacturer and the DOS version number. The third field contains a data structure, the BPB, that describes the physical characteristics of the disk. The fourth field contains the loader program.

What is the function of the Boot Sector's loader program? As described earlier, at the end of Power On Self Test (POST), ROM BIOS loads and executes the loader program that is in the Master Boot Record. The master loader uses the partition address in the Master Boot Record's partition table to search for a bootable partition. Once it finds one, it then loads and executes the loader program that is in the Boot Sector of the bootable partition. The Boot Sector loader then loads one part of the DOS kernel (IBMBIO.COM), which in turn loads and executes the remaining parts of the DOS kernel (IBMDOS.COM and COMMAND.COM).

BIOS Parameter Block

The BPB contains the following disk-specific information:

- Number of bytes per sector
- Number of sectors per cluster
- Number of File Allocation Tables (normally two)
- Number of root directory entries (same as the number of files and directories in the root directory)
- Media descriptor byte, used for identifying the disk media type
- Number of sectors allocated for the FAT
- Number of sectors per track
- Number of heads on the disk
- Number of hidden sectors

DOS uses the information in a BPB to access the corresponding hard disk or diskette. During system configuration, DOS reads the BPBs for every

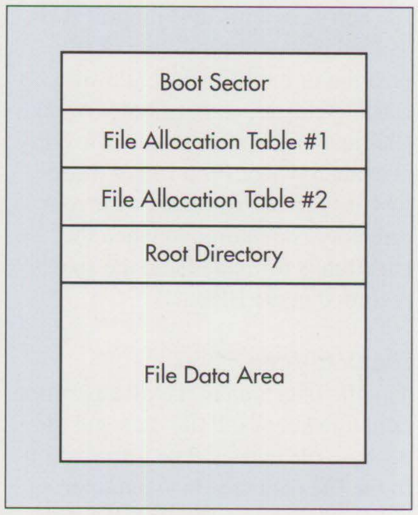


Figure 6. Disk Partition Structure

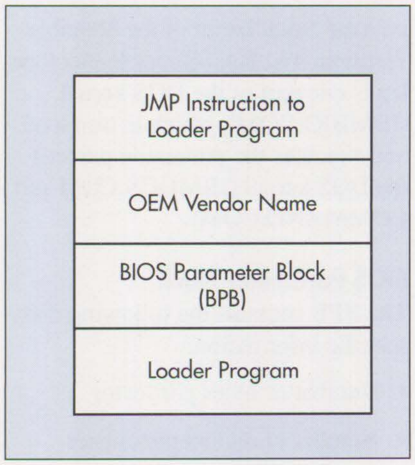


Figure 7. Structure of Boot Sector

disk partition and diskette, and creates a linked list of tables. These tables contain the disk characteristics of all the disk partitions and diskettes configured to the system. DOS searches these tables the first time it accesses a disk partition or a diskette drive.

Root Directory

The root directory area contains a list of file entries. Each file entry is a 32-byte structure that describes a file or directory in the file system's root directory. Figure 8 shows this file entry structure. (File entries for files and directories that are not in the root directory are located in the file data area of the partition.) A root-directory file entry contains information such as file name, date and time of file creation or modification, file size, starting cluster, and attributes of both files and directories. A new file entry is created whenever a file or directory is created. The size of the root directory is determined when the partition is formatted and the size is recorded in the BPB.

File Data Area

The file data area in the disk partition structure keeps the file data and the 32-byte file entries. The information in the file data area is maintained as either allocated or unallocated data blocks. Blocks that are associated

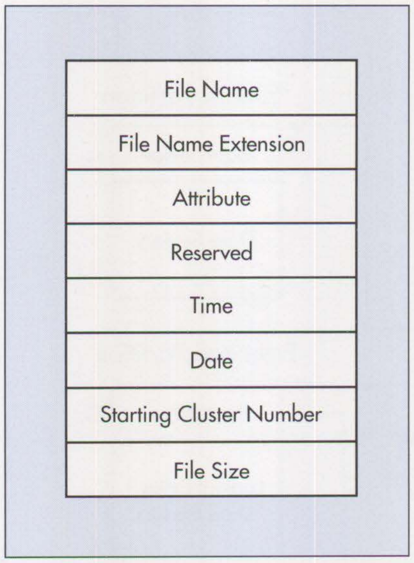


Figure 8. File Entry Structure

with either a file or file entries form a chained list. DOS creates, updates, or deletes these blocks when it processes files and directories.

File Allocation Table

DOS normally accesses disk data in a fixed number of sectors called a cluster. A *cluster* is the smallest unit of disk space that a file can have under DOS. A cluster contains from four to eight adjacent sectors. The number of sectors per cluster is based on the size of the disk partition. Normally, a partition smaller than 16 MB is given a 4 KB cluster (eight sectors per cluster), while a partition of 16 MB and above is given a 2 KB cluster (four sectors per cluster).

Under DOS, files are saved on a disk as one or more blocks scattered throughout the disk, as shown in Figure 9. A *block* is a collection of contiguous sectors. Each block has unique starting and ending cluster numbers. The starting cluster number of the first block of a file is recorded in the starting cluster field of the file entry.

To access a file, DOS must know the exact physical locations of the file blocks on the disk. The locations of the blocks are kept in the FAT. The DOS kernel continuously updates this table when files or directories are created, modified, or deleted. (A directory is simply a file that contains a list of files and subdirectories.) During a file read, DOS searches the FAT to find the physical location of each block it intends to read. The blocks are identified by a group of cluster numbers. Each cluster points to a group of contiguous sectors on the disk. Once the sectors are identified, DOS can read or write these sectors using the BIOS disk function (INT 13H) calls.

Figure 9 shows a FAT with block entries for three files: FILE.1, FILE.2, and FILE.3. FILE.1 consists of a single block that is identified by clusters 3, 4, 5, and 6. Note that the starting cluster, 3, is recorded in the file's directory entry. An entry value FFF indicates the last cluster in the chain, and 00 indicates a free cluster. Normally a cluster represents four to eight sectors, but for simplicity, assume a single sector per cluster. Therefore, FILE.1 occupies sectors 3, 4, 5, and 6. FILE.2 consists of two blocks, occupying sectors 7, 8, 9, 10, 14, 15, 16, and 17. FILE.3 consists of a single block that occupies sectors 11, 12, and 13. Note that these sectors appear between the FILE.2 sectors.

