Personal systems


# Focus on Micro Channel Architecture

Special Supplement to
IBM Personal Systems Technical Solutions

IBM®

$12.00

*IBM Personal Systems Technical Solutions* is published quarterly by the IBM United States technical support center, International Business Machines Corporation.

| | |
|---|---|
| Editor | Libby Boyd |
| Technical Consultant | Chet Heath |
| Communications | Elisa Davis |
| Design | Corporate Graphics |
| Illustrator | Bill Carr |
| 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 form number GBOF-1229.

# Contents

## Focus on Micro Channel® Architecture



*These articles have been assembled and reprinted from various sources. The original sources are noted at the bottom of each page. We hope this collection will be a valuable reference.*

# Micro Channel Developers Association

**Membership List**
**June 12, 1992**

ACCTON Technology Corp.
Advanced Computer Research Inst.
Advanced Logic Research , Inc.
Advanced Micro Devices, Inc.
ANCOT Corporation
Aox Incorporated
Apricot Computers Ltd.
AST Research Japan K.K.
Beall Technologies, Inc.
Beijing Legend Computer Group Co.
Brooktrout Technology Inc.
Brother Industries, Ltd.
Cabletron Systems Inc.
CHIPS & Technologies, Inc.
Ciprico Inc.
CMS Enhancements Inc.
Communica
Computer Elektronik Infosys GmbH
Core International
Cornerstone Technology, Inc.
Cumulus Corporation
D-Link Systems, Inc.
Dale Computer Corp.
Dataplane Technologies Inc.
Dialogic Corporation
ELSA GmbH
EnQue Corporation
Equinox Systems, Inc.
Future Domain Corporation
Headland Technology Inc.
Hitachi, Ltd.
Hypertec Pty. Limited
IBM Corporation
IBM Japan Ltd.
Infotronic S.p.A.
Intel Corporation
Interphase Corporation
Justsystem Corp.
Kingston Technology
Madge Networks, Ltd.
Matsushita Elec. Industrial Co., Ltd.
Memorex Telex
Memórex Telex Japan Ltd.
Micral Inc.
Microsysteme Co., Ltd.
Miro Datensysteme GmbH
Mitsubishi Elec. America, Inc.
MPR Teltech Ltd.
National Instruments Corp.
National Software Testing Labs
NCMIKRO
NCP Engineering
NCR Corporation
OKI Electric Ind. Co., Ltd.
Olivetti
Palyn Associates, Inc.
Parallan Computer
Pencom Systems Inc.
Phoenix Technologies Ltd.
Piiceon, Inc.
Plaintree Systems Inc.
PROTEON, Inc.
Racore Computer Products, Inc.
Radius Inc.
Reply Corporation
Research Machines Ltd.
Rexon/Tecmar Inc.
Ricoh Company, Ltd.
SANYO Electric Co., Ltd.
Seattle Telecom & Data
Sharp Corporation
Siemens Nixdorf Informationssysteme
Spea Software AG
Stac Electronics
Stargate Technologies, Inc.
Stratus Computer
Sumitomo Electric USA, Inc.
Systech Corporation
Toshiba America Electronic Components
Tseng Laboratories, Inc.
TT-Timecon OY
VLSI Technology Inc.
VME Microsystems International Corp.
XXCAL Inc.

# *Foreword*

One fall, I was enjoying a cup of coffee in the afternoon, watching mother nature with her enduring beauty and myth. High in the sky I noticed a flock of migrating geese heading south, flying reliably and efficiently for days. It amazed me to realize the effort these birds exert and the accuracy they enjoy to get them to their final destination.

Architects of computer systems have always made it their objective to produce systems embodying some of the basic characteristics found in those migrating birds. ***Reliability, efficiency, and precision*** are fundamental requirements for the survival of those migrating geese. And I contend that these are also requirements for any leading architecture in computer systems, or any other system.

One can draw from the Micro Channel architecture and its benefits and compare to those of the migrating birds. The intrinsic features embodied in the architecture can be grouped under those three basic requirements I mentioned.

***Reliability*** is increasingly becoming a must for most systems due to increased complexity, higher performance, and more responsibility. Micro Channel architecture offers a suite of reliability features that can be summarized as: higher level of noise immunity, lowest level of noise emission, data integrity checking, foolproof mechanical installation, level-sensitive electrical signals, distributed interrupt scheme, foundation to promote a higher level of integration, and a well grounded connector design.

***Efficiency*** is what the world is striving for in one form or another. Higher productivity, lower cost, and maximum return on investment are typical requirements imposed by organizations. Micro Channel architecture was designed with all those requirements in mind. Today, we have one of the highest bus transfer rates in the world. At 160 MBytes/second the Micro Channel is most likely the fastest open industry standard bus, implemented with today's economical technology. As we all know, it's only a matter of time until even higher rates are needed. This is dependent on applications and progress in compression technology. But more data traffic will still be experienced. Other features include bus mastering, distributed interrupt scheme, subsystem control block architecture, processor independent architecture, and automatic configuration.

***Precision*** is paramount. The results we all count on and often take for granted cannot be assured without attention to precision. The inherent characteristics of the Micro Channel architecture described above provide the foundation for dependable and accurate operation essential for many.

The Micro Channel Developers Association has introduced several programs aimed at ensuring the best possible level of compatibility, by design. That is "built-in" compatibility. We are creating an environment for the developers of Micro Channel-based products unmatched (or non-existent) by any other bus standard: one common version of the Micro Channel specification, common design tools (behavioral models and test vectors), a design verification facility to verify new designs, and a stringent "Micro Channel Architecture Certification" process that includes both "real world" as well as "parametric" testing. Our plans also include quality related tests such as noise emission and susceptibility.

This architecture is supported by industry leaders with substantial commitment. Micro Channel has perhaps the best acceptance of all advanced buses for concurrent systems. Over 12 million systems have been installed, and over 1200 option cards are available. This is because Micro Channel systems are very cost effective over the life of the product and the architecture transcends multiple processor platforms. Further, there is an easy licensing process similar to that of the AT and EISA buses; it is no longer proprietary, and the estimated growth rate exceeds 12% per year through 1995.

The Micro Channel Developers Association is an independent, non-profit, worldwide organization dedicated to the promotion of Micro Channel architecture. Developers of Micro Channel-based products and related activities are eligible for membership. In the short time the association has been in existence, membership has grown to close to 100 companies, and is still growing. In addition to the programs mentioned earlier, the association conducts technical seminars, data base services, technical meetings, and marketing/promotion meetings where strategic, technical and promotional discussions are held. Future generations of the Micro Channel are being formulated during those meetings. The association also disseminates information – technical and marketing – related to the architecture, participates at industry shows, and acts as a catalyst in promulgating technical issues.

The intrinsic features of the architecture, augmented by the services of the Micro Channel Developers Association, provide the ultimate recipe for more competitive products, shorter development cycles, early visibility into the future, and optimized "networking" with peers.

The migrating geese bet their lives on their capabilities. We are striving to aid developers who bet their success on Micro Channel architecture.

Ramiz H. Zakhariya
President
Micro Channel Developers Association

Micro Channel Developers Association
2280 N. Bechelli Lane #B
Redding, CA 96002
Telephone (916) 222-2262
Fax (916) 222-2528

# Requirements for Advanced Bus Architecture

*Chet Heath*
*IBM Corporation*
*Boca Raton, Florida*

**The structured channel architecture for microsystems has evolved in the same spirit as the structured channel architecture for System/360 mainframe systems. Like the System/360 channel architecture, Micro Channel architecture has become a standard, an accepted architecture for advanced microsystems. Micro Channel architecture is fully compatible with applications and operating systems for the IBM Personal Computer, and it supports a new family of adapter cards as well as new capabilities.**

## IBM Heritage

The personal computer owes its heritage to a period in 1973 when IBM introduced the 5100 series of products. The 5100 machine was a desktop computer that could execute about 30,000 instructions per second. It had a 5-inch screen and a keyboard that could not be detached. In its time, it was useful for simple, single-task, single-thread applications, typically written in BASIC or APL.

The IBM 5100 machine later advanced to the 5110, the 5120, the 5130, and the System/23 Datamaster. The Datamaster, based on Intel® 8085 processor architecture, differed principally in its channel architecture from the PC by only four lines. The PC featured additional addressing of up to one million bytes.

The PC evolved into the Personal Computer XT™ with the addition of a fixed disk. The Personal Computer AT®, initially operating at 6 MHz, produced a theoretical 750,000 instructions per second. That was later upgraded to one million instructions per second with an 8 MHz AT® implementation. For comparison, the theoretical calculations for several machines are shown in Figure 1.

All these implementations were meant to be compatible with previous products, but they lacked a general architecture or direction. It became obvious with the AT computer that a general structured architecture for advanced microsystems would be needed to move to the multitasking, multiuser environment with concurrency, typified by PS/2 Models 50 and higher.

## Previous Evolutions

A similar architecture evolution occurred in 1964 when the 1401 series of machines evolved to the IBM System/360. That transition was from single-task, single-thread operations to multitasking, multiuser operations for which the System/360, its channel architecture, and its entire system design were intended.

Another evolution occurred in 1973 in the minicomputer arena between the IBM System/3, which was a single-task, single-thread environment, and the multitasking System/3X series of machines. A transition was provided at this time – the System/32 was provided to continue the single-task applications in equipment designed for the multitasking environment.

In 1987, IBM accomplished another evolution by making the transition from the single-task, single-thread environment of the AT computer multitasking PS/2 system with Micro Channel architecture, while also providing a compatible environment for single-thread applications in the Models 25, 30, and 30 286 (Figure 2).

Each time, the needs of the future could not be met by modifications

**Approximate Performance Comparison**

| | Model | Approximate Performance |
|---|---|---|
| Single Tasking | 5100 | 30,000 Instructions/Second |
| | Personal Computer | 300,000 Instructions/Second |
| | AT (6MHz) | 750,000 Instructions/Second |
| | AT (8MHz) | 1 Million Instructions/Second |
| Multi Tasking | PS/2 - 50/60 | 1.25 Million Instructions/Second |
| | PS/2 80 | 3 Million Instructions/Second |
| | PS/2 - 70-A21 | 6 Million Instructions/Second |

**Figure 1. Approximate Performance Comparison**

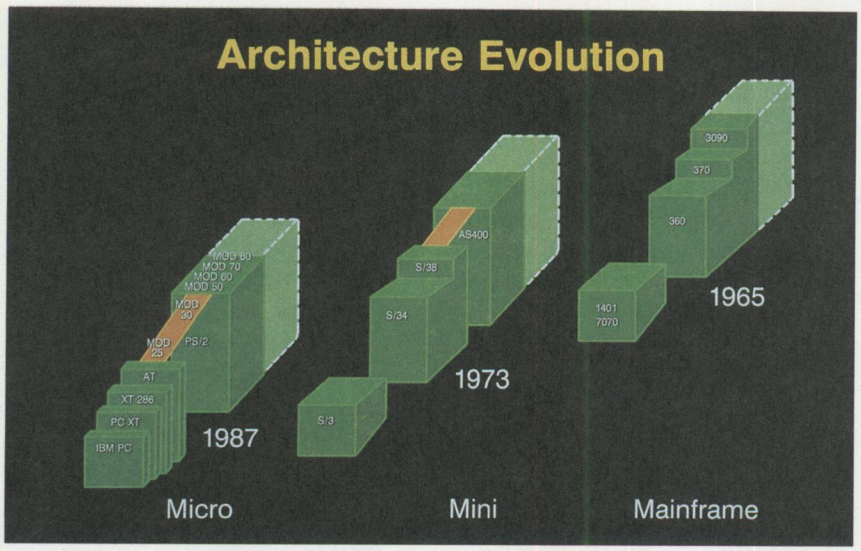



**Figure 2. Architecture Evolution**

of the past, and a new generation of systems was defined.

## Single-Tasking Versus Multitasking

In a single-task system, each operation occurs in sequence. Much the way a person can assemble a bicycle in sequential steps (first the handlebars, then the wheels, and finally the seat), so can tasks in a computer be manually performed, in sequence, by one user. The time to complete the entire sequence of operations is the sum of the times required for each task.

In the single-tasking environment, operations to the disk require a delay in processing. This is because accessing the disk, talking to the I/O, or operating the processor are each sequential operations. During the seek latency period of the file, the processor is unused.

The command to operate the disk and the time to process the data are typically short compared to the time required for the disk to respond. All other tasks wait for the first to complete, and delays accumulate from all tasks. For this reason, the latency of the disk file in a single-tasking machine is a significant factor in the performance of the system as a whole.

Single-tasking has the advantage of simplicity in system design, yielding low-cost systems. But it is not the way most of us work or think – most of us multitask in both our work and our thinking.

In a factory, for example, multiple tasks would be performed concurrently to maximize productivity. Operations would be done simultaneously whenever possible. The time to complete the entire operation is then little more than the time required to complete the longest task.

The difference between single-tasking and multitasking involves this concept of concurrency, which means that, in multitasking, I/O operations can occur at the same time rather than sequentially.

When the system is shared among a number of users or a number of tasks, it becomes much more viable economically, and the organization that it supports becomes more productive.

## Technology Trends

Looking at the trend of technological advances from 1981 until today (Figure 3), we see a 5-to-1 improvement in the clock rate of the processor. The efficiency of the execution unit for real-mode instructions has improved by a factor of 4-to-1. We have seen about a 20-to-1 improvement in the processing power of the system. I/O devices that depend on the processor to support data movement have increased their data transfer requirement (called throughput) a great deal more. Displays have moved from 128,000 picture elements to more than 6 million. Direct-access storage devices moved from the 50,000 bytes-per-second transfer rate of diskette media to the enhanced small device interface (ESDI) in excess of 1.25 million bytes per second, and most recently to the small computer system interface (SCSI), transferring data at approximately 5 million bytes-per-second.

Communications has undergone the most dramatic transition in performance. Going from 300-baud modems, which were a luxury in 1981, to 16-million-bit-per-second local area networks has produced a 53,000-to-1 improvement. I/O devices are now moving data much more quickly.

## The Need for Change

The objectives for the IBM Personal Computer were to provide an address space of 1 million bytes and a path for the transfer of one character at a time. These machines could

do only single transfers of data, which typically depended on the processor for every transfer. The AT was extended to allow a 16-megabyte address space defined by a 24-bit address bus and a 16-bit data path. The AT computer also introduced the "string mov" operation capability of the 286 processor that allowed the rapid movement of blocks of data to and from file devices.

Single-tasking machines operate their I/O interfaces one at a time, so the requirement for the bus throughput is no greater than the fastest I/O device. But in multitasking, where tasks and I/O devices operate at the same time, throughput requirements are higher.

IBM did not consider the AT bus to be a suitable base upon which to define a family of multitasking and multiuser systems. When taken together, the limitations of the AT's interface to I/O implied a number of concerns about the design of any system that supported compatibility with the cards designed for the PC, PC XT™, and AT computers. In 1984 these limitations may have been viewed as advantages for the single-task, single-thread environment; indeed, the AT is still a qualified bus for those operations. But in the multitasking environment, there were limitations in several areas:

**Addressing – 24 bits:** The AT implemented only 24 address lines for memory, so only 16 MB of storage can be addressed on the bus.

**Data Bus – 16 bits:** The AT data bus is only 16 bits wide, limiting the throughput, when existing cards are installed, to 5.3 MB instantaneously and about 3.5 MB continuously.

**Connector Design and Insertion Force:** The PC and AT connector design is a legacy from the mid-1970s' design of the IBM 5140, with insertion and removal forces approaching 40 pounds. Good design practice would keep the insertion and removal forces below 40 pounds for assembly and service operations by humans or robots. Extending the AT connector to have sufficient pins for a 32-bit interface would yield forces closer to 65 pounds during either insertion or removal. This would expose the system board to excessive stress during assembly and service operations.

**Direct Memory Access (DMA) Utilization:** The DMA channels, used as alternate paths between I/O and memory, were permanently assigned to devices early in the development of PC and PC XT machines. The AT design added more channels at a wider width, but DOS cannot use them. As a result, the DMA controller is assigned to I/O that is slow by today's standards. The processor does all the work to move data between memory and I/O.

**Electromagnetic Compatibility (EMC):** The radiation of electromagnetic energy outside the system and susceptibility to external interference are not just problems to be solved by the manufacturer, but issues of data quality and validity for the user. When energy is radiated from a data signal, the data levels are reduced, and may oscillate below valid levels. The signals then become vulnerable to outside interference. Shielding the cabinet is expensive, and only contains the energy within the cabinet. It does not limit the electromagnetic incom-

| | 1981 | Today | Ratio* |
|---|---|---|---|
| Processors | 8088 | 80386 | 4:1 |
| Instruction Unit Efficiency<br>Clock MHz | 4.8 | 25 | 5:1 |
| Maximum Memory Size | $2^{20}$ | $2^{32}$ | 4000:1 |
| Data Bus | 8 | 32 | 4:1 |
| Performance | 300 KIPS | 6 MIPS | 20:1 |
| Display (Mbits/screen) | 0.128 | 6.291 | 50:1 |
| DASD (MBytes/sec transfer rate) | 0.050 | 1.250 | 25:1 |
| Printer (Bytes/sec) | 80 | 5000 | 63:1 |
| Communications (Bits/sec) | 300 | 16,000,000 | 53,000:1 |

* Approximate performance ratios for comparison only

**Figure 3. Technology Trends**

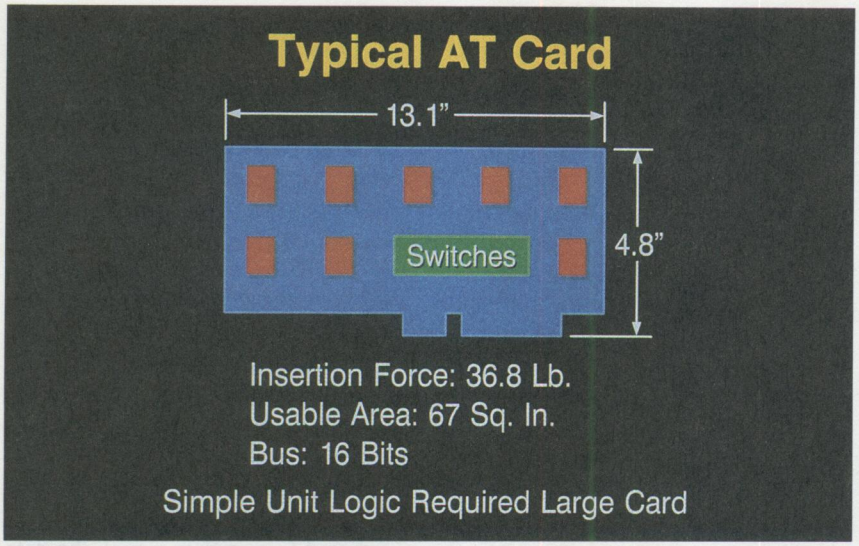



**Figure 4. Typical AT Card**

patibilities between cards or devices within the cabinet. EMC problems increase, as the square of the data rates increases, on the I/O bus. Faster I/O demands a solution to this problem.

**Configuration Conflicts:** Because the system resources required to operate a card are either fixed in the design, or very limited in selectability by switches, and because most PC, PC XT and AT designs decode only 10 of the 16 I/O address lines, many conflicting situations exist with the configuration of systems that use these cards. For example, because PC, XT, and AT interrupts cannot be shared between two device adapters, and because compatible software typically recognizes only two communications interrupts, a maximum of two cards that drive the interrupt request lines can be installed. If a user wishes to install a binary synchronous adapter, one asynchronous card must be removed. If a synchronous data link control (SDLC) adapter is installed, the binary synchronous card must be removed. This limits the number of solutions that the user can define.

**Logic-ground Isolation:** Personal computer cards have only three ground pins to connect the logic ground on the card to the system board ground. If a card pulls sufficient current from the system, the logic-ground reference on the card can rise as high as one-half volt above the system ground. Data transfer between the cards and the system can then yield misinterpretations in data level, and invalid transfers can occur. Therefore, if a branch address is sought at the instant that a noise spike or static discharge pulse from elsewhere in a network is added to the DC offset, and if the branch is taken, the system can lose its place in memory and execute variables or instructions out of order. The user is locked out of the keyboard. This places a requirement on any new architecture to solve this problem.

**Processor Workload:** Affecting performance the most, however, is the loading of a great deal of the I/O-to-memory transfer burden onto the processor by simple PC adapter cards that implement little hardware and use the processor to perform much of the interface logic through the basic input/output system (BIOS) or application drivers. This practice may be permissible in a single-tasking machine because the processor is not used elsewhere in a single-tasking system. But in a multitasking machine, such I/O designs can decrease performance because they tie up the processor when it could be dedicated to tasks and the operating system.

*The limitations that compatibility to PC, XT, and AT cards would impose on a system outweigh the relatively infrequent need to propagate old cards from an old system to a new system.*

## Getting Technical

**Mechanical Considerations:** Taking a closer look at the system design limitations just outlined, consider the typical AT card (Figure 4). When populated with early 1980s-vintage unit logic components and switches, a large card was required to perform even the simplest I/O adapter function. The personal computer 8-bit card connector consumed approximately 3 inches, or one-fourth the width of the PC system board. The AT extended the connector to slightly less than half the width of the AT system board, and tightened the mechanical reference and tolerance control in the placement of the copper tabs on the cards and in the spring contacts in the system unit connector. The force to insert or remove the card grew close to the 40-pound maximum recommended force. Large connectors consumed a large fraction of the limited system board area, yielding a large system board when the processor, memory and

I/O adapter logic were added to the layout (Figure 5). This caused the system unit to consume more area and a larger fraction of the already crowded desk. And, from a manufacturing point of view, the larger AT cards may yield unusable waste when they are cut from the standard 12-inch by 24-inch raw stock.

It became obvious that a smaller system overall, with smaller components that consumed less desk space, would be an advantage to the customer.

While this requirement may imply more development expense and time for adapter card designers to implement in a smaller space, that expense is typically amortized over tens of thousands of cards. Indeed, the techniques typically employed to implement in a small space – large-scale integration, surface-mount technology, automated manufacturing systems and computer-aided design – yield card designs that are often less expensive to duplicate and more reliable.

The developer's challenge is not the customer's problem. To the customer, small cards are better, if a smaller footprint is needed.

**Electromagnetic Compatibility:** Let's unfold a card to visualize the EMC design problems of AT cards and the systems that they support. Looking specifically at both sides of the connector interface at once (Figure 6), and highlighting the radio frequency ground signals, one can see that most of the signals do not have the close proximity of a radio frequency shield. In fact, the processor clock was placed midway between two radio frequency grounds. This was acceptable when the card format was designed in

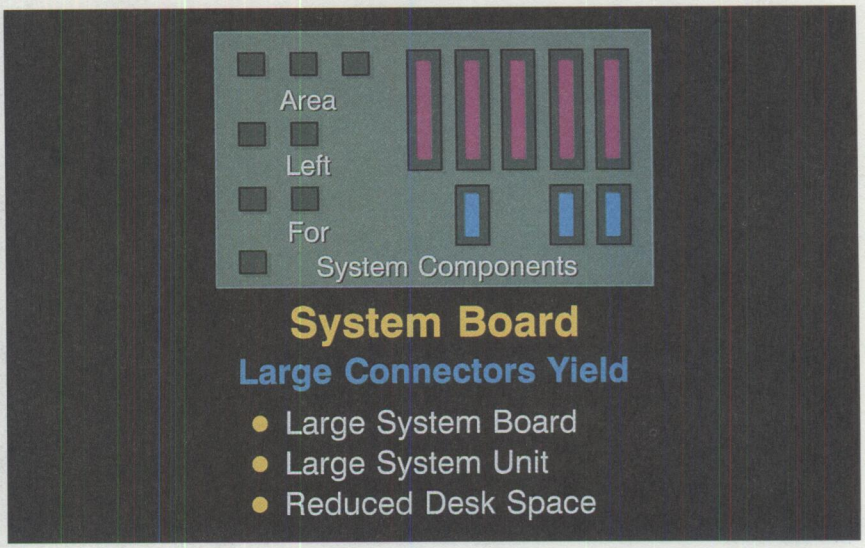



**Figure 5. System Board Layout**

1978 for the 5140 machine, from which the PC was derived. That system had a slow clock by today's standards.

Today, however, clock rates are much faster, data transfers are much faster, and I/O devices move much more data in a shorter time. The result is electromagnetic incompatibility with other nearby equipment and internal interference between elements within the system cabinet. The security of the data moving through the machine can even be compromised by others.

To prolong the life of the PC bus and allow extension to the 16-bit AT interface, IBM introduced "single-point ground" power systems, conductive plating of the power supply, frames and covers, and internal ground planes in PC

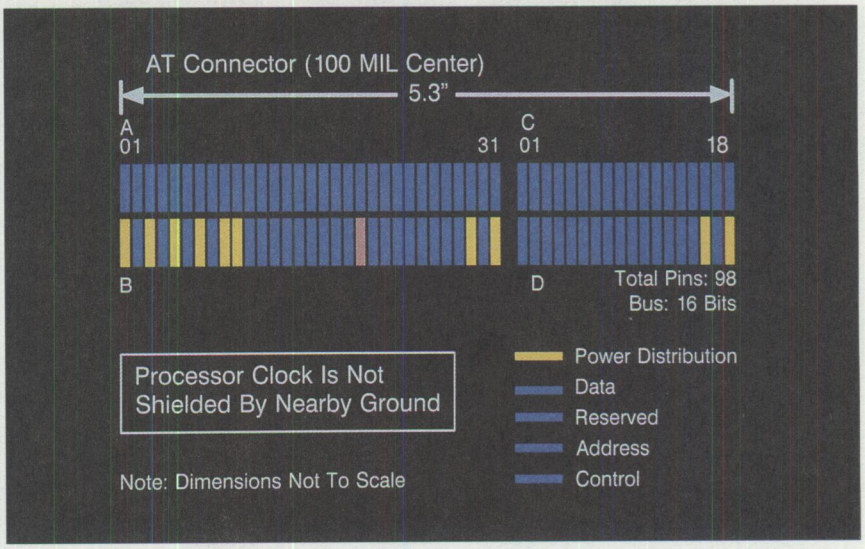



**Figure 6. AT Connector**

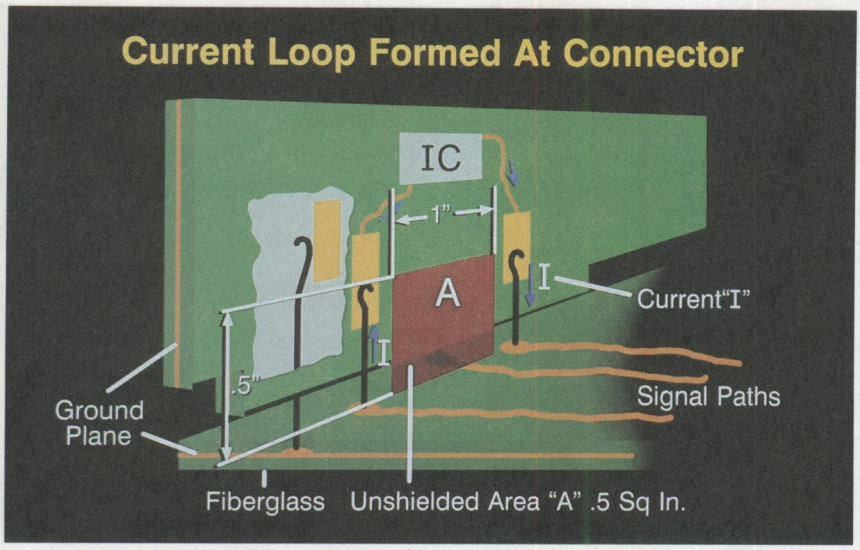



**Figure 7. Current Loop Formed at Connector**

XT and later machines. But those measures become much more costly and less effective as the speed of the systems increase. To understand why, let's go back to basic electricity.

Many of us, in our youth, built electromagnets by winding turns of wire around an iron nail or spike. We found that as we increased the area inside of loops by creating more turns or accumulating area, the magnetism increased. Also, when we increased the current by adding more batteries, we increased the magnetism.

This is true for a fixed distance from the radiating element. The electrostatic portion of the radio wave is proportional to the area inside of a loop times the current that flows through that loop, and if it is an alternating current or radio frequency, times the square of that frequency.

The data transitions on the computer bus are currents that alternate at very high frequencies and thus have the potential to transmit energy (or receive energy) as interference with other equipment either inside or outside the box.

Look at the card connector in three dimensions (Figure 7). An unshielded area is formed where the current flows on spring tabs to a card, through the card, and then returns over spring tabs back to the system. Everywhere else in the system – on the cards and on the system board – there is ground shielding, except at this one point.

Using a radio frequency quiet room, IBM engineers found that the card connector on the system board was the principal source of radiation from the I/O bus when the interface between I/O and the system was active. This unshielded area was considerably larger in the PC, PC XT and AT implementations – about one-half square inch. In all these implementations, the ground system on the cards is connected to the ground system or shield on the outside of the case.

However, this meant that when the limited number of grounds in the AT was inadequate to sink a current back to the power supply – for example, if the card bracket retaining screw was not tightened firmly – the currents could flow back through the metal system frame (Figure 8), which is the intended path for containing noise in the PC, PC XT, or AT system. Current returning to the power supply could actually flow out over the shields of cables and back through the AC power mains, forming a gigantic loop called a "ground loop." Even with a small current flowing, this would create a large loop area and very large radio signal. This is called "conducted electromagnetic interference."

The ground mechanism used in the PC, PC XT, and AT also makes the system susceptible to noise from outside the system. Energy from equipment connected to the system could be coupled through the shield system. If the retaining screw was not tightened, the current could sink through the card and through the system rather than safely returning to the AC mains. Typically, this would manifest itself as a data error when static discharges, perhaps from someone touching adjacent equipment in a low-humidity environment. This data error would not necessarily occur in the equipment that the person touched. In extreme cases, a lightning strike could actually destroy local area network (LAN) cards or communications cards where the screw is not tightened.

**Requests for Processor Attention:** The AT system unit typically had a few devices connected to it, enough for one user and a few applications. It was designed to support simple file serving and very limited multitasking configurations for only one user. Yet, increasingly, we are

using desktop systems as small mainframes. With multiple users often attached through networks, and multiple tasks per user, these systems often perform the same duties as the mainframe machine in a raised floor environment.

In a mainframe environment, each user or each task may have a favorite piece of I/O that it operates. For example, if two tasks were to talk to the same printer at the same time, the text would be mixed between the two, and we would see gibberish printed on the paper. Obviously, BIOS or an operating system would prohibit this error by sequentially scheduling access to the printer – but this would result in one task delaying another. Still another task might require a special printer or plotter for output. This is one of the reasons that the total amount of I/O and types of I/O attached to a system can grow when multitasking and multiuser operating systems are introduced.

To receive requests for attention for I/O devices, the AT has 11 interrupt request lines. These signals on the AT card interface cannot be shared between device adapters. This means only 11 different interrupting devices can be configured in an AT system.

In a multitasking system such as the Personal System/2 Model 80, where there are eight connectors and six adapters on every system board, we are talking about 14 functions. Here, eleven interrupts would be insufficient. A redefinition in the AT interface with at least 14 interrupts would be required to solve this problem alone.

Why can't interrupt requests from PC or AT cards be shared? There are two reasons. First, let's understand how the edge-triggered interrupt request mechanism works in systems that support personal computer cards. A transition from low to high on a PC bus interrupt request line is interpreted by the system as a request for attention by an I/O adapter. To create this transition, the request line is first driven low, then high. The drivers on adapter cards may be inactive at other times, but they, too must drive low, then high to get the transition.

A driver with a pull-down transistor and a pull-up transistor is required to drive the interrupt request line quickly from one state to the other. A bipolar or a tri-state type of driver may be used, as long as the high state immediately follows the low state. First the bottom transistor becomes active, "on" like a switch. Then it turns off as the top transistor becomes active, causing the transition.

Assume two such drivers on separate adapter cards are connected to the same interrupt request line. Eventually one adapter will make a transition. While the first adapter is still driving the line high, a second adapter drives the line low as it begins to present an interrupt. This type of situation is much more likely when multiple I/O adapters are operated simultaneously by concurrent tasks in a multitasking or multiuser situation.

During the overlap of the two simultaneous requests, a temporary short circuit across the power supply is created. This situation would also occur if one of the drivers were an open collector driver, as used in level-sensitive interrupt systems. This may cause either one or both of two problems:

1) The interrupt-request line may not achieve a valid level as the two transistors fight to pull it both low and high at the same time. The interrupt may thus be lost to the system. A lost interrupt can be a catastrophe in a multitasking operating system and may require a reset and restart of the system hardware, to regain synchronism between the I/O and the system.

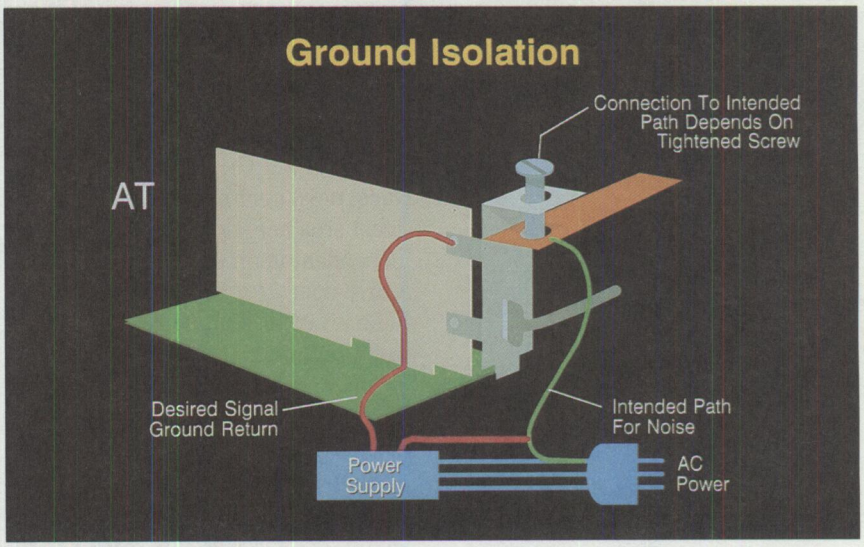



**Figure 8. Ground Isolation**

2) With time, perhaps milliseconds, or perhaps much longer, the drivers may be damaged and the lifetime of the components would be shortened, requiring the replacement of a card or a system board that contains the driver.

A resistor between the card and the bus could limit the current, but would then decrease the noise margin for the logic and aggravate the noise problems in the system.

Schemes have been devised to inhibit one card driver, if an edge is detected from another interrupt driver on the same request line. They may work when two interrupts are unlikely at the same instant in time, but they are also sensitive to noise spikes on the interrupt-request lines. The noise spikes may be detected by many or all such circuits, inhibiting some or all adapters. No interrupt is present, none will be serviced, and the system can hang.

In the opinion of IBM engineers, mixing the drivers on existing PC, PC XT, and AT cards, or mixing existing interrupt drivers with new cards that employ only open-collector drivers, could be detrimental to both data and system integrity, as well as to the reliability of the system. Consequently, we concluded that any system that mixed level-sensitive and edge-triggered interrupts on the same request line may be unreliable for users and very difficult and potentially expensive to support under warranty and service agreements. Even if all the possible warnings are given, normal human errors would still result in unreliable configurations.

A satisfactory design point could be achieved only if all existing PC, PC XT, and AT cards were excluded from an interrupt-sharing mechanism and were defined with an all open-collector, level-sensitive interrupt sharing system. *One of the greatest advantages of such a system would be that it does not support existing PC, PC XT, and AT cards.* Given that all new cards would have to be designed, and a new system would have to be designed to support them, the best technical solution could be chosen without compromise for PC, PC XT, and AT card support.

*The best technical solution could be chosen without compromise.*

## Requirements of the Multitasking Environment

Many other limitations imposed by support of PC, PC XT, and AT card designs could be fixed as well.

**The Single-Tasking Environment:** One limitation is the excessive dependency of most I/O devices on the processor for data movement and control. The displays, all the files, most communications, most local area networks, and printers are dependent upon the AT processor to move data. An 80386™ processor, with all of its complexity and sophistication, is reduced, for many of the operations in the machine, to simply moving data and operating I/O devices. In a single-tasking environment, it may be acceptable to operate each I/O device in sequence and to dedicate the processor to support that device. Such a system may even look very favorable in single-tasking benchmarks. The I/O adapters can be relatively simple and inexpensive to build.

**The Multitasking Environment:** However, in a multitasking environment, all of the interrupt-driven I/O adapters tax the system performance before the applications and operating system instructions can be executed. Before PS/2 systems with Micro Channel architecture were developed, the solution to supporting fast, processor-dependent I/O was typically to build faster processors and faster memory systems – a very cost-ineffective solution.

For example, let's look at what happened with printers. In 1981, an 80-character-per-second printer was considered cost-effective and adequate for a personal computer environment. Today, we have moved to laser printers that can transfer 5,000 bytes per second in graphics mode for desktop publishing. In a DOS environment, this original load of 80 bytes per second on the processor was rather modest. DOS applications consume about 200 instructions to service the interrupt, and to move a byte to the printer. These instructions save the state of the machine, do the housekeeping required to figure where the next print character will come from, operate the interface, and resume processing at the place that it was prior to the interrupt. This requires one interrupt per character in the existing printer adapter. The asynchronous and binary communications adapters, and many other adapters for the personal computer, are of this same basic design, with one character transfer per interrupt. This means a load, under DOS, of 16,000 instructions per second. This is a modest load,

because there are 300,000 instructions per second available in the PC.

The same function today for a single interrupt is a load of 5,000 characters per second, times 200 instructions per character, or 1 million instructions per second – about all of the capability of a 80286™ processor, running at 10 MHz.

Under a multitasking operating system, however, the environment is much much larger and more complex; it can take 1,000 instructions to service the interrupt. This means a 5,000 character-per-second printer will use 5 million instructions per second. That is approximately all of the capability of a 25 MHz, zero-wait-state 80386-based system. So a $10,000 computer is tied up 100 percent of the time when the printer is operating.

**Processor Bottleneck:** A requirement of concurrency is that multiple devices can operate at the same time. If each device depends 100 percent on the processor, then two devices cannot operate at the same time at full performance. One solution to the burden on the processor is to offload that burden to a direct memory access (DMA) controller. Then multiple channels of the DMA controller can each simultaneously support an I/O adapter while the tasks and operating system get maximum access to the processor. This would be the ideal solution. Unfortunately, it is not possible with the AT bus and may be severely limited in a system that supports AT cards.

When the personal computer evolved in 1981 from the original mid-1970s Datamaster product, it implemented a DMA control function. In 1981, that was considered an advantage over competitive designs. The diskette moving data at 50,000 transfers per second was the fastest I/O device. It was assigned to DMA channel 2.

Cards using the DMA controller used the same drivers as interrupt request drivers and consequently could not share a DMA request line.

The connection to the DMA request and acknowledge signals on the PC bus was also hardwired on the card, to assigned DMA channels. The result was that each new DMA adapter design would require that a new DMA channel be defined. Otherwise, an installer would need to understand which card designs, which DMA channels, and which configurations to avoid, to protect the drivers.

*Multiple devices can operate at the same time.*

The SDLC adapter was similarly assigned to DMA channel 1. Dummy-read transfers from memory using DMA channel 0 did the refresh operations. This left only DMA channel 3 available for further implementations. When the PC XT was introduced, DMA channel 3 became the interface for the fixed disk. The PC Network was defined to allow the cost-effective sharing of fixed-disk storage.