DOS accesses these files as follows. Accessing a file requires two pieces of information: the file entry and the location of the file blocks. Assume DOS wants to read FILE.1. DOS searches for the FILE.1 entry in the root directory and the file data area. If the file entry is found, DOS reads it to find the starting cluster of the file. Using the starting cluster, DOS locates other file clusters by tracing the FILE.1 cluster chain in the FAT. It then converts the clusters into sectors, and calls the DOS disk driver to read the sectors into the system buffer.

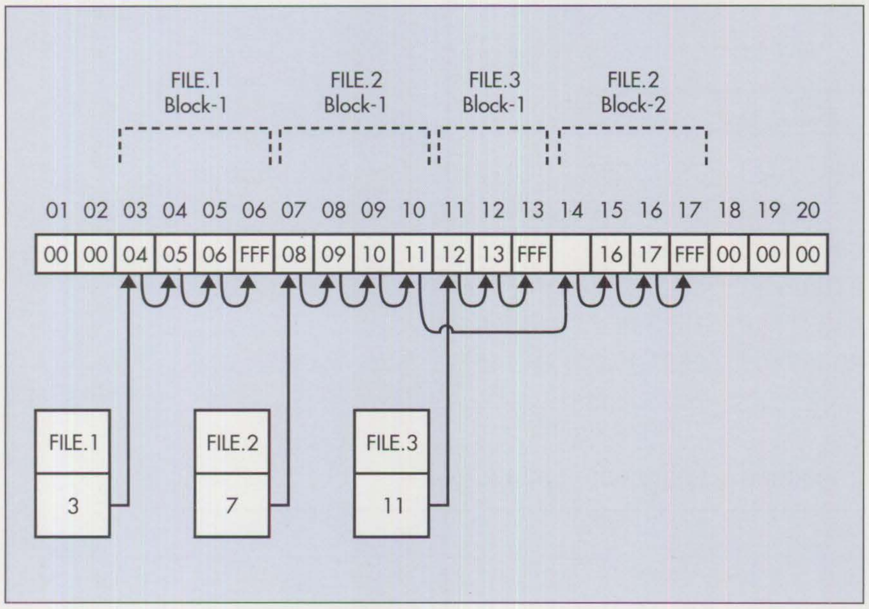
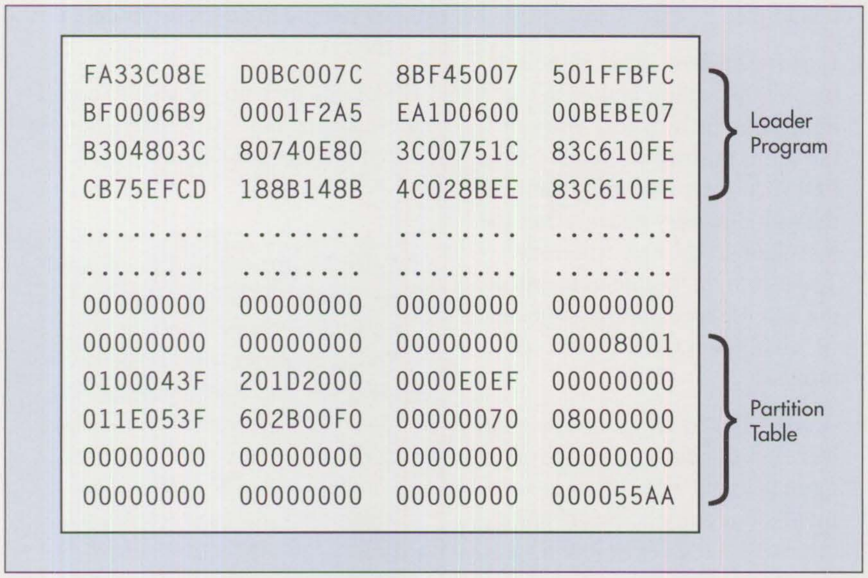


Figure 9. Structure of File Allocation Table



```
FA33C08E  D0BC007C  8BF45007  501FFBFC  }
BF0006B9  0001F2A5  EA1D0600  00BEBE07  } Loader
B304803C  80740E80  3C00751C  83C610FE  } Program
CB75EFCD  188B148B  4C028BEE  83C610FE  }
........  ........  ........  ........
........  ........  ........  ........
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00008001  }
0100043F  201D2000  0000E0EF  00000000  }
011E053F  602B00F0  00000070  08000000  } Partition
00000000  00000000  00000000  00000000  } Table
00000000  00000000  00000000  000055AA  }
```

Figure 10. Dump of Master Boot Record

Suppose the second file, FILE.2, needs to be lengthened. In this case, DOS searches the FAT for the required number of free clusters. If found, DOS makes the current last cluster point to the first of the newly allocated clusters, and marks the last newly allocated cluster as FFF. If a file is to be truncated, the clusters to be deleted are marked as free (00) and the last cluster of the allocated clusters is marked as FFF.

Disk Management Commands

Now that we have discussed the DOS disk structure and how the DOS kernel manages the file system, let us look at the DOS disk commands FDISK, FORMAT, and CHKDSK.

FDISK Command

The FDISK command configures the hard disk by dividing it into partitions. The major functions of FDISK are to create, delete, and update disk partitions, and to display disk partition information.

The DOS disk partition information is described in the Master Boot Record's partition table. FDISK creates a Master Boot Record only if it does not already exist on the disk. A partial disk dump of the Master Boot Record is shown in Figure 10. When a user creates a DOS primary or extended partition, FDISK adds new partition information to the first or second entries in the partition table. Changing or deleting partitions involves updating the appropriate entries.