It may seem as though a rule has been violated here, where the PC Network and fixed disk share a DMA request line. To avoid problems, the PC Network BIOS turned off the fixed disk when the network was active, preventing two drivers from simultaneous activity on the same line. The DMA request was sequentially shared, and file and network activity became mutually exclusive. This was permissible. Remember that the PC XT was a single-tasking design, like its predecessors, and did not support application concurrency.

When the AT computer was introduced, it could offer concurrency between the fixed disk and PC Network by moving the fixed disk adapter to the processor and using the "string mov" capability of the 80286 processor. The display controller, asynchronous communications, printers, and bisynchronous communications were getting faster and were an increasing burden, but the overhead to support them was still within the capability of the faster 80286 processor.

Networks evolved connections to other networks called "gateways" or "bridges." This implied two network functions in one system. The mechanism of having the PC Network BIOS turn off another device would not work when the other device was the PC Network itself. Either more DMA channels would have to be defined, or the network responsibility would have to be moved to the processor, where the attention could be moved between I/O adapters with interrupts, albeit with considerable software overhead.

The AT computer did both. It defined three new 16-bit DMA channels and moved networks to the processor along with second and third fixed-disk devices, using the power of the processor to support I/O. Yet the processor was now the single element that could move data

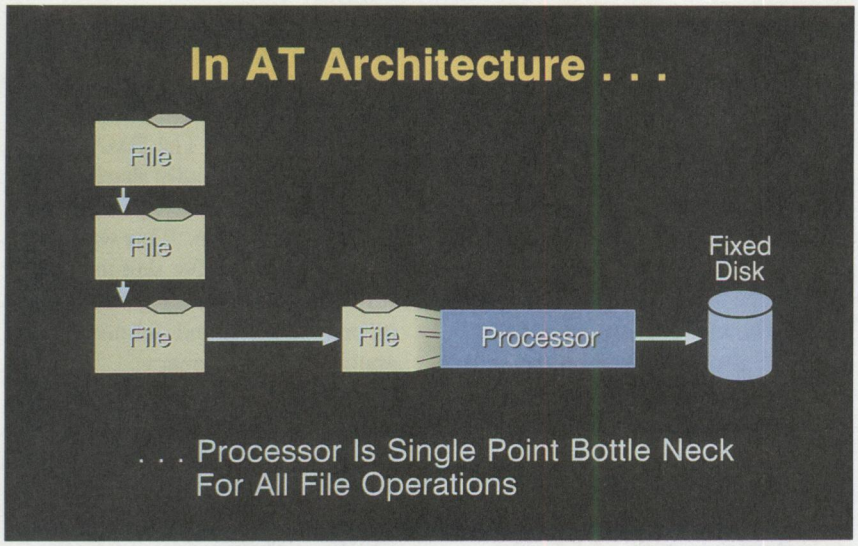



**Figure 9. In AT Architecture**

for most I/O. As long as faster processors and memory systems were cost-effective, the operating system could schedule concurrent tasks as sequential tasks, and the speed of the processor would make up for the fact that all I/O would need to wait in line for the one processor (Figure 9). The AT also removed the memory refresh from DMA channel 0. This left four DMA channels free for use by I/O devices. But for the large part, none of the new channels has been used. Why?

Originally intended to support only sector-oriented files, the new 16-bit channels were incompatible with DOS. The reason is that the DOS file system is byte oriented, and it can place a buffer of data anywhere on an 8-bit boundary in storage. It can create buffers that are even or odd, and they can end or start on even or odd boundaries.

The Intel 8237 DMA module was implemented to count words, not bytes, on the AT and would only allow DMA transfers from even boundaries. This is a characteristic problem of many systems, derived from the AT architecture, that implement the 16-bit DMA ports.

DMA channels 5, 6, and 7, if implemented in those systems, are incompatible with DOS because they cannot place a buffer on any given boundary as defined by the operating system. DMA channel 0, which was freed up in the AT because refresh responsibilities went to dedicated logic, has not been used. Perhaps this is because card designs would have to be specific to the AT, and the AT only, and could not operate on personal computer designs.

To summarize these points, the AT system has three active, or usable, DMA ports for which there exists a business case to build cards that can also be used in the PC and PC XT implementations. This limitation on the ability to move data transfer responsibility from the processor is a major limiting factor in the concurrent environment of a multitasking operating system such as IBM Operating System/2™ (OS/2).

## All Things Change

**The Complete System:** In a concurrent environment, typically a number of applications are operating at the same time. This is the intent of a multitasking operating system – to do more than one thing at a time. It is what can make the Personal System/2 more productive, as much as five times more productive, than previous machines.

We have accepted that a new operating system is required to run these multiple applications concurrently. We have accepted that new processors are required to do multitasking and multiuser operations. The 80286, 80386 and 80486™ processors allow the efficient switching of tasks because they have hardware dedicated for those functions.

Multiple tasks create simultaneous activity to a number of I/O devices. Each application may operate a unique device; the number of devices increases. More types of devices are connected, and the amount of data traffic, or throughput, to and from these devices expands. The number of requests per second for processor attention, interrupts from these devices, grows as well.

The path between I/O and memory, the channel, must be redesigned to reflect the change. Every element of the system – adapters, processor, operating system, applications and the channel – must be redesigned.

These characteristics of concurrent systems are not new. We have seen them in every multitasking and multiuser system that IBM has produced since 1964 (Figure 10).

OS/2 is not the first operating system that we have used for multitasking. In fact, multitasking is at least

25 years old. The problems that micro system designers are finding today are often similar to problems that minicomputer designers saw 15 years ago and mainframe designers saw a decade before that. What *is* new is that the rules have changed and innovative concepts have been applied. Logic can now be more complex; it is faster, less expensive, and far more reliable. The processors are more powerful. Memory is far less expensive. We can be creative and use our experience as well.

**Video Graphics:** The reason that the change can occur now is technology. Figure 11 shows a comparison between an Enhanced Graphics Adapter (EGA) card that was designed around 1984, and a Video Graphics Array (VGA) chip that replaces that function. In fact, the chip package consumes almost 80 percent of the chip's area; only 20 percent of that component is the actual VGA silicon chip. The VGA chip has almost twice the function of the EGA card.

The display function is one of the most frequently upgraded features in a system. Unfortunately, another limitation of systems designed for AT cards is that the investment in the display adapter will probably be discarded within a few years. Why?

When the EGA card was installed, the monochrome and color cards were removed. The original investment was lost as the new EGA function made the old cards obsolete. The user paid again for the monochrome and color functions because half of the EGA logic was dedicated to reproducing the old display modes. AT users upgraded to VGA and paid again for the EGA display function. Each time, approximately one-half of the investment in the new card was to replace

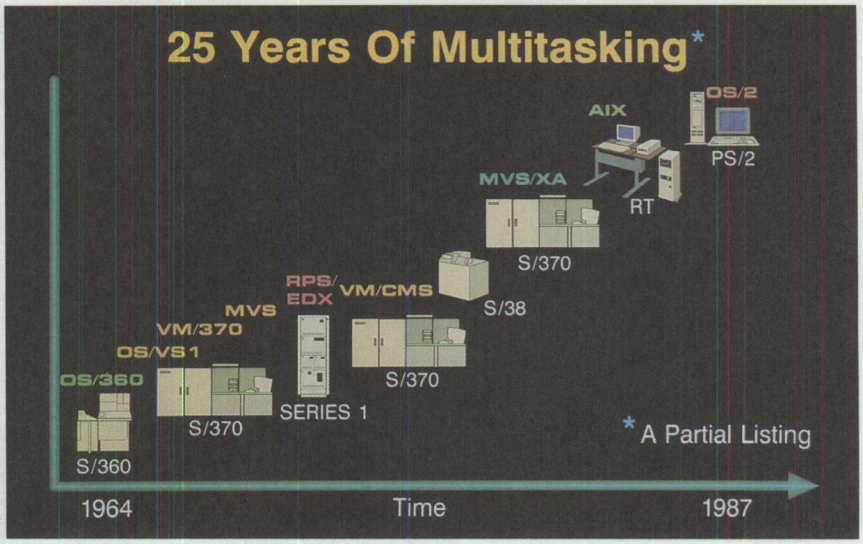



**Figure 10. 25 Years of Multitasking**

the function just removed. Micro Channel architecture puts an end to this loss of investment, as we will see later.

**File Servers:** A file server is a potential single point of failure for what is, in effect, a multiuser system. It operates I/O concurrently, and is an example of the type of system that needs Micro Channel architecture today.

The server will probably grow in its throughput and configuration as more I/O devices and users are connected to the LAN. The desktop will use Micro Channel architecture to run multiple tasks at the user station and to allow network asset management or control of the network configuration. Network asset management allows an electronic census of installed options, prior to distribution of applications, that

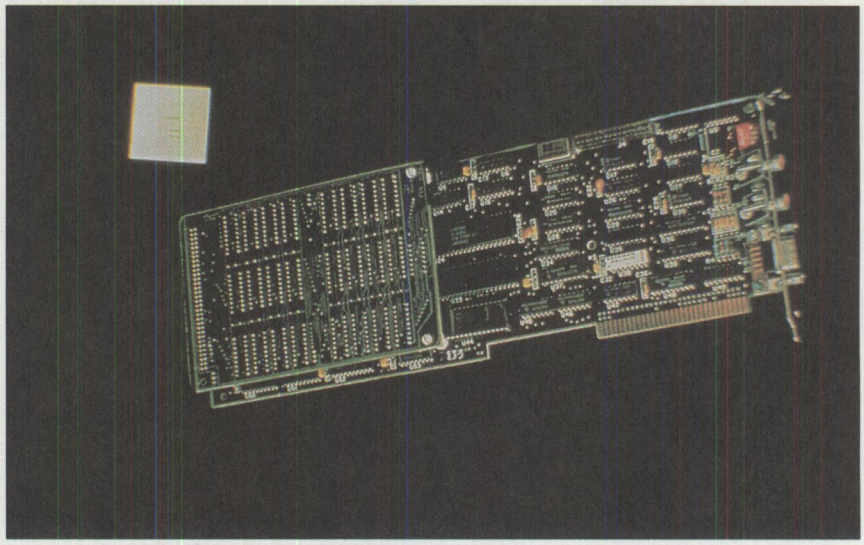


**Figure 11. Video Graphics Technology**

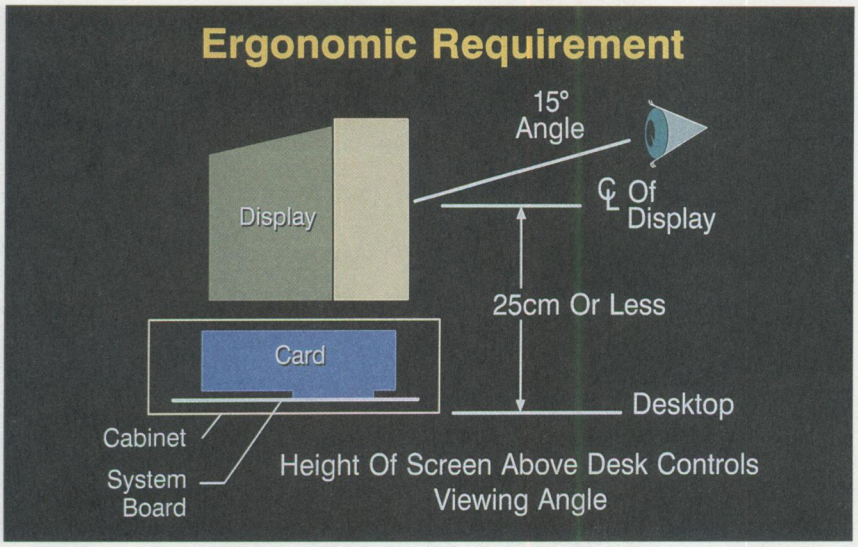



**Figure 12. Ergonomic Requirement**

may be I/O device-dependent. Additionally, the network manager can cycle adapter functions in the LAN stations on and off to accommodate application I/O needs and remote diagnostics. These functions can be supported by specialized software applications in all the stations and the network manager or by the operating system itself.

## A New Foundation

An architecture for the 1990s has 32 bits for both I/O and memory, can be expanded to wider data bus widths, allows higher throughput, and still supports the 80286 and 80386SX™ and other processors with 16-bit interfaces. It is independent of both the processor and operating system. Specifically, IBM's intention was to create a foundation for a broad range of machines, with an architecture that could support single or multiple bus system architectures (for example, the PS/2 Model 70-A21).

**EMC and Signal Quality Characteristics at Higher Speeds:** The channel would be expandable to wider data paths and faster throughput. It would allow advanced error detection, recovery and diagnostics.

To support the faster data transfer, the data quality must be predictable and not lose integrity through radiation.

**Automatic Configurability:** Complex multitasking and multiuser systems would need automatic setup to support conflict-free configuration of multiple I/O interfaces. If IBM had kept switches, resolution of conflicts in configuration would have been very complex indeed. Assume that a system had eight sockets; then one could install as many as eight cards of the same design. That would mean eight I/O address ranges for the adapter control registers. Three switches are needed to select one of eight options.

To provide enough DMA ports to move processor-dependent I/O off the processor, four switches would be required to select one of 15 DMA ports. There are 16 address locations, for "on-card ROM," used for power-on self-test (POST) and BIOS extension – another four switches. Without interrupt sharing, three switches would duplicate the seven "PC" interrupts, but four would be needed to select one of all 11 non-shared AT interrupts. This is a total of 15 switches for each adapter:

```
      3 I/O addresses
      4 DMA channels
      4 On-card ROM/RAM addresses
3 or 4 1 of 11 interrupts
----------
     15 Switches
```

Designing for the worst case, with six adapters defined on the system board and four adapters per combo card in each socket, as many as 570 switches would be required for foolproof, switchless set-up on the AT computer:

```
(6 x 15) System board
    +
 4 x 15 Each card (combo)
   x  8 Cards
----------
    570 Switches
```

This is beyond the average person's clerical ability and patience, but well within the capability of the computer. Programmable Option Select (POS) can replace 32 equivalent switches (expandable to 256,000 switches) and turns what would have been an ordeal for a human into a minor task for the computer.

**Smaller Footprint for Desktop:** The smaller card can yield a smaller system, and also can assist in the usability of the system. In several countries, industry regulations and public laws define certain ergonomic characteristics of data entry terminals and personal sys-

tems. One part of these requirements is the placement of the centerline of the display no more than 25 centimeters above the desktop to reduce glare and neck-muscle fatigue (Figure 12). With the common practice of pancake-like stacking of the display and system unit to conserve desk space, the designer has a choice of a smaller screen or a smaller card.

Micro Channel cards are 3.5 inches high, the same height as the connectors to peripherals on the rear bracket of PC, PC XT, and AT cards. PS/2 systems, therefore, maintain connection to most AT peripherals while permitting larger-size displays to be installed where ergonomics is a consideration.

However, Micro Channel architecture does not restrict the card height to 3.5 inches. While this is a specification for existing PS/2 computers, systems with capability for larger cards are feasible.

**Increased Flexibility for Interrupt Structure:** The AT interrupt structure had to be improved. Micro Channel systems support up to 256 adapters to share a common interrupt-request line. With 11 compatible interrupt-request lines, this allows more than 2,800 device adapters to be installed in a system:

256 x 11 = 2816 Interrupting adapters

Certainly this is enough for an advanced microsystem; it is also enough for minicomputers and mainframes as well.

The I/O device's request for attention is interlocked with the system's response to the request. Lost or spurious interrupts can be detected, and diagnostics can locate defective adapters and/or interrupt routines more easily.

**Increased Data Rates:** The Micro Channel interface supports a base burst speed, in its initial definition, of 20 million byte transfers per second. Even other microcomputer manufacturers specify the AT bus at no greater than 3.5 million byte transfers per second. There have been implementations of the Micro Channel standard that move data at a rate of approximately 17 million byte transfers per second, or on the order of four times the capability of the AT bus.

IBM has defined all the timing, loading, capacitance, and electromagnetic characteristics of the system so that designers building cards, systems, and applications to the Micro Channel specification can define equipment that can be integrated properly.

The performance of the Micro Channel bus can be extended well beyond its present level. We have preserved the ability to extend Micro Channel functionality, in a compatible fashion, so that cards developed today to Micro Channel specifications can run on Micro Channel systems tomorrow.

**Expanded DMA and Bus Masters for Concurrency:** We have expanded the DMA and bus master capability so that a number of devices can be intelligent and no longer be dependent upon the processor. Up to 15 devices can be supported independently of the processor.

## Building In the Future

To see how this has been done, look at a physical comparison of an AT card and a Micro Channel card (Figure 13). On initial inspection, the Micro Channel card is physically smaller. This allows for a smaller system. However, the card format is optimized for surface-mount technology (SMT). The connector dimensions match the tab spacing on SMT components, facilitating wiring of the card.

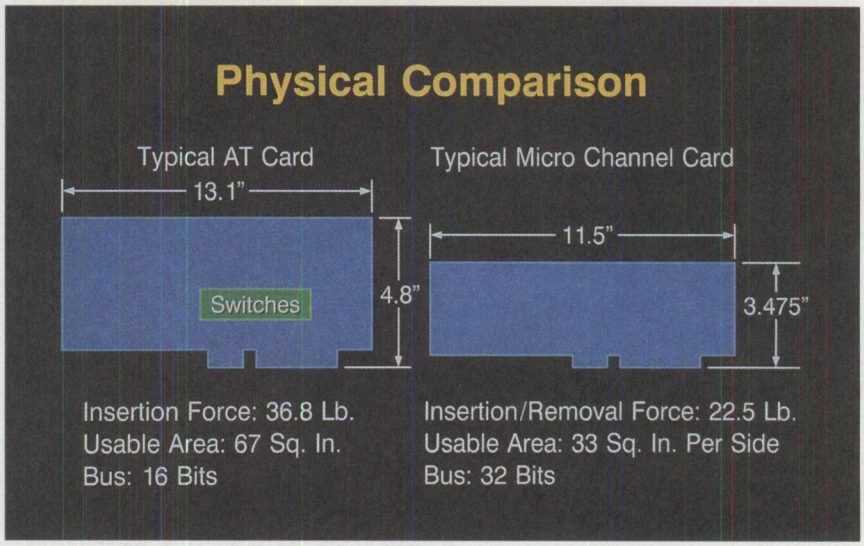



**Figure 13. Physical Comparison**

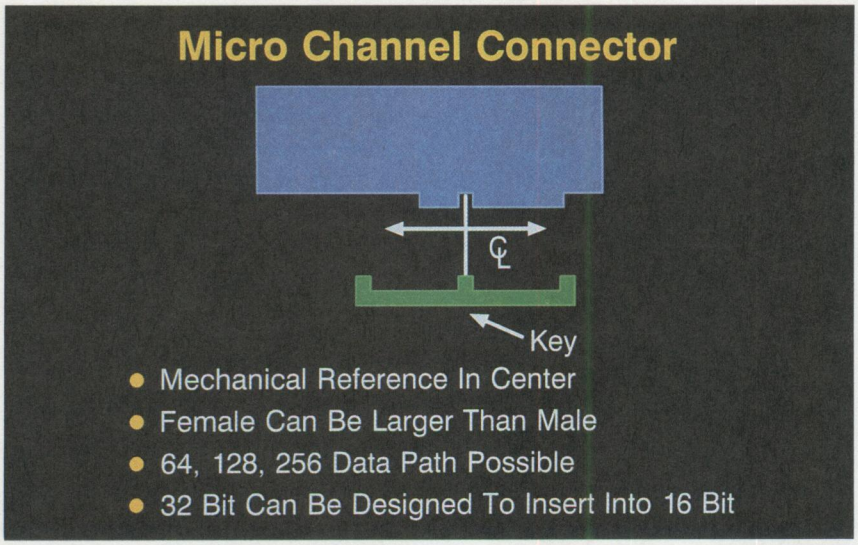



**Figure 14. Micro Channel Connector**

SMT allows an increase in the vertical and horizontal densities, each by a factor of 2-to-1, for an overall aerial density improvement of 4-to-1. Surface mounting also allows the implementation of components on both sides of the board for a total improvement of 8-to-1 in component density.

The Micro Channel connector uniquely references mechanical alignment from a keyway centered in the connector, allowing tighter control of mechanical tolerances (Figure 14). When inserted in a 32-bit socket, neither end of a 16-bit male connector is in contact with the end of the female socket. Likewise, a 16- or 32-bit card could be inserted into a 64-bit or larger socket and align perfectly. A 32-bit card has been designed that leaves clearance at the end bulkhead position for a 16-bit connector. Therefore, this 32-bit design can be used in 16- or 32-bit or larger systems. A single design can be made for all systems and still be capable of 32-bit throughput.

The spacings on the connector have been cut to one-half of the spacings used on the AT connector. A Micro Channel 16-bit socket is approximately the same size as a PC 8-bit socket. The Micro Channel 32-bit connector is actually smaller than the AT 16-bit connector. This follows the trend of other components in the system, such as processors that have grown in power from the 8-bit interface of the 8088 processor to the 16-bit 80286 and the 32-bit 80386 without growing significantly in physical size.

The number of signals in the connector has been doubled, and the insertion and removal forces have been cut almost in half. This is extremely important because in manufacturing and service situations, had the length of the connector increased or more tabs been defined, either the insertion force or removal force would have exceeded 40 pounds.

We have designed out the electromagnetic compatibility problems of the AT connector. One can see by the pattern in the lower half of Figure 15 that each signal in the Micro Channel bus is situated between a ground, across from a ground, or adjacent to a ground signal. There is a radio frequency shield on every fourth pin. On the opposite side of the connector, the pattern is identical but offset by two pin spacings. The area between the closest shield and a signal is thereby diminished significantly.

The loop area of the Micro Channel bus is one-twenty-eighth of that for the AT bus. The signal transitions can then increase in frequency, by a factor of the square root of 28, or approximately five times (Figure 16). This has been experimentally verified outside of IBM – studies of the Micro Channel bus have shown a cutoff frequency above 80 MHz. Studies of the AT bus have shown a cut-off frequency of approximately 16 MHz, or again the same factor of 5-to-1.

Micro Channel cards now make positive ground contact with an intended path for ground return in the AC power lines. This occurs when fingers attached to the shield bracket on the back of the connector make direct contact on insertion.

The card itself is isolated electrically, so that currents on the card cannot escape that card through the shield system and must return directly to the ground of the power supply. External static discharge currents will be returned through the intended path to the AC power system, not through the card itself.

**More DMA:** Remember the problems with DMA utilization on AT cards? Only one channel on the AT was usable. The DMA channels that were added to the AT were not compatible with DOS. Micro Channel architecture has solved these problems. Instead of three

usable DMA channels supporting just three permanently assigned devices, we can support up to 15 devices with no fixed assignments.

All channels are compatible with DOS. Micro Channel DMA controllers can handle the even transfers (as does the AT) as well as all data formats involving odd transfers (those beginning or ending on even and odd boundaries). The DMA controller automatically adjusts the address boundaries upon transfer.

Processor-dependent I/O can now move to DMA because usable DMA channels are available; no assignments are fixed. Data can move faster. Single-transfer operations are employed for compatibility with the AT operations, but we also defined a burst capability in the Micro Channel interface that allows 312 transfers to occur with the same arbitration overhead that normally would have allowed only one data transfer in the AT.

**More Designs:** Existing designs for the AT system can be migrated to the Micro Channel standard because all of the AT signals can be derived from the Micro Channel interface. However, new functions not possible in the AT can also be defined.

Potentially, more Micro Channel adapter designs are possible. Recall that, on the majority of the cards designed for PC, PC XT, and AT systems, only 10 of the 16 I/O address lines defined by the processor are decoded. This means that only 1,024 registers can be selected by the processor instead of the 65,536 defined by the full 16-bit I/O address.

Even the most rudimentary adapter will define three registers – one for commands to the attached device, another to transmit data to or from the device, and one last register to read status. If these registers are only 8 bits wide, then a maximum of 341 unique designs could be defined for a system that could install PC, PC XT, and AT cards with 10-bit decodes.

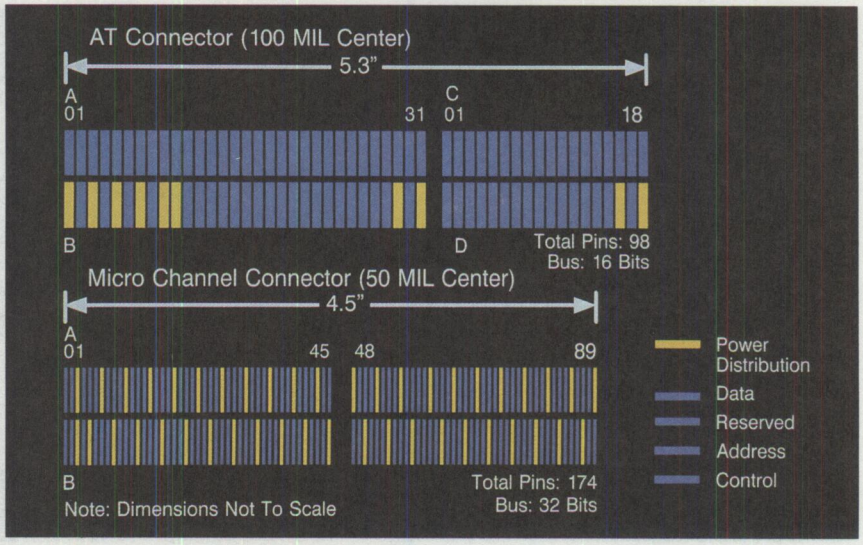



**Figure 15. Card Connectors**

However, all cards complying with the Micro Channel specification fully decode the 16-bit address, so that theoretically about 22,000 unique designs are possible for Micro Channel systems.

**Faster DMA:** Micro Channel architecture also defines a serial DMA capability, much like the "string mov" operation in some Intel processors. Separate read and

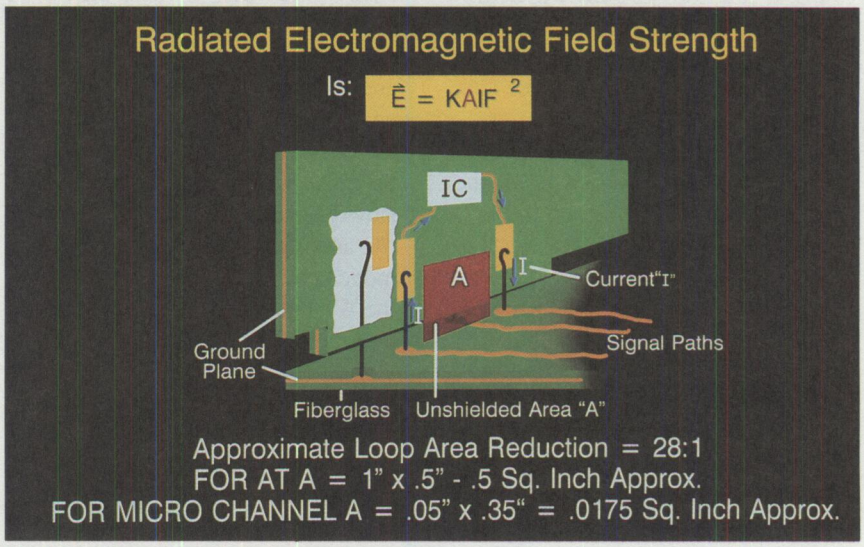



**Figure 16. Radiated Electromagnetic Field Strength**

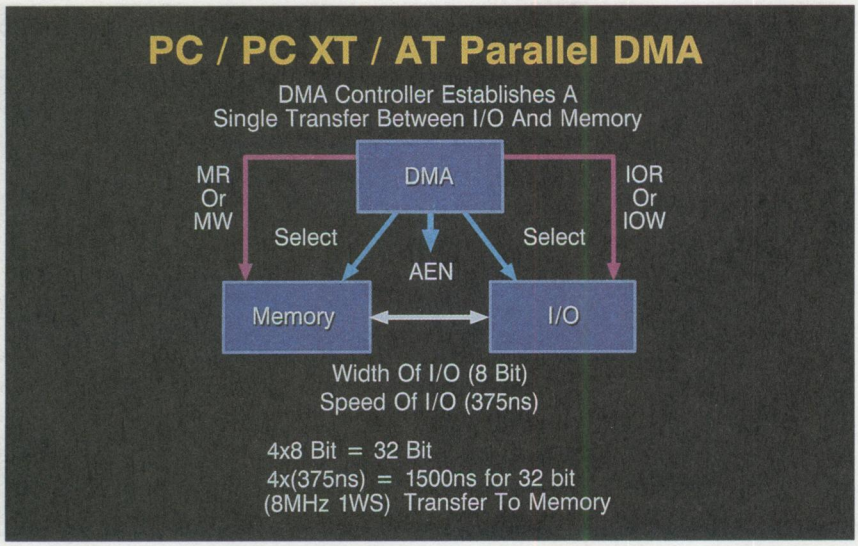



**Figure 17. PC/XT/AT Parallel DMA**

write operations allow memory-to-memory operations, I/O caching, and inline processing to be defined in the DMA controller. Yet, on first inspection, serial DMA might appear to be slower than the parallel DMA, defined for PC, PC XT, and AT implementations, that moves data with just one operation.

First, let's understand the way the PC, PC XT, and AT handle a DMA transfer (Figure 17). The PC DMA controller does not actually transfer data. It is a third party that arranges a transfer out from a source (for example, memory) and into a destination (in this example, an I/O adapter). The transfers occur as one operation, in parallel, hence the name "parallel DMA."

The parallel transfer begins when the DMA controller selects the memory by address, selects the I/O adapter by a signal called "DMA acknowledge," and asserts a signal called "AEN." AEN says to all the non-DMA adapters in the system, "I will drive two data strobes at once," in this case, memory write and I/O read. Only the parties in the DMA transfer respond, and data moves out of memory, across the data bus, and into the I/O adapter, all as one operation.

If everything occurs as one operation, then the width of the transfer must be the width of the narrowest party. That is, an 8-bit I/O device could not receive 32 bits of data from a 32-bit memory in one operation; it would take four operations, each 8 bits wide, to move the 32-bit value. The most common I/O width is 8 bits, and today most systems are being defined for 32-bit memory width, so this is a realistic example. Likewise, the duration of the transfer must be the length of the slowest party. Here again, it is the I/O adapter on the AT bus that moves data at the default speed of the AT bus – 375 nanoseconds (ns), or 375 billionths of a second. It will take four 375 ns, 8-bit-wide transfers to move a 32-bit quantity of data from memory to I/O, or 1500 ns. The memory and the I/O are both delayed 1500 ns to complete the four operations.

Now let's understand serial DMA as it is used in the "string mov" operation of the 80286, 80386, and 80486 processors and by Micro Channel DMA as well (Figure 18). In a Micro Channel DMA controller, a register is directly involved in the transfer. Data is read into the register in one or more operations and written to a destination with one or more operations. The transfers occur sequentially, hence the name "serial DMA."

Here again, four 8-bit transfers would be required to move a 32-bit value from an 8-bit I/O adapter to the DMA controller, and at least one more to read the data out. There are still more operations than with PC-, PC XT-, and AT-style parallel DMA, but they occur much faster on the Micro Channel interface. The four 8-bit transfers can occur at the default speed of the Micro Channel interface, 200 ns each. The total time is 800 ns.

The transfer with the I/O adapter was at its optimal speed, and the transfer with memory can be at the optimal speed of a 32-bit cached storage (as used in the Model 70-A21). Assume an average duration of 50 ns. If the operations occur on one bus, the total duration is 850 ns, or faster by just slightly over half the time of the AT bus. If the memory is on a separate bus, the I/O transfers do not impede its operation, and the serial DMA transfer with memory will occur 30 times faster than the AT-style parallel mode.

**Greater Latitude for System Design:** Whether a computer is designed with one bus or more than one bus is a function of the system organization, but not of the channel architecture. Cost-effective Micro Channel systems can be defined

with all the memory and I/O on one bus, as in the PS/2 Models 50 and 60. In addition, systems with the memory on one bus and the I/O on another, such as the PS/2 Model 70-A21, can also be defined with the same Micro Channel specification. It is important that high-function, advanced architecture does not imply higher complexity and cost – and it does not with Micro Channel architecture. The business case for card development is built on the large number of sockets installed for a wide range of products and not on the limited number of sockets for high-end systems alone.

One thing that allows this flexibility is that each of the parties in the system moves data at its individual optimal rate rather than in lock step with a system processor. Asynchronous transfer allows the straightforward Micro Channel implementation of processors of different types and speeds than the one that may have been soldered to the system board at the time of purchase.

**Full Specifications:** A full specification of all channel timings, current and capacitance loadings, mechanical and electrical considerations, and full definition of the functions are available in the *IBM Personal System/2 Hardware Interface Technical Reference* manual for the PS/2 Models 50, 60, 70386 and 80386 computers.

*Note:* IBM regularly holds education sessions and offers design assistance to adapter card developers.

**Micro Channel Reliability:** The Micro Channel interface has significantly improved the reliability, availability and serviceability of the system unit as well. Micro Channel design assists in the implementation

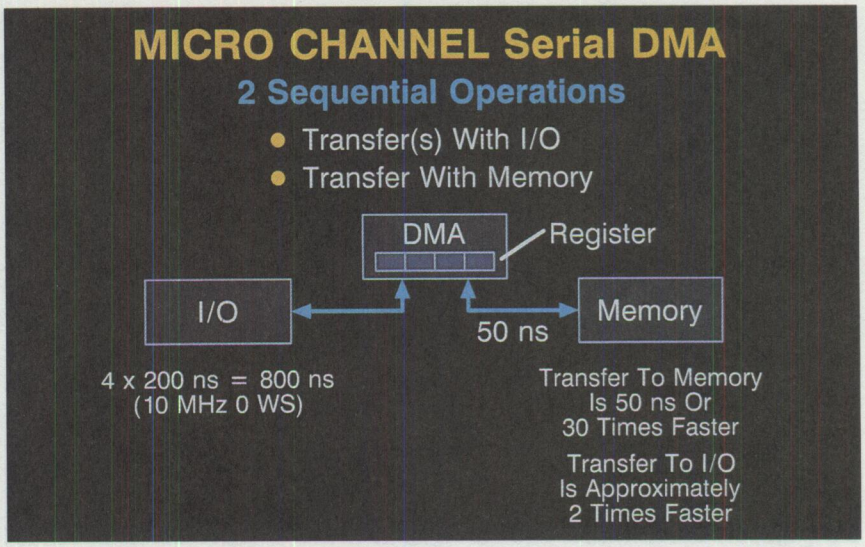



**Figure 18. Micro Channel Serial DMA**

of integrated circuits (ICs). IC designers most often run out of pins in the package before the silicon circuitry inside is completely used. More information is packed into fewer signals with the Micro Channel interface, and fewer pins are required to define the same function in an IC. A larger portion of the circuity in a system can be included inside the more reliable silicon chip, and the reliability of the entire system increases.

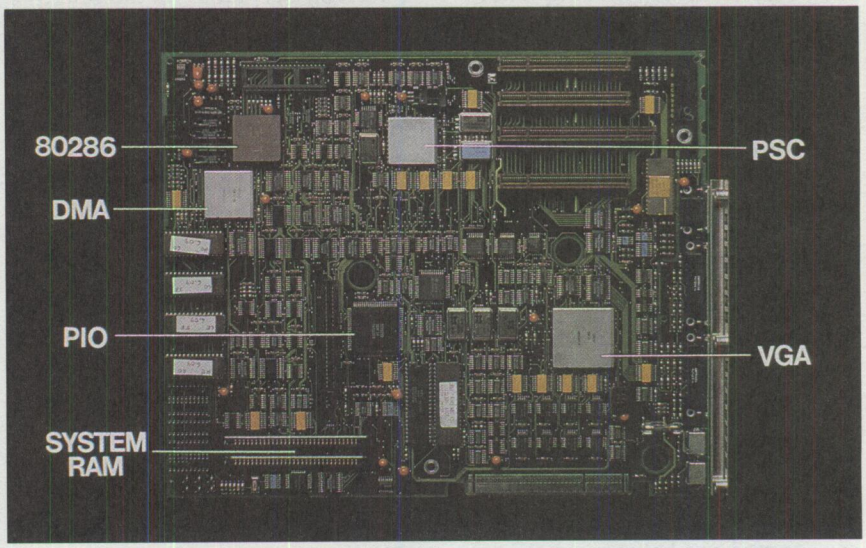



**Figure 19. PS/2 Model 50 System Board**
**80286: Intel 80286 Processor**
**VGA: Video Graphics Array**
**DMA: Direct Memory Access Chip**
**PSC: Processor Support Chip**
**PIO: Planar (System Board) I/O Chip**

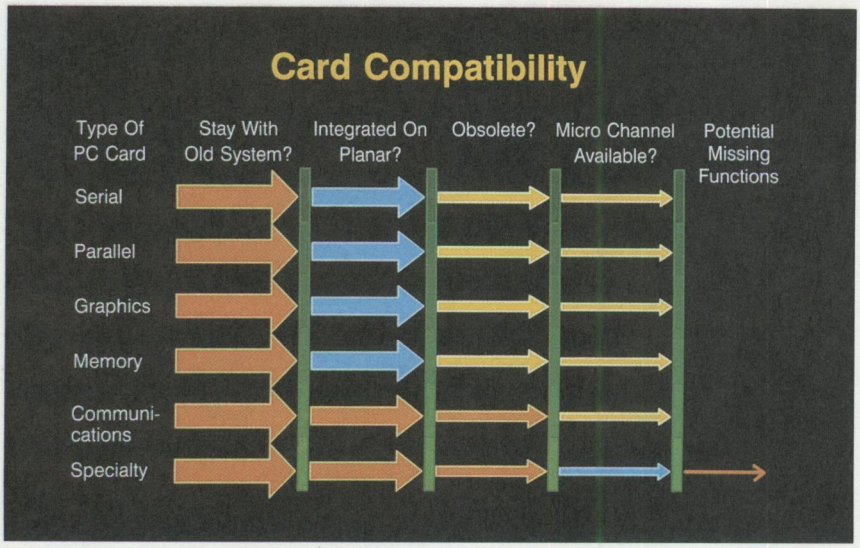



**Figure 20. Card Compatibility**

Because the Personal System/2 Model 50 (Figure 19) uses integrated circuits, it has one-eighth of the board area of the AT 339 that it replaces, is 25 percent faster, and includes six I/O adapters on the planar. It has a mean time between failure (an index of reliability) that is two to three times better than that of the AT 339 system, and has a service cost that is approximately one-third that of the AT 339. It also consumes only 94 watts, or half the power of most AT-based systems. These are all advantages to the customer.

Micro Channel architecture was planned from the ground up to support tomorrow's applications. It has the data integrity and reliability required of a system that multiple users may depend on.

**Apples and Oranges:** We often see comparisons between what Micro Channel architecture was in its first 16-bit implementation, and the maximum configuration of what implementation of another architecture might offer in the future. The AT computer might be extendable, just as the Micro Channel computer will be expanded. While we will not describe what extensions will be made to Micro Channel architecture here, one comparison between today's implementations is often not recognized: the DMA transfer rate for 16-bit operations on the Micro Channel bus is often 10 times the DMA transfer rate of AT systems that employ 5 MHz Intel 8237-compatible DMA controllers.

Compare the what-might-be cases – for example, what if the AT and Micro Channel DMA controllers each were implemented as bus masters. In both cases, the maximum burst rate of the Micro Channel bus would be nearly four times that of the AT bus – with both buses operating at default rates. This is because a system that exchanges data with existing AT cards could move two bytes in 375 ns. The Micro Channel interface defines 4-byte transfers between master and slave in 200 ns. The 32-bit SCSI interface bus master has been implemented and demonstrated with 4-byte transfers every 240 ns, or nearly 17 million bytes per second. This approximates the calculated continuous rate of 18.7 MB. That is approximately five times the rated speed of the AT bus as given in other manufacturers' specifications.

As for claims that lack of AT card compatibility limits a system designer's options, many of the cards in systems today must stay right where they are because the diskette, fixed disk, display, and memory cards are matched to the data width, speed, and peripheral interfaces of those systems (Figure 20). If the cards are removed, the system peripherals will not work.

Most of these interfaces have been upgraded. Where they are implemented on the PS/2 system board, no card need be purchased at all. Furthermore, many of the communications and other functions, purchased for the old system when it was new, are now obsolete, and more current designs are now available for the PS/2 Micro Channel computers. It is important to remember that all Micro Channel cards are current – with none made before April 1987.

**Card Portability:** Studies have shown that it is indeed rare for a user to "cannibalize" a system to recover even a few cards for installation in a new system. Businesses need the most current and productive systems to be competitive. In the workplace, the old system is typically given to a new user whose needs are met by the capability of the old system. Almost 98 percent of the time, all the cards in a system remain in that system. Little more than 2 percent of the cards are brought forward (Figure 21).

LAN cards are the cards most frequently drafted into new duty. It is

important to note that this data was obtained before Ethernet® LAN adapters were widely available for Micro Channel systems. Today there are several Micro Channel Ethernet bus master cards, and as many slave adapters, available. There is a similar situation for 3278 emulation boards. Token-Ring LAN has recently gone through a generational change from 4-megabit to 16-megabit interface as well.

Today there are over 1,000 Micro Channel adapter cards available. (*See page 103 for catalog of adapters.*) IBM has issued more than 2,700 design IDs worldwide, indicating that as many as 1,700 more cards are under development. More than 650 vendors have the Micro Channel standard in their plans today. Of the 1,000 cards, 27 are advanced bus master cards; 197 bus master IDs have been assigned. Indeed, this is a better story for Micro Channel advanced I/O card designs than for AT cards.

## Dollars and Common Sense

*Editor's note: Because this is a reprinted 1989 article, the cost comparisons are 1989 values.*

It's easy to get excited about advanced technology, but dollars and common sense are what influence most business decisions. The reason to buy Micro Channel architecture is that it comes for free, in the highest-quality personal systems available today - the PS/2 Models 50 and above. They are also among the least expensive to own!

Assume you intend to own your new system for five years. Let's analyze the cost of ownership. We'll use the AT 339 system for comparison, but data for any other AT-bus machine can be used instead.

During the first year, both systems are under warranty, so the service cost is identical – zero. Based on IBM's annual onsite maintenance charges as of July 1, 1989, years 2 through 5 show that the Micro Channel system has a much lower IBM service cost. This is a reflection of higher reliability and more efficient diagnostics, as well as a reduction in "no trouble found" service calls due to mis-set switches.

Diagnostics in Micro Channel systems can help prevent costly and unnecessary replacement of functional components due to a greater ability to accurately isolate and replace only the failed components; this can further reduce the service cost.

When you upgrade the system display, you do not need to pay a second time for the VGA function as you would have to in an AT system. This is because both the input and output of the display controller are connected to the Micro Channel interface. Cards that extend the display capabilities do not need to discard the existing controller or replace the function on the extension card. This can save the original VGA investment that need not be added to cost of the extension display adapter. This amounts to a savings of several hundred dollars.

PS/2 system users also won't have to spend as much on electric power. Check the name plate power rating on the back of most AT-based systems. The AT 339 is 192 watts, and, fully configured, it can cost you $650 to run it five years for 24 hours per day, 292 days per year, at 10 cents per kilowatt hour. Adjust this number if you use the machine more or less, or if your local power costs more or less.

While some may say that the lower power allotted to adapter designs in Micro Channel systems makes the designer's job more challenging, those challenges can be overcome by large scale integration (LSI) and computer-aided design techniques that reduce size and power consump-

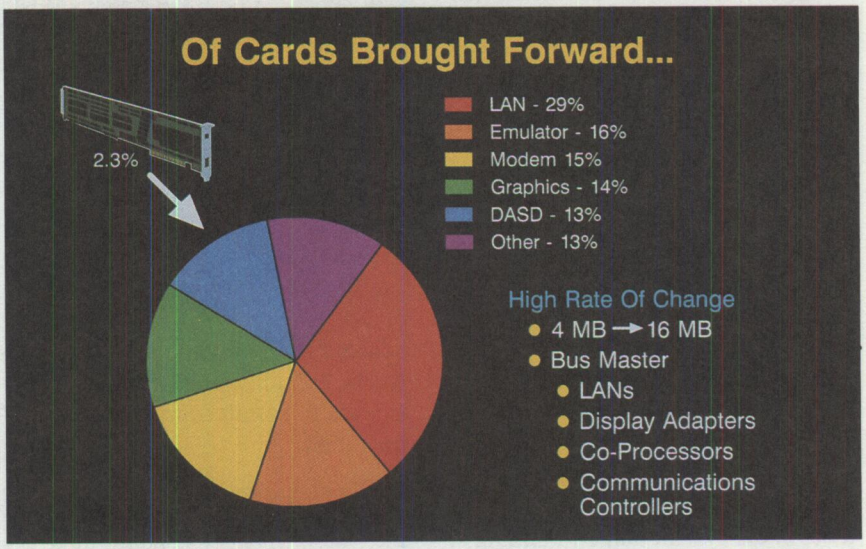



**Figure 21. Cards Brought Forward**

tion and make the design less expensive to reproduce and more reliable in operation. The reduced power in PS/2 Micro Channel systems is an advantage to the customer. The name plate rating for a maximally configured PS/2 Model 50 Z is 94 watts, about one half that of the AT system it replaced. The result: you'll save about $325 in electric bills. (The display also uses electricity but is not included in this system unit comparison.)

Add up all the savings you can demonstrate, and then consider comparative depreciation in the resale value of the system, and the value of increased productivity from a system that is more available and needs less maintenance time. You'll find you save a sizable fraction of the purchase price. Your calculations would look something like this:

On-Site Maintenance:

| Year | AT 339 | PS/2 50 Z | You Save |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 2 | 442 | 192 | 250 |
| 3 | 442 | 192 | 250 |
| 4 | 442 | 192 | 250 |
| 5 | 442 | 192 | 250 |
| Display Upgrade | 400 | 0 | 400 |
| Electricity | 650 | 325 | 325 |
| Total | | | $1,725 |

In fact, if you keep the system a few more years, it will have been free – the savings in ownership will have paid for the original purchase.

The moral is to look past the cost of purchase to the real cost of ownership before you decide which system offers the lowest overall cost. This is why Micro Channel architecture also stands for "Maintain the Customer's Assets" – maintain the asset in the system unit by building the future in and "designing in" quality, and maintain the asset in the feature cards by assisting in the portability of those functions across a broad range of machines.

## Evolution Follows Revolution

The future will see the Micro Channel standard grow and evolve gracefully. New functions are already defined for advanced systems. They will include further improvement in the ability of the system to detect, in some cases to recover from, and perhaps even to correct, errors.

Although implementations of the Micro Channel interface have already defined data transfers at 32 megabytes per second, the demands of multitasking and multiuser systems will further increase throughput requirements. By adjustments in the efficiency of transfer protocols and wider-width data transfers, we are confident that the throughput can be significantly increased over time.

The initial Micro Channel specification defined 32-bit DMA, and it will appear in future systems. Most exciting of all, control block architectures, not unlike those in use on IBM minicomputers and mainframes, are enabled by the Micro Channel bus master definition. Looking many years ahead, the future has been built into these systems by the definition of far-reaching standards – standards that help complex systems to be defined and sourced from multiple manufacturers, and integrated with confidence of sustained trouble-free operation.

A large infrastructure now supports the Micro Channel standard interface, and that trained design base is growing. Micro Channel architecture can grow into the future because it is part of a system of standards, and a vision that does the same with the documented interfaces for Advanced BIOS and Systems Application Architecture™ (SAA™), the operating system's interface to applications.

Micro Channel architecture is simply an element of a structure. Philosophically it is not just an extension of the now dated PC bus for advanced microcomputer systems. It is indeed the opposite – Micro Channel architecture incorporates mainframe architecture principles, innovatively made cost-effective and compatible for advanced microsystems.

*ABOUT THE AUTHOR*

*Chet Heath is a senior engineer at IBM's Entry Systems Division laboratory in his twentieth year with IBM. He holds a bachelor of science degree in electrical engineering from the New Jersey Institute of Technology and a master of science degree in electrical engineering from the IBM LSI Institute at the University of Vermont. He refers to himself as "the oldest living survivor of Micro Channel architecture" because he was involved in its definition since inception and gave the architecture its name. He has received IBM Quality, Outstanding Technical Achievement, Outstanding Innovation, and Corporate Technical Achievement Awards. Chet also has attained the eighth level of invention awards. He is presently assigned to development of Micro Channel strategic enhancements.*

# Micro Channel System Configuration Considerations

*Carl Grant, Don Ingerman and Robert Robinson*
*IBM Corporation*
*Boca Raton, Florida*

**This article addresses Micro Channel system configuration considerations to enable those who are responsible for selection of Micro Channel products to understand the basic systems and adapter structures, the basic system performance parameters, and the basic configuration considerations to make intelligent choices in the selection of Micro Channel systems and feature adapters.**

The capabilities of personal computers have increased significantly over the last several years. During this same period, there has been a dramatic change in the environments in which personal computers are used.

When first introduced, the personal computer was exactly that, a computer intended for the personal use of a single user. Response times were related to the perception of the person sitting at the keyboard. Word processing, for a single user, was one of the primary applications. In this environment, there was little, if any, need for overlapping operations. The input/output (I/O) for these systems was normally handled by what is called programmed I/O. In this mode, when an operation was initiated to or from an I/O adapter (for example, a keyboard, display or file), a special program took control of the system processor to handle the movement of the data to and from the adapter. The application program in progress was suspended until the I/O operation was completed. Thus, the I/O operation and the application execution were serial, not overlapping. This type of structure is called single-tasking.

It soon became apparent that some overlapping of operations would improve the productivity of the system and the user. An example of this is the ability to print one file while updating a second file. One way to support this type of operation is to restructure the software running on the system so that it can support multiple tasks running at the same time. This type of structure is called multitasking. More efficient handling of I/O operations was needed to realize the full benefits of the multitasking operating environment. A new method of controlling I/O, direct memory access (DMA), which frees the system processor from being involved in the movement of data, was introduced.

New uses for personal computers have rapidly developed due to improved performance and capabilities of the new systems and the availability of multitasking operating systems. Personal computers are now commonly used as file servers, print servers, network nodes and in other applications that require the simultaneous support of multiple users. This type of operation is called multiuser. An increased emphasis on the ability of the overall system to provide a consistent level of satisfactory performance for all users is now required. This in turn has resulted in further changes in the interface between the system processor and its attached I/O adapters. To reduce the amount

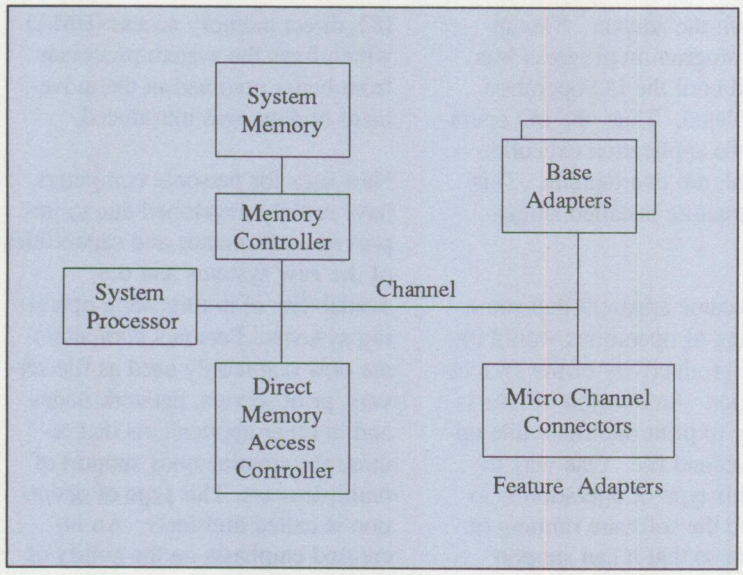



**Figure 1. System Hardware**

of time that the I/O adapter is using the I/O bus, the data bus width has been increased and the data transfer time has been decreased. More intelligence is incorporated in the adapters, further reducing the amount of control that is required from the system processor. Micro Channel architecture allows the adapter to become the bus master. In this mode, once the adapter has gained control of the channel, it controls all of the activity associated with the data transfer, without impacting the system processor.

New systems structures have been introduced that improve overall performance in the multitasking environments. An example of this is the memory cache that has been implemented (in some systems) between the system memory and the system processor. The addition of this memory cache reduces the potential contention between the system processor and the adapters for access to system memory. This results in improved performance by the system processor and by the adapters attached to the system.

These new levels of sophistication in personal computers make it more important to select the proper components to obtain a balanced and efficient system. There are a large number of adapters available for use on Micro Channel systems. Many of these adapters perform similar functions. In order to select the best adapter for a given system and its applications, not only the function provided by an adapter must be considered, but also how that adapter will perform in the total system.

The purpose of this article is to present a discussion of the considerations that are necessary in selecting components for a Micro Channel system. These include system hardware and feature adapters.

## Micro Channel System Hardware

The Micro Channel system hardware consists of the:

- System processor
- Channel
- Memory subsystem, which includes the memory controller and the system memory
- Micro Channel connectors
- Direct memory access (DMA) controller
- Adapters
- System board
- Support logic required for basic operation that is furnished with the system board

This structure is shown in Figure 1.

### System Structure and Components

The basic system structure consists of the:

- System processor
- Processor bus
- Memory bus
- I/O bus, called the *channel*
- System memory

This structure is shown in Figure 2.

The processor bus, the memory bus and the channel are the same logical path. Therefore, both the system processor, for instruction and data fetching, and the adapters, for data transfers, use the same resource.

An enhanced system structure has been introduced to improve overall

system performance. This enhanced system structure places a high-speed memory, called a *memory cache*, between the system processor and the system board memory. Figure 3 shows that for this structure, the memory cache is connected to the system processor by the processor bus, and to the system memory by the memory bus. The processor bus is isolated from the memory bus/channel.

When instructions or data are fetched from the memory cache, called a *cache hit*, the processor bus is used and transfers can be occurring simultaneously on the memory bus/channel. When the instructions or data required by the system processor are not in the memory cache, called a *cache miss*, the information must then be fetched from system memory using the memory bus. When a cache miss occurs, the system processor may contend with an adapter data transfer for the memory bus/channel.

The enhanced system structure offers two advantages. The first advantage is that when instructions or data are fetched from the memory cache, it takes less time to fetch this information because of the faster memory access times. The second advantage is that when information is fetched from the memory cache, the memory bus/channel is not used, and an adapter data transfer can overlap the system processor instruction execution. When there is a cache hit, the potential for contention with an adapter data transfer is eliminated.

## System Memory

System memory is memory addressable by the system processor memory address space. The system software and data structures are resident in system memory. The system memory furnished with the system is resident on the system board.

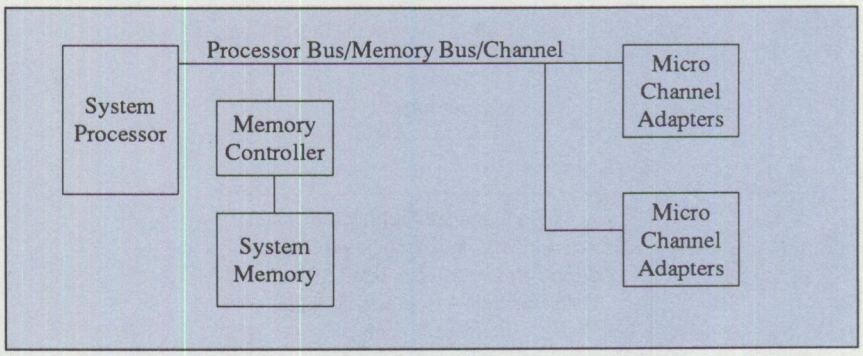



**Figure 2. Basic System Structure**

In an IBM PS/2 computer system, the system processor memory address space and the Micro Channel memory address space are a single address space. This allows system memory to be resident on an adapter that occupies a Micro Channel connector. System memory that resides on an adapter is called Micro Channel system memory. This offers the option to use Micro Channel system memory to expand, or to upgrade, system memory beyond the capability provided by system board memory.

## Adapters

Adapters control the additional functions that are needed to give the system the desired functional capability. These adapters are known as base adapters, if they are furnished as part of the system, or feature adapters, if they are purchased separately. Base adapters enhance the basic system with generally required functions (for example, video graphics array, serial port, parallel port, keyboard controller, diskette controller, fixed file adapter). Base adapters are either on the system board or in a Micro Channel connector. Feature adapters are inserted into Micro Channel connectors. Figure 1 shows base adapters and feature adapters in relation to the rest of the system

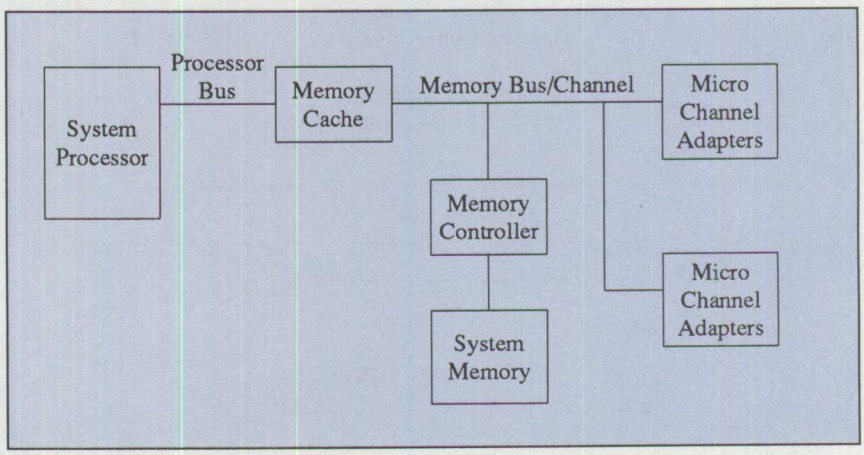



**Figure 3. Enhanced System Structure**

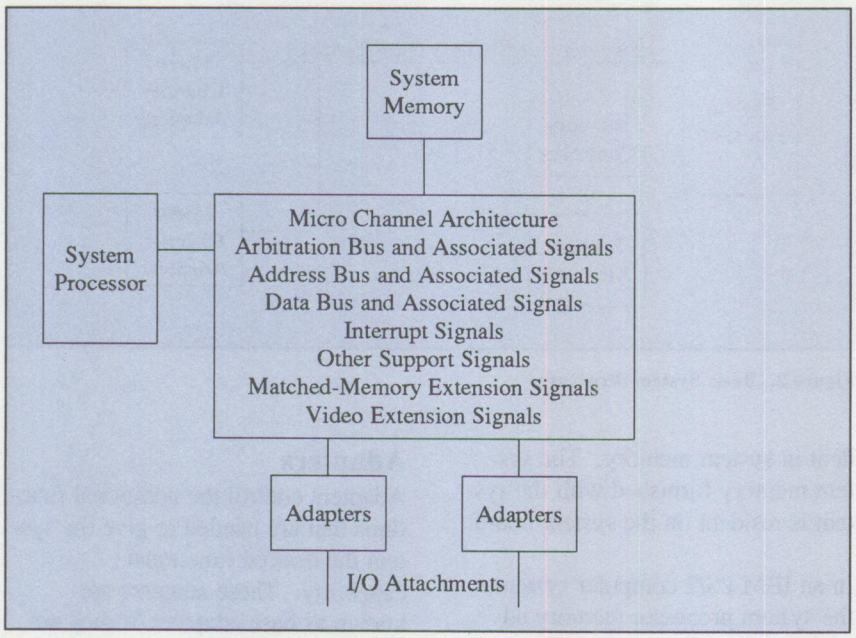



**Figure 4. Micro Channel Structure**

hardware. Examples of feature adapters include coprocessor adapters, serial port adapters, memory expansion adapters, host communications adapters, and network adapters.

This article discusses the adapter attachment interface to the channel. Devices attached to the non-channel side of an adapter, for example, a printer attached to a serial port adapter or a DASD attached to a SCSI adapter, and the related protocols are not included.

***Micro Channel Architecture***

Micro Channel architecture provides a high-speed data highway that connects the system processor, the system memory and the adapters.

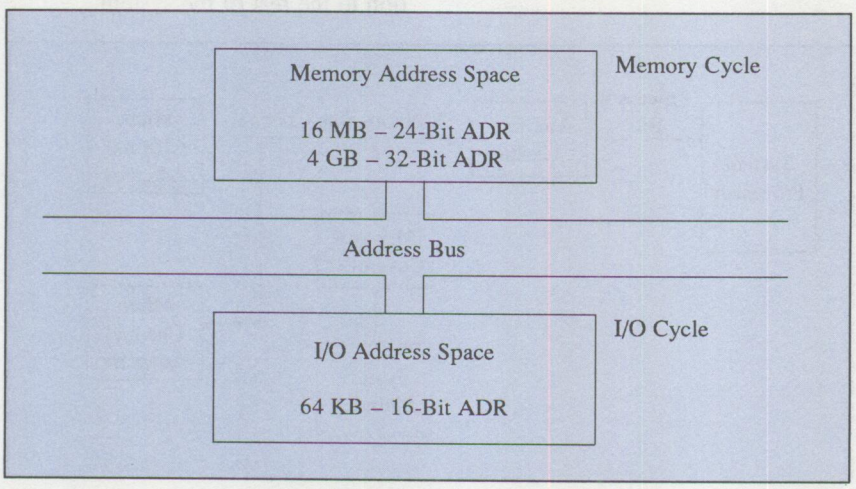



**Figure 5. I/O and Memory Address Space**

Micro Channel architecture consists of an arbitration bus and associated signals, an address bus and associated signals, a data bus and associated signals, interrupt signals, other support signals, and optional extensions for matched-memory extension signals and video extension signals. This structure is shown in Figure 4.

Micro Channel addressing consists of two separate address spaces: a memory address space and an I/O address space. The address bus and associated signals provide either a memory address or an I/O address on the channel (see Figure 5).

The I/O address space consists of 64 KB (KB = 1024 bytes) I/O addresses. During an I/O cycle, the I/O address space is addressed by 16 bits of the address bus.

The memory address space can be as large as 4 GB (GB = 1,073,741,824 bytes). During a memory cycle, the memory address space is addressed by the address bus. A 24-bit address can address 16 MB (MB = 1,048,576 bytes) of memory and a 32-bit address can address 4 GB of memory.

**Arbitration Bus and Associated Signals:** Micro Channel architecture provides a hardware arbitration procedure to control bus ownership. The arbitration bus and associated signals are used in the arbitration procedure, which prioritizes and resolves up to 16 requests (arbitration levels), for control of the channel. The winner of the arbitration is the temporary bus owner, called the *controlling master*. The arbitrating participants can be adapters or the system processor (see the section, "Micro Channel Participants"). The system processor is assigned an ar-

bitration level, leaving 15 arbitration levels available for adapters.

The arbitration procedure supports a burst mode. Burst mode allows an adapter to retain control of the channel to perform multiple data transfers without arbitrating for the channel for each data transfer. The data transfer continues until the data transfer is complete, or for up to 7.8 microseconds after another arbitrating participant requests the channel. At this point, channel ownership is released. The arbitration procedure is serial with data transfer, either an arbitration procedure or a data transfer can occupy the channel at any one time. Therefore, the use of burst mode provides for the maximum efficiency in the use of the Micro Channel bandwidth.

**Address Bus and Associated Signals:** The address bus and associated signals are used by the controlling master to select a slave during a data transfer procedure. Either an I/O space address or a memory space address is indicated. The address, on the address bus, is also used to select the storage location for the data transfer.

**Data Bus and Associated Signals:** The data bus is used to transfer either 8, 16, 24 or 32 bits of data. The associated signals indicate the width of the transfer and the direction of the data transfer, read or write.

**Interrupt Signals:** A request for processor attention, called an *interrupt*, is presented by an adapter through an interrupt signal. Each interrupt signal is on a system interrupt-priority level. The adapter request for service is processed by the system processor at the requested priority. Micro Channel architecture supports 11 interrupt levels. To allow configuration flexibility, each interrupt level may be concurrently shared by multiple adapters.

**Other Support Signals:** The other support signals include the capability for integrating a single-channel audio signal on the channel. Adapters may exchange and process audio information using this audio signal.

**Matched-Memory Extension Signals:** Matched-memory extension signals support a unique data transfer procedure between the system processor and its channel-resident memory (see the section, "Matched-Memory Extension").

**Video Extension Signals:** The video extension signals provide the capability for a Video Graphics Array (VGA) feature adapter to transfer video data between the video subsystem and the feature adapter. For example, the IBM PS/2 Display Adapter allows upgrade of VGA performance beyond that provided by the base VGA adapter.

***Micro Channel Participants***

The system processor, system memory and adapters attached to the channel contain entities called *participants*. This section describes the Micro Channel participants. See Figure 6 for a diagram of the participants and the data transfer paths supported.

The Micro Channel participants are either masters or slaves. A master, or a DMA slave, arbitrates for access to the channel and supports data transfer to or from a slave. A slave sends and receives data under the control of a master, called the controlling master.

**Masters:** There are three types of implemented masters: the system processor, called the *system master*, the DMA controller and a bus master. The system processor and the DMA controller are provided with the system hardware. The bus master is an adapter implementation.

*System Master (System Processor):* The system processor is the processor provided with the system hardware. The system processor

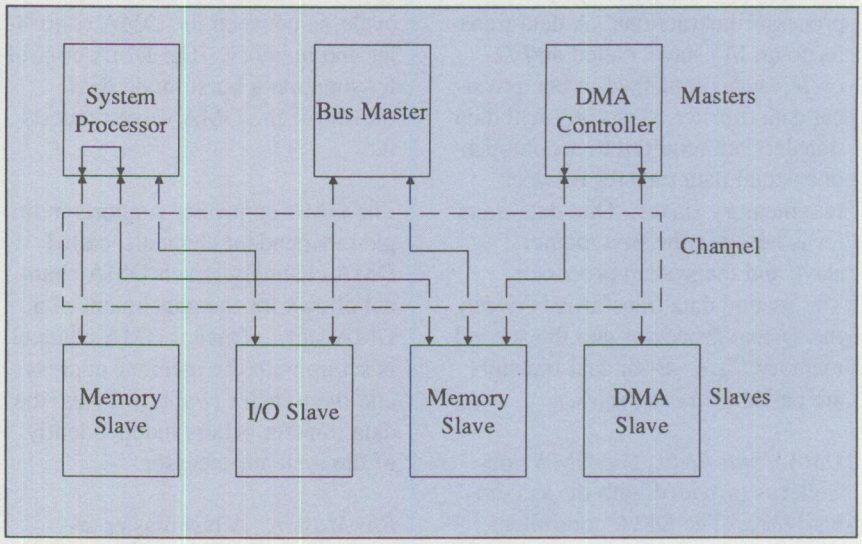



**Figure 6. Micro Channel Participants and Data Transfer Paths**

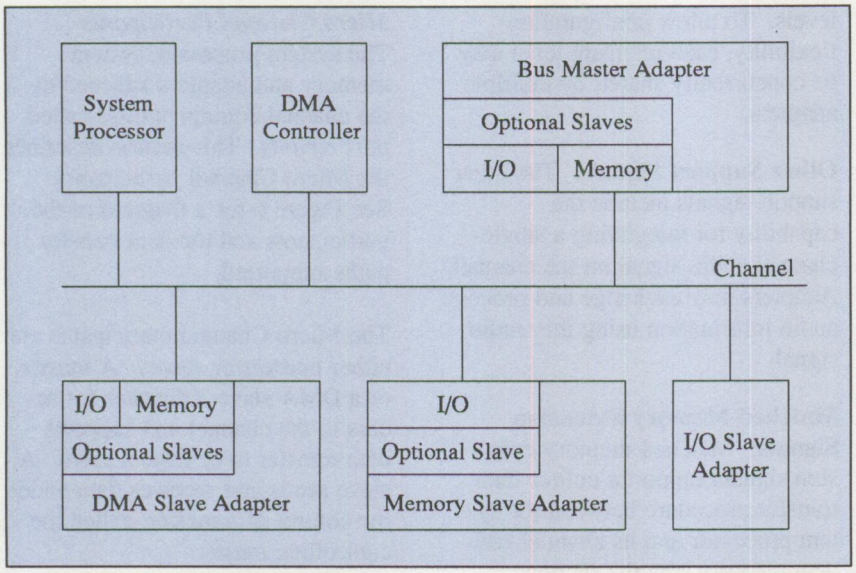



**Figure 7. Adapter Participants**

controls and manages the system configuration. It may initiate a request for use of the channel. The system processor owns the channel when no other master owns the channel; it is also known as the *default master*.

The system processor supports data transfers to or from an I/O slave or a memory slave. The data transfers are directly controlled by system processor instructions. A data transfer to an I/O slave, called an *I/O cycle*, is accomplished in one physical data transfer. Two physical data transfers are required to accomplish one actual data transfer between two memory slaves. One data transfer is between the first memory slave and the system processor. The second data transfer is between the system processor and the second memory slave. Both data transfers are called *memory cycles*.

*DMA Controller:* The DMA controller is provided with the system hardware. The DMA controller does not arbitrate for the channel. A DMA slave that requires a data transfer will perform the arbitration.

The DMA controller manages the transfer of data between a DMA slave and a memory slave. This operation requires two physical data transfers to accomplish one actual data transfer. One data transfer, the I/O cycle, is between the DMA slave and the DMA controller; and a second data transfer, the memory cycle, is between the DMA controller and memory. The DMA controller supports a burst mode data transfer if the DMA slave requests it.

The DMA controller supports multiple independent channels, called DMA channels. Each DMA channel allows for the attachment of a DMA slave. Once a DMA channel is set up with the memory address and the transfer count, in bytes, the data transfer occurs independently of the system processor.

*Bus Master:* A bus master arbitrates for the channel and manages the data transfer to or from an I/O slave or a memory slave. The data transfer is accomplished in one physical data transfer. The bus master can, optionally, support a burst mode data transfer.

A bus master can receive an interrupt signal from an I/O slave adapter. Either the system hardware or a bus master will be enabled to receive an interrupt level, but not both. It is not possible for two masters to receive the same interrupt. Therefore, the interrupt level that is received by the bus master must be disabled in the system hardware by a software procedure before it can then be enabled in the bus master.

**Slaves:** There are three types of slaves: I/O slave, memory slave and DMA slave.

*I/O Slave:* During an I/O cycle, an I/O slave is selected by an address in the I/O address space. The system processor or a bus master is required to perform the data transfers. Either the master will poll the adapter to initiate the data transfer or the I/O slave will interrupt the master to initiate the data transfer.

*Memory Slave:* During a memory cycle, a memory slave is selected by an address in the memory address space. A master (system processor, DMA controller, or a bus master) is required to perform the data transfers. The two basic applications of a memory slave are Micro Channel system memory and non-system memory.

Micro Channel system memory is memory that is installed in a Micro Channel connector. The system memory is assigned a range of addresses within the Micro Channel memory address space. The Micro Channel system memory may be

used as shared memory among adapters or as instruction memory for a bus master adapter. For an IBM PS/2 computer system, the Micro Channel system memory also can be used to expand system memory beyond that supported by the system board (see the section, "System Memory").

Non-system memory is memory in the Micro Channel memory address space that is on an adapter. The non-system memory is used for data transfer between the memory slave and a controlling master. This is referred to as memory-mapped I/O.

*DMA Slave:* A DMA slave arbitrates for the channel. The DMA slave relies on the DMA controller to be the controlling master that manages the data transfer between the DMA slave and a memory slave. The DMA slave is associated with a DMA channel in the DMA controller. The DMA slave can, optionally, support a burst mode data transfer.

***Participant Usage***

The functional capability of one or more Micro Channel participants is incorporated into an adapter. The primary mode of operation (*operational mode*) of the adapter determines its primary participant when there is more than one participant. The adapter type is the same as its primary participant. There are four basic adapter types: an I/O slave adapter, a memory slave adapter, a DMA slave adapter, or a bus master adapter (see Figure 7). If an adapter performs a data transfer between an I/O device and memory using the functions of a DMA slave participant, it is a DMA slave adapter. If an adapter performs a data transfer between an I/O device and memory using the functions of a bus master participant, it is a bus master adapter. In the selection of an adapter, see the section, "Adapter Participant," for the adapter performance considerations.

As shown in Figure 7, an adapter may incorporate one or more secondary participants in addition to the primary participant. The secondary participant can be used to support adapter initialization and control functions required in the execution of its operational mode. A second usage is adapter to adapter data transfer between two bus master adapters (see the section, "Adapter-to-Adapter Data Transfer"). The controlling bus master performs the data transfer to a memory slave participant in the second bus master adapter. The operational mode of both adapters is a bus master.

The adapter system resource requirements (for example, interrupt level, DMA channels, arbitration levels, I/O address space addresses, and memory address space addresses) are the total of the primary participant requirements and any additional requirements of one or more secondary participants.

***Address Bus Support***

A Micro Channel adapter implements a 16-bit, 24-bit or 32-bit address bus. Figure 8 provides the currently supported address bus widths for each participant.

***Data Transfer***

Micro Channel data transfers can be between the system processor and an adapter, the system processor and system memory, an adapter and system memory, or an adapter-to-adapter. The significant performance parameters are the data bus width and the data transfer time, which yield the data transfer rate.

**Data Bus Width:** A Micro Channel adapter can implement an 8-bit, 16-bit or 32-bit data bus width. For example, a 32-bit data bus means that the data transfer width can be up to 32 bits, and that each data transfer can be 8 bits, 16 bits or 32 bits. The data transfer width is equal to the lesser of the data bus width of the controlling master and the selected slave, or of the data width of the selected I/O address (it can be less than the data bus width).

Figure 8 provides the currently supported data bus widths for each participant.

**Data Transfer Time:** The data transfer time is determined by the data transfer procedure and the capability of the two participants involved in the data transfer. The data transfer time is the greater of the data transfer times supported by the controlling master and the selected slave.

*Basic Data Transfer Procedure:* Micro Channel architecture supports

| Participant | Data Bus Width | Address Bus Width |
|---|---|---|
| System Processor | 16-bit or 32-bit | 24-bit or 32-bit |
| DMA Controller | 16-bit | 24-bit |
| Bus Master | 16-bit or 32-bit | 24-bit or 32-bit |
| DMA Slave | 8-bit or 16-bit | 16-bit |
| I/O Slave | 8-bit, 16-bit or 32-bit | 16-bit |
| Memory Slave | 8-bit, 16-bit or 32-bit | 24-bit or 32-bit |

**Figure 8. Participant Data Bus and Address Bus Width**

| Transfer Width | Transfer Time | | |
|---|---|---|---|
| | 200 ns | 300 ns | 3800 ns |
| 8-bits | 5 MB/second | 3.3 MB/second | 0.26 MB/second |
| 16-bits | 10 MB/second | 6.6 MB/second | 0.52 MB/second |
| 32-bits | 20 MB/second | 13.3 MB/second | 1.05 MB/second |

**Figure 9. Data Transfer Rate**

a range of data transfer times for a basic data transfer procedure. The default data transfer time is a minimum of 200 nanoseconds. Either the system or the adapter may extend the 200 nanosecond minimum time. An adapter may implement an extended data transfer time from 300 nanoseconds to 3800 nanoseconds.

*Matched-Memory Extension:* A matched-memory extension is a unique data transfer procedure between the system processor and the selected Micro Channel system memory adapter. The procedure provides for a data transfer time that is less than the default data transfer time of that system. Both the system hardware and the Micro Channel system memory adapter must support the matched-memory extension, or the data transfer will be performed with the basic data transfer procedure.

**Data Transfer Rate:** The instantaneous data transfer rate of an adapter on the channel is determined by the data transfer width and the data transfer time. Figure 9 provides the range of capability of the Micro Channel architecture for the basic data transfer procedure.

The DMA controller furnished with IBM PS/2 computer systems has an 16-bit data bus width capability that is independent of the system data bus width. As two physical data transfers are required to accomplish one actual data transfer (see the section, "DMA Controller"), the maximum data transfer rate is limited to 5 MB per second (MB/sec) for a DMA slave adapter with a 16-bit data bus, or 2.5 MB/sec for a DMA slave adapter with an 8-bit data bus. (This does not include the enhanced functions described in the article, "Overview of Extended Micro Channel Functions" in this issue.)

**Adapter-to-Adapter Data Transfer:** Micro Channel architecture supports an adapter-to-adapter data transfer that does not involve system board memory or the system processor. The transfer is between a bus master adapter and a memory slave or an I/O slave adapter, or between a DMA slave adapter and a memory slave adapter through the DMA controller.

A bus master adapter may optionally support data transfer to or from another bus master acting as a slave. The controlling bus master initiates the data transfer to or from the selected memory slave participant supported by the other bus master. Either bus master may be the controlling master, providing a symmetrical data transfer capability. This method of data transfer is referred to as *peer to peer*.

***Buffered Adapters***

A *buffer* is storage in an adapter that is used to temporarily hold data. The purpose is to compensate for the difference in rate of flow of data or time of occurrence of events, when transferring data between the adapter and memory. The buffer permits the transfer of data using burst mode, which provides for the most efficient usage of the Micro Channel bandwidth.

A buffered adapter minimizes or eliminates the critical service time requirement of the adapter (see the section, "Critical Service Time").

***Intelligent Adapters***

An adapter can furnish the basic connectivity to a device, and it also can perform some of the tasks necessary for the device to operate. When an adapter furnishes more than just the basic connectivity, it requires some intelligence. This intelligence is generally in the form of an onboard processor. An onboard processor can perform functions that would normally be accomplished by the system processor. This offloading of function allows the system processor to perform concurrent execution for other tasks.

A programmable, intelligent adapter provides the user with the capability to develop applications that are executed on the adapter. The programmable adapter may be an I/O subsystem or a coprocessor. A coprocessor adapter may allow for system performance improvement of an existing system.

***Adapter Functional Packaging***

**Single-Function Adapter:** A single-function adapter is one that has a single Micro Channel interface. The adapter may have one or more operational modes. However, for an adapter with more than one operational mode, only one of these operational modes is active at any one time. The active operational

mode determines the performance characteristics. A bus master SCSI adapter is an example of a single-function adapter that supports a single operational mode. A parallel port adapter is an example of a single-function adapter that may support more than one operational mode. This adapter may act as an I/O slave as operational mode 1 and a DMA slave as operational mode 2.

An adapter can present a single Micro Channel interface, but support multiple I/O attachments. This is also a single-function adapter if the multiple attachments are managed transparently to the Micro Channel interface. An intelligent I/O subsystem that supports multiple serial ports and a SCSI adapter that attaches multiple devices on the SCSI interface are examples of this type of single-function adapter.