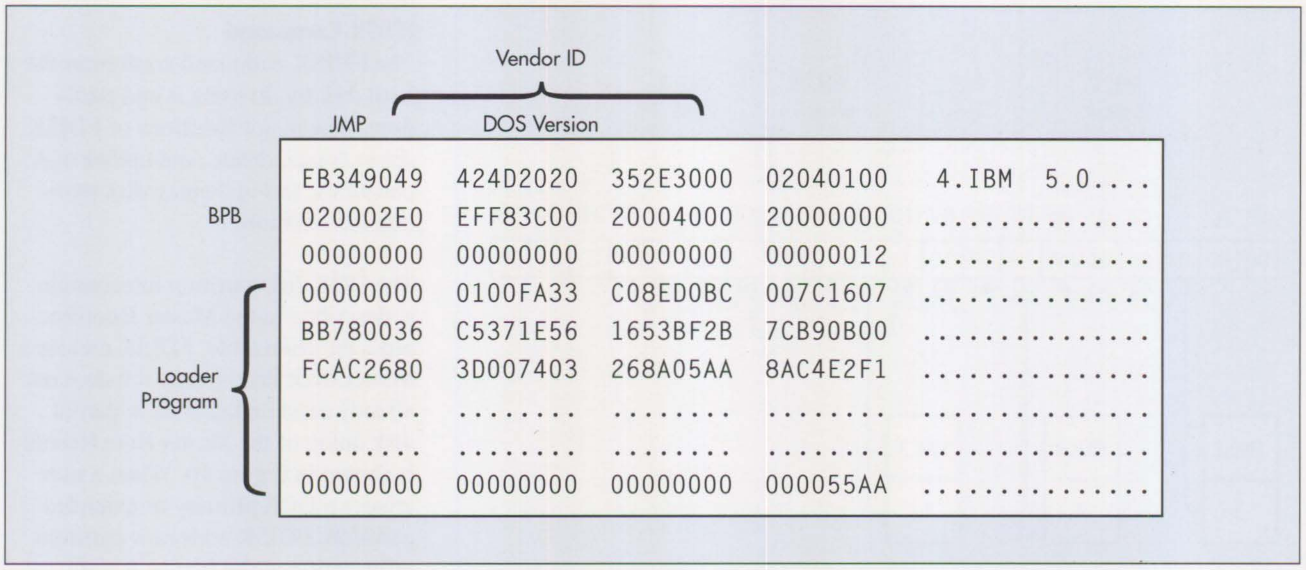
FORMAT Command

The major function of the FORMAT command is to initialize partition structures: the Boot Sector, Root Directory, and FAT. The partition must be formatted before it can be used. Formatting the partition with FORMAT erases the existing data. Formatting a partition involves the following steps:

1. Check for defective sectors in the partition. If found, mark them as unusable.
2. Create a Boot Sector, as shown in Figure 11.
3. Create an initial root directory.
4. Create an initial FAT.

CHKDSK Command

The CHKDSK command analyzes and can correct inconsistencies in the DOS file system. How does a file system get corrupted? The file system consists of files and directories. Information about files and directories, including their physical layouts, is in the disk directory and the FAT. If the computer is inadvertently turned off (usually because the elec-



```
EB349049  424D2020  352E3000  02040100   4.IBM  5.0....
020002E0  EFF83C00  20004000  20000000  ...............
00000000  00000000  00000000  00000012  ...............
00000000  0100FA33  C08ED0BC  007C1607  ...............
BB780036  C5371E56  1653BF2B  7CB90B00  ...............
FCAC2680  3D007403  268A05AA  8AC4E2F1  ...............
........  ........  ........  ........  ...............
........  ........  ........  ........  ...............
00000000  00000000  00000000  000055AA  ...............
```

Figure 11. Dump of Boot Sector

tricity fails) while a file is being accessed, DOS does not copy the updated file information into the disk directory and the FAT. This results in one or more of the following file system errors:

- Damaged directory entries that make it impossible to trace the directory path
- Invalid file attributes in the directory entries
- Lost file cluster chains
- Cross-linked clusters between file cluster chains
- Invalid file size

CHKDSK may be able to repair some of these file system errors, or at least notify the user of the inconsistencies. For example:

- Lost file clusters occur if the file block information in the FAT is intact, but the directory entry of the file is destroyed. In this case, FDISK creates a new file name in the root directory and attaches the lost chain to the new file entry. The user may be able to identify the lost file by examining the new file FILEnnnn.CHK in the root directory.
- A cross-linked cluster means that identical file clusters are found in more than one file cluster chain. Invalid directory entries occur if directory entries exist, but no associated file cluster chain can be found in the FAT.
- Invalid file size means the file size described in the directory entry does not match the size computed by tracing the file cluster chain in the FAT.

CHKDSK may not be able to correct these errors, but it will notify the user about the inconsistencies.

Pylee Lennil *is a staff programmer in the IBM Entry Systems Technology Laboratory in Boca Raton, Florida. He is presently involved in the development of AIX. Previously, he spent several years in PC DOS development. Pylee joined IBM in 1983. He received a BS in physics from Kerala University in India, and an MS in computer engineering from the University of Massachusetts at Lowell.*

Customizing Alphanumeric Screen Dimensions

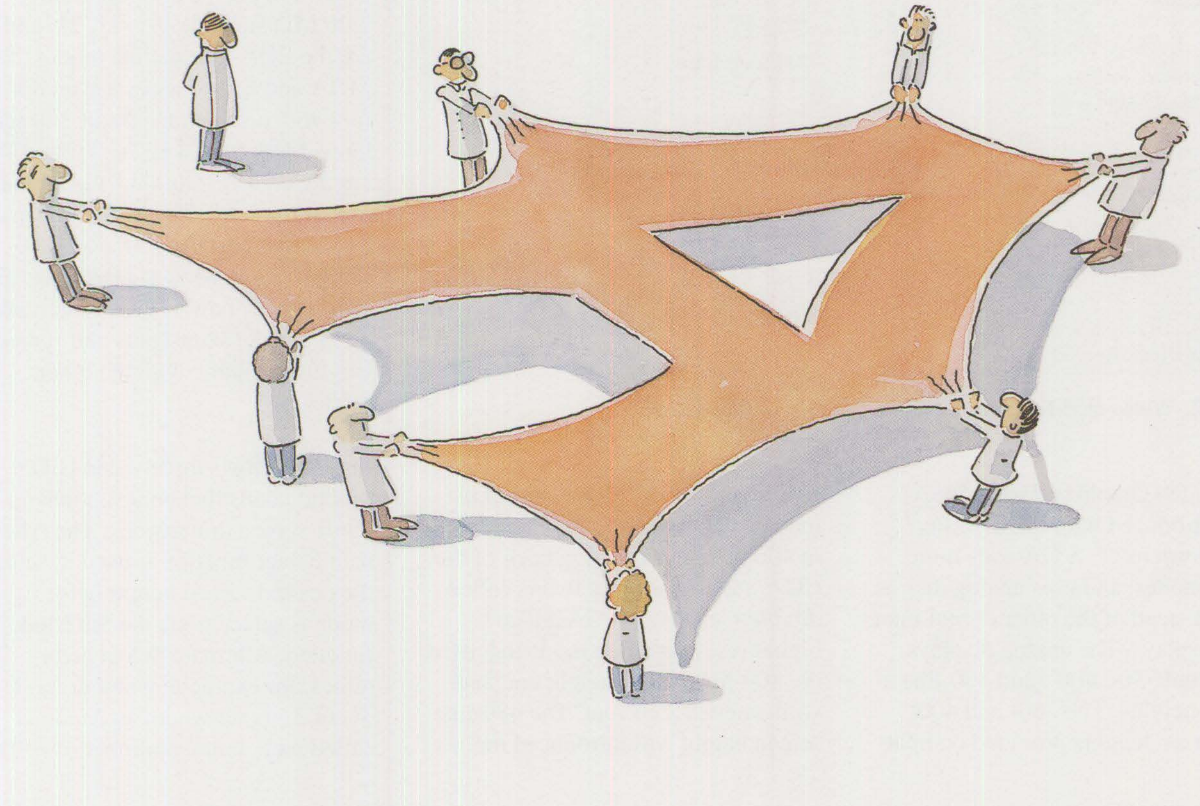
Steve Thompson
IBM Corporation
Boca Raton, Florida

This article shows how video BIOS functions can control the dimensions of alphanumeric screens by varying the screen and font heights used. Coding examples in PC assembler and C illustrate how to incorporate these BIOS functions into applications.

Graphical user interfaces have been adopted for many applications that were originally text-based. These applications now mix text and graphics to attain exciting new capabilities. Many applications, however, are still better suited to the traditional text or alphanumeric interface. Applications such as airline reservation terminals, computer input/output editors, and banking or point-of-sale terminals benefit from the performance, ease of programming, and low data-storage requirements of the traditional alphanumeric mode.

Most video adapters in IBM personal systems support alphanumeric mode. IBM's first PC video adapters, the Monochrome Display Adapter (MDA) and the Color Graphics Adapter (CGA), used a fixed-size character font to support alphanumeric mode dimensions of 25 rows by 80 columns. The MDA's screen addressability, using a 9 x 14 font, was 720 pixels wide by 350 scan lines high; the CGA's addressability was 640 pixels wide by 200 scan lines high, using an 8 x 8 font.

Next came the Enhanced Graphics Adapter (EGA), with added support for programmable and variable-sized fonts. This allowed a variable number of rows to be displayed on the screen. The EGA's pixel addressability was 640 x 350. To support the older monochrome and color displays, the video BIOS provided with the EGA contained both the CGA (8 x 8) and the MDA (8 x 14) fonts. (While the MDA used a true 9 x 14 font, the EGA stored an 8 x 14 font and produced the ninth pixel of each character by using an algorithm built into the hardware.) Either the CGA or MDA font could be loaded into the adapter by using BIOS calls.



Mode Number	Screen Size	Font Size	Screen Height
0, 1	40 x 25	8 x 8	200
2, 3	80 x 25	8 x 8	200
14h	132 x 25	8 x 8	200
0*, 1*	40 x 25	8 x 14	350
2*, 3*, 7	80 x 25	8 x 14	350
14h*	132 x 25	8 x 14	350
0+, 1+	40 x 25	8 x 16	400
2+, 3+, 7+	80 x 25	8 x 16	400
14h+	132 x 25	8 x 16	400

Notes:

1. The 200-line modes are double-scanned to occupy 400 lines on PS/2 displays.

2. Mode 14h is available only on XGA and 16-bit VGA systems.

Figure 1. Standard BIOS Text Modes

Register Setting	Denotes
Alternate Select (AH = 12h)	
BL = 30h	Select scan lines for text modes
AL = 0	200 scan lines
AL = 1	350 scan lines
AL = 2	400 scan lines
Set Mode (AH = 0)	
AL = mode	
Character Generator (AH = 11h)	
AL = 11h	Load 8 x 14 font
AL = 12h	Load 8 x 8 font
AL = 14h	Load 8 x 16 font
BL = Block to load	

Figure 2. Video BIOS Functions for Controlling Screen Dimensions

The Video Graphics Array (VGA) was introduced with the IBM Personal System/2®. VGA adds more video modes, and uses analog displays instead of the older digital interface displays. The analog displays supported 350-, 400-, and 480-line screen heights. The 350- and 400-line screen heights provided compatibility with CGA, MDA, and EGA modes. (CGA modes were displayed in 400 lines by repeating each of the CGA's 200 scan lines twice, called *double scanning*.) VGA added a higher resolution text mode that used the 400-line screen height coupled with a new 8 x 16 font. The 480-line screen height was introduced mainly for graphics modes. A 480-line text mode is possible, but not available directly through BIOS.

IBM's newest video subsystems are the 16-bit VGA and the Extended Graphics Adapter (XGA). These subsystems allow text modes with 132 columns per row. The XGA was introduced with the PS/2 Models 90 and 95, but BIOS support was not available for the 132-column text modes until the introduction of PS/2 Models 35, 40, 56, and 57. The procedure for setting a 132-column text mode on an XGA without 132-column BIOS support can be found in *Personal System/2 Hardware Interface Technical Reference – Common Interfaces* (S84F-9809).