**Multifunction Adapter:** An adapter can contain multiple independent functions. The interface to the channel may have a common physical appearance, but there will be an independent logical appearance for each function. The functions may be identical with a single operational mode, such as a serial port adapter that supports two serial ports; or the functions may be different, such as an adapter that has a serial port and a parallel port. There is a unique operational mode for each function, in the second case. In both cases, each independent function can be considered a separate adapter, except for the physical resources. Therefore, the system resource requirements for the multifunction adapter are the total of the system resource requirements for each independent function.

## Software

There are three classes of software that will support a given hardware configuration. These software classes are application, operating system and device driver. Figure 10 shows the interrelationship among

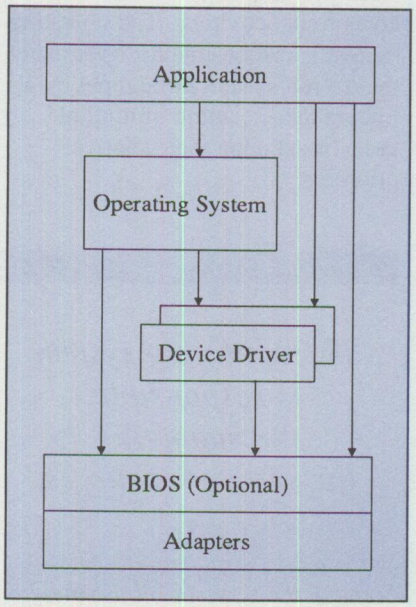



**Figure 10. Software Systems Structure**

the software classes. It also shows the different levels of interfaces used to control the transfer of data to or from an adapter.

When selecting the components for a system, both software and hardware, it is important to know that the best performance is realized when the software complements the hardware. In addition, the software, operating system, application and device driver should be matched for best performance. This section will address the aspects of software that affect system performance.

### Applications

Application software performs a function for the user. Examples of applications are a spreadsheet, a financial model, or an engineering calculation.

***Tasking Environment***

When choosing an application, it is necessary to determine whether it is suitable for your environment. It is also necessary to determine the type of operating system that the application was designed to run under. Most applications that are designed to run under a single-tasking operating system will run under a multitasking operating system, but this operation may be inefficient. Such applications may monopolize resources at the expense of other tasks. However, applications specifically designed to run under a multitasking operating system in order to exploit the multitasking capabilities, use the resources in an efficient manner. This allows other tasks access to the resources.

Most applications fall into one of two basic categories, processor-intensive or I/O-intensive.

***Processor Utilization***

A processor-intensive application is one which spends at least 80 percent of its time performing calculations. Examples of this are engineering and scientific programs and financial modeling applications. Most business programs are only processor-intensive for short periods of time, for example, for the recalculation of a spreadsheet or the compiling of a program. The effect of a processor-intensive application may be observed as delays between a request and the response to the request.

***I/O Utilization***

An I/O-intensive application is one that spends at least 80 percent of its time performing I/O operations. Text editors, word processors or

data base programs are examples of programs that are I/O-intensive. The throughput supported by an I/O-intensive program is determined by its design. A program that performs character I/O is generally at the low end of the performance spectrum while a program that performs block I/O is generally at the high end. The size of the request blocks and the frequency of the requests will influence overall throughput.

***Application Performance Consideration***

The application response time, in a single-tasking environment, is determined by the hardware. In a multitasking environment, the resources required by the other tasks that are being executed concurrently can affect the application response time.

## Operating Systems

The operating system provides system resource management and the support structure for executing application software and device drivers. The operating systems are general purpose and are designed to support a wide variety of application software.

***Resource Management***

The operating system is responsible for managing the system resources. This management involves the ownership and sharing of the resources. When the resources are examined, it is done relative to the environment, single-tasking or multitasking.

In a single-user, single-tasking environment, such as IBM DOS, the operating system is responsible for handling interrupts and I/O services. The I/O services handled by the operating system are console I/O and file I/O. All other I/O is performed by device drivers that are separate from the operating system.

A single-user, single-tasking environment handles one task at a time.

A single-user, multitasking environment is more complex; IBM OS/2 is an example of a single-user, multitasking operating system. This environment allows a number of concurrent activities. The operating system is responsible for optimizing the overall system throughput by allowing the execution of multiple tasks based upon their relative priorities.

*The operating system is responsible for managing the system resources.*

Some of the services furnished by the operating system are the handling of interrupts, memory management, and task management. The operating system determines whether a given interrupt level can or cannot be shared and calls the appropriate interrupt handler.

The operating system will try to balance the throughput of system activities depending on task priority. This is accomplished through its task management services, the services used to switch from one task to another. Task switching can result in a call to the memory management services, which may result in I/O activity on the channel.

Multitasking operating systems rely on subsystems and device drivers to manage their respective I/O requirements. These subsystems and device drivers are responsible for handling I/O requests from the applications running in the system, and are also responsible for performance optimization. These subsystems can be acquired with the operating system or they can be acquired as separate software products and installed as a part of the operating system.

A multiuser, multitasking operating system takes the complexity of the single-user, multitasking operating system and adds the support for multiple users, generally by supporting multiple terminals. This requires operating two or more terminal sessions concurrently. There are different versions of multiuser operating systems, and applications written for one may not necessarily run on another. IBM AIX is an example of a multiuser, multitasking operating system.

Multitasking operating systems that are executing I/O-intensive applications may fully utilize the resources available in a system. The user interface will reflect performance by its responsiveness to the requests.

Some application software packages can run a form of multitasking using a single-tasking operating system.

## Device Drivers

Device drivers are written to handle the specific requirements of a particular device. They are designed to work with a specific operating system. There is at least one interface that the device driver uses to communicate with the operating system. Other interfaces may be provided to allow the device driver to communicate with programs running under the operating system. These interfaces allow the application software

(above the device driver) to be device independent. The device driver makes it possible to support new devices after an operating system has been made available.

The device driver's characteristics and capabilities are dependent upon the device that is being supported and the environment that they are operating under.

***Execution Environment***

In a single-tasking environment, a device driver will support a device either through interrupts or through polling by the system processor. Events in a single-tasking environment are usually synchronous; for example, an application sends or requests data and then waits for an interrupt, or polls for the response. The device driver typically uses interrupts to support a device in a multitasking environment. It is expedient in this environment, because when one task is waiting for a request to be serviced another task can be executing instructions. Events in a multitasking environment are asynchronous, and the device driver must be able to support multiple requests. These device drivers use system resources more efficiently.

***Devices Supported***

A device driver can support a single device, or it can support multiple devices. The performance derived from a device driver may be a function of the capabilities of the device. If a device driver is supporting a character I/O device, it may buffer the requests coming or going to the application, but it still has to transfer one character at a time to the device. If a device driver is supporting a block I/O device, it will be able to transfer data a buffer at a time. The size of the buffer will depend on the capabilities of the device.

## BIOS

The Basic Input/Output System (BIOS) was developed to provide a compatible interface between software and hardware. BIOS allows the applications programmer to request an I/O operation without having to consider the physical details of hardware operation.

Two types of BIOS exist, BIOS and Advanced BIOS (ABIOS). BIOS is used in single-tasking environments such as IBM DOS. ABIOS is used in multitasking environments such as IBM OS/2.

# Performance: Parameters and Considerations

The performance of a system is determined by a combination of the hardware capability, the operating environment and how it is used. Performance is a measure of system productivity.

It is necessary to determine the capacity that is needed to adequately perform the work required of the system. Then it is necessary to define a configuration that has the required capacity. To ensure that a system will meet its throughput and response time performance objectives, it is necessary to configure the components consistently with their capacity requirements. Since the channel provides the fundamental interconnection among the system processor and multiple adapters, the interaction of the system processor with the adapters becomes the most important part of the configuration process.

## Performance Parameters

The parameters that define the hardware capability are system processor performance and the Micro Channel data transfer rate. The parameters that measure system performance are throughput and response time.

***System Processor Performance***

The system processor performance is the measure of its ability to do work and is referred to as the *system processor power*. When viewed over the short-term, in the order of hundreds of microseconds, it is referred to as *instantaneous system processor power*; when viewed over the long-term, in the order of several seconds, it is called the *average system processor power*. In addition, in multitasking environments where there can be concurrent I/O, it is referred to as *effective system processor power*.

**Instantaneous System Processor Power:** The instantaneous system processor power available in a system is determined by measurements taken over the short-term, several hundred microseconds at a maximum. It is dependent upon a number of factors. These factors include the operating environment, the system structure, and the adapters. In a single-tasking environment, the instantaneous system processor power available when the system processor is active is the rated power of the system processor. In a multitasking environment, the instantaneous system processor is affected by by concurrent I/O and system structure.

The available instantaneous system processor power varies with time; this variability will be discussed under "Effective System Processor Power."

**Average System Processor Power:** The average system processor power available in a system is determined by measurements taken over the long-term, several seconds or more. It is an indication of the overall system capability that is available to the user. It is not an indication of the system capability available to a user at any specific instant of time.

**Effective System Processor Power:** The effective system processor power available to run applications is a function of the operating environment, the system structure and the amount of concurrent I/O. In a single-tasking environment, the system processor is either processing an application or waiting for an I/O operation to complete. When the system processor is processing an application, the rated power of the system processor is available. When the system processor is waiting for an I/O operation to complete, the system processor instruction execution is suspended. In a multitasking environment, two or more tasks can be active at the same time, and the system processor power can be affected by concurrent I/O requiring access to the channel.

Effective system processor power is the power available to an application after the effect of concurrent I/O has been taken into account. In the basic system structure as shown in Figure 2, the channel is used by the system processor to fetch instructions and data and by the I/O to transfer data. Therefore, the system processor's ability to fetch instructions or data from system memory can be delayed by I/O operations that share the channel. With the enhanced system structure, when the system processor fetches instructions or data from the memory cache, it uses the processor bus as shown in Figure 3. There is no contention with concurrent I/O, and the effective system processor power is the rated power of the system processor. This is one of the reasons that the enhanced system structure can exhibit more system processor power than the basic system structure. When the system processor fetches instructions or data from system memory, it requires the memory bus/channel and its operation can be inhibited by I/O operations that share the channel.

The potential contention between the system processor and the I/O for use of the channel makes feature adapter selection an important consideration. In multitasking environments it becomes more important to select feature adapters that use the channel efficiently. These adapters use less Micro Channel time to transfer the same amount of data, and, therefore, reduce the potential for contention.

***Micro Channel Bandwidth***

The Micro Channel bandwidth is the data transfer capability of the channel. It is the throughput of the channel measured in MB/sec. When viewed over the short-term, this bandwidth is called *instantaneous bandwidth*, and when viewed over the long-term, it is called *average bandwidth*.

**Instantaneous Bandwidth:** When the system processor, or an adapter, uses the channel, the instantaneous bandwidth is of interest. Figure 9 shows the ranges of instantaneous bandwidth available for different adapter capabilities. When a participant is using the channel, the instantaneous bandwidth of the channel is based upon that participant's data transfer rate.

**Average Bandwidth:** The average bandwidth of the channel is the bandwidth that is delivered over a long period of time. When two, or more, participants require the Micro Channel, each participant is allocated a portion of the bandwidth. (Micro Channel architecture permits 15 adapters, plus the system processor, to simultaneously request bandwidth.) Therefore, the average bandwidth is the summation, over time, of the active participants' instantaneous bandwidths.

***Throughput***

Throughput is a measure of the amount of work that a system can produce. It is the number of tasks that can be performed in a unit of time. Throughput is affected by the capabilities of the hardware, the application software and the operating system.

***Response Time***

Response time is the length of time required to service a request. It is measured from the time that the request is placed until the time that the response to that request is received.

## Considerations

There are a number of performance-related considerations involved in selecting a system. These include the operating environment, the application software, the system structure and the adapters.

When the components of the system are matched, it will deliver a satisfactory level of performance. The performance may be affected by mismatched components. Therefore, care is necessary in matching the capabilities of the system hardware, the adapters and the software to each other and to the functions to be performed.

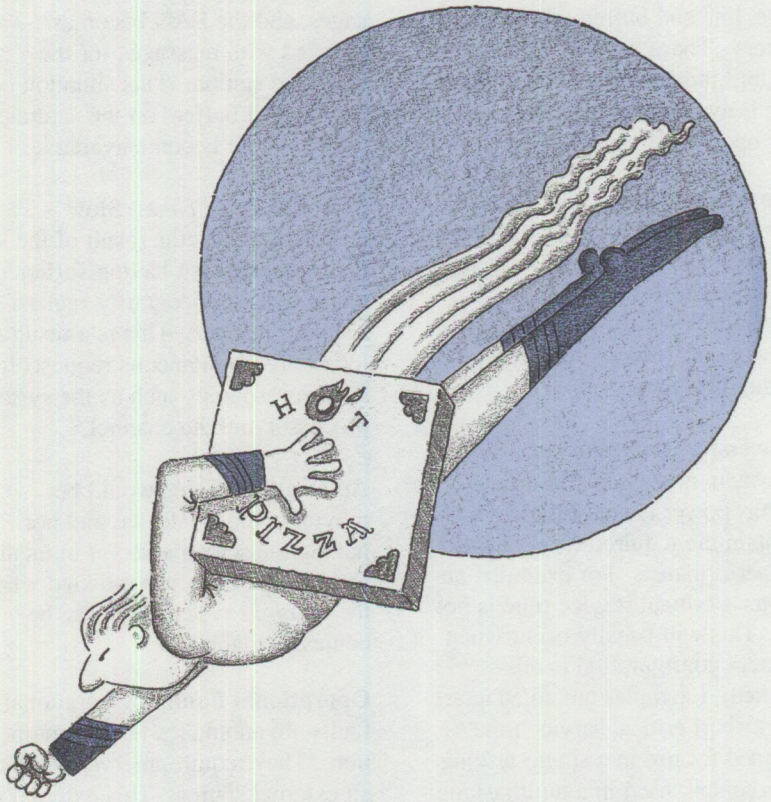
Some hardware configurations work better in a single-tasking environment. One such configuration has adapters that have a critical service time dependency on the system processor. These adapters depend upon the resources being available when needed. When one of these configurations is used in a multitasking environment, there is the possibility that degraded performance, recoverable errors or operational faults might occur. This problem can be avoided by selecting adapters that do not have a critical service time dependency on the system processor. This configuration should increase the system capacity and improve the system performance.

A properly loaded system, with matched components, can exhibit degraded performance at specific instants of time. These systems will slow down when the instantaneous workload peaks and temporarily exceeds the system capacity. The system load stabilizes, and the performance returns to normal when the workload peak is reduced.

***Critical Service Time***

An adapter that does not arbitrate for the channel must be serviced by a controlling master. The controlling master can poll the adapter to initiate a data transfer, or the adapter can signal the controlling master with an interrupt to indicate it is ready for service. The time required for a controlling master to service an adapter is referred to as the *service time*. The amount of time that an adapter can wait for service is called the *critical service time*.

Micro Channel architecture describes the arbitration process and the maximum delay that a participant can experience when the




channel is active. It is recommended that an adapter have a buffer large enough to overcome the arbitration delays. If an adapter does not have sufficient buffer capacity, there is the possibility of a *data overrun*. A data overrun occurs when an adapter is receiving data from a source and exhausts its local buffering capability before it is serviced. The time interval before the adapter is serviced is the Micro Channel service time. If the adapter does not have sufficient buffer capacity, the time interval before this capacity is exhausted is the Micro Channel critical service time.

A *control overrun* occurs when an adapter does not receive control information in a timely fashion. This overrun can occur when a critical service time or a Micro Channel critical service time is not met.

When a critical service time is not met, an exception condition occurs, and the system will experience a performance degradation, a recoverable error or an operational fault.

**Performance Degradation:**

Performance degradation occurs when a device does not receive service within its critical service time period. The result of this service anomaly is that there is a perceived slowdown in system operation. User intervention is not required, and operations return to normal when the instantaneous power reaches a level that is adequate to handle the current workload.

Examples of this are character mode, line and buffered page printers. These printers stop operating until the next character, line or page is available, and then resume their operation.

**Recoverable Errors:** Recoverable errors are anomalies in system performance that do not require intervention from an external source. These problems are handled by the hardware or software that experiences the difficulty.

Recoverable errors can occur when the system processor's instantaneous power is not sufficient to maintain the required level of system performance. For example, an adapter's critical service time is not met. This will usually occur when a system configuration is mismatched. Examples are an adapter with a short critical service time designed for use in a single-tasking environment, used in a multitasking environment, or several adapters with critical service times used in a multitasking environment with a low capacity system processor. The result can be performance degradation, network congestion or slow response time.

*Network Congestion:* Network congestion occurs when a communication facility does not receive service within a critical service time. The result of this service anomaly is a slowdown in network operations. This is caused by the network retrying the failed operation. User intervention is generally not required. Operation returns to normal when the instantaneous power reaches a level that is adequate to handle the current workload.

An example of this is a local area network (LAN) adapter where the buffers are full. The LAN adapter cannot receive any additional messages, and the LAN becomes clogged with messages for the saturated station. This situation continues until buffers on the saturated LAN adapter become available.

*Slow Response Time:* Slow response time is the result of the system processor not having sufficient capacity to service a user request in a timely manner. This is a result of too many simultaneous requests for system resources such as the system processor and the channel.

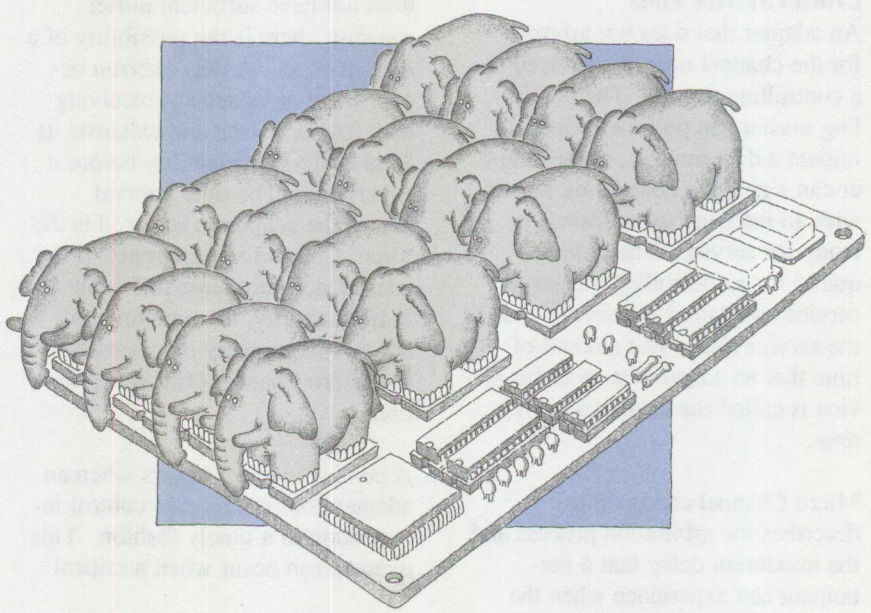
An example of this would be moving a mouse device and not having the screen respond immediately. The request is satisfied when the required system resource becomes available.

**Operational Faults:** Operational faults are anomalies in system operation. They require intervention by an external source. This external intervention, which is typically manual, could be restarting a communications line or other device in order to recover from the effects of the anomaly.

***System Memory Considerations***

System memory can have a major impact on system performance. Systems that have more memory generally perform better. Additional memory reduces the need for paging, or swapping, information from a disk file, thereby reducing the delays incurred waiting for disk access. Also, additional memory can be used as a disk cache, where frequently used disk data is stored for fast access, or a RAM disk, which is a form of fast access disk file. There are other similar uses for additional system memory.

These performance gains are achieved whether the additional system memory is system board memory or Micro Channel system memory. On some IBM PS/2 computer systems, the memory subsystem operation may provide the system processor a faster access to system board memory than to Micro Channel system memory. Therefore, it is advisable to completely populate the system board memory before adding Micro Channel system memory. The performance



gains achieved by using additional memory, either on the system board or on the channel, are far greater than the performance differences that might exist between system board memory and Micro Channel system memory.

***Address Bus Considerations***

A 32-bit system processor with a 32-bit address bus has performance advantages over systems that support 24-bit addressing. The 32-bit address bus supports an address space of 4 GB. The 24-bit address bus supports an address space of 16 MB. When an adapter in a 32-bit operating system, (for example IBM AIX), has a 24-bit address bus, the software must manage the 16 MB data space to assure that the address for a data transfer between an adapter and system memory resides within the 16 MB space. This software overhead is eliminated, yielding better system performance, when the adapter has a 32-bit address bus.

***Data Transfer Considerations***

Data transfer can be affected by a number of factors. These factors include the data transfer time, the data transfer width, the adapter buffer and burst capability, and the number of physical moves required.

An adapter that can support the minimum default data transfer time of 200 nanoseconds will move data faster than an otherwise equivalent adapter that supports an extended data transfer time of 300 nanoseconds (see Figure 9). The wider data transfer (for example, 32 bits as opposed to 16 bits) will have a higher data transfer rate for the same data transfer time.

The Micro Channel arbitration time is a minimum of 300 nanoseconds. An adapter that supports burst mode can perform multiple data transfers after one arbitration. This can cut the arbitration overhead up to 97 percent for long bursts and greatly increase the Micro Channel data transfer efficiency.

The actual data transfer efficiency is affected by the number of moves required to get the data to the proper place. For example, to move data from a disk file to a LAN adapter could use any of the adapter types. Because this type of data transfer typically involves large files, only the DMA slave adapters and the bus master adapters will be considered because they can support burst mode. An adapter that supports burst mode will provide a buffer to sustain the burst. The size of the buffer may affect the length of burst that the adapter can sustain. DMA slave adapters require four physical transfers to move the data from the disk file adapter to the LAN adapter. The following physical moves are required to complete the move from the disk file: first from the disk file adapter to the DMA controller and then from the DMA controller to system memory. The following physical moves are now required to complete the move to the LAN adapter: first from system memory to the DMA controller and then from the DMA controller to the LAN adapter. Bus master adapters will require two physical transfers to move the data from the disk file adapter to the LAN adapter: one move from the disk file adapter to system memory, and a second move from system memory to the LAN adapter. Bus master adapters that support adapter-to-adapter data transfer will require only one physical move to accomplish this transfer. This move would be from the disk file adapter directly to non-system memory on the LAN adapter.

The bus master adapter in the previous example has an advantage over the DMA slave adapter in addition to using fewer physical moves. The bus master adapter can support a 32-bit data transfer while the DMA slave adapter is currently limited to a 16-bit data transfer. Therefore, the bus master adapter has a greater channel efficiency than a DMA slave adapter; it requires fewer physical moves to complete a data transfer, and each physical move can contain twice the data.

***Adapter Participant***

The adapter participants reflect the Micro Channel participants, (see the section, *Micro Channel Participants*), and are either masters or slaves. Each participant has capabilities that make it suitable for use in different operating environments and for performing different tasks.

**I/O Slave Adapter:** The I/O slave adapter has a critical service time dependency on the master (system processor or the bus master). It generally does not provide a data buffer and requires a controlling master to perform data transfers.

The I/O slave adapter is most suitable for use in single-tasking environments. It may also be adequate for use in multitasking environments that have low data transfer rate requirements.

**Memory Slave Adapter:** A memory slave adapter may be a Micro Channel system memory adapter or a memory-mapped I/O adapter. A Micro Channel system memory adapter does not have a critical service time dependency on the master (system processor or the bus master) or the channel. A memory-mapped I/O adapter typical-

ly does not have a critical service time dependency on the master (system processor or the bus master) or the channel.

The memory-mapped I/O adapter, which provides memory for use as a data buffer, requires a controlling master to perform data transfers. The controlling master can be the DMA controller, a bus master or the system processor. If the controlling master is either the DMA controller or a bus master, this adapter is suitable for use in any operating environment.

When the system processor is the controlling master, the system processor is required to execute instructions to perform each data transfer. Therefore, this adapter is most suitable for use in single-tasking environments and for multitasking environments where the data transfer requirements are low. The memory-mapped I/O adapter can use an excessive amount of the system processor resources when it is required to transfer large amounts of data. In a multiuser environment, this excessive use of the system processor resources can adversely affect system performance.

**DMA Slave Adapter:** A DMA slave adapter may be buffered or non-buffered. Typically, a buffered DMA slave adapter does not have a a critical service time dependency on the master (system processor or the bus master) or the channel, and a non-buffered DMA slave adapter has a critical service time dependency on the master (system processor or the bus master) or the channel.

The DMA slave adapter relies on the DMA controller to be the controlling master to perform data transfers. This adapter requires two physical moves for each data transfer. There is one move between the adapter and the DMA controller and another move between the DMA controller and system memory. The DMA slave adapter supports burst mode data transfer, which allows multiple data transfers for each arbitration.

The DMA slave adapter is suitable for use in any environment. It is best used when the data transfer requirements are low to moderate because of its Micro Channel efficiency.

**Bus Master Adapter:** A bus master adapter may be buffered or non-buffered. Typically, a buffered bus master adapter does not have a critical service time dependency on the master (system processor or the bus master) or the channel, and a non-buffered Bus Master adapter has a critical service time dependency on the master (system processor or the bus master) or the channel.

The bus master adapter is generally the most sophisticated Micro Channel participant. It is a controlling master and does not require any system processor intervention to perform data transfers. The Subsystem Control Block (SCB) architecture provides a software protocol for a bus master to transfer control information and data between bus master adapters and between a bus master adapter and the system processor. This results in a further reduction of system processor dependency on the part the bus master adapter.

The bus master adapter can move data very efficiently. It requires only one physical move for each data transfer. Burst mode is supported where multiple data transfers can be completed for each arbitration. In addition, bus master adapters can support adapter-to-adapter data transfer, which allows data to be transferred directly between two adapters without an intermediate transfer to system memory.

The bus master adapter is suitable for use in any environment. Its Micro Channel efficiency makes it desirable for use when large amounts of data are to be transferred in a multitasking environment. The bus master adapter is recommended for use in all multitasking environments, and it is especially useful in high data rate demand environments.

## Product Selection

The previous sections have discussed the various components of a Micro Channel computer system, and the performance parameters and considerations of those components. This section builds on that base and provides help in selecting and configuring Micro Channel systems and feature adapters.

### Operating Environment

The most important consideration in selecting and configuring a hardware system is the intended usage, and, based on the intended usage, the operating environment. The operating environment can be either single-tasking or multitasking. A single-tasking environment is for a single user. A multitasking environment may be either for a single user or for multiple users. The software is selected based on the established operating environment and the intended usage.

**Applications:** Applications are selected that meet user requirements and are compatible with the operating environment. The applications

must be selected in conjunction with the operating system to assure compatibility.

**Operating System:** The operating system is selected based upon the operating environment, and the available applications. For example, IBM DOS supports a single-tasking environment; IBM OS/2 supports a single-user, multitasking environment; and IBM AIX supports a multiuser, multitasking environment.

**Device Driver:** The device driver must be compatible with the operating system and the adapter. The device driver can be provided with the software, with the adapter or as an independent entity. If the device driver is provided with the software, the adapter hardware requirements are specified. If the device driver is not provided with the software, it must be provided with the adapter or purchased separately.

## Hardware Selection

Hardware system selection consists of determining the hardware requirements of the selected software, the operating environment, the system capacity, and the usage environment, home or business.

Based on the above, the system hardware is selected. The considerations are the system performance, the amount of installed system board memory, the maximum capacity of system board memory, the I/O capability of the base system, the FCC classification and the resources available.

After the system hardware is selected, the optional features required are determined. These optional features include memory expansion, system board memory or Micro Channel system memory, and other feature adapters (for example, coprocessor, ESDI, SCSI, tape, serial port, display, printers, plotters, facsimile). The considerations in the feature adapter selection are the functional capability, the FCC classification, the performance attributes of the adapter and the resource requirements.

If the device driver is provided with the operating system or the application, it is necessary to select an adapter that is compatible with the requirements specified by the software.

After the initial selection of the feature adapters, the total physical resources and system resources required by the system configuration must be reviewed. Any mismatches may require the selection of a different system, and/or feature adapters.

**FCC Classification:** The FCC has two classifications for computing devices. These are class A for commercial or industrial use and class B for residential or home use.

## Physical Resources

There are physical resources that must be considered when a system is being configured. These resources include the maximum amount, or capacity, of system memory that can be supported, the amount of system board memory that can be accommodated, the number and type of Micro Channel connectors on the system board available for feature adapters, the number and type of Micro Channel connectors required by an adapter, and the number of adapters of the same type that can be coresident in a system.

**System Memory Capacity:** The hardware system specifies the maximum system memory capacity that may be accommodated. A system with a 24-bit address bus is limited to supporting a maximum of 16 MB of system memory. A system with a 32-bit address bus optionally may support greater than 16 MB of system memory, up to a maximum specified by the system. The system specifies the maximum system memory that may be accommodated on the system board. The expansion of system memory beyond that accommodated on the system memory board is accommodated by a Micro Channel system memory adapter.

**Micro Channel Connectors:** Each system board has a physical limit to the number of adapter cards that can be accommodated. This limit is specified in the number of connectors that are available. If a system board has three connectors available for options, then three feature adapter cards can be accommodated. Likewise, if a system board has seven connectors available for options, then seven feature adapter cards can be accommodated. When a greater number of feature adapter cards are required than the number of available connectors, either a system with additional Micro Channel connectors or a multifunction feature adapter card is required.

**Micro Channel Connector Types:** There are five types of Micro Channel connectors:

- Type 1 - 16-bit connector
- Type 2 - 16-bit connector with video extension
- Type 3 - 32-bit connector
- Type 4 - 32-bit connector with video extension

- Type 5 - 32-bit connector with matched-memory extension.

Based on the adapter connector type, Figure 11 indicates which of the five Micro Channel connector types that an adapter can use on the system board.

*Note 1:* A 16/32-bit adapter connector is an adapter with a 32-bit connector that is designed to allow insertion into into a 16-bit Micro Channel connector. In this case, the 32-bit adapter will operate as a 16-bit adapter.

**Multiple Occurrences of an Adapter:** Multiple occurrences of an adapter is the maximum number of adapters of the same type (part number) that may be coresident in a system configuration. If configuration requirements cannot be met based upon the multiple occurrence capability of the adapter, it will be necessary to select a different adapter to provide the the required function.

## System Resources

The Micro Channel system and adapters support a Programmable Option Select (POS) facility that eliminates the need for switches on the system board and adapters. An automatic configuration utility and a change configuration utility are provided with the system on the System Reference Diskette. The automatic configuration utility identifies the installed hardware and automatically tracks and allocates system resources (interrupt levels, arbitration levels, DMA channels, I/O address space - I/O ports, and memory address space). Information is used in the creation of the configuration data from the adapter description file, the adapter description program, and, if required, an initialization program. The programs are provided with the system or the feature adapter. When more than one adapter is configured to the same system resource and that resource cannot be shared, only one of the conflicting adapters is enabled.

In special cases, it may be necessary for the user to change the default configuration settings from those set by automatic configuration. The change configuration utility supports a manual procedure that is used to resolve unusual conflicts or for performance tuning.

The above procedure simplifies the management and consideration of the system resources in adapter selection. However, it is advisable for the user to review the following system resource considerations in product selection to prevent a conflict, or to prevent exceeding the available system resources. Both the base adapter and the feature adapter requirements are included in the total system resource requirements for product selection.

**DMA Channel:** Each DMA slave adapter requires one or more DMA channels. The total of the base adapter and the feature adapter DMA channels is limited to the total provided by the system DMA controller. If multiple base or feature adapters require a fixed assignment for a DMA channel, then it is necessary to check for potential conflicts in the selection of a DMA slave adapter.

**Arbitration Level:** Each DMA slave adapter requires one arbitration level per DMA channel assigned to the adapter. Each bus master adapter requires one or more arbitration levels. The total number of arbitration levels for the base and feature DMA slave and bus master adapters is limited to 15. If multiple base or feature adapters require a fixed assignment for an arbitration level, then it is necessary to check for potential conflicts in the selection of a DMA slave or a bus master feature adapter.

| Adapter Connector | System Board Connector | | | | |
|---|---|---|---|---|---|
| | Type 1 | Type 2 | Type 3 | Type 4 | Type 5 |
| 16-bit | X | X | X | X | X |
| 16-bit with video extension | | X | | X | |
| 32-bit | | | X | X | X |
| 16/32-bit (see note 1) | X | X | X | X | X |
| 32-bit with video extension | | | | X | |
| 32-bit with matched-memory extension | | | | | X |

**Figure 11. Micro Channel Connector Type**

**Memory Address Assignment:** A memory slave adapter, or an adapter that supports a memory slave participant, requires the assignment of one or more ranges of addresses within the Micro Channel memory address space. There are two types of memory slave participants that must be considered, Micro Channel system memory and non-system memory. Non-system memory is memory on an adapter that is contained in the Micro Channel memory address space.

*Micro Channel system memory:* The memory address range can be assigned between 1 MB and 16 MB where the system or the adapter limits the address bus to 24 bits. The memory address range can be assigned between 1 MB and 4 GB where both the system and the adapter have a 32-bit address bus. Automatic configuration assumes that the memory address is assignable on 1 MB boundaries. If the memory card does not support assignment on 1 MB boundaries, it may be necessary to perform manual configuration through the use of the change configuration utility. If the Micro Channel system memory adapter requires the support of an initialization program, a fixed disk is required in the system configuration.

*Non-system memory:* For an adapter that contains non-system memory, the memory address must be assigned in the adapter ROM address range (HEX 000C0000 to 000DFFFF), the address range between 1 MB and 16 MB, or the address range between 16 MB and 4 GB. The adapter specifies the number of memory addresses required for each memory address range that is supported.

If an adapter memory address requirement is restricted to the adapter ROM address range and the starting address is programmable on 8 KB boundaries, there is no addressing conflict. This scheme supports up to sixteen 8 KB segments without an addressing conflict. However, for an adapter that requires a fixed address range, it is necessary to check for potential conflicts of the memory addresses. If multiple adapters that are restricted to the adapter ROM address range require greater than 8 KB of addresses each, then it is necessary to confirm that they do not exceed the 128 KB that is available. In addition, it is necessary to confirm that the address range can be configured in a contiguous block without an addressing conflict.

If an adapter supports a programmable range of addresses for non-system memory that is not constrained to the adapter ROM address space, there is no addressing conflict. The memory address can be assigned between 1 MB and 16 MB where the system or the adapter limits the address bus to 24 bits. The memory address range can be assigned between 1 MB and 4 GB where both the system and the adapter have a 32-bit address bus.

**I/O Address Assignment:** An I/O slave adapter, or an adapter that supports an I/O slave participant, requires the assignment of a block of one or more addresses within the Micro Channel I/O address space. The block of I/O addresses may be either fixed, a fixed base I/O address with a programmable Adapter I/O Address Select, or selectable by the Device I/O Address Assignment. If the adapter supports the Device I/O Address Assignment, up to 59 adapters of the same type can be supported in a system. An I/O address space conflict cannot occur for this case. If the adapter supports a fixed block of I/O addresses with a programmable Adapter I/O Address Select, up to eight adapters of the same type can be supported in a system. If the adapter supports a fixed block of I/O addresses, or a fixed block of I/O addresses with the programmable Adapter I/O Address Select, it is necessary to check for potential conflicts of the I/O address blocks between adapters.

## Product Selection Summary

The proper selection and configuration of a system requires the matching of the system hardware and feature adapters to the selected software. The system hardware should have the performance and the capacity to execute the applications of interest. Information concerning IBM PS/2 computer system hardware specifications is in *Personal Systems Hardware Interface Technical Reference*, form number S68X-2330, available from IBM. This technical reference manual will also have information regarding the resources, both physical and system, available to each system.

The information of interest for system units includes:

- System board devices and features
- Performance specifications
- Physical specifications

Adapter information, performance and resources (both physical and system) required, should be obtained from the adapter manufacturer. A catalog of Micro Channel adapters, with manufacturers name, address, telephone number, and a brief product description is avail-

able from IBM. This catalog is *Catalog of Micro Channel Hardware Products*, form number G360-2824.

The information of interest for adapters includes:

- Adapter participant type
- Micro Channel connectors, number and types required
- DMA channels, number and assignment
- Arbitration levels, number and assignment
- Memory addresses, number and assignments
- I/O addresses, number and assignments
- Address bus width
- Data bus width
- Data transfer type and time
- Buffer capacity
- Burst mode support
- FCC classification

Refer to the article, "Information for Developers," in this issue for more information.

It is necessary to determine the resource requirements for each adapter after selecting the feature adapters of interest. The resource requirements are then summed by subcategory within the physical and system categories. The configuration is valid if none of the system hardware resources has been exceeded and there are no conflicts for fixed resource requirements.

If a resource has been exceeded, or there is a conflict for a fixed resource requirement, it is necessary to make new adapter selections and reassess the resources required, as above. If a valid configuration is not obtained, it is necessary to make new adapter selections or to select system hardware with greater resources.

Reconfiguration should not be necessary under most circumstances. Typically, only a system configured for heavy-duty, multitasking operation will exceed system resources. It is more probable that a configuration problem can occur when an adapter requires a fixed resource.

*ABOUT THE AUTHORS*

*Carl H. Grant is a senior technical staff member at IBM's Entry System Division in Boca Raton, Florida and is currently with PS/2 Systems Architecture. He is involved in the development of Micro Channel architecture. Carl joined IBM in 1960 and has previously been associated with development of communications products and architecture, IBM System/360 and 370 Channel and I/O development, IBM Distributed Systems early definition of IBM Token-Ring, and 9370 and AS400 communications IOP development. Carl has an Outstanding Contribution Award for the 2880 Channel, 2305, 3330 Integration and an Outstanding Innovation Award for 8100 Direct Attach Loop. He has two patents and several invention disclosures published in the "IBM Technical Disclosure Bulletin." He has a bachelor of science degree in electrical engineering from the University of Vermont.*

*Donald Ingerman is an advisory engineer in IBM's Entry Systems Division in Boca Raton, Florida. He joined IBM in 1984 and is a member of the PS/2 Systems Performance Evaluation Department. Before joining the Entry Systems Division, Don did performance work on the SPD I/O Bus and several communications projects. He has a bachelor of science degree in electrical engineering from Fairleigh Dickinson University and a master of science degree in operations research from New York University. Don has been in the performance field for over 20 years with Norden Systems, Bell Telephone Laboratories, ARINC Research Corp, and the Library of Congress. He was a communications engineer with the Western Electric Company prior to his work in the performance field. His current responsibilities include PS/2 system performance and Micro Channel system performance.*

*Robert F. G. Robinson is a staff programmer at IBM's Entry System Division in Boca Raton, Florida and is currently with the OS/2 Market Planning group. He teaches classes and seminars on OS/2 and conducts customer briefings and disclosures on OS/2 Standard Edition. Robert joined IBM in 1974 and has previously designed and developed 3270 communications products for the IBM PC and minicomputers. He is coauthor of a book,* OS/2 Presentation Manager Programming, *to be published by Wiley and Sons in November, 1989. He has a bachelor of science degree from Florida Atlantic University.*

# Overview of Extended Micro Channel Functions

*Chet Heath*
*IBM Corporation*
*Boca Raton, Florida*

**In September of this year, IBM released information about extended functions of Micro Channel architecture. This article provides an overview of those functions. Further detail is provided in subsequent articles in this issue.**

The functions that have been disclosed for the Micro Channel architecture standard advance its capabilities to communicate data and control operations between memory, I/O adapters, and processors in a wide variety of applications. Although these new functions will be introduced in systems as the implementations require, no architectural limit specifies high- or low-end exclusivity. The functions are optional extensions of the "basic" Micro Channel specification that are, in general, upward-compatible from existing design points – and in specific situations, they are also applicable to augment existing systems.