Figure 1 shows the standard alphanumeric modes that are set directly by the video BIOS Set Mode function. Modes 0, 1, 2, and 3 provide compatibility with CGA through an 8 x 8 font and a 200-line screen size double-scanned to 400 lines. Mode 7 is the monochrome alphanumeric mode that provides compatibility with MDA. Modes 0*, 1*, 2*, and 3* are the EGA-compatible modes. The MDA and EGA modes use an 8 x 14 font with a screen height of 350 scan lines. Modes 0+, 1+, 2+, 3+, and 7+ are the high-resolution VGA alphanumeric modes. These modes use the 8 x 16 font and the 400-scan line screen height. Modes 14h, 14h+, and 14h* are the new 132-column modes. They use the same fonts and screen heights available with the other modes.

Figure 2 shows three video BIOS function calls that control alphanumeric screen dimensions. The Alternate Select function is used to select the desired screen height before a mode is selected via the Set Mode function. Alternate Select determines, for example, whether mode 3, 3*, or 3+ is set when Set Mode is called with a mode number of 3. The

Font Height	Screen Height			
	200	350	400	480
16	12	21	25	30
14	14	25	28	34
8	25	43	50	60

Figure 3. Text Rows Possible with Various Font and Screen Heights

```
; Select 350 scan lines for text modes.

mov     ah, 12h      ; video BIOS Alternate Select function
mov     bl, 30h      ; select scan lines for text
mov     al, 1        ; we want 350 scan lines
int     10h          ; invoke video BIOS

; Now select 132-column text mode.

mov     ah, 0        ; video BIOS Set Mode function
mov     al, 14h      ; mode 14h is the 132-column text mode
int     10h          ; have video BIOS set the mode

; Now load the appropriate font to get 43 lines.

mov     ah, 11h      ; video BIOS Character Generator function
mov     al, 12h      ; select the 8 x 8 font
mov     bl, 0        ; load block 0
int     10h          ; have video BIOS load the font

; The 132 x 43 text mode is now set.
```

Figure 4. Assembler Code to Set a 132-Column by 43-Line Text Mode through the BIOS

Character Generator function allows an application to explicitly load one of the three BIOS fonts (8 x 8, 8 x 14, or 8 x 16), or an application-supplied font, after a mode has been set by Set Mode. There are more sub-functions under Character Generator, but the ones shown are particularly useful since they also adjust certain VGA control registers to ensure that the new font displays properly.

In Figure 2, AH, AL, and BL are register names. AH denotes register A, high byte; AL denotes register A, low byte.

By varying the screen and font heights, it is possible to select from a wide range of character row counts. Figure 3 shows the number of rows that are available for each combination of screen height and BIOS font height. For example, if the 8 x 8 font was loaded using a 350-line screen height, there would be 43 rows of text displayed on the screen. Note that 480-line text screen heights are not available through the Alternate Select BIOS call. Reprogramming

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

/* The offsets below define offsets in the BIOS data area that BIOS */
/* uses to control the screen.                                      */

#define BIOS_VBUFLENLO 0x4c  /* offset of video buffer length low    */
#define BIOS_VBUFLENHI 0x4d  /* offset of video buffer length high   */
#define BIOS_ROWS      0x84  /* offset of row count minus one        */

void set_132x60_text_mode(void)
 {
  union REGS regs;          /* software interrupt register variables /*
  union
   {
    char far *ptr;          /* BIOS data area pointer                */
    struct
     {
      unsigned off;         /* ... pointer offset portion            */
      unsigned seg;         /* ... pointer segment portion           */
     } x;
   } bios_data;
  unsigned temp;            /* temporary storage variable            */
```

Figure 5. C Source Code for a 480-Line Alphanumeric Mode (continued)

```
   /* Initialize BIOS data pointer to 40:0                           */

   bios_data.x.off = 0;     /* offset is zero                        */
   bios_data.x.seg = 0x40;  /* segment is 0x40                       */

   /* Step 1: Select the desired number of columns.                  */

   regs.h.ah = 0;           /* select BIOS mode set function         */
   regs.h.al = 0x14;        /* select mode 14 hex to get 132 columns */
   int86(0x10,&regs,&regs); /* call BIOS                             */

   /* Step 2: Load the required font.                                */

   regs.h.ah = 0x11;        /* set up for char gen load BIOS call    */
   regs.h.al = 0x12;        /* code for 8x8 font                     */
   regs.h.bl = 0;           /* block zero                            */
   int86(0x10,&regs,&regs); /* load char gen and set CRTC registers  */

   /* Step 3: Reprogram the CRT Controller for a 480-line screen size. */

     /* Reset the CRTC register 0-7 write protect bit to allow writing */
     /* to these registers.                                            */

     outp(0x3d4,0x11);               /* set index                      */
     temp = inp(0x3d5);              /* get register contents          */
     outp(0x3d5, temp & 0x7f);       /* clear protect bit              */

     /* Next, select the 480-line display size by setting the horizontal */
     /* and vertical sync polarities to the appropriate values.          */

     temp = inp(0x3cc); /* read misc. output reg to set sync polarities  */
     temp &= 0x3f;      /* mask existing bits                            */
     temp |= 0xc0;      /* OR in new bits                                */
     outp(0x3c2, temp); /* write register                                */

     /* Now reprogram the CRTC for 480 scan lines.                       */

     outp(0x3d4, 0x06); outp(0x3d5, 0x0b); /* vertical total register    */
     outp(0x3d4, 0x07); outp(0x3d5, 0x3e); /* overflow register          */
     outp(0x3d4, 0x10); outp(0x3d5, 0xea); /* vertical sync start        */
     outp(0x3d4, 0x11); outp(0x3d5, 0x8c); /* vertical sync end          */
     outp(0x3d4, 0x12); outp(0x3d5, 0xdf); /* vertical display end       */
     outp(0x3d4, 0x15); outp(0x3d5, 0xe7); /* vertical blank start       */
     outp(0x3d4, 0x16); outp(0x3d5, 0x04); /* vertical blank end         */

   /* Step 4: Update the BIOS data area to reflect the new screen size.   */

   temp = 60 * 132 * 2;                       /* calculate new buffer size */
   bios_data.ptr[BIOS_VBUFLENLO] = temp & 255; /* set low  byte of length */
   bios_data.ptr{BIOS_VBUFLENHI] = temp / 256; /* set high byte of length */
   bios_data.ptr{BIOS_ROWS] = 60 - 1           /* enter row count         */