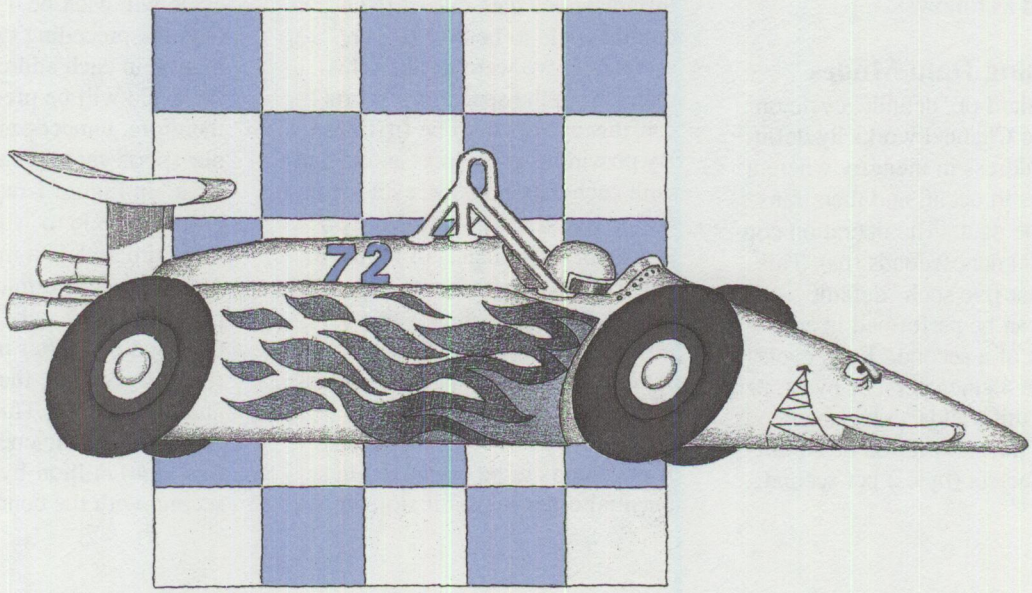
## The Advanced Functions

- Streaming Data Mode – Improved efficiency in the use of time during data transfers yields a potential four- and perhaps eight-fold improvement in the speed of data transfer. While 80 MB data transfer is impressive, no adjustments to the physical structure of the systems, mechanical or electrical definition of the connector interface, or redefinition of existing protocols was required to achieve the increase. While not proven, the potential for 160 MB throughput is promising.

- Parity checking – Parity checking of data and/or address information during channel operations has been standard operating procedure in mainframes for many years. It is used to support the mainframe-like environment that is evolving for advanced microsystems with large memory systems, extremely fast I/O devices, and complex I/O configurations. Parity is typically employed in large-system architectures.

- Synchronous exception signaling – This allows the association of the specific channel cycle with individual errors and enables error data collection to be synchronous with occurrence.

- Subsystem Control Block (SCB) architecture – This capability is similar to functions that have been available in IBM mainframe or minicomputer designs since 1965. It is the enabling software and hardware structure that can coordinate the operation of multiple bus masters in Micro Channel systems. It provides for system control of complex configurations with multiple intelligent subsystems and extends the foundation to support distributed multiprocessing in the future.



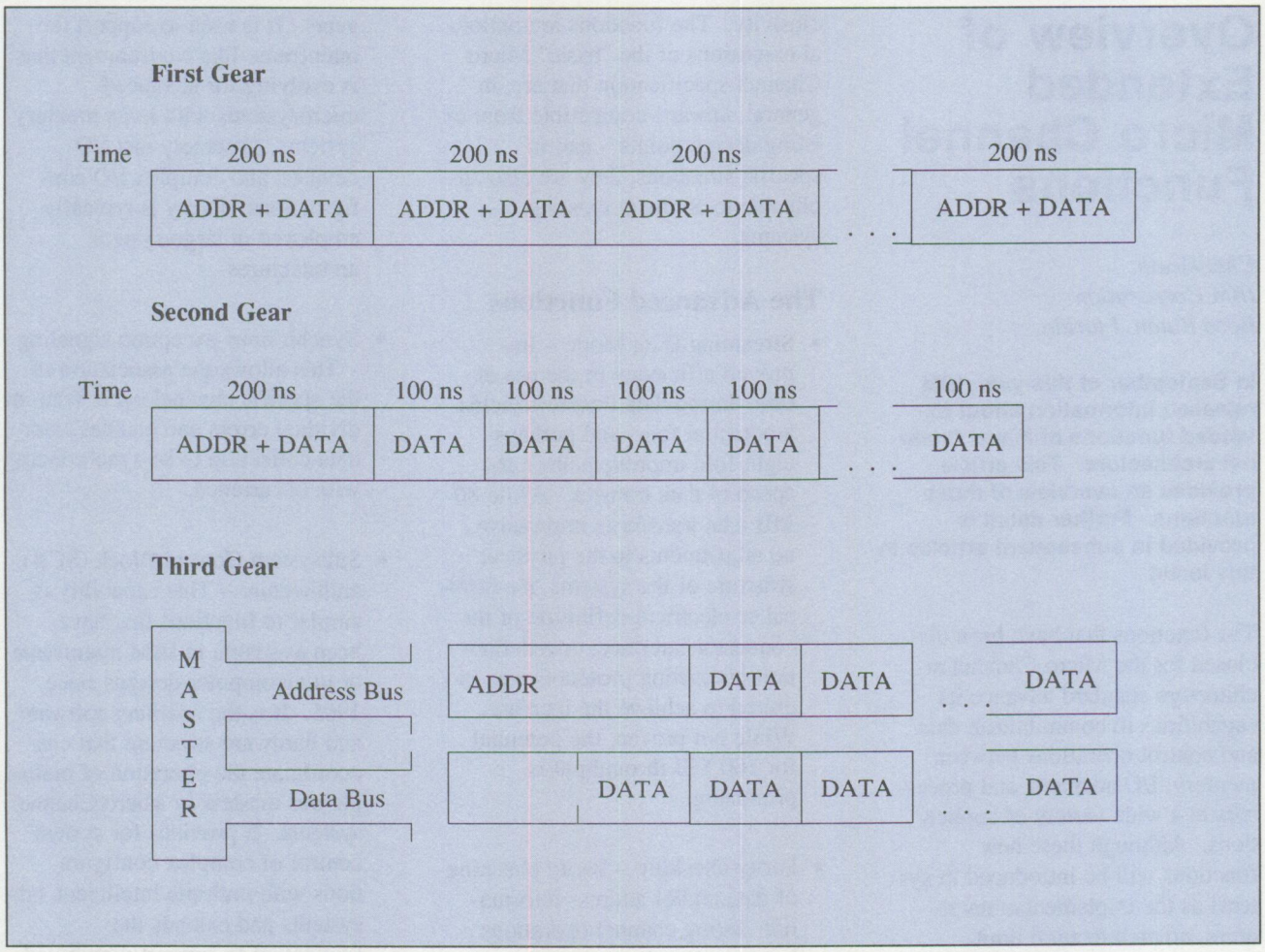



**Figure 1. Streaming Data Mode**

More specifically the functions are explained as follows:

## Streaming Data Modes

The standard or "default" cycle on the Micro Channel works by defining the address in memory where a transfer is to occur, and then transferring the data. The operation consumes 200 nanoseconds (ns). This means that five such "default cycles" can be performed in each millionth of a second. Each cycle can move 4 characters, or bytes, per transfer; this yields an instantaneous rate of 4 times 5 million or 20 million characters (bytes) per second.

This is defining the "throughput," or instantaneous data rate of the default cycle to be 20 MB per second. If we were to liken this data rate to the speed of a sports car, this cycle would be first gear. By providing an address associated with each data transfer, cycles can define RANDOM operation by a processor on storage. In Figure 1, "First Gear" represents this graphically. A block of data is transferred, using burst mode as a "stream" of address and data transfer operations.

If the data is to be organized in sequential order in 32-bit storage, then the address portions of the cycle will each be 4 bytes higher than the preceding cycle. The information in each address portion of the cycle will be predictable and, therefore, unnecessary. To continue our sports car analogy, "Second Gear" in Figure 1 represents a more efficient cycle to "stream" sequentially ordered data into memory. The address information would be presented once only in the beginning and thereafter maintained as a running count by the data's source and destination. This thereby doubles the efficiency of long transfers to 40 million-byte transfers per second with the condition that data

typically be sequentially ordered in memory.

It may be observed that the 32-bit address bus is not imparting information after the first 200 nanosecond period. It is, therefore, available as an additional 4-byte path for data transfer. Four bytes of data moving on the address bus are coordinated with 4 bytes of data transferring on the 32-bit data bus. This yields 8 bytes of data transferred in one operation. This additional mode is called "Multiplexed Streaming Data Mode" and is capable of transferring sequentially ordered data at a rate of 80 million bytes per second. The mode is represented as "Third Gear" in Figure 1. This is important because rates that high can occur when many very high-speed devices (SCSI files, FDDI LAN) transfer data "concurrently" – all at the same time. It can also allow a block of data to be fetched from bus-attached memory fast enough to be usable for a high-speed memory cache elsewhere in the system.

Micro Channel architecture is enabled for additional procedures that will yield up to 160 MB. (Now you're in Fourth Gear!)

**Random versus Sequential Transfers:** A faster cycle might have been designed around matched-memory protocol. When matched-memory protocol was originally defined in the technical reference manual for PS/2 systems in April 1987, it used the asynchronous cycle capability of the Micro Channel interface to define faster cycles to bus-attached system memory.

Matched-memory protocol is simply a modification of the default cycle that can be accelerated to higher speeds, that is, with less time per period. With a 16 MHz, zero wait state system, the time for each period is reduced to 62.5 nanoseconds or one period of the 16 MHz processor clock (Figure 2). It yields a potential throughput of 32 million bytes per second. This is not nearly as fast as streaming data mode, but the data can be fetched randomly. Since no specific order is defined for the data, processor instructions can be fetched from high-speed memory on the bus at usable rates for the processor. There is then a trade-off between the ability to "stream" data at the highest speed and the ability to change the address randomly on every transfer. It is, therefore, important to classify the transfer type – random or sequential – whenever throughput comparisons are made. Random capability is very desirable when a processor fetches instructions and data from bus-attached memory; high-speed sequential data transfers are ideal for high-speed interface to fast I/O devices. A bus-attached processor, operating from a local cache, might use both modes to replenish the cache or control I/O.

In all cases, faster cycle times allow more time for other operations in the machine. The net is that the implementation will determine which is the best mode of transfer.

## Parity Checking

*The motto for the "MG" sports car was "safety fast." It wasn't sufficient to go fast from point "A" to point "B"; you also had to get to point "B" in one piece.*

When we humans communicate, our oral data usually has meaning only in context. The words in a sentence often verify the meaning of other words. For example, if we say "the dog took several bytes out of my leg," the sentence makes no sense unless we exchange the word "bites" for "bytes." If we heard the sentence, we would assume the use of "bites," because it is the only choice that makes sense.

When computers exchange information, all combinations of data are typically valid; eight bits of data, called a byte, that define a data character, can yield 256 combinations of alpha, numeric, punctuation, and graphic characters. To put the data in context, a ninth bit can be added. The state – one or zero – of this ninth bit can be determined mathematically by counting the total number of ones in the data byte, regardless of position, and determining if it is odd or even (see *Note* at the end of this section). By calculat-

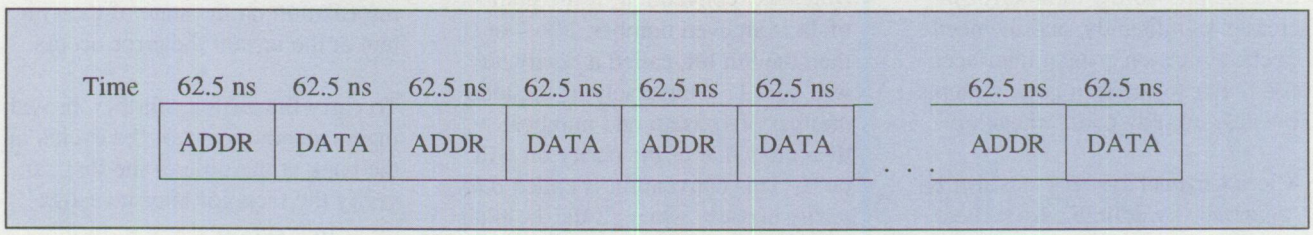



**Figure 2. Matched-Memory Protocol**

ing what the parity bit in the data should be and comparing it to what is actually received, the receiving party can determine if it has been sent good data.

Returning to the example of oral conversation, if someone with a heavy foreign accent were to say "the dog took several bites out of my leg," we might ask them to spell "bites" to verify the meaning; with others we might be more confident of what we have heard, and we would accept the verbal data without checking. We would be "adaptive" in that we would check the meaning for some, but not all, situations. What if the consequences of error were severe? One might imagine a large dog sitting next to a foreigner with a diskette in his mouth; is the foreigner warning us, or making a joke?

In the computer, some interfaces or adapters may have more severe reactions to distorted data than others. An error to the display might yield an invalid symbol in a word, or just one dot that is slightly off-color. Our brains would probably detect and either correct and ignore the error. However, data destined to be sent to a large data base in a mainframe file system might not be reused for months – or worse, it could be propagated into other files without ever being checked. If the data throughput is very high and the storage capability of the system is also very high, as in a mainframe, then the probability of an error increases significantly, and the need to check data on critical interfaces rises. For this reason large systems liberally employ parity checking.

Micro Channel systems can also be "adaptive" by defining error checking capability, only on a vulnerable interface. Alternately, parity could be eliminated where limited consequences of error do not justify the expense. With adaptive parity, the bus master drives a special signal on the interface, which indicates that valid parity information is included in the data transfer. This signal allows the slave adapter to know if the transmitting party will include the ninth parity bit in the transfer or not. If it is there, then the receiving party can check the data. If an adapter receives parity, usually it also generates parity when it is the originating party in a data transfer.

*The bus master drives a special signal on the interface, which indicates that valid parity information is included in the data transfer.*

This definition of adaptive parity allows systems and options to be compatible, even if only part of the system, such as one master adapter and one slave adapter, implements the parity check logic. Furthermore, parity checking can be applied to data transfers and/or address information as well.

*Note:* By convention, if the number of 1s is an even number, like 4 or 0, then the 9th bit, called a *parity bit,* will be a 1. Conversely, if the total number of 1s is an odd number, then the value of the parity bit will be 0. This convention is called *odd parity* because when all the 1s in all 9 bits are counted, the number will always be odd. For example, if the 8 bits that make up the character byte of data are 0,0,0,1,0,1,0,1 or 1,1,1,0,1,1,0,0 each have an odd number of 1s, so the value of the 9th or parity bit is 0, yielding an odd number. By contrast, the 8-bit value 0,1,0,1,0,1,0,1 would yield a parity value of 1.

## Synchronous Exception Signaling

When errors (for example, parity errors) occur in high-speed systems, delays in detection can cause status of the error condition to be presented after a transient condition that caused the error has passed. This is analogous to the situation of a burglar alarm in the bank. If the authorities arrive after the perpetrators have departed, the money may be gone forever. If, however, a guard stationed in the building can catch them in the act, the money can be retained, that is, the error and its source can be "logged," and the error causing condition can be, perhaps, arrested. In the computer it is the data that is valuable although it may represent the customer's money.

For example, if a communications interface is used to transfer data into a system, identifying an error in a long transmission would allow retransmission of just the segment containing the error and not the entire record. A similar example exists for tape input/output. Exact identification and reporting of the error condition may require detailed information on the state of the system at the instant the error occurs.

To carry the analogy further, providing a camera record of the events in the bank at the time of the loss can verify the facts for later investigation. In a very real sense, further similar events can be reduced if the

diagnosticians can pinpoint the events that lead to the to error and prevent future similar "loss" situations. Synchronous exception signaling is then enabling the "arrest" of participants at the instant of potential loss, making collection of the evidence as to the nature of events leading to the potential loss of data or system control possible, and hopefully allowing full recovery of the data and the system from the situation.

This function is in addition to the asynchronous channel-check capability in the basic Micro Channel definition. The basic function works more like a "bank examiner" and will not necessarily identify the error but will alert the system that an error has occurred. It is saying, "the books don't balance at XYZ bank," allowing a log of errors and associated cards to be created.

## Basic SCB Architecture Concepts: Control Blocks and Chaining

**What it does:** SCB architecture defines the hardware and software control means for:

- attachment of multiples of bus master attachments
- error and status reporting protocols
- dynamic assignment of system resources required to support concurrent activity of multiple masters.

It also introduces a powerful concept of "command chaining" where masters can decide, according to prescribed algorithms, which of several potential control paths to follow. This allows the paths for coordination of bus master operations to be predefined as "macro" operations while giving the subsystems the latitude to respond to changing conditions at the time of operation. The SCB architecture greatly avoids the overhead of interrupt operation and anticipates the migration of mainframe operating system principles into the personal systems environment.

*"Command chaining" allows masters to decide which of several potential control paths to follow.*

Control block architectures have been defined for IBM mainframe and minicomputer products for approximately 25 years. However, personal computers, including the PC, PC XT, and AT series of products, depended on interrupt requests and the Basic Input/Output System (BIOS) as the interface between the operating system and I/O adapters. Interrupts and BIOS allowed adapters to evolve to more powerful designs but still placed a heavy burden on the processor.

*Avoids Interrupts:* Systems with processors designed for rapid interrupt handling are called real-time systems and typically employ a separate set of processor registers for each interrupt level. This is one area where the single-tasking legacy of the personal computer is most telling. Typically, it can take 200 processor instructions in these single-tasking machines to respond to an interrupt. One thousand or more instructions per interrupt are not uncommon for a multitasking operating system. Why so many? Without a real-time interrupt structure, it can take at least that number of instructions to save the environment (environments, in a multitasking operating system), to operate the I/O and restore the original environment.

If only one user and one task are active at a time, as in most PC applications under DOS, processing will wait on slow I/O. In this environment, large interrupt overheads can be hidden by the I/O delays. The existing scheme for operating system to I/O interface is inadequate as we move to a "multi" environment, where I/O adapters can be as concurrently active as the tasks that stimulate them. The cumulative overhead from many concurrent interrupts can severely load a system's performance. For example, if you want to increase the performance of an asynchronous communications interface, what do you do?

Most often the solution has been to keep the asynchronous adapter card and the communications application, and replace the system processor. This increases the rate at which the communications application runs. If a 6 MHz machine won't do, then get an 8 or 10 or 12 MHz system to run the asynchronous adapter faster. Soon we have a lot of money tied up in expensive memory to hold the application for fast processors to run the serial adapter at a faster rate. A better solution to this problem is to employ a bus master to control the serial interface and maintain the user's investment in the system unit.

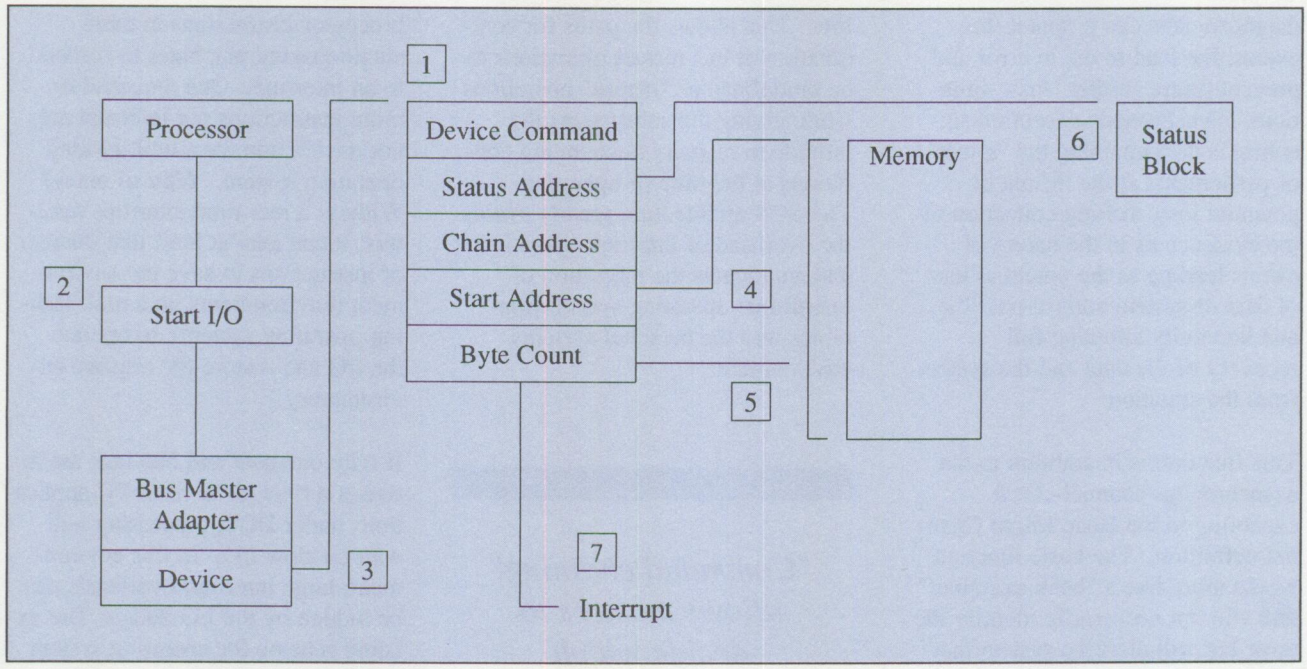



**Figure 3. Process of a Generic SCB**

This releases the processor for operating system and applications.

But what about the software interface to control bus masters?

So far, discussion about the feasibility of bus masters in systems has focused on arguments of throughput to main storage, efficiency of arbitration, or what the best algorithm is to ensure that each adapter gets a fair share of the channel time. Many of these arguments center on assumptions about system structure and elevate the importance of performance over issues of system integrity, error recovery, diagnostics and intersystem interoperability. These are the real issues! The software interface between the operating system and I/O adapters is the key to the resolution of those issues.

*Builds in the future:* Subsystem Control Block (SCB) architecture can enable the design of ultra high-speed, peer-to-peer communications between adapters and simpler and faster operating systems, and can reduce much of the burdensome interrupt overhead that is the real bottleneck to performance. SCB architecture is defined to allow intelligent bus master systems to work together smoothly and to allow a graceful evolution to mainframe-like operating systems in workstations and personal systems. And that IS the future!

Quite simply, the SCB architecture depends on the ability of the Micro Channel bus master adapter to randomly access and control memory or I/O directly, rather than requiring the processor to move data or control the adapter.

*Frees the processor:* With SCBs, the processor creates a block of data in common memory where a bus master can read it and follow the commands defined in the block. A generic SCB might work as shown in Figure 3.

1) The processor prepares an SCB (it may be prefabricated and stored on a disk or be constructed on the fly).

2) The processor issues commands to the bus master adapter, indicating the location of the SCB in memory.

3) The bus master arbitrates for control of the system and fetches the SCB. The command from the SCB can be delivered by the adapter to the device without processor overhead.

4) The bus master then transfers data with memory starting at the address specified in the Start Address field.

5) The transfers continue until the length of the data specified by the byte count has been reached.

6) Should an error occur, recovery is facilitated by automatically writing the adapter and/or device status to the address specified in the Status Address field.

7) After all operations are complete, the processor is interrupted to indicate the adapter is now free for another command.

If this is all that SCB architecture could do, it would be an improvement over interrupts and BIOS because it would typically amortize the interrupt overhead over many data transfers and free the processor to perform other duties after the initial command. Yet SCB architecture can do more.

A much more powerful function called "command chaining" is enabled by SCB. The operation begins as in the simple case above, but omits step 7 and continues on to steps 8 through N (Figure 4).

**What SCB Chaining Can Do:** SCB data chaining can allow the concatenation of data buffers that have been fragmented by virtual

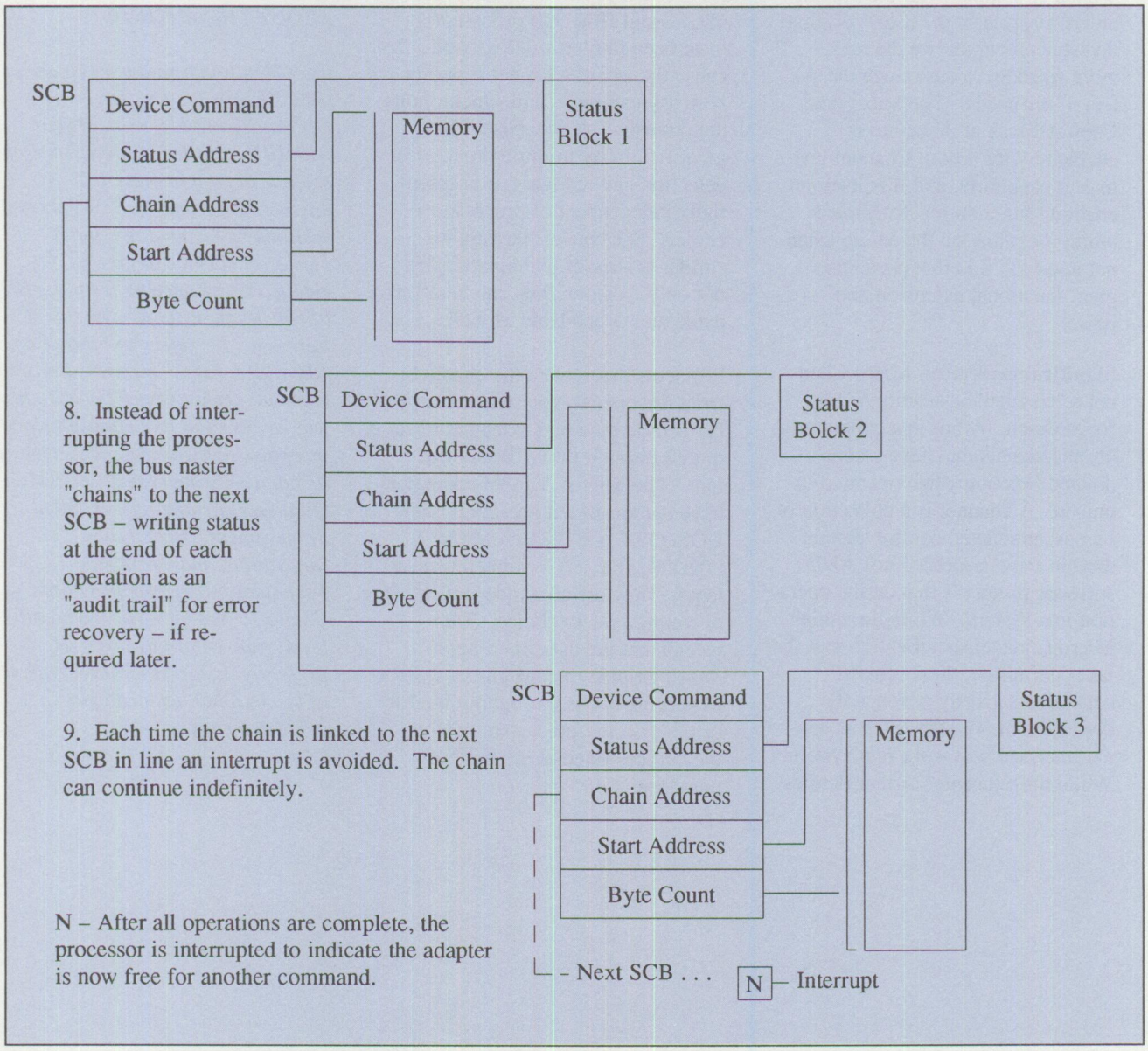



**Figure 4. Command Chaining**

addressing, and can define macro operations, like background backup, and device-to-device offline copy. It can enable many powerful functions that do not involve the processor except to initiate and terminate operations or to handle recovery procedures should an error occur. The performance of the system as a whole is not dictated by the capability of the processor that originally shipped with the system unit; this protects the user's original investment and allows the user more freedom to customize the system to his needs! The Subsystem Control Block architecture is enabled by the Micro Channel bus master capability, and it is itself an enabling function for distributed multiprocessing on the Micro Channel interface, and that portends great functional extension and power.

**Significance:**We call Micro Channel a "channel" instead of a "bus" for a reason. A bus is a collection of data signals that have a coordinated function when operated in unison. A channel is a collection of busses, associated control signals used in transfer procedures AND software protocols that define operation in a system. While the initial Micro Channel specification met the latter definition, the disclosed capabilities greatly advance the definition of how the Micro Channel interface will work in a system. While the extension further defines the "rules" that govern data movement in the system, the designer is offered more, not less, latitude in design alternatives.

It is important to remember that the Micro Channel interface has already supported designs that attach 8088, 8086, 80188, 80186, 80286, 80386 and 80486 Intel processors. Attachment of Intel i860 RISC and IBM RT® RISC processors, Motorola 68030, and IBM 370 processors have been implemented as well. Because the Micro Channel supports both asynchronous and synchronous data transfer, and because it has great flexibility in throughput, error detection and recovery, it is essentially independent of processor choice. Systems of like and dissimilar processor combination are not only feasible; they can be affordable and dependable as well.

Protocols that isolate the channel from the operating system can now be well-defined and extended as required through the SCB architecture. The Micro Channel computers have demonstrated operation with DOS, OS/2, OS/2 EE, VM386, UNIX®, and 370 VM operating systems. They are also independent of operating system choice. This means that the user has a maximized latitude to configure a system to varying needs throughout the life of the system, rather than having the choices fixed at the time of purchase.

Enhanced design latitude, extendability, and adaptability allow the user to maintain the value and capability of the system over time. This is, of course, good news. The best news of all is that the advanced functions have been disclosed for Micro Channel architecture without altering or obsoleting the originally published specification.

*ABOUT THE AUTHOR*

*Chet Heath is a senior engineer at IBM's Entry Systems Division laboratory in his twentieth year with IBM. He holds a bachelor of science degree in electrical engineering from the New Jersey Institute of Technology and a master of science degree in electrical engineering from the IBM LSI Institute at the University of Vermont. He refers to himself as "the oldest living survivor of Micro Channel architecture" because he was involved in its definition since inception and gave the architecture its name. He has received IBM Quality, Outstanding Technical Achievement, Outstanding Innovation, and Corporate Technical Achievement Awards. Chet also has attained the eighth level of invention awards. He is presently assigned to development of Micro Channel strategic enhancements.*

# SCB – An Architecture for Micro Channel Bus Masters

*Frank Bonevento, Ernie Mandese, Joe McGovern and Gene Thomas*
*IBM Corporation*
*Boca Raton, Florida*

**The Subsystem Control Block (SCB) architecture was developed by IBM to standardize the engineering task of designing Micro Channel bus master programming interfaces as well as the programming task of supporting them. This article provides an introduction to the architecture, discusses some of its underlying concepts, and describes its delivery service facilities.**

The Micro Channel bus master facility provides a structure for the independent execution of work in a system. Independent execution of work by bus masters frees a system unit, for example, to perform other tasks. Doing more work in parallel typically improves the response time and throughput of the system as percieved by the end user.

The SCB architecture builds on the Micro Channel bus master capabilities by defining the services and conventions needed to design, implement, and use bus masters effectively. The SCB architecture supports many functions found in larger IBM systems designed to facilitate multiprocessing. Command chaining, data chaining, signaling masters, and a free-running duplex control block delivery service have been provided. Finally, the ability to support user-defined control blocks provides a means for dealing with the requirements of existing, current, and future applications.

## Purpose

The SCB architecture provides a programming model for the Micro Channel and a definition of the logical protocols used to transfer commands, data, and status information between bus master feature adapters. The SCB architecture employs Micro Channel architecture as its underlying physical transport mechanism.

The term "control block" in the architecture name refers to the organization of the control information (commands) into areas that are separate from the data areas. The separation of control and data is used to increase performance, raise the level of functional capability, and provide the implementation independence required by today's applications.

The SCB architecture defines a control block structure for use between entities in a system unit and entities in feature adapters that have Micro Channel bus master capabilities. "Entity" in this context is a collective term referring to both device drivers and/or resource support found in a system unit, as well as device interfaces and/or resources found in feature adapters. The architecture also defines how control block delivery is provided between entities in a system unit and a feature adapter, as well as between entities in feature adapters on the Micro Channel.

The contents of the control blocks have been defined so that they offer a great deal of functional capability while, at the same time, providing implementation independence between the entities in the system unit and the entities in the feature adapter. This allows entities in a system unit to offload or distribute work to entities in feature adapters, thereby freeing them to perform other tasks. It also allows entities in a feature adapter to optimize their implementations without concern for entity interactions or internal implementation details.

The level of interaction between entities of the architecture is at a logical protocol level in order to insulate users from the details of, and interactions with, the underlying implementations. This means that applications written to adhere to the architecture and the underlying implementations that conform to it will remain viable even as technology advances.

Thus, as a feature adapter or system unit is enhanced to take advantage of new technology (higher data rates on the Micro Channel, the existence of data and/or address parity checking, and so on), it will be possible to continue to use existing applications with little or no change.

## Scope

The SCB architecture has been designed to have broad application and be used in a variety of areas, including the following:

**Traditional I/O Protocols:**Traditional I/O protocols typically have a single system unit that requests a feature adapter to perform work on its behalf. This type of feature adapter might be found in an intelligent file subsystem on a personal system or a workstation.

For the traditional I/O systems, application of the architecture would

probably mean use of the Locate mode form of control block delivery. In Locate mode, only the address of the control block is delivered to the subsystem. The subsystem uses this address to locate the control block and to fetch it into its own private storage area for execution. In these I/O systems, a close synchronization is usually maintained between the request and the response to the request.

**Peer-to-Peer Protocols:** In peer-to-peer protocols, requests flow not only from the system unit to feature adapters, but also from feature adapter to feature adapter without direct involvement of the system unit. This might be found in feature adapters that serve as local area network (LAN) bridges or gateways, providing routing for file servers on LANs. Peer-to-peer protocols may also be found in loosely coupled multiprocessing systems.

**Communications Protocols:** Communications protocols may require routing to network servers within a feature adapter and the handling of replies that arrive out of sequence and are interspersed with requests. They may also require the ability to handle the movement of large amounts of data at very high speeds.

**Response-Time-Critical Applications:** Response-time-critical applications require fast response time. The architecture defines a full-duplex, free-running delivery capability to support multiple work requests and replies. The amount of data transferred as well as the speed of the data transfer are critical in this environment.

In the last three categories, application of the architecture would probably mean the use of the Move mode form of control block delivery. In Move mode, control blocks are moved from the requesting, or client, entity to the providing, or server, entity using a shared storage area. This shared storage area behaves like the named pipe function found in several of today's operating systems. Delivery using pipes is free-flowing and defined so that separate pipes exist for both inbound and outbound flows to and from the server (full duplex operation). This form of delivery is commonly used by controllers that have very high arrival rates of work requests and/or run in an asynchronous manner.

The SCB architecture is intended for use by both IBM and non-IBM developers designing bus master feature adapters for the Micro Channel.

## Architecture Overview

The following sets forth some basic concepts used in developing the SCB architecture, its hardware context, system context, and its service structure.

### Concepts

In the personal systems and workstation environments, there is a need to establish a higher-level, control block-oriented interface between support operating in a system unit (system-side entities) and support operating in a feature adapter (adapter-side entities). There is also a need for the same higher-level interface between support operating in two different feature adapters.

The nature of this interface is such that the control block information and the data can be considered as two separate parts of the communication between two cooperating entities.

Control block delivery service is used by each entity to communicate control information. Based upon this control information, there may also be data to be communicated between the two entities. This is referred to as "data delivery." Additionally, there may be information required during system initialization to tailor the delivery support to a particular configuration. This is configuration information.

Figure 1 gives an overview of the control block, data, and configuration delivery support.

### Capabilities

Understanding the SCB architecture starts with understanding the requirements that are common to designers of Micro Channel bus master hardware and engineering software. From these requirements, the capabilities of a bus master feature adapter can be identified. The following are some of the most commonly requested capabilities.

**Provide a Programming Model for the Micro Channel:**

- Provide flexibility
- Support function distribution
- Support signaling among all bus masters and their users
- Provide a higher functional level of interface
  - — Support command chaining
  - — Support data chaining

- Provide request / reply protocol between users
  - — Provide means of correlating requests to replies
  - — Provide source and destination identification
  - — Allow multiple outstanding requests
  - — Allow unsolicited operations
- Allow data to be carried within control blocks

**Support Bus Masters on the Micro Channel:**

- Provide a processor-independent architecture
- Provide full duplex operation
- Support feature adapter-to-feature adapter operation (peer-to-peer)
- Support asynchronous operations

**Improve Performance and Throughput:**

- Reduce interrupt overhead
- Support expedited requests

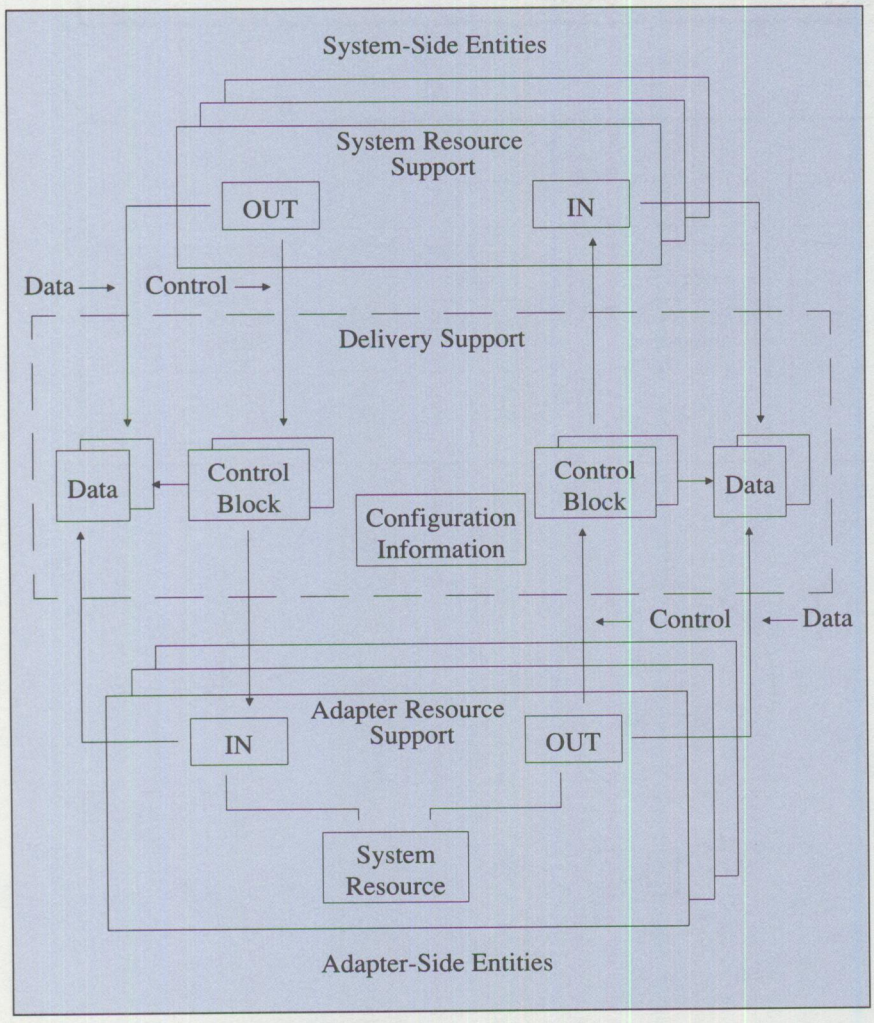



**Figure 1. Overview of Delivery Support**

## Hardware Context

To better understand the structure of control blocks and the overall structure of the control block delivery service, it may be helpful to provide an example of the hardware within which the delivery service is expected to operate.

Figure 2 is an example of a hardware configuration that would benefit from using control blocks and the control block delivery service. This example shows several types of bus master feature adapters present on the Micro Channel, each using the bus master capabilities of the Micro Channel, and each providing support for one or more resources or devices. The example shows that there are multiple types of I/O system support within the system unit, each providing access to and support for resources and/or devices associated with a particular subsystem and feature adapter.

The functions available on bus master feature adapters are also used to establish peer-to-peer relationships between feature adapters, as well as between system units and feature adapters.

## System Context

It is helpful to have an example that shows where control blocks and control block delivery fit relative to a system unit or feature adapter operating environment. This can easily be done for a system unit, but is much more difficult to do for the many different types of feature adapters. This is due to the nature

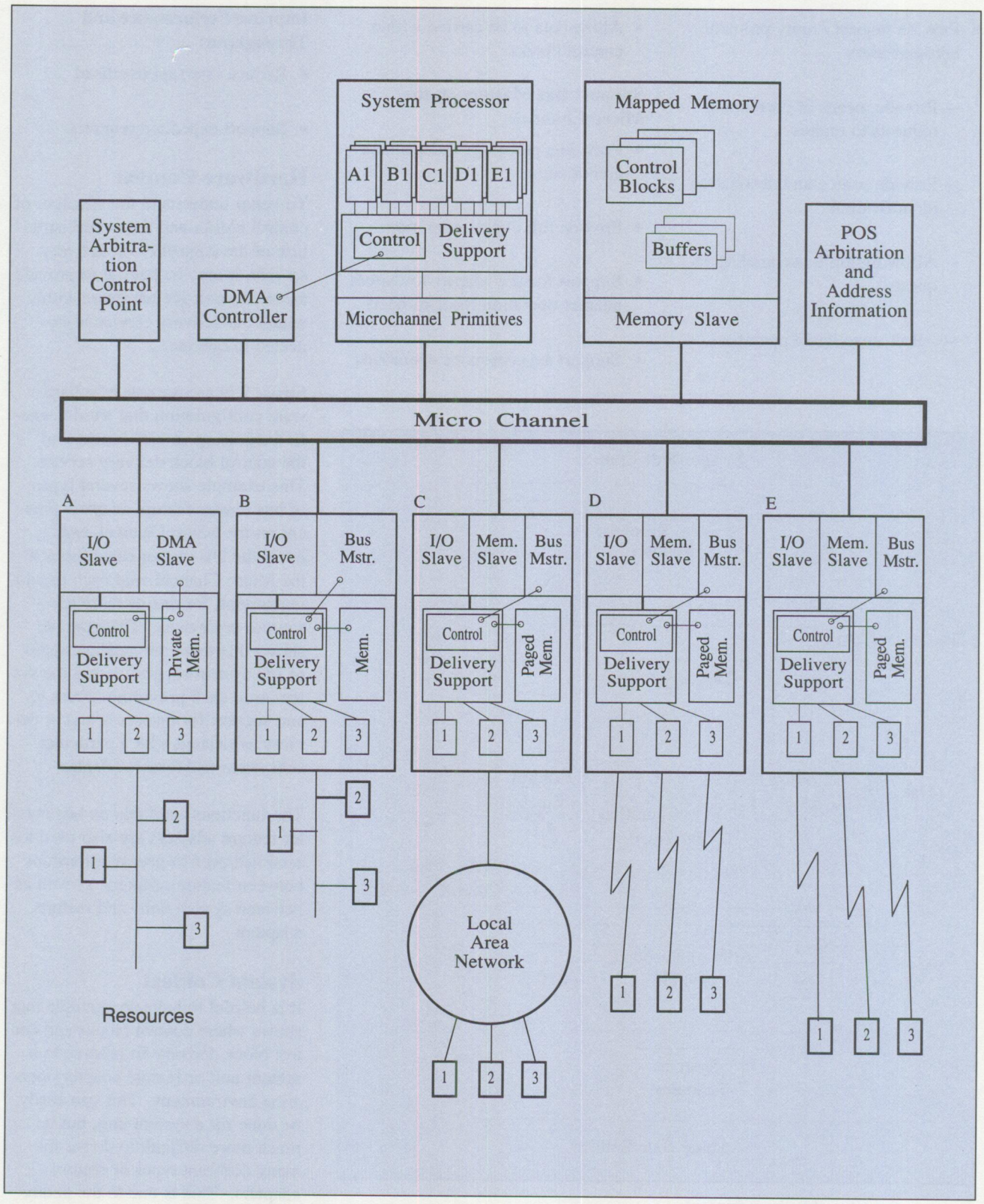



**Figure 2. Hardware Environment**

of feature adapter implementations. At the low end, the operating environment may consist of nothing more than hardware logic and state machines. At the high end, it may consist of a powerful microprocessor with paged memory and a multitasking kernel or operating system.

Figure 3 is an example of how the delivery service maps into a system unit in a DOS or OS/2 type of operating environment.

## Service Structure

The control block delivery service may be viewed as supporting communications between client entities located in the system unit and server entities located in a feature adapter. The delivery service itself is distributed between the system unit and the feature adapter. The portion of the delivery service local to each is the local delivery agent. Delivery agents communicate with each other using the services provided by the Micro Channel. Micro Channel services include memory and I/O space that is shared between the system unit and the feature adapter. This view of the delivery service is shown in Figure 4.

The delivery service supports the delivery of control blocks between pairs of entities (a client and a server). These entities build and interpret the control blocks. The ac-

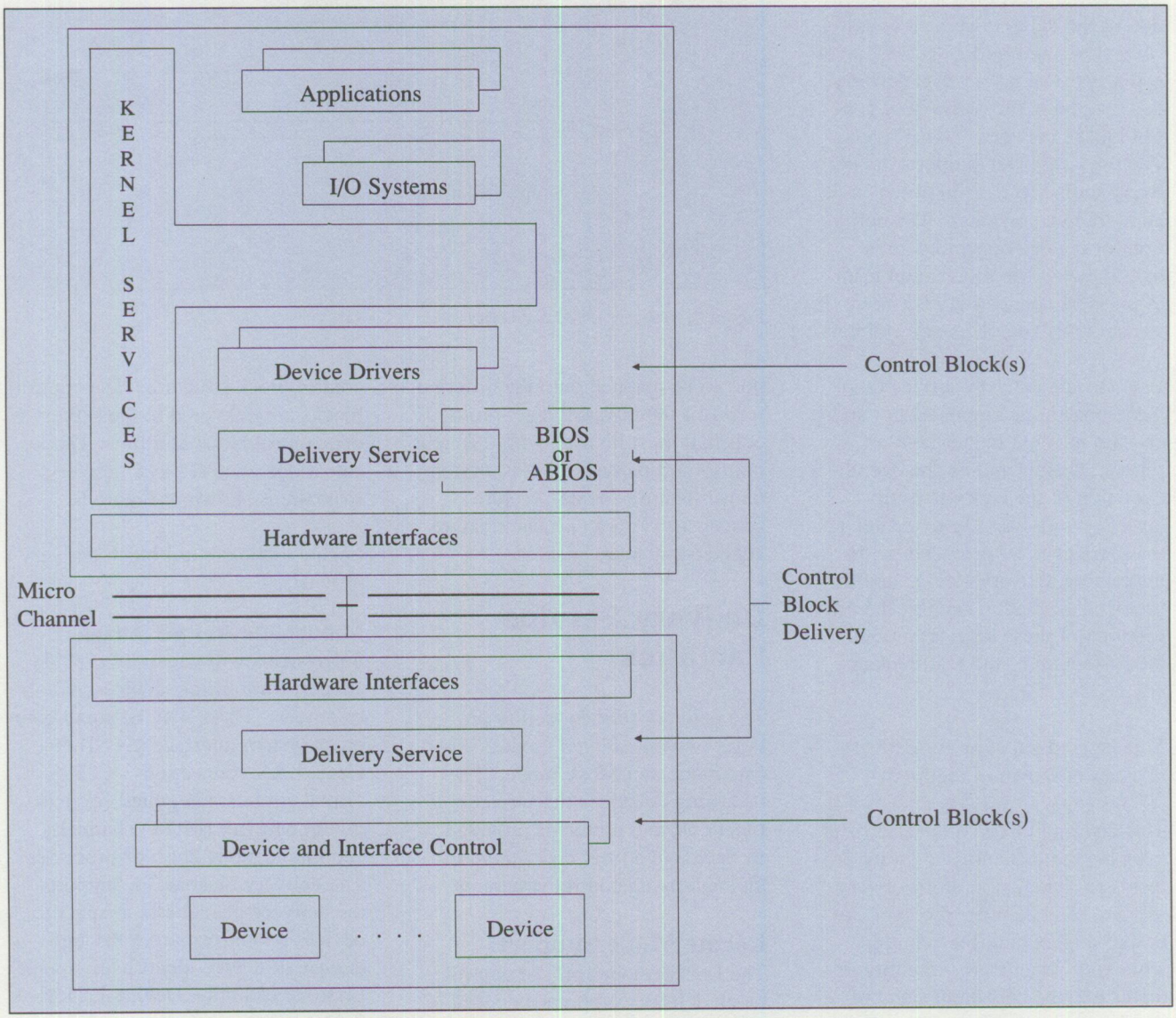



**Figure 3. Example of System Unit Operating Environment**

tual control blocks, their content, and their sequence determines the specific entity-to-entity protocol being used between a client and server.

Both the control block delivery protocol and the entity-to-entity protocol(s) may use the shared memory to physically pass control information and data between the system unit and feature adapters.

The internal operation and distribution of the delivery service is logically structured into a delivery layer and a physical layer. The delivery layer supports the delivery of control blocks between "entity" pairs. The physical layer supports the delivery protocols used between "unit" pairs (delivery agents). The definition for the delivery protocol is based upon a Micro Channel form of physical connectivity between system units and/or feature adapters.

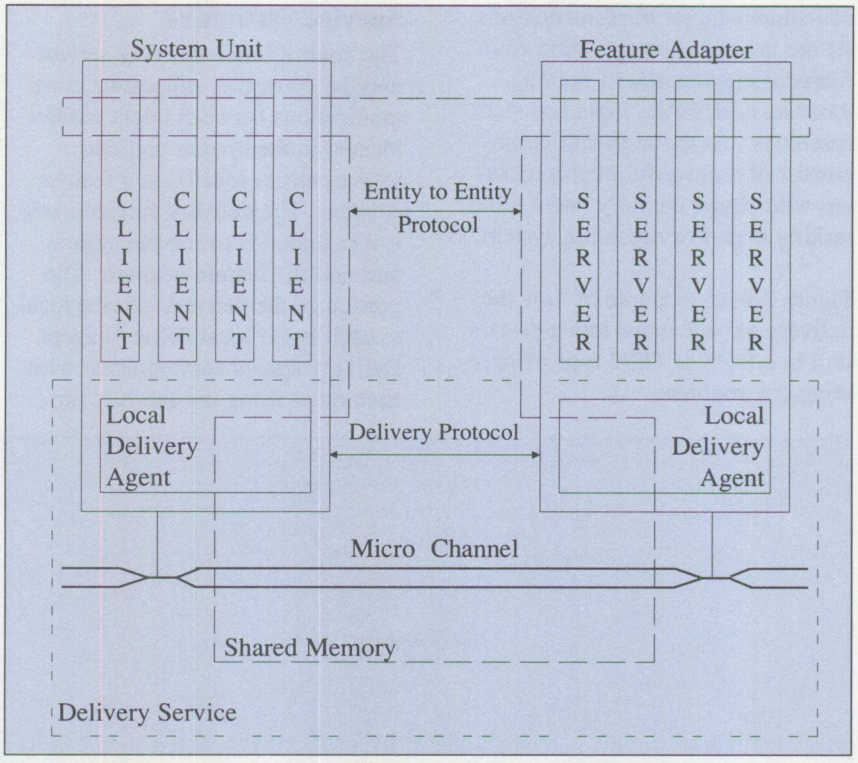



**Figure 4. Generic Delivery Service**

Users of the delivery service (entities representing clients and/or servers) are the next higher layer of service. Understanding the overall operation of the entity-to-entity layer services was key to defining the services to be provided by the underlying delivery service and the protocols needed to support the distribution of these services among the system units and feature adapters.

This layered structure for delivery services is shown in Figure 5.

This layering of delivery services allows the various entity-to-entity protocols to share a common delivery service. The delivery protocol can be full-duplex and free-running while individual entity-to-entity protocols may be half-duplex and of the master / slave form. The delivery protocol allows the delivery support to be mapped onto the different forms of the physical level protocols that must be used with different unit-to-unit pairings; that is, system unit-to-feature adapter, feature adapter-to-system unit, and feature adapter-to-feature adapter.

## Delivery Service Facilities

The delivery service facility provides two operational modes: Locate mode and Move mode. The following describes the services, functions, and protocols provided by each and a brief description of the underlying control structures.

### Locate Mode Support

The Locate mode form of control block delivery supports a control structure that has a relatively fixed control block structure. The control blocks are delivered to the server one control block at a time. The Locate mode control block delivery structure is shown in Figure 6.

Locate mode control block delivery provides:

**Multiple Devices per Adapter:** Subsystems will generally provide support for multiple devices and/or resources. These may be small computer system interface (SCSI) devices, LAN connections, X.25 virtual circuits, integrated-services digital network (ISDN) channels, communications lines, or processes. The delivery mechanism supports the delivery of requests to specific devices and/or resources through the use of device identification numbers (for example, Device 1, Device 2, Device n).

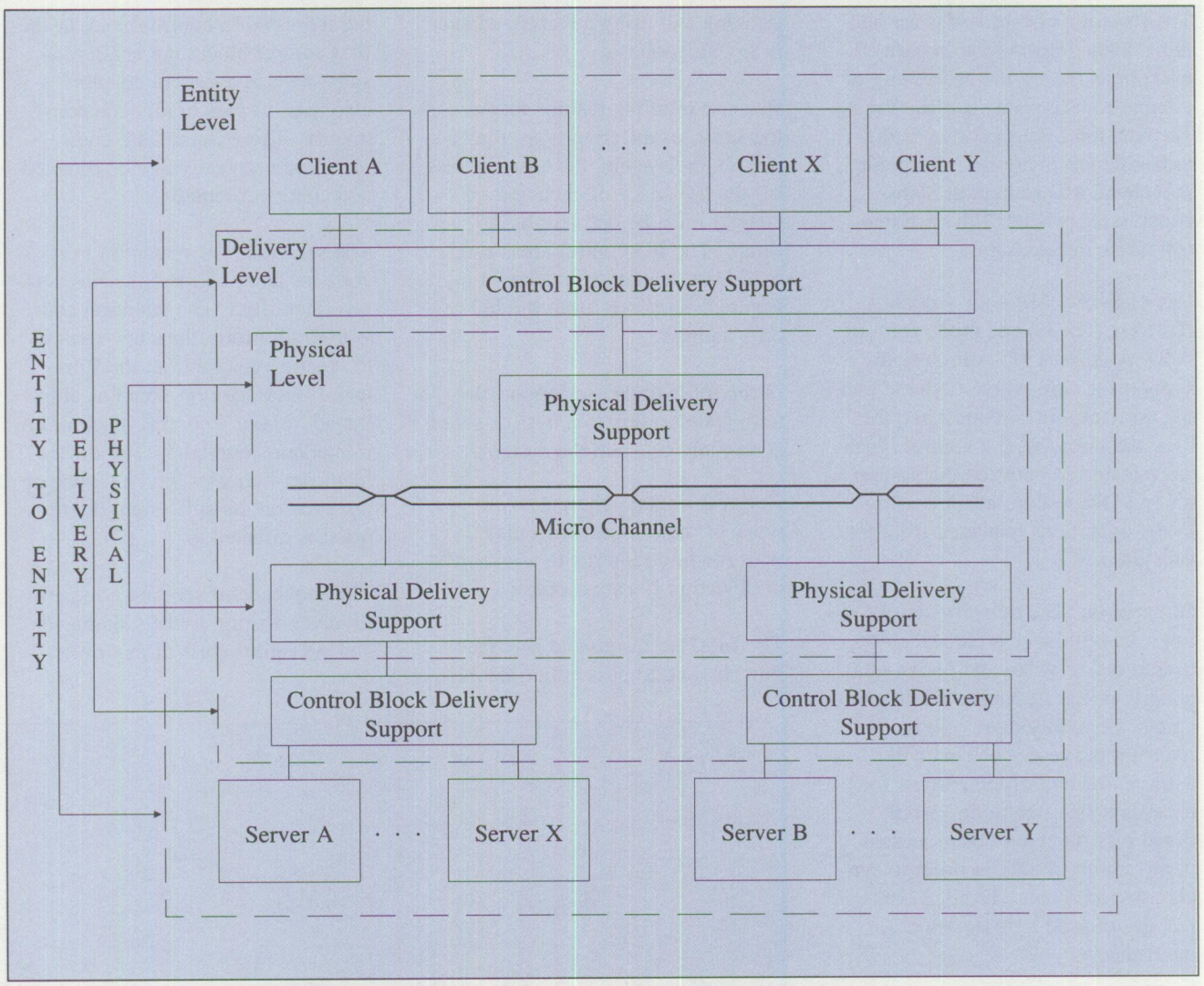



**Figure 5. Delivery Service Structure**

**Subsystem Management:** There is a requirement to deliver subsystem management information as well as control information to the subsystem. The subsystem manager is assigned Device identification number 0 and receives all subsystem management information.

**Requests to Devices:** To use a device or resource, a program (the client) sends requests in the form of control blocks to a specific device or resource (the server) and receives replies from the device or resource for those requests. A single program or multiple programs operating in a system unit can be using a device or resource in a subsystem. These programs may be using the same or different devices or resources in the subsystem.

**Command and Data Chaining and Detailed Status:** The control structure defined for Locate mode provides for an immediate request, a request made up of multiple control blocks chained in a specific order and treated as one logical request, or using an Indirect List for chaining multiple buffers associated with a given control block.

Figure 6 shows a sample control request structure that consists of two control blocks (command chaining). The first control block uses an Indirect List to reference multiple buffers (data chaining). The other has a single buffer. The commands and parameters are contained within the individual control blocks.

The structure also provides for handling status information in case of exceptions during the processing of a request. The status is placed in a Termination Status Block. In order to handle termination at any point in a chain, a Termination Status Block is associated with each control block in the chain.

**Use of Direct Memory Access (DMA):** The Locate mode form of delivery defines the interfaces to support the case where both the control structure for a request and the data associated with a control block are transferred between the system unit and the feature adapter using DMA operations managed from the subsystem.

**Interrupts:** The delivery service allows the builder of a request structure to define when and under what conditions interrupts should be generated. Generally there will be one interrupt per request. This will occur at the end of the request for exception-free operation. Additional interrupts may be requested in any control block in order to synchronize activities. Explicit commands are used to reset device interrupts.

The flow of the interrupt processing is from the hardware to the operating system kernel to the device driver (interrupt portion) and, when ABIOS is used, to the Advanced Basic Input / Output System (ABIOS) interrupt entry point. The interrupt processing uses information provided as part of the request/reply interface to determine which of the devices or resources in the subsystem caused the interrupt.

***Locate Mode Control Areas***

The architecture identifies the specific control areas in I/O space to be used, as well as the protocols for initializing and using a feature adapter in Locate mode.

Because multiple feature adapters of the same, or different types may be used in the system, the base address for the I/O space of each feature adapter must be defined during setup. The I/O control areas used by a feature adapter are shown in Figure 7 as offsets from the I/O base address.

In the following descriptions, the term "port" refers to a byte or set of contiguous bytes in the system.

**Request Ports:** There are four types of request ports associated with sending requests to a resource or device in Locate mode.

The first, the Command Interface Port, is used to pass either the 32-bit address of a control block or the first control block in a chain to a subsystem in a feature adapter. It is also used to pass immediate commands. These immediate commands are typically device-directed and control-oriented.

The second is the Attention Port. It contains an attention code and a device identifier. The attention code is used to inform the subsystem in the feature adapter that the Command Interface Port contains either the address of a control block or an immediate command. The device identifier indicates which device or resource on the subsystem the request is directed to.

The sequence of sending a request involves writing to the Command and Attention Ports in that order.

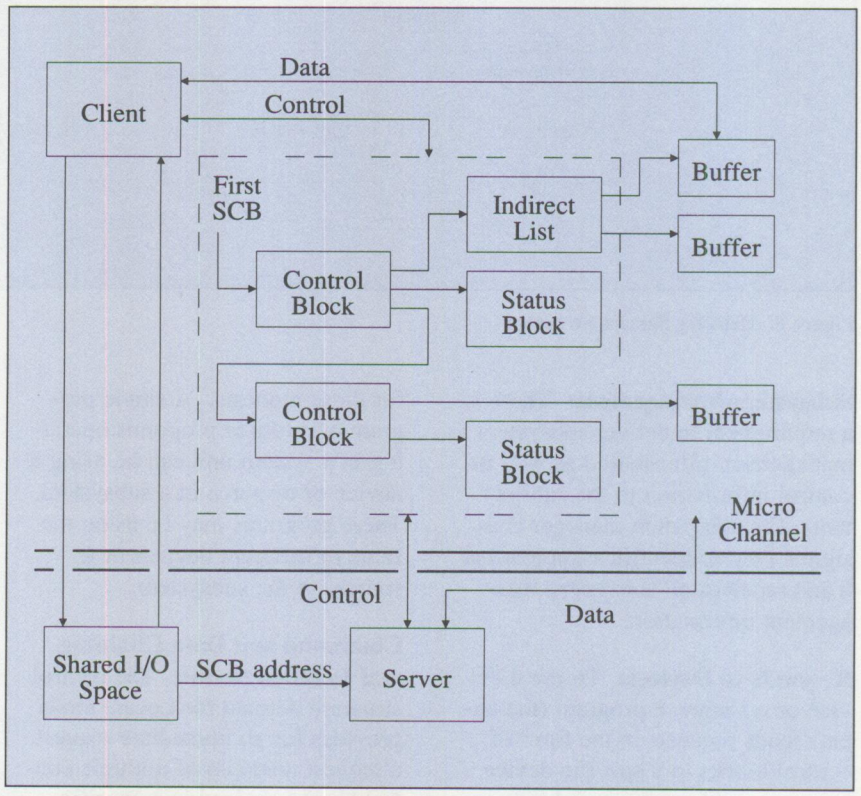



**Figure 6. Locate Mode Control Block Delivery Structure**

The third, the Interrupt Status Port, is used when the subsystem has completed processing a request or immediate command. It provides information needed by the system unit to associate a Micro Channel interrupt with a specific device on the subsystem. In addition to identifying the interrupting device, it also indicates whether or not an exception condition exists.

The fourth is the Command Busy/ Status Port. It is used by the subsystem in a feature adapter to serialize access to the shared logic of the control block delivery service, the subsystem, or the device/resource. The port contains the following indicators:

- Busy - indicates that the subsystem is busy (using the shared logic). Commands submitted while Busy are ignored by the subsystem in the feature adapter.
- Interrupt Valid - indicates that the contents of the Interrupt Status Port are valid and that the subsystem has requested an interrupt on behalf of one of its devices or resources.
- Reject - indicates that the subsystem has rejected a request (a Reset is needed to clear a Reject and allow the subsystem to resume accepting requests).
- Status - indicates the reason for the rejection.

**Subsystem Control Port:** The Subsystem Control Port is used to pass control indicators directly to a subsystem that cannot be easily handled by requests to subsystem management. The port contains the following control indicators:

- Enable Interrupts - indicates that interrupts should be enabled or disabled for all devices attached to the subsystem in the feature adapter.
- Enable DMA - indicates that DMA operations should be enabled or disabled.
- Reset Reject - indicates that a reset of the reject state of the subsystem should be performed.
- Reset - indicates that a reset of the subsystem and all devices in the subsystem should be performed.

**Device Interrupt Identifier Ports:** Interrupt status for all devices and resources associated with a subsystem are reported to the system unit through the Interrupt Status Port. However, when the optional Device Interrupt Identifier Port(s) are used, only the interrupt status for immediate commands will be reported through the Interrupt Status Port. All other interrupts will be reported using the Device Interrupt Status Port(s). When a device or resource completes processing a control block, it sets the bit in the Device Interrupt Identifier Port corresponding to its device or resource identifier, and then interrupts the system unit.

By using these optional ports, an interrupt handling program can process multiple control block interrupts on a single Micro Channel physical interrupt. This is accomplished by reading the Device Interrupt Identifier Port(s), and using the special immediate com-

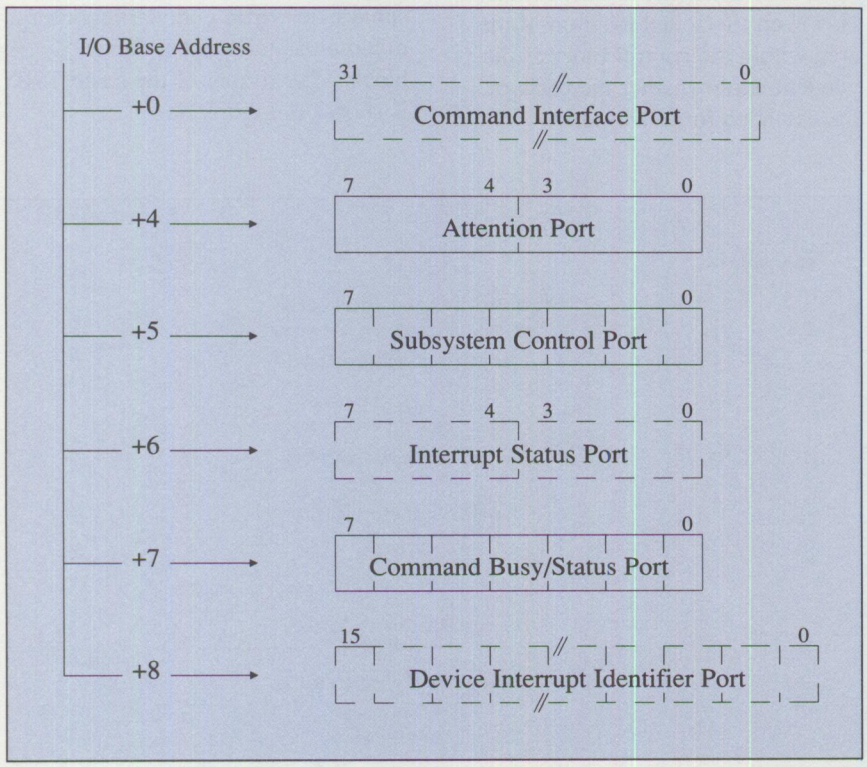



**Figure 7. I/O Control Areas**

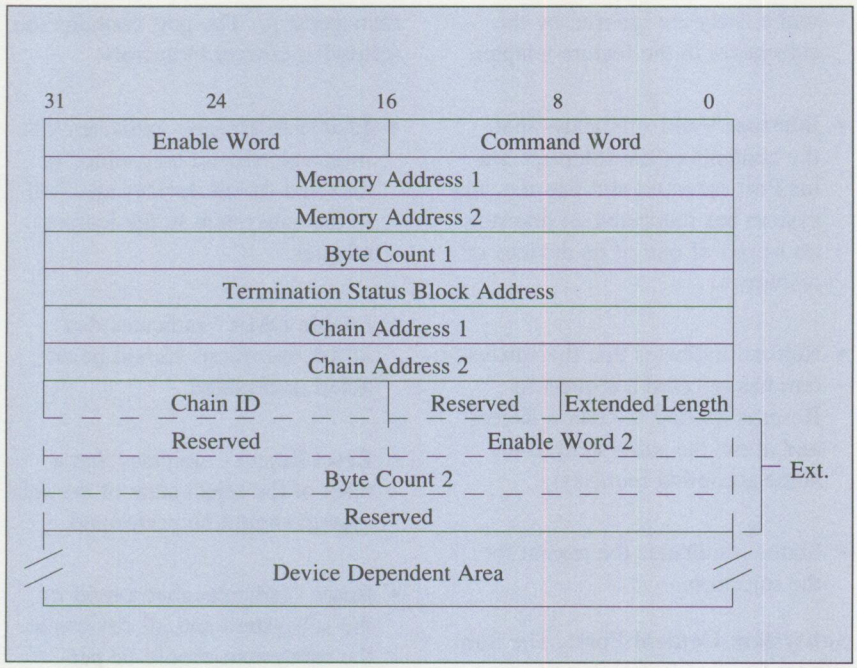



**Figure 8. Control Block Format**

mand, Reset Subsystem Control Block Interrupts, to clear the interrupt requests for the device.

Before issuing requests, the system-unit software must ensure that the subsystem is enabled to accept new requests, that is, not Busy or in the Reject state.

If virtual memory is being used, the system-unit software must ensure that all control blocks, Termination Status Blocks, Indirect Lists, and data areas associated with a request are locked into memory.

***Control Blocks***

The structure and content of a typical control block is shown in Figure 8.

There are two formats for the control block: basic and extended. Both forms share all of the fields shown in solid lines. The remaining fields (shown in dashed lines) are present only in the Extended Format. The Device Dependent Area is also present in both forms. However, the actual location of the area within the control block is dependent upon whether the basic or the extended form is used.

The physical address of the control block must be placed in the Command Interface Port and the device address and attention code in the Attention Port, in that order.

***Indirect Lists***

An Indirect List is a variable-length list consisting of address-count pairs used to support data chaining. Both the address and the count are four bytes. The length of the list is contained in the control block that points to the list. The format of the Indirect List is shown in Figure 9.

***Termination Status Blocks***

In addition to the exception/no exception indication in the Interrupt Status Port, the SCB architecture provides for detail status information to be reported for each command. The status information is reported in the Termination Status Block (TSB). Each control block includes a TSB address to which a subsystem writes completion or termination status for that control block. The format of the basic TSB is shown in Figure 10.

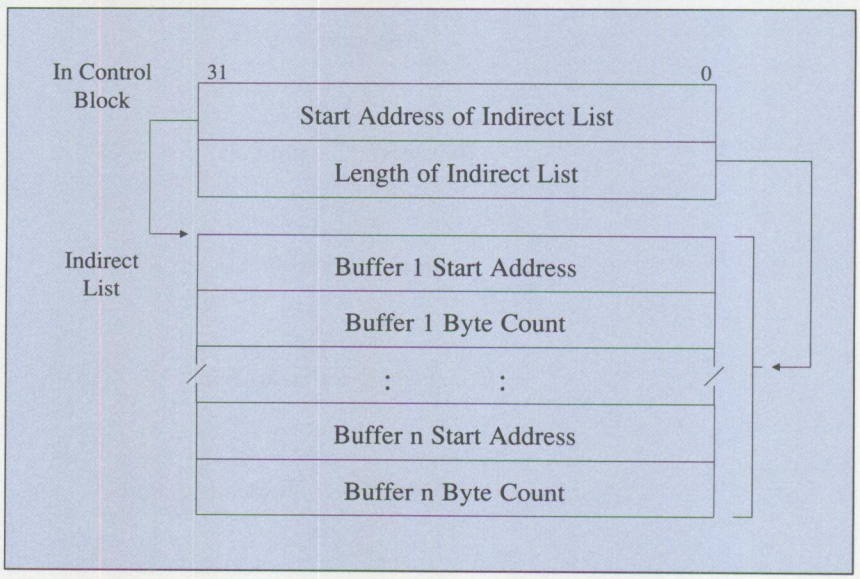



**Figure 9. Indirect List Format**

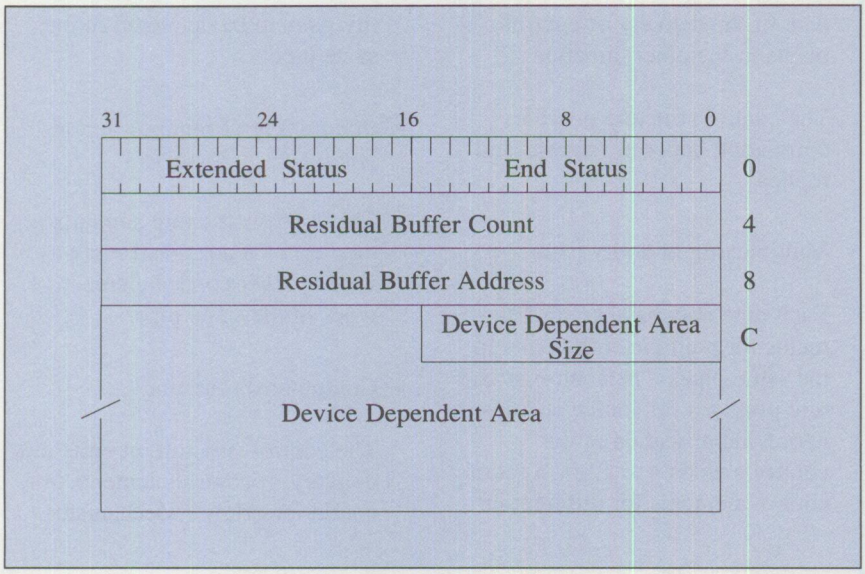



**Figure 10. Termination Status Block Format**

## Move Mode Support

The Move mode form of the control block delivery supports a control structure designed to allow a variable-length list of control-element primitives to be used to deliver control information to a server. This variable-length list may contain requests, replies, error, or event notifications for a specific device or resource in a subsystem, or for different devices or resources in the same subsystem.

The Move mode control block delivery structure in Figure 11 shows the interface to the delivery support and the various memory spaces related to control element delivery.

The Move mode facility has an overall structure similar to that of the Locate mode facility; that is, clients build requests, requests are delivered to server, server builds replies, and replies are delivered to client. There are a number of additional capabilities that have been defined for the Move mode delivery facility.

The following are the additional Move mode capabilities.

**Request/Reply Extensions:** Request/reply extensions define support for error and event control elements in addition to request and reply control elements and the ability to have multiple outstanding requests between entity pairs. The flow of control elements is independent of the physical layer protocols. The two parties in a specific entity pair have been defined as the client and the server. In general, the client sends requests to a server and the server sends replies back to that client. Events may flow in either direction. The delivery mechanism supports the delivery of these requests, replies, errors, and events among source and destinations having multiple client server pair relationships.

**Shared Memory:** Shared memory allows for the use of memory in feature adapters as well as memory in the system unit. This allows the control structure used to convey control elements to be located in either the system unit or the feature adapter. It also allows for various options when determining how the control structure is built and moved to the destination entity. (The server is the destination entity for requests and the client is the destination entity for replies.)

Figure 4 depicts the fact that memory is shared between the units and feature adapters. How this memory is shared and accessed will be different for different system environments (move/copy, DMA, and so on). The definition of the Move mode control structure provides for the fact that different forms of memory addressing will be needed.

**Variable-Length Requests and Replies:** The Move mode support provides a control structure that allows a request to be made up of a set of variable-length control elements. At the entity interface, these elements are contiguous. This allows for chaining of control elements within a request to be done with a minimum of address manipulation. It also allows for a minimum-size control block for passing simple requests and replies.

Variable-length control elements addresses the need to have smaller control structures, to be able to pass a variable number of request parameters and to have a simple way to support a number of different subsets. Some client / server entity pairs may choose to use complex combinations of primitives while others use a simple set. They are all implemented from the same set of primitives using the same delivery services.

**Lists of Requests and Replies:** Figure 11 shows a sample set of control elements flowing from a client to a server and another flowing

from a server to a client; both flow on the shared memory delivery service pipes.

The delivery pipes defined for the Move mode control element delivery service have the following attributes:

- Full duplex

  A pipe for each direction of delivery between units. This allows for the delivery of control elements in one direction independent of the delivery of control elements in the other direction.

  The control structure provides correlation between requests and replies.

- Multiplexing of entity pairs

  Each pipe may have control elements for multiple entity pairs in the same pipe. The control structure provides for source and destination identification in the control elements to allow a set of control elements for different entity pairs to be delivered in the same pipe.

- Intermixing of requests and replies

  The control structure supports a mixture of request and reply as well as other control element types in the same pipe.

- Continuously running

  The control structure permits the delivery of control elements in a continuous flow. Mechanisms

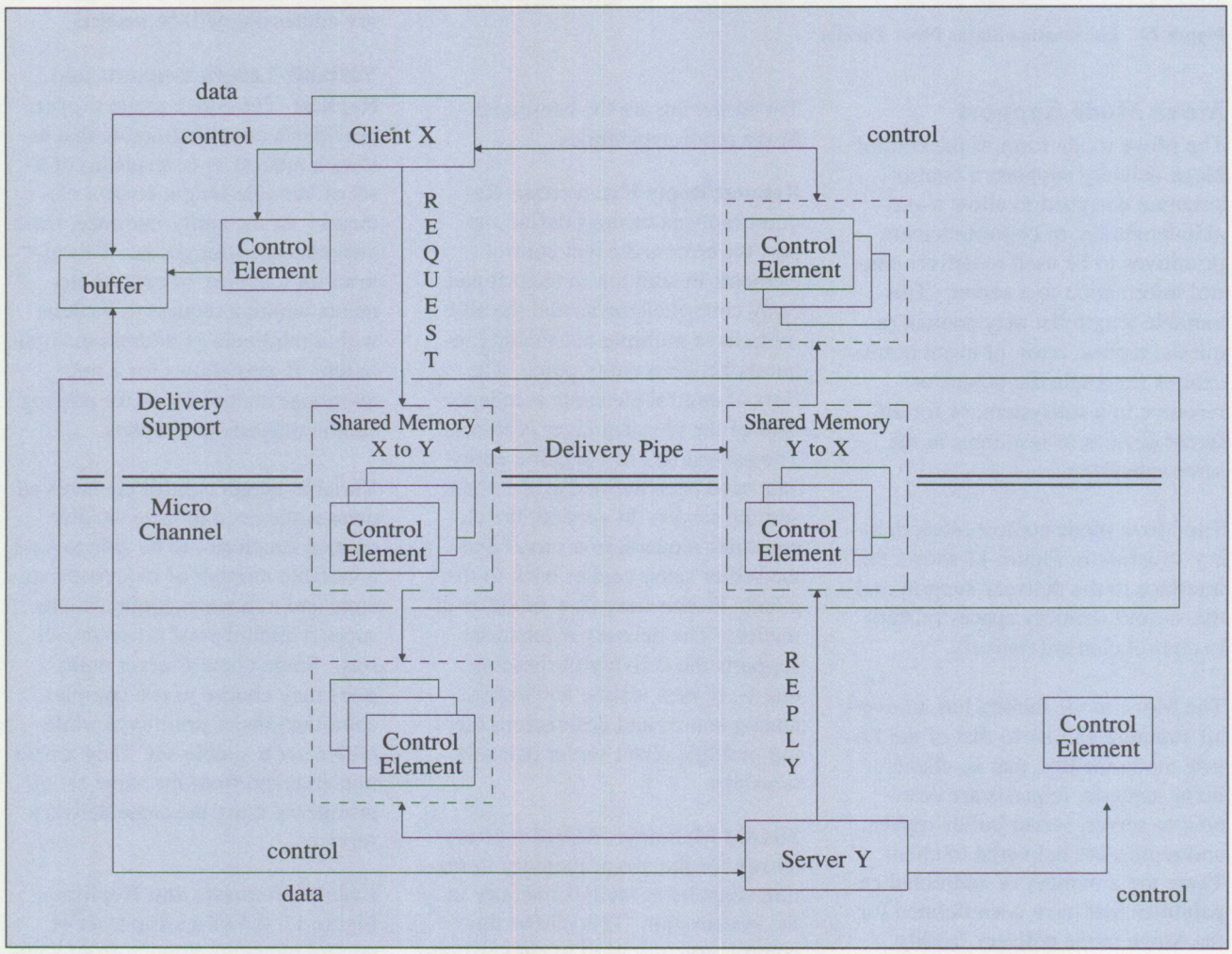



**Figure 11. Move Mode Control Block Delivery Structure**

are defined to provide a common way for entities to suspend the delivery of control elements at the entity-to-entity level, to notify the destination entity that a control element(s) is available and to provide for synchronization between the entities in the source and the destination.