  }
```

Figure 5. C Source Code for a 480-Line Alphanumeric Mode

some VGA registers is required to invoke a 480-line alphanumeric screen.

Changing Screen Dimensions Using BIOS

Figure 4 shows a PC assembler-code fragment that sets a 132-column by 43-line alphanumeric mode using the BIOS functions discussed. This code fragment requires that the video BIOS recognize mode 14h, which is the new 132-column text mode. Currently, PS/2 Models 35, 40, 56, and 57 are the only systems that recognize this mode number.

Notice the value of register BL when the Character Generator function is invoked. The value supplied in BL determines which VGA character generator is loaded with the specified font. VGA can store up to eight fonts in its font memory. Any two fonts can be displayed through combinations of attribute bit 3 and the VGA Character Map Select Register. Further details about this subject can be found in *Personal System/2 Hardware Interface Technical Reference – Common Interfaces*.

Figure 4 also shows that invoking an alphanumeric mode with a given number of rows and columns using only the BIOS calls involves the following three steps:

1. Select the required screen height with the Alternate Select BIOS call.
2. Invoke the appropriate video mode with Set Mode to select the desired number of columns.
3. Load the appropriate font via Character Generator to get the desired number of character rows.

Changing Screen Dimensions Using VGA Registers

Figure 5 contains a C-language function that sets a 132-column by 60-line screen text mode, using a 480-line alphanumeric mode. To do this, the function must reprogram some VGA registers, in addition to using the BIOS calls. Consequently, it does not follow the three-step procedure outlined above. Instead, the following procedure is used:

By varying the screen and font heights, up to 60 rows can be displayed.

1. Invoke the appropriate video mode, via Set Mode, to select the desired number of columns.
2. Load the appropriate font via Character Generator.
3. Reprogram the indicated VGA registers to set the screen height to 480 lines.
4. Update the BIOS data area to reflect the new screen dimensions.

Conclusion

This article explores ways to use video BIOS functions to display other than the 25 rows of characters normally displayed when an alphanumeric mode is set via the BIOS Set Mode function. By varying the screen and font heights, up to 60 rows can be displayed. This capability, combined with the 132-column text modes available with IBM's newest video subsystems, allows a large amount of data to be shown on a single screen. Other benefits of alphanumeric video modes are ease of programming, high performance, and the ability to select a wide range of screen sizes.

Related IBM Publications

- Update to *Personal System/2 Hardware Interface Technical Reference – Common Interfaces* (S04G-3281).
- *Personal System/2 and Personal Computer BIOS Interface Technical Reference* (S04G-3283).

***Steve Thompson** is an advisory engineer in graphics development in the Entry Systems Technology laboratory in Boca Raton, Florida. He is currently designing VLSI chips for IBM's Extended Graphics Adapter (XGA). Since joining IBM in 1983, Steve has primarily done graphics development; he has also designed memory boards and power supplies. He received a BS in electrical engineering from Wayne State University in Detroit, Michigan.*

Little Solutions

We invite you to share your "little solutions" in this column. Send them to us in care of the editor.

Subclassing Made Easy! (Part 2)

In the first part of this "solution" in the January 1992 issue, I explained that subclassing is used to modify the default behavior of a window. This is done by placing a "detour" in the message path for the window you want to control. In most detours, by following the signs, you eventually find yourself back on the familiar route. However, in some detours, you may travel a route that finishes at an unintended destination because someone erred in placing the detour signs.

Take a listbox as an example. When a listbox is created in an application, you expect it to behave in a certain predefined, documented way. For example, when you click the left mouse button on an item inside the listbox, the window procedure receives an LN_SELECT message, and the item is highlighted. However, when you double-click on an item (which is equivalent to pressing the Enter key), the default listbox procedure sends an LN_ENTER message to the application, and the window procedure then defines the action to be taken. You can also have the window procedure ignore the LN_ENTER message when it arrives, which results in no action.

Now point to another item in the listbox and click on it with the *right* mouse button. Nothing happens. Double-click on the same item and you get the same result: nothing!

Let us define an application that requires the listbox to respond to the right mouse button. We will select an item by clicking once (causing it to be highlighted), and we will cause an action to occur when we double-click the right mouse button.

The basis for our application is a simple phone book/dialer program that I wrote. My "PhoneBook" application is simply a listbox that displays a list of names, and a flat-file database that contains a name, address, and two phone numbers per entry. I have defined some buttons above the listbox that enable me to select a subset of the contents, to display or edit the address and phone numbers, and to dial either of the numbers.

The normal behavior of the listbox makes it easy to use the left mouse button to select and dial the telephone. However, what if we want to "point and shoot" – to dial either number with the mouse? To do that, we need to enable the right mouse button.

First, the application needs to define the prototype of the subclass window procedure for the listbox:

```
MRESULT EXPENTRY List1SubProc
(HWND, USHORT, MPARAM, MPARAM);
```

Second, create the listbox that becomes the "page" in the "phone book" in ClientWindowProc in response to the WM_CREATE message as shown in Figure 1.

Next, add the following statement to put up the detour sign.

```
hwndList1 = WinCreateWindow(
                                    hwnd,
                                    WC_LISTBOX,
                                    "Names",
                                    WS_VISIBLE|LS_NOADJUSTPOS,
                                    0, 0, 0, 0,
                                    Window,
                                    HWND_TOP,
                                    255,
                                    NULL,
                                    NULL);
```

Figure 1. Creating a Listbox

```
ptrNormListProc =
WinSubclassWindow(hwndList1,
List1SubProc);
```

Now, as I explained in the last issue, we have obtained the pointer of the normal listbox procedure and stored it in ptrNormListProc. We also have another pointer for our own listbox procedure, List1SubProc.

There is nothing strange or different about the procedure List1SubProc when compared to other window procedures. Remember, though, I wrote the original procedure to change the default behavior of my listbox. With that in mind, look at the program code in Figure 2 while reading the explanation of the operation and effect of each statement.

It is important to remember that not all messages are passed through to the normal window procedures from the predefined window classes. For example, the procedure for the listbox window class does not pass on messages pertaining to mouse button 2. This does not mean that they are not generated or sent; it simply means they are just thrown away.

All we must do is look for button 2 messages in our subclass procedure. When the program receives one, provide the logic so the desired behavior can be obtained. The first deviation from the normal behavior is to recognize clicks of mouse button 2, as lines 110 and 112 in Figure 1 show. Build a case statement for the WM_BUTTON2DOWN and WM_BUTTON2UP messages, then write all the logic to do the following:

- Locate the mouse pointer.