***Move Mode Control Areas***

The architecture identifies the specific control areas in I/O space to be used as well as the protocols for initializing and using a feature adapter in Move mode. The control areas are essentially the same as those used for Locate mode, except the Command Interface Port, Interrupt Status Port, and Device Interrupt Identifier Ports are not used.

Because multiple feature adapters of the same or different types may be used in the system, the base address for the I/O space of each feature adapter must be defined during setup. The I/O control areas used by a feature adapter are shown in Figure 7 as offsets from the I/O base address.

***Control Elements***

Control elements are like control blocks. They are used to exchange control information between a client and a server. However, control elements differ from control blocks in the following ways:

- Control elements are variable in length.
- Control elements are self-describing.
- Control elements provide a means of specifying the destination, identifying the source, and indicating the type and urgency of the control information they contain.
- Control elements may optionally contain data as well as control information.
- Control elements contain information for correlating requests with replies.
- Control elements may be processed asynchronously.

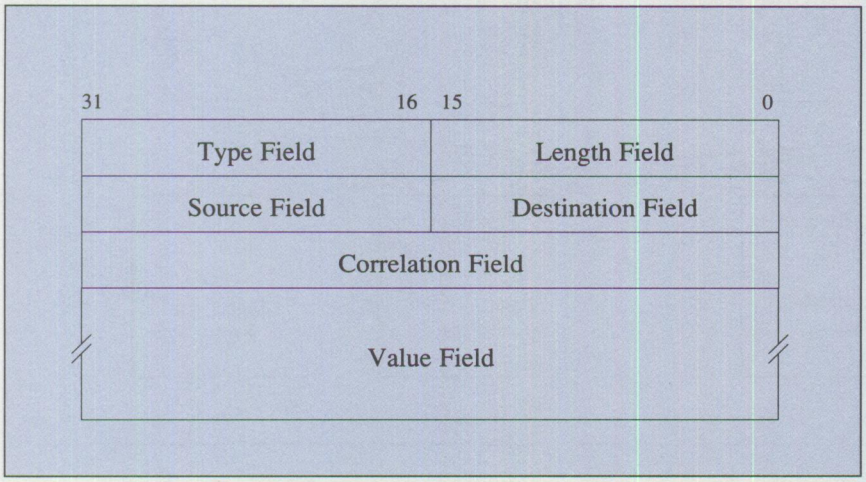



**Figure 12. Control Element Format**

To draw an analogy, a control element is like an envelope with a see-through window, while a control block is more like a post card. Both have a purpose and a use.

Delivery services use information in the window to deliver control elements, without knowing or understanding what is contained within the body of the control element.

Clients and servers use information in the window to specify the destination, identify the source, indicate the type and urgency of the control information, and correlate replies with previous requests.

Figure 12 shows the format of a typical control element.

The type, length, source, destination, and correlation fields are used by the delivery service as well as the client and server. They constitute the information visible through the window in the envelope. The remaining variable-length field, the value field, represents the contents of the envelope (that is, the control information). The structure, content, and length of this field is determined by the particular protocol being used between a client and server and is meaningful only to them.

***Queueing Control Elements***

With the use of delivery pipes in Move mode (which are implemented as circular queues), there is a tendency to want to mix the queueing capability defined to support the delivery of control elements between units and the queueing of control elements to a specific server. Figure 13 shows a request flow example of a delivery pipe as defined by SCB architecture. The delivery protocols support a free-flowing pipe between entities. Each

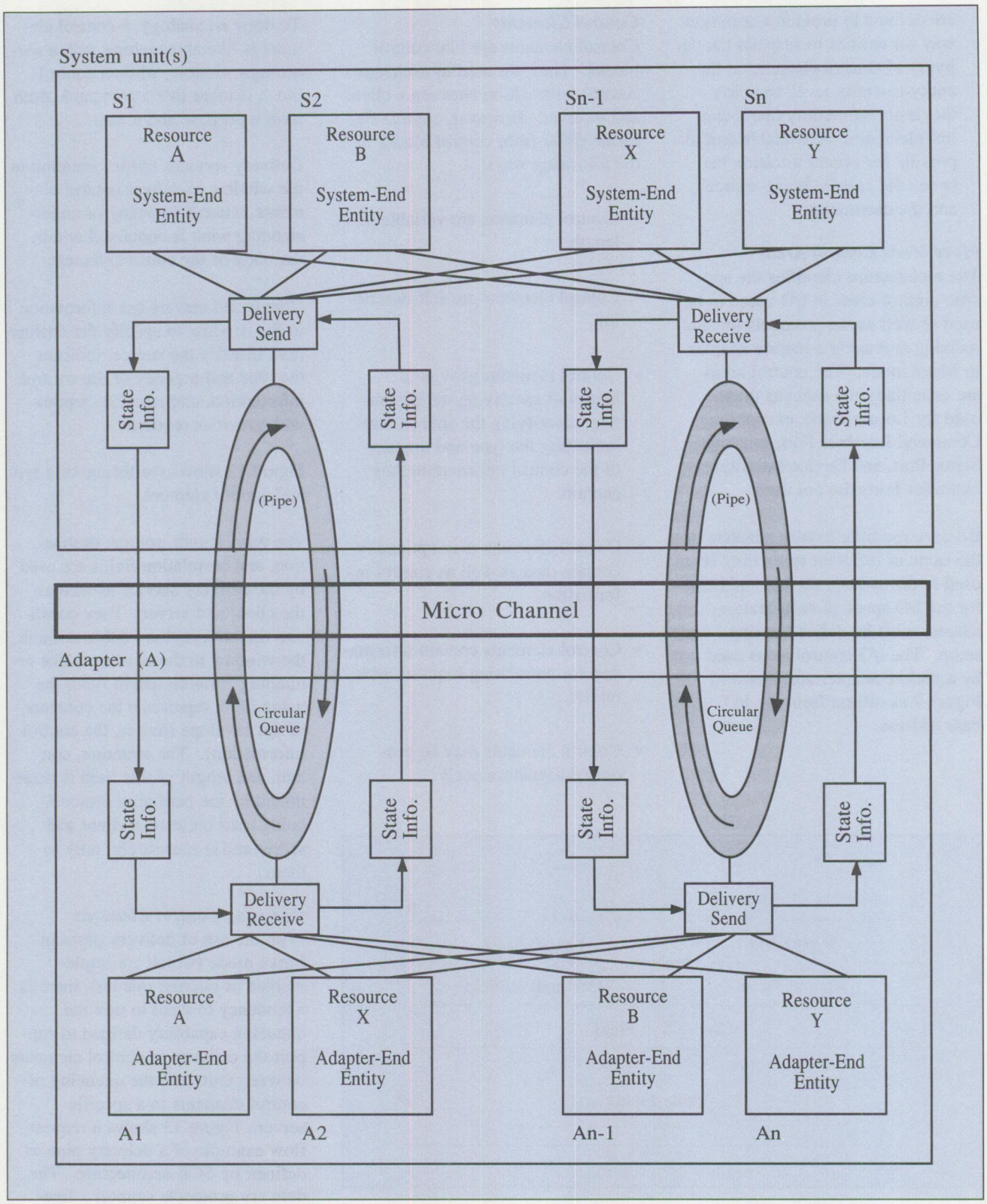



**Figure 13. Handling Control Elements in Move Mode**

entity pair is responsible for ensuring that there is a pending receive for elements sent so that one entity cannot block others from using the pipe. If there is no pending receive, the delivery service can discard the element and notify the source entity (sender of the element).

## Entity-to-Entity Relationships

For the Move mode form of control block delivery, the generic configuration shown in Figure 4 must to be expanded to include both system unit-to-feature adapter and feature adapter-to-feature adapter delivery. The feature adapter-to-feature adapter operations are referred to as "peer-to-peer."

### Peer-to-Peer Relationships

From a delivery point of view, the term "peer-to-peer" implies that there are no restrictions about where clients and servers may be located. It does not say anything about the relationship between the various client and server entities (which may be operating in a non-peer relationship). It also refers to the fact that control elements may be delivered directly between any two system unit and/or feature adapter that are physically connected by the Micro Channel.

In order to operate in a peer-to-peer relationship, the delivery support must allow requests and replies to flow in either direction and to be mixed on the same delivery-level flow. In the Move mode form of control element delivery, this is supported by having independent delivery of control elements in either direction and by allowing clients in system units or feature adapters and servers in feature adapters or system units. Peer-to-peer also requires support in both system and feature adapters to resolve contention when

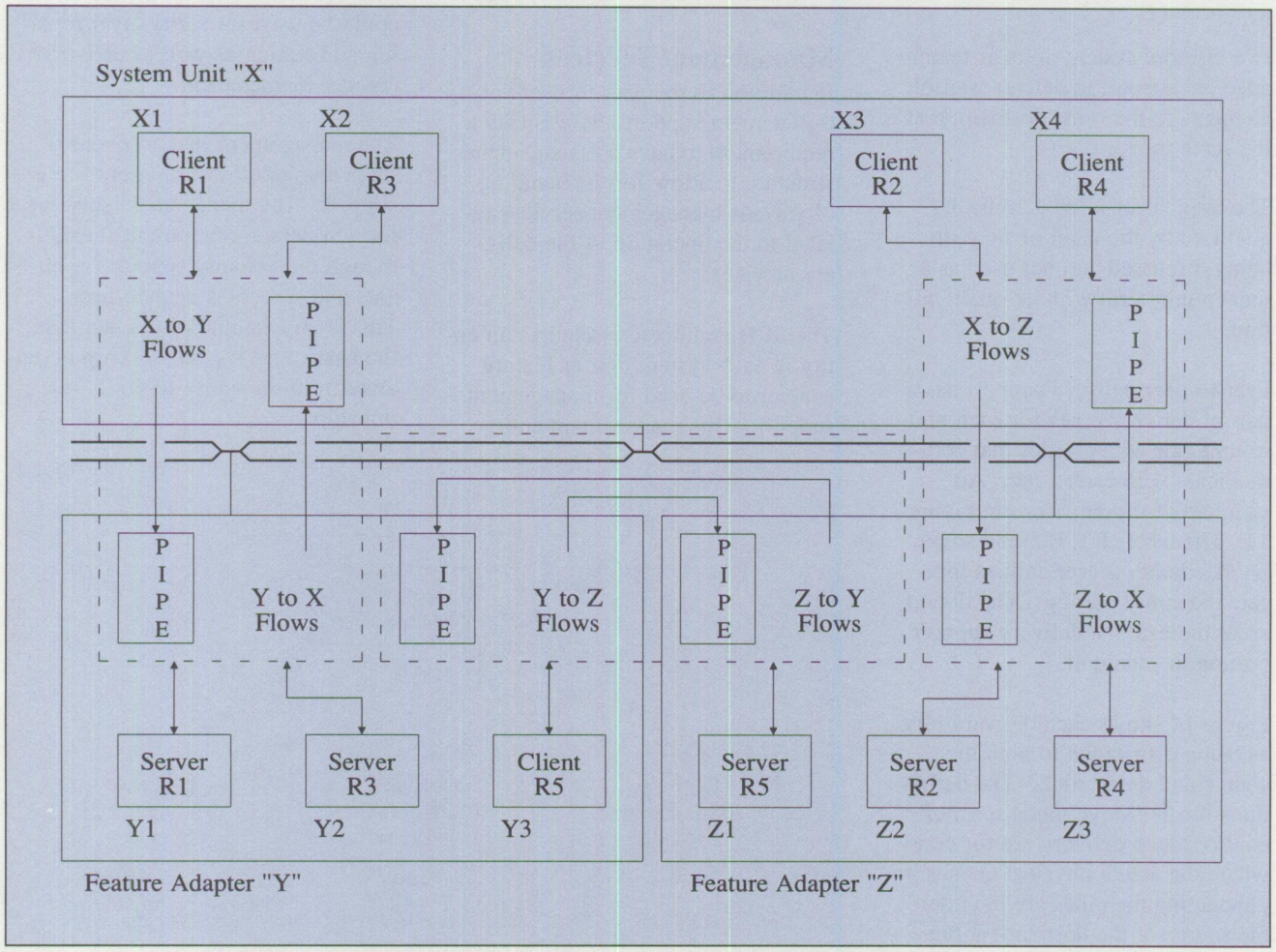



**Figure 14. Peer-to-Peer Delivery Model**

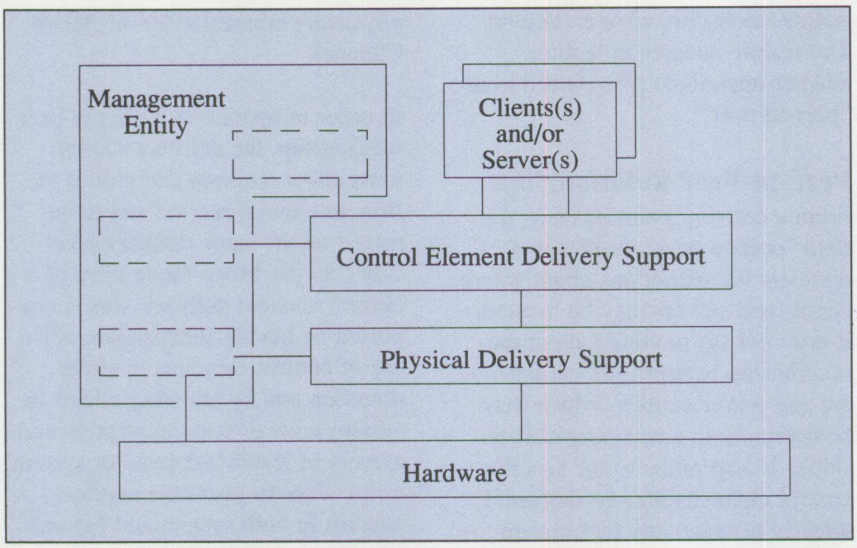



**Figure 15. Delivery Management Services**

two or more system units or feature adapters attempt to deliver control elements to the same destination at the same time.

The term "peer-to-peer" must be qualified by the level of support being discussed and not used as a unit-to-unit term without qualification.

Peer-to-peer delivery support has a pair of delivery pipes for each unit-to-unit pair with entities that communicate with each other. An example of this is shown in Figure 14. The labels R1, R2, and so on, on the client / server entities indicate the entity pairings. The dotted areas indicate the delivery support portion in each unit.

Figure 14 shows each delivery pipe as being distributed to both the source and destination. The definitions for the Move mode form of control block delivery are for cases where the actual physical queues implementing the pipes are in either the source or the destination, but not both.

## Management Services

In addition to the peer-to-peer control element support, there is also a requirement to have a management structure to allow for the handling of various management services related to the operation of the delivery services.

The SCB architecture requires an entity in each system unit or feature adapter to be used for management support. This management entity, which is shared by all the other entities, has an entity identifier of zero. The management entity structure is shown in Figure 15.

These management entities may use the delivery services to send control elements to or receive control elements from a system-level management entity. The system management entity, which does not have an entity ID of zero, provides system-wide management services (Figure 16).

This management structure may support management services of various types and is not unique to the delivery service. This means that it could be used for entity layer reporting and testing as well as delivery layer(s) management.

The management structure represents a unique form of client server support. The management services for a system are hierarchical even though the delivery support for clients and servers is peer-to-peer. This is an example of the fact that the basic delivery relationship is separate from the entity-to-entity relationship.

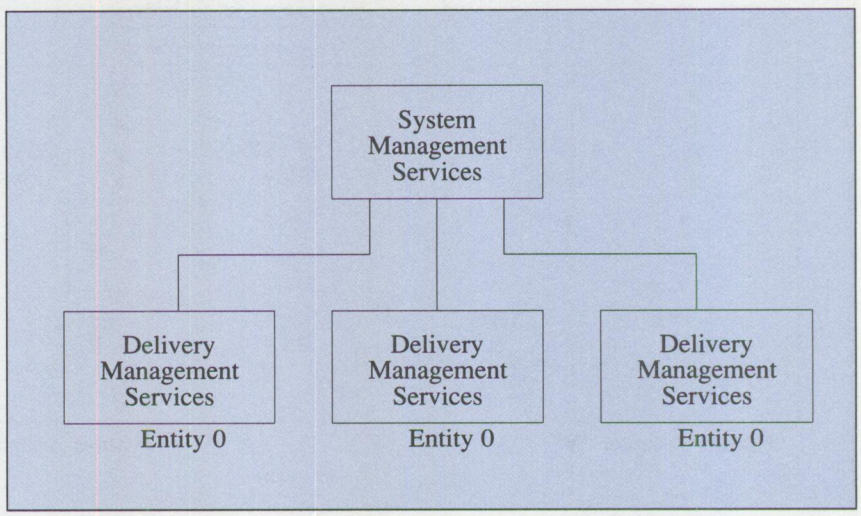



**Figure 16. System Management Services Structure**

## Summary

The SCB architecture was developed to standardize the programming task of supporting Micro Channel bus master feature adapters, as well as the engineering task of designing the bus master programming interface. It gives the adapter designer the freedom to define the meaning and content of the protocols that the feature adapter supports, while at the same time providing the programs using these feature adapters with standards for important interfaces.

*ABOUT THE AUTHORS*

*Frank M. Bonevento is a Program Manager with IBM. He joined IBM in 1964 and has held a number of technical and management positions in the areas of Systems Architecture, and in the design and development of systems software. He was a member of the design team that developed PL/S, a systems programming language widely used in IBM. Later Frank was a Programming Development Manager for systems software for the IBM Series/1. He received an Outstanding Innovation Award for the design and implementation of the Series/1 PL/1 software products, and a Division Excellence Award for work on Advanced Systems Architecture for the Personal System/2. Frank has had inventions published in the* IBM Technical Disclosure Bulletin *and has patent applications in process on several others.*

*Ernie Mandese is an Advisory Engineer with IBM's Personal Systems Architecture group. He has been with IBM since 1979. He holds a bachelor of science degree in electrical engineering from University of South Florida. Ernie participated in the development of the IBM Personal Computer and the Personal System/2. He is co-author in the development of the Base Subsystem Control Block Architecture. He has several patents pending.*

*Joseph P. McGovern is an Advisory Programmer at IBM's Entry Systems Division development laboratory where he is involved in the development of architectures for the Personal Systems. Joe joined IBM in 1984 after spending 15 years with Sperry Univac, where he was responsible for the development of the Distributed Communications Architecture (DCA) and Office Information Systems (OIS) architecture. He was a charter member of the ANSI/SPARC Distributed Systems Study Group and later served as the Chairman of ANSI and ISO/TC97/SC16 working groups responsible for developing the service protocol and definitions for the Session and Transport layers of the ISO Reference Model for Open Systems Interconnection.*

*Eugene M. Thomas is a Senior Technical Staff Member currently responsible for developing architectures for Personal Systems. Gene joined IBM in 1957 and has been involved in numerous design and architecture activities related to graphics, communications and distributed systems. These activities include work on early definition of software support for 2250 and 3270, and the definition of SNA.*

# Design Alternatives with Micro Channel Systems

*Chet Heath*
*IBM Corporation*
*Boca Raton, Florida*

**The developer's choice of attachment design will affect the cost, performance, power consumption, and reliability of peripheral device adapters, and may impact other elements of the system. While the personal computer was a single-tasking system design, and only one adapter and device were typically active at one time, Micro Channel computer systems can additionally support simultaneous activity of multiple tasks and/or users. This implies that many adapters and devices can now be active at the same time. The concept is called I/O concurrency and is a major new element of design for these systems.**

All of the simpler personal computer design types are retained in Micro Channel systems in order to allow support of logically identical adapter designs for PC applications. Five times as many direct memory access (DMA) adapters can be supported on the Micro Channel standard as are supported on the IBM Personal Computer. A powerful new type of adapter, the bus master, is defined to extend the capabilities of systems beyond the limitations of a single processor that controls all I/O.

## Primitive Designs

Personal computer designs typically depended heavily on the support of the processor for even the most rudimentary operations. This was permissible, because the processor was dedicated solely to one adapter function at a time. The most cost-effective approach was to implement the minimum of logic and depend on complex and lengthy support software to operate the device. Adapter examples of this type are:

- Continuous polled
- Interrupt per character
- Memory mapped

In general, these adapter types can induce limitations in the system. They consume processor time, at the expense of application processing, and require that the system processor be in control when I/O devices operate. Depending on the degree to which the I/O devices are active, such adapters can also degrade system performance in a multitasking or multiuser system. When other masters and DMA adapters compete for bus ownership, significant delays can occur when the processor must first gain control of the channel before operating the adapter. Yet these adapters are often implemented in a system because they provide register-level compatibility to existing personal computer (DOS) applications.

**Continuous Polled:** In a continuous polled adapter, such as the game port, the processor periodically requests information, or "polls" adapter registers for status of the attached device. Repetitive polling can consume much, if not all, of the computer's processing power. Such adapter designs are best suited to the only purpose for which they are used today – playing computer games.

**Interrupt per Character:** Interrupt-per-character designs are a step better than polling; the processor is free to support other duties until an interrupt request for processor attention is placed on the channel. The interrupt request to move a character of data between the adapter and storage, can consume 200 or more processor instructions just to change the processor's attention. This overhead may be acceptable in support

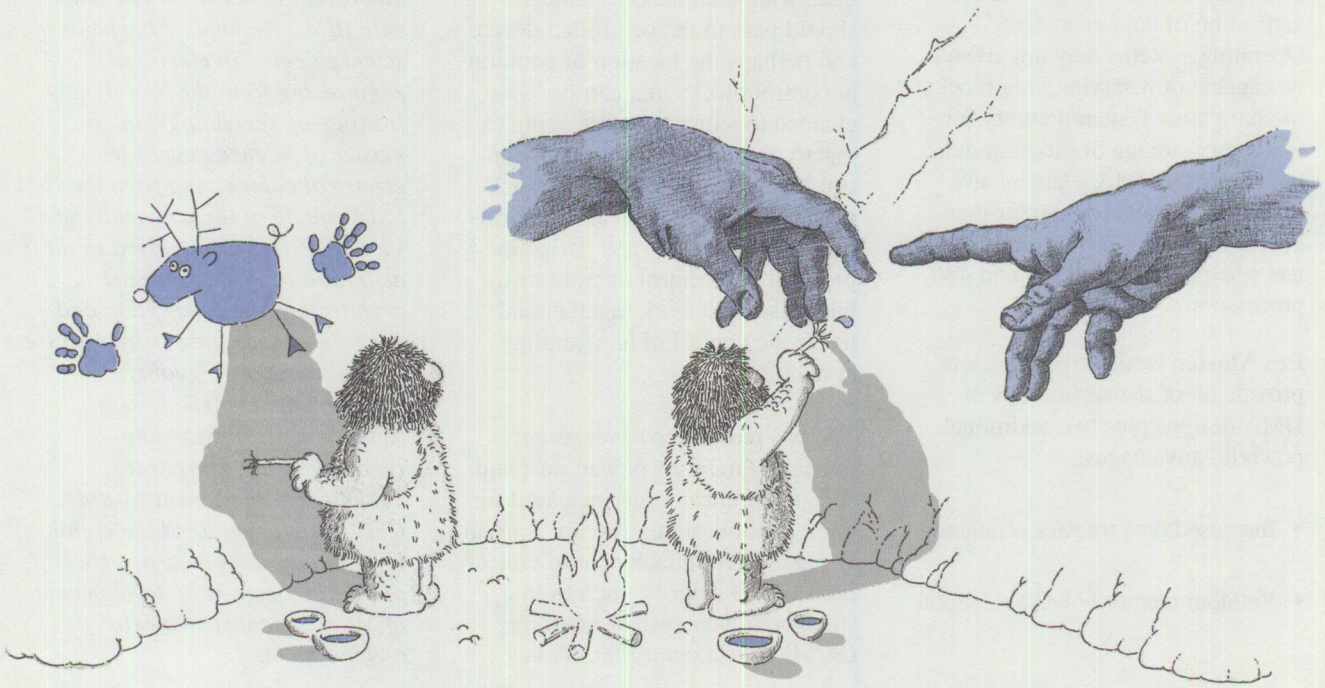
of low performance devices (such as a keyboard adapter) where the requests are infrequent and represent only a small percentage of the total capability of the CPU. Faster devices, such as high speed communications ports and printer interfaces, can easily consume all of the processor's power as the rate of interrupts and the cumulative overhead increases.

**Memory Mapped:** A memory-mapped adapter shares a segment of the storage area in the system with the processor. Large blocks of several thousand characters can be written to the mapped-memory area without disturbing the processor. An interrupt can signal the processor that the block is now available to the processor and the interrupt "overhead" is thereby "amortized" over many transfers. To some degree, this design type can be more efficient than most interrupt-per-character designs.

Unfortunately, there are other problems. The shared memory is assigned to a fixed address in an area that is reserved for input / output operations. The data often must be copied to or from an area of memory, dynamically defined by the operating system, for the storage of program data. The copy operation takes time and can negate much of the performance advantage. While the two memories can be controlled in a similar way by the processor, the shared memory is not physically part of the general memory system. It is implemented on the adapter card, adds cost to the card, takes up space and consumes power. The arbitration for access to the memory by the system can further diminish performance by adding time to every data transfer operation. Furthermore, there are a limited number of the fixed assignments of shared I/O memory. Consequently the number of configurations possible with other designs may be limited as well.

## Advanced Designs

Two important design alternatives that do not depend on the processor are now available to the designer: DMA designs and bus masters. The number of direct memory access designs was limited in PC, PC XT, and AT computers due to prior fixed and permanent assignment of the DMA data transfer channels. With Micro Channel architecture, as many as 15 DMA adapters can now be supported. Bus masters – adapters that can directly control memory and other adapters in the system – provide the greatest freedom in application design, are the most efficient with system resources, and allow the minimum dependence on the system processor. As many as 15 bus masters can be concurrently installed and active on the Micro Channel standard interface. While primitive adapters are limited by the instruction processing power of the system processor, the advanced types are typically limited by the capacity of the channel to transfer data, called throughput; the



throughput of the Micro Channel is very high indeed!

**DMA Designs:** A DMA design can free the processor from responsibility to transfer data, and is an ideal choice for low cost, medium performance adapters that transfer records of a few thousand bytes or less at a time. The adapter requests control of the bus, and one DMA channel which has been assigned to support the adapter then moves the data to or from system storage. Such adapters may implement the burst capability of Micro Channel architecture to move large blocks of data directly to the memory area defined by the operating system for storage of program data.

The DMA design does not require the shared memory of memory-mapped designs, may be less expensive, transfers data directly to the program data storage area, does not depend on processor activity, and still amortizes the interrupt overhead over a large number of transfers. The DMA adapter is, however, limited to sequential organization of data in storage. Operating systems may not always be capable of reserving long blocks of contiguous system memory for temporary storage of program data; consequently DMA adapters are best suited to records shorter than approximately four thousand characters (the page size for 386 and 486 processors).

**Bus Master Designs:** Bus masters provide all of the capabilities of DMA designs plus two additional, powerful advantages:

- Increased data transfer efficiency
- Random memory or I/O selection

A DMA adapter may require as many as four operations on the Micro Channel interface to transfer data with a data destination; a bus master will require only one operation. Even more important, a bus master can randomly address memory and I/O devices and is, therefore, not limited to sequential data transfer. It can intermix operations in support of a number of devices, or a number of data storage areas. Bus masters are optimal for the attachment of concurrent processors, intelligent subsystems, or any adapter that transfers either long lengths of data and/or high-speed data.

The ability of a bus master design to directly control and randomly read and write storage is an enabling characteristic for a Subsystem Control Block (SCB) architecture. With SCB architecture, the processor places formatted blocks of control information in storage where a bus master can access them. These blocks specify the location of data buffers in storage, the length of the data, what command an adapter should pass to the controlled device, and perhaps the location of additional control blocks that can be chained together to create complex macro operations. Subsystem control blocks complete the other half of the equation in distributed multiprocessor design. They define the protocol and control operations for bus masters to work together and under the control of an operating system.

The bus master, however, exacts one tax. Often the power, cost and design complexity are increased for this type of design. The tax is usually justified for attachment of collections of I/O devices, such as in a small computer system interface (SCSI) device controller. A bus master may also be cost-effective for the connection of expensive and perhaps complex functions, such as a processor, local area network, fixed disk, or display adapter.

The developer of products for Micro Channel systems is presented with a number of enhanced and new choices for the attachment of peripheral devices. Within each type of design, there are subdivisions and refinements that further extend the design options. The broad range of choices makes it possible for Micro Channel systems to be designed for simple single-tasking situations, for advanced microsystems, and even for minicomputer or mainframe applications. Having a broad range of applications is the greatest advantage of all to the developer; it ensures the greatest possible market opportunity for the design.

*ABOUT THE AUTHOR*

*Chet Heath is a senior engineer at IBM's Entry Systems Division laboratory in his twentieth year with IBM. He holds a bachelor of science degree in electrical engineering from the New Jersey Institute of Technology and a master of science degree in electrical engineering from the IBM LSI Institute at the University of Vermont. He was involved in the definition of Micro Channel architecture since inception and gave the architecture its name. He has received IBM Quality, Outstanding Technical Achievement, Outstanding Innovation, and Corporate Technical Achievement Awards. Chet also has attained the eighth level of invention awards. He is presently assigned to development of Micro Channel strategic enhancements.*

# 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:

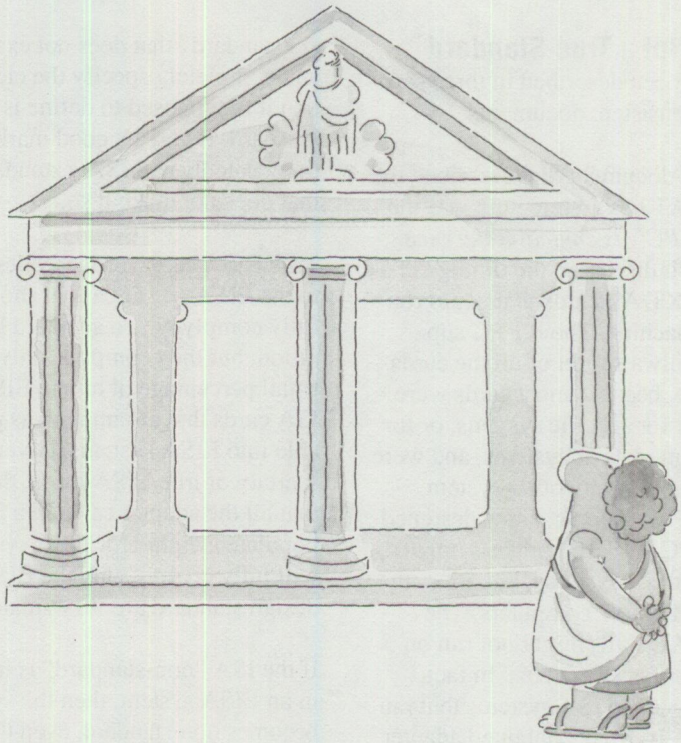
1. Removing the covers
2. Identifying the card
3. Setting the switches on the card
4. Reinstalling the card to complete the configuration of the system
5. Replacing the covers

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 74.) 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 system 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.

---

[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.

## 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.

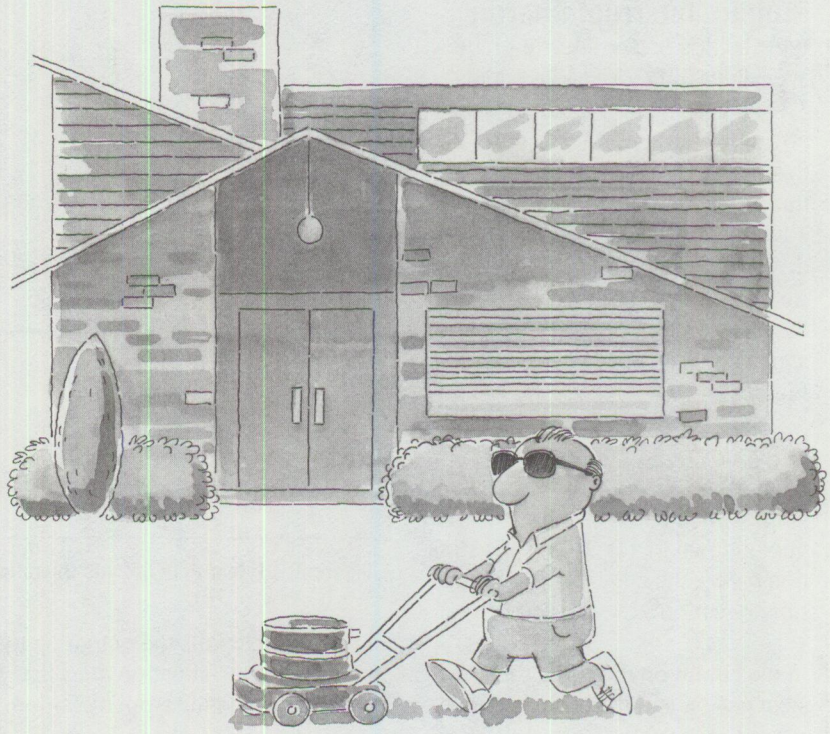
## Is PC/XT/AT Card Portability an Advantage?

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, hard-wired 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:

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

## 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.

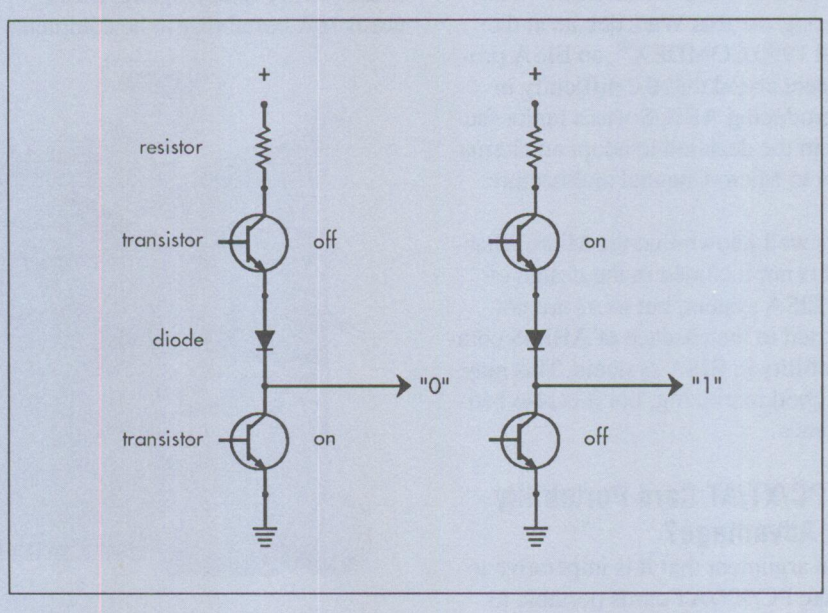



**Figure A. Totem-Pole Circuits Operating at Logic 0 and Logic 1 Levels**

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

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 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

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

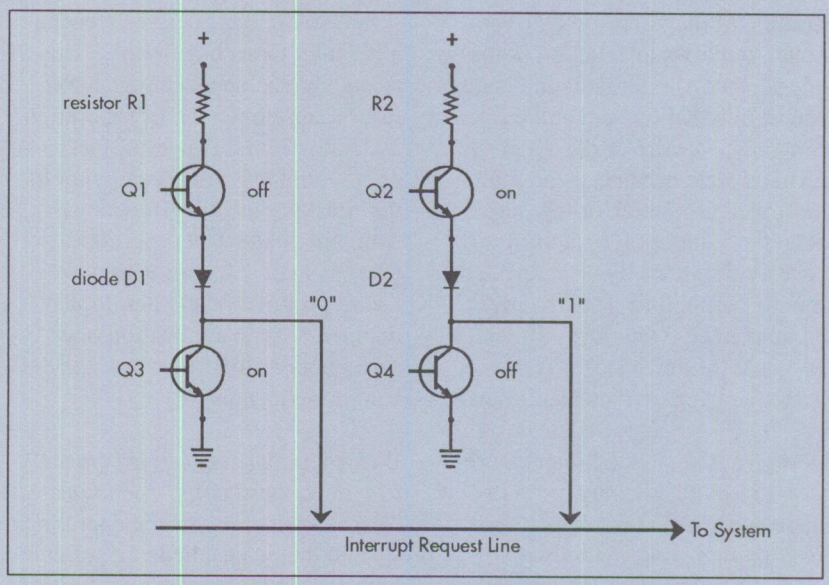



**Figure B. Two PC/XT/AT Cards Connected to the Same Interrupt Request Line**

Some PC/XT/AT cards have a second type of driver that is called *tri-state* or *3-state*. This design is a

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 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

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.

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 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 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,

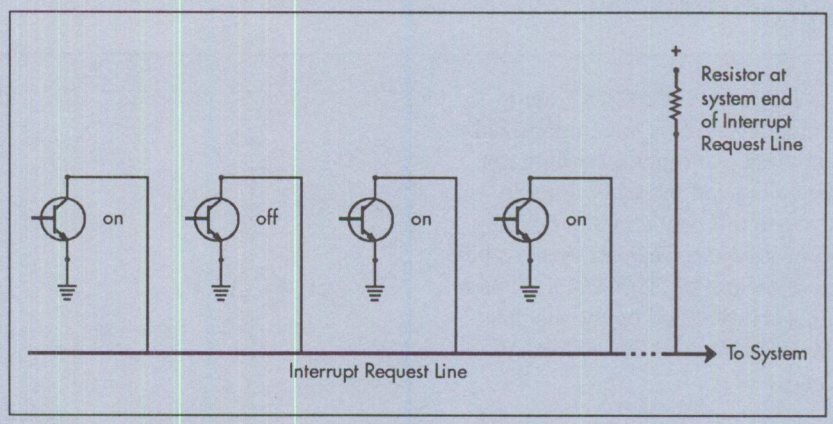



**Figure C. Multiple Open-Collector Drivers on the Same Interrupt Request Line**

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

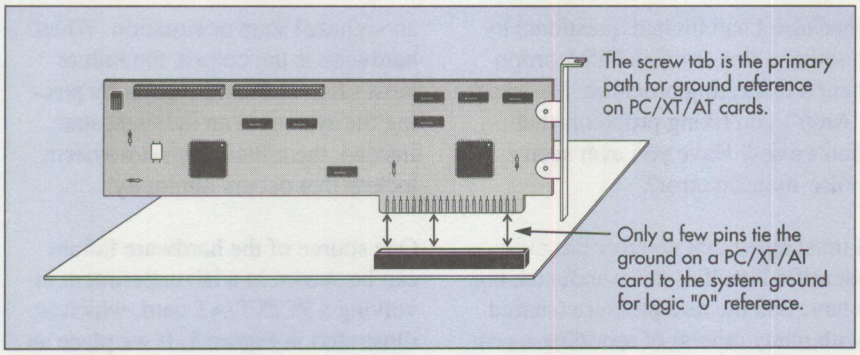



**Figure 1. How PC/XT/AT Cards Are Grounded**

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?