```
100 MRESULT EXPENTRY List1SubProc (HWND hwnd, USHORT msg,
MPARAM mp1, MPARAM mp2)
102 {
103    switch (msg)
104    {
105       case WM_BUTTON1DBLCLK:
106          WhichMouseButton = BUTTON1;
107       case WM_BUTTON2DBLCLK:
108          WhichMouseButton = BUTTON2;
109          msg = WM_BUTTON1DBLCLK;
110       case WM_BUTTON2DOWN:
111          msg = WM_BUTTON1DOWN;
112       case WM_BUTTON2UP:
113          msg = WM_BUTTON1UP;
114    }
115    return((*ptrList1SubProc) (hwnd, msg, mp1, mp2));
116 }
```

Figure 2. Listbox Procedure

- Identify the first item displayed in the listbox.
- Determine the height of the items in the listbox.
- Compute the offset of the item under the mouse pointer at the time button 2 was clicked.

Finally, invert the colors so the item is highlighted. Not so simple!

Just a minute: button 1 already does that. All we want is to have a single click of button 2 duplicate the behavior of button 1. So, in lines 111 and 113, we need to change the value of MSG from WM_BUTTON2UP/DOWN to the corresponding button 1 messages. Then, send the messages to the normal listbox procedure via the pointer ptrNormListProc. That saves us much work, and it is also easy to do.

Now we can concentrate on the more difficult task: defining the action to be taken when button 2 is double-clicked. We can save ourselves a lot of work here as well. What we want from button 2 when it is double-clicked is the same thing that happens when button 1 is double-clicked. The only difference is that we want to dial a different phone number. We can do the same thing we did before, but we also have to let the dialer function know which phone number to dial. Lines 106 and 108 set the variable WhichMouseButton to BUTTON1 or BUTTON2, respectively, and the dialer function uses that information to make the selection. Line 109 changes MSG to WM_BUTTON1DBLCLK and sends it on. The only reason for line 106 is to reset WhichMouseButton for a real button 1 double-click.

The rest of the application still does not know anything about mouse button 2. It sees only a variable (that might not otherwise be used) and makes a selection based on the contents. By intercepting the button 2 messages, we have succeeded in modifying the way a listbox normally works. That is subclassing!

— Larry Pollis, IBM, Dallas, Texas

PS/2 Models 90 and 95 Reference Diskettes

The newest models of the IBM Personal System/2 8590 and 8595 come with a new Reference Diskette that is different from the one used by all the other 8590 and 8595 models. You can no longer carry one diskette around and expect it to work on all models of the 8590 and 8595.

The new Reference Diskette supports the 8590 and 8595 models that have 25 MHz 80486SX processors and optional 80487SX coprocessors. These models, -0H5, -0H9, -0HF, and those with the 487SX/25 processor upgrade option must use the new Type Two Reference Diskette.

The new Reference Diskette became necessary because additional Reference files were created for the 25 MHz SX systems, and these new files could not fit on the original Reference Diskette for 8590 and 8595 systems. That original Reference Diskette, which is called *Type One*, works in all 8590 and 8595 systems except the newest 25 MHz models. A Type One Reference Diskette is labeled Level 1.10 and has a small "1" above the part number. The newest Reference Diskette, called *Type Two*, supports the 25 MHz SX 8590 and 8595 systems listed above. A Type Two diskette is labeled Level 1.00, and it has a large "2" above the part number.

Figure 1 shows the part numbers for all Reference Diskettes for the 8590 and 8595.

	PS/2 System	
	8590	8595
Type One diskette	92F3042	92F3036
Type Two diskette	04G4713	04G4727

Figure 1. Part Numbers for 8590 and 8595 Reference Diskettes

The model byte and the submodel byte in the computer determine which Reference Diskette should be used. The computer's model byte and submodel byte can be found by writing a program that uses interrupt 15H, function C0H (Return System Configuration Parameters), or by using the Reference Diskette (once you know which one to use) and viewing the system configuration. Figure 2 is a cross-reference between system type, model byte, submodel byte, and type of Reference Diskette.

— Chuck Hanford, IBM, Dallas, Texas

System	Model Byte	Submodel Byte	Reference Diskette Type	Processor
Model 90				
8590-0G5, -0G9	F8	2F	1	80486SX 20 MHz
8590-487SX/20	F8	2D	1	80487SX 20 MHz
8590-0H5, -0H9	F8	57	2	80486SX 25 MHz
8590-487SX/25	F8	59	2	80487SX 25 MHz
8590-0J5, -0J9	F8	11	1	80486 25 MHz
8590-0KD, -0K9, -0KF	F8	13	1	80486 33 MHz
8590-50 MHz Processor	F8	2B	1	80486 50 MHz
Model 95				
8595-0G9, -0GF	F8	2E	1	80486SX 20 MHz
8595-487SX/20	F8	2C	1	80486SX 20 MHz
8595-0H9, -0HF	F8	58	2	80486SX 25 MHz
8595-487SX/25	F8	5A	2	80487SX 25 MHz
8595-0J9, -0JD, -0JF	F8	2A	1	80486 25 MHz
8595-0KD, -0KF	F8	14	1	80486 33 MHz
8595-50 MHz Processor	F8	16	1	80486 50 MHz

Figure 2. 8590 and 8595 Systems, Model Bytes, Submodel Bytes, Reference Diskettes

New Products

IBM LinkWay Live! Version 1.0

LinkWay Live!™ Version 1.0 (20G2106) is a multimedia-enabled version of the LinkWay™ product. In addition to providing a low-cost application development environment, IBM LinkWay Live! Version 1.0 enables the simplified creation and use of multimedia applications. Other major enhancements are support for DVI® technology using the PS/2 Action-Media® II Display Adapter and IBM PS/2 ActionMedia II Capture Option, use of the extended memory feature of IBM DOS Versions 4.0 and 5.0, and support for two new graphics modes, XGA and VGA-8.

Highlights:

- Enables easy integration of multimedia objects into applications without programming
- Supports the DVI Technology via the IBM ActionMedia II Display Adapter and IBM ActionMedia II Capture Option
- Provides support for DOS extended memory
- Provides support for VGA-8 and XGA graphic modes
- Supports networked multimedia

Letter # 292-048, January 21, 1992

IBM OfficeVision/2 Version 1 Release 2.0

The OS/2 Office Feature of Office-Vision™/2 has been enhanced to include support for NetWare 3.11 and 2.2 network operating systems, a gateway interchange between Network Courier users and Office-Vision/2 users, installation process improvements, performance and other functional improvements.

Highlights:

- Represents a Systems Application Architecture (SAA) application, based on OS/2, providing business solutions for mail, address book, correspondence editing, file system, and telephony.
- Enables users on a LAN to exchange information with each other, IBM OfficeVision/2 users on another LAN, and users of VM, MVS, Operating System 400®, and VSE office systems.
- Presents a Common User Access (CUA) interface in which everyday office objects are represented pictorially. In the OS/2 environment, the IBM OfficeVision/2 office objects behave as the objects they represent, promoting ease of learning and use.
- Allows Network Courier users to coexist with OfficeVision/2 users.
- Provides installation enhancements with Online Installation Template and creation of LAN aliases.

Letter # 291-735, December 17, 1991

IBM VoiceType Version 1.0 Speech Recognition Software

IBM VoiceType is a versatile productivity tool. When used with the appropriate IBM M-Audio Capture and Playback Adapter, it allows speech to be used as an alternative to keyboard input for text entry or control of applications, DOS, and VoiceType itself. Both Micro Channel and AT bus architectures are supported on systems having a 386, 386SX, 386 SLC, 486, or 486SX processor with a clock speed of 16 MHz or faster and meeting the memory and fixed disk requirements of VoiceType.

Highlights:

- User productivity can be enhanced by using speech recognition with a large, expandable vocabulary, an 80,000-word backup dictionary, voice commands, and voice macros.
- VoiceType offers a business solution for employing persons who cannot easily use a keyboard.
- Multiple user speech models combined with voice-and-keyboard input capability to a variety of software applications and DOS offer growth potential.
- Supports both AT bus and Micro Channel architectures and works with a variety of DOS applications.
- A microphone is the only external attachment.

Letter # 292-064, February 4, 1992

NetWare Network Computing Products from IBM

NetWare Services for OS/2 from IBM is being announced as an additional product under a product distribution, licensing, and support relationship with Novell, Inc. This is consistent with IBM's interoperability strategy. Under the terms of this relationship, IBM markets, services, and supports NetWare computing software. Net-Ware Services for OS/2 from IBM includes the NetWare Requester for OS/2 2.0, as well as graphical desktop utilities, enhanced installation program, and network management tools utilizing Presentation Manager.

Letter # 292-065, February 4, 1992

Personal Systems

What's *your* opinion?

We want to provide you with the information you need about IBM personal systems products. You can help by letting us know what topics are important to you.

Use this form to give us feedback. First, make a copy of the form. After completing it, fax it to (817) 961-7218 or (817) 961-6520, or mail to:

IBM Personal Systems Technical Solutions
IBM, Mail Stop 40-A2-04
1 East Kirkwood Boulevard
Roanoke, TX 76299-0015

Check those topics that interest you.

- ☐ OS/2
- ☐ OS/2 Presentation Manager
- ☐ OS/2 Database Manager
- ☐ OS/2 Communications Manager
- ☐ OS/2 Application Development
- ☐ Object-Oriented Programming
- ☐ APPC
- ☐ DOS
- ☐ LAN
- ☐ OS/2 LAN Server and Requester
- ☐ DOS LAN Requester
- ☐ LAN Installation
- ☐ Novell NetWare
- ☐ Token-Ring and Ethernet
- ☐ LAN Problem Determination
- ☐ 3270 Emulation
- ☐ PS/2 Micro Channel Computers
- ☐ Other PS/2 Computers
- ☐ PS/1
- ☐ Printers and Printing
- ☐ Displays
- ☐ SCSI
- ☐ CD-ROM
- ☐ Multimedia
- ☐ PC Security
- ☐ Directions and Strategy
- ☐ Other__________

What do you think of this issue?

	Very Useful 1	2	3	Not Useful 4	Not Applicable
Overall	☐	☐	☐	☐	☐
Hardware Articles	☐	☐	☐	☐	☐
OS/2 2.0	☐	☐	☐	☐	☐
DOS	☐	☐	☐	☐	☐
IBM and Novell	☐	☐	☐	☐	☐
Little Solutions	☐	☐	☐	☐	☐

Comments____

How many years have you used personal computers? __________ IBM employees check here ☐

What is the size of your company? ☐ Under 100 ☐ 100–500 ☐ 501–1,000 ☐ Over 1,000

Thanks for your help!

Libby Boyd

Libby Boyd
Editor

Index to Past Issues of IBM Personal Systems Technical Solutions

“**The advantage in not being tied to ISA is that Micro Channel can truly be a consistent standard, whereas EISA cannot.** (page 4)

“**The clear choice for leading-edge portables is active-matrix LCD.** (page 16)

“**TFT LCD panels offer several advantages over STN panels for color, contrast ratio, viewing angle, and response time.** (page 21)

“**Working with objects on your OS/2 Desktop is easy.** (page 24)

“**Operating System/2 2.0 comes with 29 new productivity and entertainment applications.** (page 32)

“**The ExtendedInstall keyword allows the OS/2 Install program to be chained with another installation program.** (page 41)

“**IBM and Novell communications software share a LAN adapter with the many different protocols (IEEE 802.2, NetBIOS, TCP/IP, IPX, and so on).** (page 48)

“**ARCserve backs up or restores an entire LAN.** (page 63)

“**The DOS protected-mode environment is a DOS environment created by control programs that run in the protected mode of the processor.** (page 66)

“**By varying the screen and font heights, up to 60 rows can be displayed.** (page 81)

G325-5015-00