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 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.

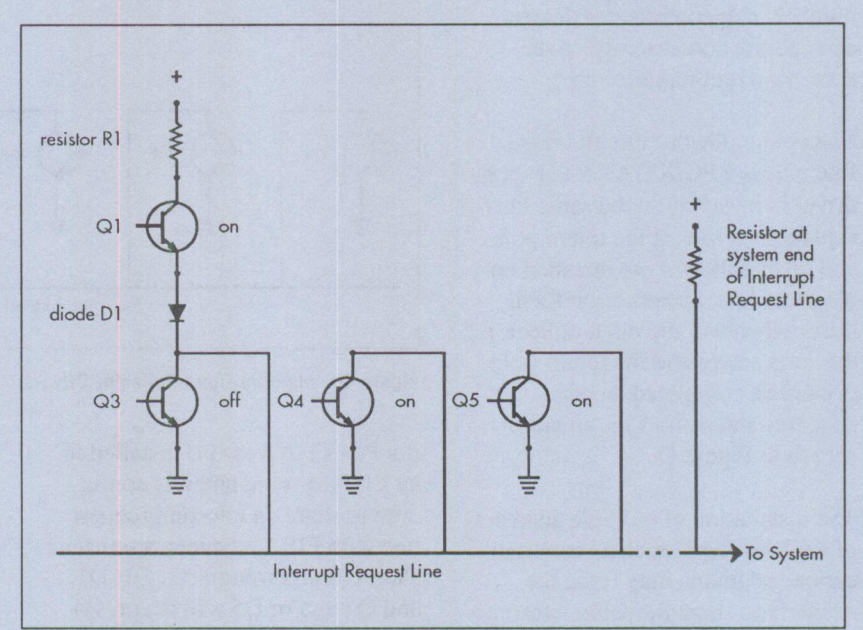



**Figure D. Totem-Pole and Open-Collector Drivers on the Same Interrupt Request Line**

by 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 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 Chan-

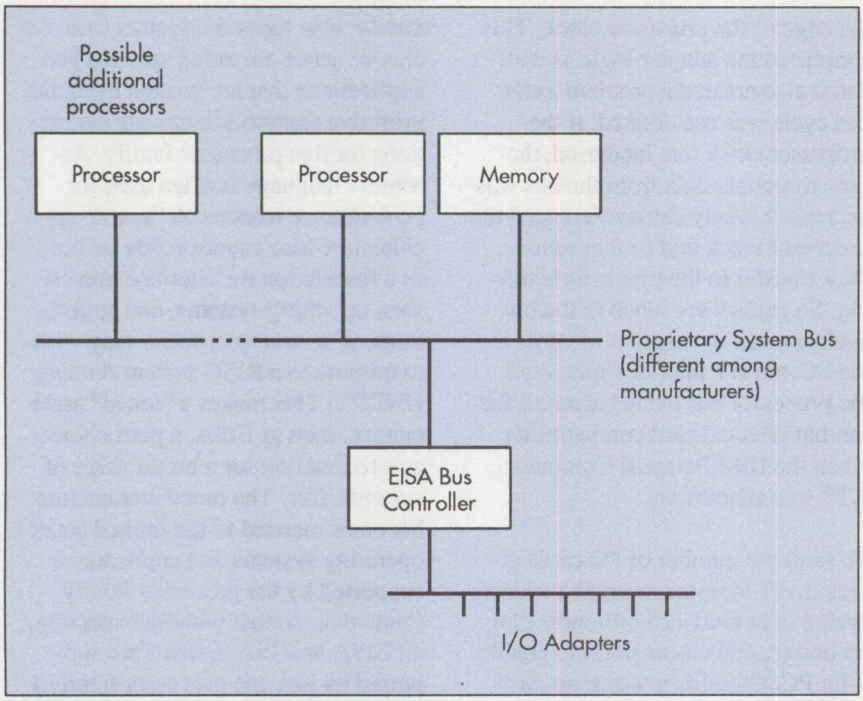



**Figure 2. EISA System Structure**

nel 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.*

# Comparing Architectures: Micro Channel and EISA (Part 2)

**Chet Heath**
**IBM Corporation**
**Boca Raton, Florida**

*This is the second 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. In Part 1, which appeared in the April 1992 issue of* Personal Systems Technical Solutions, *EISA advanced function – automatic system configuration, interrupt sharing, and high-speed transfer – were shown to be incompatible with PC/XT™/AT® adapter cards that are installed in EISA systems. Part 1 explained that because Micro Channel does not support PC/XT/AT cards, it does not inherit the limitations of PC/XT/AT systems. Finally, Part 1 gave the technical explanation of interrupt-sharing problems. Part 2 discusses the restrictions on EISA's DMA and bus arbitration that are imposed by PC/XT/AT compatibility, and shows the pitfalls of EISA's design and strategy compared to the design and strategy of Micro Channel architecture.*

When the IBM Personal Computer was introduced in 1981, it had a major advantage over the current market leader, the Apple® II computer: the PC's Direct Memory Access (DMA) enabled some devices to operate at higher speeds than were possible with 8-bit processors alone. A "DMA channel" acted like a second processor with only one instruction: block move. The PC hardware used bus cycles efficiently, moving one byte at a time across the 8-bit PC bus between memory and I/O adapters. The protocols for DMA data movement were embedded in the design of the silicon, and did not involve the processor except to start and end movements of blocks of data. The IBM PC had four DMA channels: one used by the system and three available to cards.

## Conflict Between PC/XT/AT Cards: Unusable DMA Channels

It was not long until the three DMA channels available to adapter cards were permanently reassigned to specific system functions. After that, newly developed adapter cards were either precluded from using DMA, or if they attempted to use DMA, they caused conflicts and possible permanent damage.

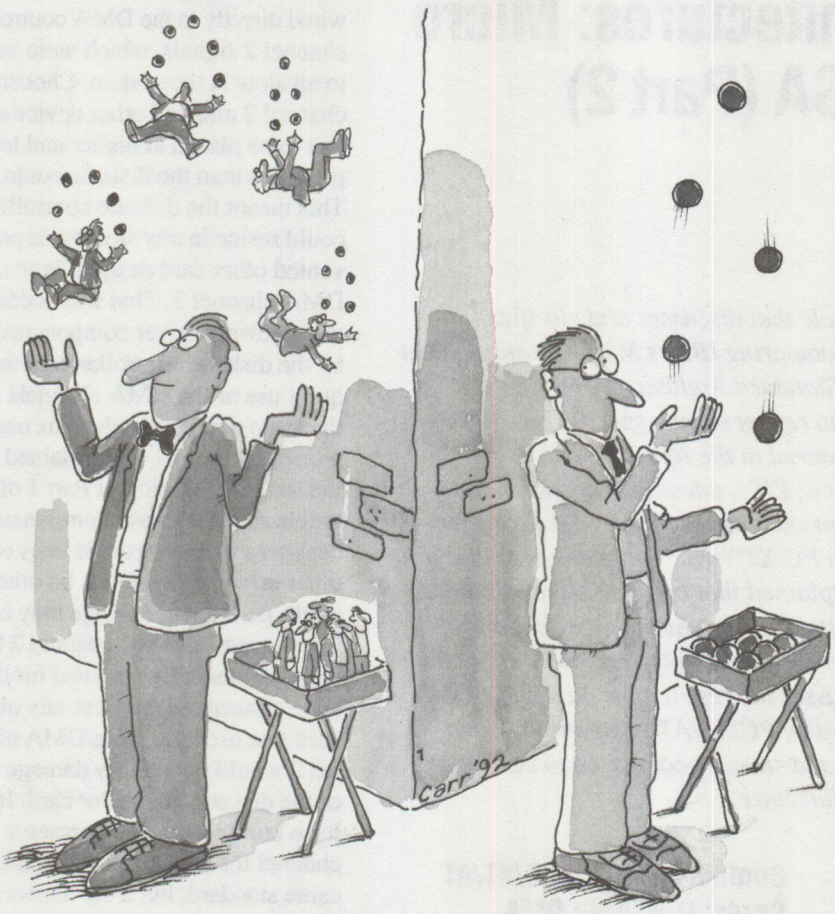
### DMA Channel 2: Diskette Controller

For I/O adapters that move large blocks of data, DMA was more efficient and faster than the processor. For example, the diskette controller, initially an option in the original IBM PC, needed to move 512-byte blocks of data at 50,000 bytes per second. DMA supported this rate as well as the much higher speeds of larger capacity diskette and hard disk drives that were to come.

The PC's diskette controller card was wired directly to the DMA controller's channel 2 signals, which were bused to all slots in the system. Choosing channel 2 allowed other device adapters to be placed at higher and lower priorities than the diskette controller. This meant the diskette controller could reside in any slot, but it prevented other card designs from using DMA channel 2. That was because the hardware driver components used by the diskette controller card to request use of the DMA channels were the same component designs used to request interrupts. As explained in the technical section of Part 1 of this article, if these driver component designs share the request lines with other driver components on other cards, permanent damage may ensue. Consequently, DMA channel 2 had to be permanently reserved for diskette controllers, because any other card that used the same DMA channel 2 would potentially damage itself or the diskette controller card. It may have been acceptable to reserve one channel for a function that soon became standard, but it set a precedent for the use of other channels that came later.

### DMA Channel 0: DRAM Refresh

Dynamic Random Access Memory (DRAM) is an extremely compact design for large-scale memories in computers. Using Very Large Scale Integration (VLSI), DRAM, with capacity sufficient to support the original IBM PC, can now be packaged as a single unit about the size and shape of a stick of gum. Despite the technical advances, DRAM memory components have one intrinsic characteristic: they "forget." Their memory is volatile if not refreshed every few thousandths of a second. In DRAM, minute amounts of electrical charge remain in one place inside the silicon, to indicate the presence of a logical one or zero. The electrical charge dissipates within a few thousandths of a second, so memory con-




tents will disappear quickly unless they are continuously and rapidly refreshed. Fortunately, the memory system can refresh DRAM by periodically reading its contents. The refresh and the restoration of the charge inside the silicon occur as a result of the read operation.

Because data integrity is among the highest priorities in system design, the highest priority channel of the DMA controller – DMA channel 0 – was wired directly to a timer on the system board. The timer requested a dummy read from memory every few thousandths of a second, and that operation refreshed the memory. Thus, the second of four channels on the DMA controller in the PC and PC/XT became permanently dedicated to memory refreshment.

### DMA Channel 1: SDLC

Synchronous Data Link Control (SDLC) adapters also characteristically move data as blocks. While they could not justify the high priority of DRAM refresh, SDLC adapters required that a block be retransmitted when the system could not keep up with SDLC's needs. SDLC adapters needed a priority lower than that of DRAM refresh, but higher than that of the diskette controller; therefore, they were hardwired to a fixed assignment at DMA channel 1.

### DMA Channel 3: Hard Disk Controller

A diskette drive can wait one more rotation to access data if the system cannot keep up with its needs. This appears as a performance degradation rather than a data integrity problem. Most users never notice that the diskette drive goes back to its master index, finds the data's starting point again, and goes to the appropriate track to catch the data as it passes the read head a second time. This is called a retry operation.

The IBM Personal Computer XT® added a 10 MB hard disk that vastly increased the speed and capacity of the system beyond the relatively slower 360 KB diskette drives of the day. Compared to the XT's diskette drive, a retry operation on the hard disk was far less noticeable. The XT's fixed-disk adapter card was hardwired to the DMA channel 3 request line on the PC/XT bus interface.

At this point in 1983, each DMA controller channel was permanently reserved for one and only one I/O adapter design.

### Sharing Channel 3 with a PC Network

The 10 MB hard disk of the IBM PC/XT could support 30 user files, each equivalent to a diskette in capacity. It would be more cost effective if many PC users could somehow share the more costly XT hard disk. Networking many PCs to a common XT server was the answer. But with only a few thousand instructions per second from the 8088 processor, and with all DMA channels assigned, it was difficult to find resources in the system to support a network.

The "temporary" solution was to break a rule and to share DMA channel 3 between the fixed disk and a PC network. It was the only feasible choice, because all the other DMA assignees would fail miserably if forced to share with a slow network. (A more permanent solution would have to await the IBM Personal Computer AT®, which had more DMA channels and a much faster processor that could also move blocks of data.)

To enable the XT's fixed disk adapter and the PC network adapter to share

a DMA channel, it was necessary to develop a special case in which both devices could not be active at the same instant under any circumstances. Sequentially, the network could exchange data with memory, then the fixed disk could exchange data with memory, and so on. The BIOS for the PC network would electrically turn off the fixed disk in an XT, thereby preventing concurrent operation of both adapters. Also, DOS was programmed to complete all transactions with the fixed disk before communication with a network could begin. This mutually exclusive situation was inefficient compared to the more desirable concurrent design, in which both the fixed disk and a network can operate at the same time. Indeed, the XT was totally unsuited to be a network server, but it was the best that technology could offer a decade ago.

The DMA channel utilization at the end of 1983 is summarized in Figure 1.

| DMA Channel | PC or PC XT Use | Width |
|---|---|---|
| 0 | DRAM Refresh | Byte |
| 1 | SDLC | Byte |
| 2 | Diskette | Byte |
| 3 | Other Fixed Disk or PC Network used sequentially | Byte |

**Figure 1. DMA Channel Utilization in IBM PC and XT**

### Concurrent Operation in the PC AT

The IBM PC AT expanded the simple DMA structure in the PC and XT. The AT implemented dedicated hardware to refresh DRAM, thereby freeing DMA channel 0 for use by adapters. This made the refresh operation more efficient. The free DMA channel – the one with the highest priority – was assigned to tape backup systems. Retry operations in tape systems are very slow because the tape must be stopped rather than continuing at high speed. The beginning of the missed record must be searched for and found, and the tape must then regain speed to read or write the record a second time. Placing a large data buffer on the tape adapter itself would, at best, reduce the probability of a retry operation. Therefore the highest priority DMA channel had to be assigned to the task.

Tape backup of larger files in AT systems became advisable because the AT was the first system that defined concurrent operations, albeit for just two adapters. Those adapters were the PC network and the hard disk adapters – exactly the two devices that should run concurrently in a minimal file server.

To handle concurrent operation, the more powerful 80286 processor moved the fixed disk data, while the DMA controller supported the PC network. Today, faster network cards such as Token Ring and Ethernet use the processor rather than DMA. This has increased the demands on the processor in PC/XT/AT systems to the point where very fast processors and associated fast memory systems are used, at great expense, largely to support high-speed data transfer through the processor.

DMA channel 3 is still permanently reserved for PC network cards, because they may still be installed in such systems. No other adapter card design should use channel 3 because, as explained previously, it might damage itself or the drivers on another card. This is one of the major penalties a system architecture must pay for AT card compatibility: the number of PC/XT/AT DMA adapter card designs that can exist without causing a system integrity problem or requiring much higher skills for installers is limited to the number of DMA channels in the system.

As a consequence, compared to the number of programmed I/O adapters for PC/XT/AT systems, very few DMA adapters exist beyond the few listed here. Some DMA adapters have been defined to use the "allocated" DMA channels, but they are typically installed in custom system configurations with knowledge of the potential conflicts and system integrity problems. Users of any new DMA card designs must be aware of the DMA resources already used in any system before installing such cards, or risk permanent damage to the system unit or cards.

Thus, technical constraints limit the market for many more DMA adapter cards for PC/XT/AT systems. These constraints, however, have not deterred the marketing of new adapters that use DMA.

### DMA Expansion in the PC AT

When the IBM PC AT was announced in 1984, it defined three more DMA channels that could double data-transfer throughput by moving 16-bit words, rather than just one 8-bit byte, at a time. Plans were made to enable DOS to support addressing blocks of words set on boundaries that were multiples of two bytes. The two-byte boundary restriction occurred because the AT incremented (by two bytes at a time) the address that the DMA controller used to select data in memory. The DMA controller had to start transfers on a memory address that was a multiple of 2, end transfers at an address that was a multiple of 2, and move only even numbers of bytes in each block transfer.

The plans for DOS – and for using the three new DMA channels – never materialized. DOS was unable to ensure that data buffers in memory would always reside on word boundaries. In fact, DOS would most likely organize buffers so that data would begin or end on odd boundaries. Any read from memory by a 16-bit DMA chan-

| DMA Channel | PC or PC XT Use | PC AT Use | Width |
|---|---|---|---|
| 0 | DRAM Refresh | Tape Backup | Byte |
| 1 | SDLC | SDLC | Byte |
| 2 | Diskette | Diskette | Byte |
| 3 | Fixed Disk or PC Network | PC Network | Byte |
| 4 | | Cascade[1] | Word |
| 5, 6, 7 | | Not compatible with DOS and unused except for busmaster upgrade | Word |

[1] When channels 5, 6, and 7 were added for AT systems, DMA channel 4 was consumed as the result of cascading one DMA controller into the input of another.

**Figure 2. DMA Channel Utilization in IBM PC, XT, and AT**

nel may miss the first or last byte of a buffer, or get an extra byte from an adjacent buffer. Even worse, write operations by a 16-bit channel could destroy a byte of data in an adjacent buffer by overwriting it. For this reason, DMA channels 5, 6, and 7 have seldom been used to move data in PC AT systems.

The DMA channel utilization at the end of 1984 is summarized in Figure 2.

The restrictions in Figure 2 apply to all PC/XT/AT compatible systems and in EISA systems that accept PC/XT/AT cards. Plugging an adapter card into an EISA system that already has a card using the same DMA channel can damage the card or other cards in the system. The purported advantage of EISA – its ability to accept PC/XT/AT adapter cards – represents an exposure to costly problems.

## Micro Channel: A Fresh Start

In 1987, IBM announced a fresh start called Micro Channel architecture. A fresh start was necessary to discard the legacy of limitations developed by PC/XT/AT cards and because the demands of even the near future could not be met by modifications of the past.

What kinds of demands? Multitasking, multi-user systems typically have more I/O devices than PC/XT/AT systems can configure. The I/O demands of adapters and devices are advancing faster than a single processor can support. Multimedia, in particular, requires much higher throughput than Industry Standard Architecture (ISA) systems can handle. The demands of the "multi" environment are also much greater than EISA systems with ISA cards installed may be able to support (see Part 1 of this article).

Although the need for an advanced bus was initially satisfied by the Micro Channel architecture, a consortium that formed 18 months later endorsed that need with a competing proposal. This consortium developed the EISA design specification, which cloned some of the functionality already available in Micro Channel architecture. Apple's NuBus architecture, Micro Channel, and EISA defined auto-configuration, interrupt sharing, higher throughput, and busmaster capabilities – all requirements of multitasking, multi-user, or multimedia systems.

Figure 3 shows that EISA systems, which accept PC/XT/AT cards, have only DMA channels 5, 6, and 7 available for expansion. A single DMA serial/parallel adapter card can then consume all the DMA resource in the EISA system. EISA is a 32-bit-only architecture, so its busmaster adapter designs are 32-bit only, and a maximum of eight can be installed.

In contrast, Micro Channel PS/2 systems can accommodate 15 DMA adapters, 15 busmaster adapters, or any combination totaling 15 (16 in some multiple bus systems). A Micro Channel system could theoretically have 4, perhaps 5, DMA serial/parallel adapters (each such adapter uses three DMA channels), where **EISA systems are limited to one** because there are only three spare DMA channels, as shown in Figure 3. This gives Micro Channel greater expandability to create balanced processor and I/O expansion capabilities over the life of the system.

### Arbitration Schemes

EISA systems centrally arbitrate (prioritize) requests by busmasters and DMA adapters. Each slot in an EISA system has a request signal to notify the central arbitration bus that a busmaster or adapter card in that slot is requesting control of the system. All requests are given the same priority level with a scheme called *strict fairness* or *rotating priority*, because each adapter or busmaster has the same chance to control the system. This arbitration scheme defines a minimum of seven DMA request signals, seven DMA acknowledge signals, eight busmaster request signals, eight busmaster acknowledge signals, and one signal for PC/XT/AT-compatible DMA called Address Enable (AEN). This requires 31 signals on the bus.

Micro Channel systems, on the other hand, distribute the arbitration function. They use an encoded request-and-acknowledge structure where arbitration is done between the adap-

| Priority (Arbitration) Level | EISA | | | Micro Channel Architecture with PC Software Compatibility | | |
|---|---|---|---|---|---|---|
| | Use | Reservation | Width | Use | Reservation | Width |
| 0 | DMA | Tape Backup | Byte [1] | DMA/BM | Available | 8/16/32-bit |
| 1 | DMA | SDLC Comm | Byte [1] | DMA/BM | Available | 8/16/32-bit |
| 2 | DMA | Diskette | Byte [1] | DMA/BM | Diskette | 8/16/32-bit |
| 3 | DMA | DMA Network | Byte [1] | DMA/BM | Available | 8/16/32-bit |
| 4 | DMA | Optionally implemented | | DMA/BM | Available | 8/16/32-bit |
| 5 | DMA | Available | 8/16/32 bit | DMA/BM | Available | 8/16/32-bit |
| 6 | DMA | Available | 8/16/32 bit | DMA/BM | Available | 8/16/32-bit |
| 7 | DMA | Available | 8/16/32 bit | DMA/BM | Available | 8/16/32-bit |
| 8 | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| 9 | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| A | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| B | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| C | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| D | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| E | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |
| F | BM | Available | 32-bit only | DMA/BM | Available | 8/16/32-bit |

[1] Shown with PC/XT/AT portability fixed assignments on DMA channels 0 to 3

BM = Busmaster

*Note*: In some single-bus Micro Channel designs, the processor is a busmaster at priority (arbitration) level F.

**Figure 3. Configurability of Arbitrating Devices in Micro Channel and EISA Systems**

ters, but coordinated centrally. In a Micro Channel system, an adapter card or busmaster issues a request using a signal called "Preempt" and places a 4-bit prioritized value on an arbitration bus. If a requesting card detects that another arbitrating adapter has a higher priority value on the arbitration bus than its own bid, the requesting card removes its bid but retains the preempt signal. In a few billionths of a second the auction is over, and the winner is the only card whose bid remains active. A line called "Arbitrate/Grant" signals the end of the arbitration bidding auction. This requires a total of only six signals on the bus.

The number of signals used to communicate the arbitration mechanism is an important consideration in Large Scale Integrated (LSI) circuit designs used in Micro Channel and EISA adapters. Micro Channel architecture requires fewer lines, making it less expensive to implement in LSI – both on the adapter and the system board. The superior LSI affinity of Micro Channel now helps a modern generation of card development produce less costly LSI designs.

**Card Size**: EISA advocates have claimed an advantage due to the greater space on EISA cards making it possible to implement more function on a card. EISA needs more space than Micro Channel on some cards to implement a function. In addition, many Micro Channel systems now support cards of the same size.

**Multiple Priorities per Card**: EISA systems provide only one busmaster request-and-acknowledge signal to each card slot. This makes it impossible to place multiple busmasters on a card at different system-priority levels. Micro Channel can package multiple busmasters per card, with each at a separate priority level.

This is important in applications such as serial communications. In full-duplex mode, data is transmitted and received at the same time; for ex-

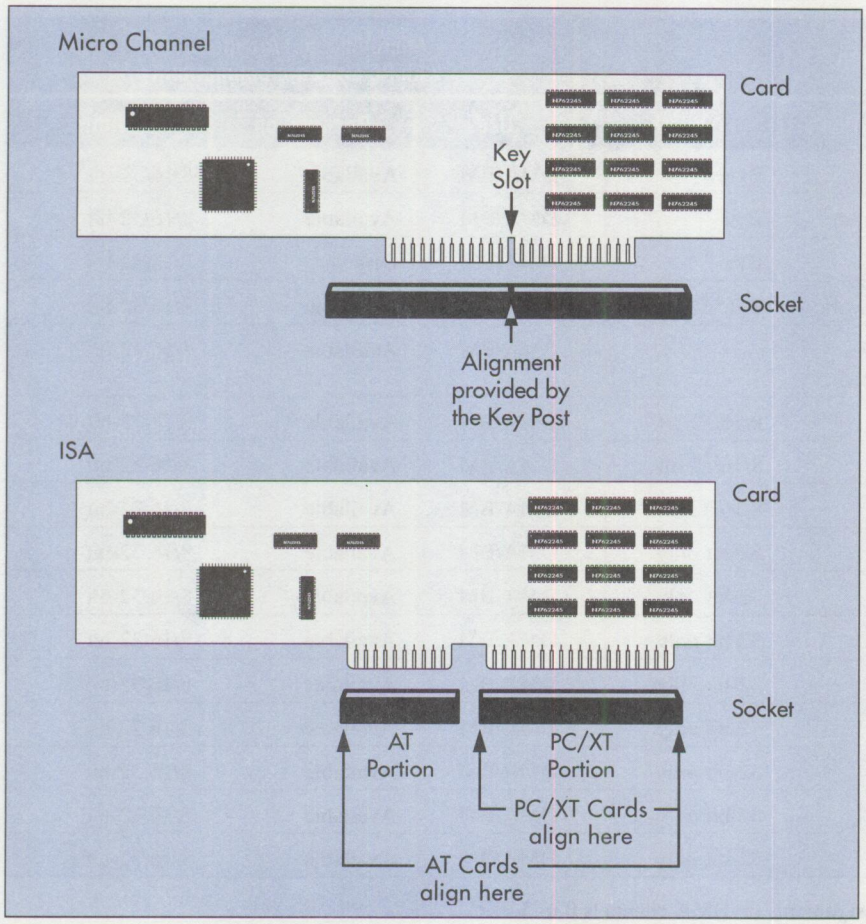



**Figure 4. Mechanical Alignment of Adapters in Micro Channel and ISA Systems**

ample, a keyboard transmits data into a system while a display receives characters from that system. Many terminals doing very high-speed duplex serial operations make efficient use of dedicated lines to remote mainframe computers, or between departments or regional centers. A busmaster or DMA interface that handles both sending and receiving would be ideal in the server that drives these lines. Whereas outbound data can wait until the transmitting end is ready to operate, data being received must be dealt with immediately; otherwise data may overflow available buffers. This means that the inbound channel must have much higher priority than the outbound channel. Micro Channel systems can accommodate these different priorities on a single card or busmaster; **EISA systems cannot.**

**Programmable Fairness**: Micro Channel supports *arbitration fairness* as a default mode for all adapters in the system, but also allows *programmable fairness* when adapters, such as full-duplex serial communication adapters, have one function that requires much greater priority than the other. Multiple priorities are needed on tape controller cards, where the length of an outbound record is known and can be written slowly to a buffer, but the inbound records are of unknown length and may exceed a buffer. This imbalance requires higher priority inbound and lower priority outbound. Network adapters, whose performance is higher than that of serial communication adapters, likewise have different priorities for inbound and outbound transfer. Micro Channel has been designed to accommodate situations of disproportionate priorities on the same card; **EISA's design cannot**. This means that Micro Channel cards can apportion system priorities where they are needed rather than forcing all cards to a lowest common priority.

## Bus Throughput

The mechanical design of the connector used in Micro Channel systems may not seem significant when comparing the architectures of Micro Channel and EISA, but it is. As shown in Figure 4, a PC or XT card plugs into an ISA slot and aligns against the end bulkheads of the PC/XT portion of the ISA system socket. Likewise, an AT or EISA card aligns against the end bulkheads of the full EISA socket. As a consequence, the EISA socket can never be wider than the card, because alignment is from the ends of the card. Because all signal tabs are dedicated, it is very difficult, perhaps impossible, to add signals.

In contrast, a Micro Channel-based system can expand to a wider data bus by expanding its socket length to add more signal tabs (accommodating a bus width of 256 bits). In fact, IBM is already using such a connector – it is the connector that the processor upgrade cards plug into PS/2 Model 90 and 95 systems. Using no more than the 160 MB per second protocols (already defined and shown publicly at BUSCON West 92), expanded to a width of 256 bits, a peak throughput of 640 MB per second can be attained. This is not the only option available to extend Micro Channel throughput; it is simply one of the more easily explained options. It shows that there is already at least one way that will support existing 64-bit cards.

Bus throughput is important because it determines the ability of a system to keep up with the growing demands of I/O. Historically, the performance of I/O devices has grown faster than the capacity of processors to process information. Growth comparisons between 1981 and 1991 are shown in Figure 5.

| Component | 1981 | 1991 | Growth Ratio |
|---|---|---|---|
| Processor Execution Rate | 0.3 MIPS | 15 MIPS | 50:1 |
| Display | 64 KB/second for still image | 1 MB/second for still image | 16:1 |
| | | 30 MB/second for motion | 480:1<br>16:1 with 30x "lossless data compression" |
| Fixed Disk Speed | 0.625 MB/second (ST-506) | 5.0 MB/second (SCSI) | 8:1 single file<br>64:1 file array |
| Communication | 300 bits/second (asynchronous) | 100 Mbits/second (FDDI LAN) | 333:1 |

**Figure 5. The Increasing Demands on Bus Throughput from 1981 to 1991**

I/O requirements are growing so fast that even 640 MB per second may be inadequate within a decade. IBM's demonstrated capability to support 160 MB per second now, and four times that without redesign, gives confidence that Micro Channel can meet that challenge. On the other hand, EISA's maximum is 33 MB per second (see Part 1 of this article), with no way to widen the data path. Thus, Micro Channel can be easily expanded for more throughput or advanced functions, where apparently **EISA cannot**.

## Ensuring Data Integrity at High Speeds

There is little value to have data traveling at high speed if there are no brakes to avoid catastrophe – the loss of data integrity due to error. Fortunately, Micro Channel design includes error-detection and response protocols called *bus parity* and *synchronous channel check* that go beyond the simple mechanism defined for the PC and inherited by EISA.

| Parity | Data | Description |
|---|---|---|
| 1 | 0 1 0 1 0 1 0 1 | This is the 9-bit value (data bits plus parity bit) transmitted. If it is received like this, the receiver is assured that no single error occurred in transmission. |
| 0<br>1<br>1 | 0 1 0 1 0 1 0 1<br>0 0 0 1 0 1 0 1<br>0 1 1 1 0 1 0 1 | If any of these 9-bit values are received, then an error has been detected. The error may have happened because the parity bit itself is invalid. |
| 1 | 0 0 0 0 0 1 0 1 | If any even number of errors occurs at once, they cannot be seen from the context. Fortunately, multiple errors in one byte are very rare compared to single errors. Odd numbers of errors are detected like single errors. |
| 1 | 1 1 1 1 1 1 1 1 | The total number of 1's in the data and parity together is always an odd number for odd parity. |

**Figure 6. Examples of Data Parity**

Parity is a concept inherited from mainframe and minicomputer systems. Parity is the context that is associated with the data to indicate its validity. A parity bit is transmitted with the data so that the receiving device can check both parity and data for validity. With odd parity, if the total count of 1's in a byte is an even number, the parity bit is a 1; if the total count is an odd number, the parity bit is a 0. Examples of parity are shown in Figure 6.

Bus parity is not foolproof; multiple errors, while unlikely, can get through. Although total elimination of error is impossible, the probability of undetected errors can be greatly reduced. The objective is to increase the integrity of the system to the highest possible degree. Parity simply requires an extra pin per byte of data on the bus. Parity is one data integrity tool available in Micro Channel systems, because extra pins are already defined on the Micro Channel interface. Extra pins, and therefore parity, do not exist on the EISA bus. Thus, Micro Channel defines error-detection capabilities that **EISA cannot**.

Most of today's PCs and PS/2s do not process parity, but newer Micro Channel-based systems in the IBM family of products do process parity information along with the data. The workstation products in the IBM RISC System/6000™ family and the IBM mainframe 7437 and ES/9000™ systems use parity to check data. As personal computer products provide more memory and hard disks, processors that perform parity checking on data transfers will likely become commonplace.

### Asynchronous and Synchronous Channel Checks

Using parity, a protocol in PC/XT/AT systems that was inherited by Micro Channel and EISA, called *asynchronous channel check*, can indicate that an error has occurred somewhere within a block of data. Recovery involves retransmitting or rereading the entire block.

The need for synchronous channel check arises when the error must be identified rapidly and with great precision. For example, what if the data error occurs when instructions are fetched from various parts of memory, or as part of a long train of data moving at high speed? Does the processor have to be halted or must the entire operation be repeated? Efficient recovery becomes possible if the individual erroneous byte can be identified. Then perhaps the process can be recovered by restarting just before the error, or by retransmitting only the error in a stream of data. To do this, detection of an error must be synchronized with the receipt of data. As part of its legacy from mainframes, Micro Channel inherited synchronous channel check, which precisely identifies the failing byte within a transfer. PC/XT/AT and EISA systems, having only asynchronous channel checking, can only isolate errors to the block. They require retransmission of entire records, making error correction less efficient. EISA adapter cards do not have the pins required to implement the signals for synchronous channel checking.

## Summary

In Parts 1 and 2 of this article, the abilities of the Micro Channel and EISA architectures to meet meaningful user objectives have been compared. Part 1 shows Micro Channel to be an open and consistent standard, whereas the foundation of EISA – Industry Standard Architecture – is inconsistent between the documentation and real-world implementations. The ability of the architectures to be applied to a broad spectrum of applications and system designs has been contrasted. In addition, the ability of the architectures to grow to accommodate advancing system needs for I/O performance and integrity has been outlined. The objective of Micro Channel architecture to migrate mainframe architecture to advanced microcomputer systems has been shown to be superior to EISA's short-term objective of "Extending personal computer architecture to advanced microsystems." Part 1 shows that EISA's affinity for a single family of processors and the operating systems that support these processors effectively make EISA a "proprietary" solution.

More than 1,200 cards are available from various Micro Channel manufacturers. There are approximately 100 members of the Micro Channel Developers Association, including many system vendors in addition to IBM.

Part 1 shows Micro Channel to be technically superior in its ability to offer advanced automatic setup of the system as a whole, regardless of feature card configuration. Micro Channel architecture's interrupt sharing capability is uncompromised by system integrity exceptions, unlike EISA. High-speed data transfer in Micro Channel systems is accomplished by existing products, implemented at very high throughput, whereas EISA systems' high-speed modes are subject to interference from installation of PC/XT/AT cards. Part 2 shows that direct memory access, and busmaster configurability and flexibility, are far superior in Micro Channel architecture compared to EISA. Finally, the data integrity features needed in advanced systems are defined in Micro Channel, yet are nearly absent in EISA. Not one function in EISA is absent in Micro Channel, but examples of the converse abound.

EISA is left with only one advantage over Micro Channel: the ability to install existing PC/XT/AT cards. Yet the advanced functions (that mimic Micro Channel) expected from the architecture are lost when PC/XT/AT cards are plugged into an EISA system. EISA can only come close to Micro Channel when all installed cards are EISA cards. Even then, the legacy that EISA inherits from compatibility with the PC, XT, and AT makes the comparison heavily favor the Micro Channel architecture.

## References


- *EISA Technical Reference Guide*, 2nd Edition. Compaq Computer Corporation, 1990. (130584-002)
- *IBM Personal System/2 Hardware Interface Technical Reference* (S84F-9808)



***Chet Heath** is a Senior Technical Staff Member and engineer at IBM's Entry Systems Technology Laboratory, in his twenty-second 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 was named one of the ten most influential people on PCs in the '80s by* PC User *magazine in the U. K.*

# Synergy by Design

**Chet Heath**
**IBM Corporation**
**Boca Raton, Florida**

*This article shows how the synergy between Micro Channel architecture and Operating System/2® 2.0 produces a demonstrable benefit for users of PS/2 Micro Channel systems. In the everyday function of printing, there is dramatic improvement in the time needed to transfer print files, and dramatic improvement in process efficiency, when OS/2 2.0 and Micro Channel architecture are used together.*

*The article discusses an experiment and gives instructions for conducting it yourself. There is no magic involved other than that which is designed into OS/2 2.0, IBM Advanced BIOS (ABIOS), and Micro Channel architecture. Everything is either right off the shelf – in announced IBM hardware and software – or printed in this article.*

The experiment discussed in this article shows the benefits and value of the synergism of IBM's PS/2 Micro Channel architecture, Direct Memory Access (DMA), IBM's Advanced BIOS (ABIOS), and OS/2 2.0 operating in concert. This experiment undertakes a common function: printing. In the experiment, a batch program copies a large file from diskette to fixed disk, prints the file from the fixed disk to a high-speed laser printer attached to the parallel port LPT1, and displays the start/stop times for printing the file. The experiment is conducted in two identical parts, with each part using a different operating environment. The first part runs under DOS; the second runs under OS/2 2.0 on a PS/2 Micro Channel system with ABIOS and an enabled DMA parallel printer port. The object is to demonstrate the significant reduction in elapsed printing time under the latter environment. The significant difference in the elapsed times is especially noticeable when both parts of the experiment are started simultaneously on two side-by-side computers.

## The Components

The computer used in the experiment is an IBM Personal System/2 Model 57 SX, which contains an Intel 80386 SX processor running at 20 MHz and a hard disk large enough to hold the data to be printed.

The data to be printed is in a file of about 200,000 bytes – roughly 40 pages of graphics-mode or spreadsheet text. The data consists of null characters (00 hex), chosen because null characters do not actually print, so they use no paper.

The time needed to transfer data to a printer depends on the performance of the printer's computer interface. The printer used in the experiment is an IBM 4029 LaserPrinter Model 5E running in Fast Bytes transfer mode. This mode, which is selectable from the front panel of the 4029, is designed for high-speed data transfer. In Fast Bytes mode, the 4029's internal microprocessor gives priority to the printer's computer interface while its print buffer is being loaded. The result is that the printer accepts the transferred data faster, and the experiment concludes sooner. Fast Bytes mode does not skew the experiment; it has the same effect on both parts of the experiment.

## Disabling Print Spoolers

To ensure that the experiment measures only the performance of the printing function, any print spoolers installed in DOS and OS/2 must be disabled. Print spoolers simply send data elsewhere in memory; they do not actually perform the I/O, but they signal that the I/O is complete before it is done. Print spoolers can cause inaccurate results in the concurrent I/O test, so they must be disabled. Terminating print spoolers in a DOS environment requires removing them from the `AUTOEXEC.BAT` file, then rebooting the system. Alternately, the system may boot DOS from diskette. The instructions for disabling the OS/2 2.0 print spooler are given in Figure 1.

## Part 1: DOS and Programmed I/O

DOS uses the system's microprocessor to move data between memory and I/O ports according to programmed instructions. This is called Programmed Input/Output (PIO). In Part 1 of this experiment, the print file transfer takes place in PIO mode under DOS.

Programmed I/O printing is typical in PC/XT/AT systems. Print data transfer in PC/XT/AT systems takes place in two sequential steps. The steps are sequential because both use a common element: the processor. In step 1, the data moves from the hard disk, through registers in the processor, to the system memory. In step 2, the data moves from system memory, back through the processor, to the printer port. The function of DOS is to sequentially schedule (that is, program) these steps.

The PC's Basic Input/Output System (BIOS), resident in Read-Only Mem-

### How to Disable the OS/2 2.0 Print Spooler

1. Boot OS/2 2.0 from diskette.
2. Double-click the left mouse button on the OS/2 System icon.
3. Double-click the right mouse button on the System Setup icon.
4. Click the right mouse button on the Spooler icon.
5. If the displayed context menu shows the Disable Spooler option, the spooler is currently enabled. Select that option, then click the left mouse button. A pop-up message will remind you to reboot the system to disable the spooler.
6. Click the left mouse button on the OK icon.
7. Press Ctrl-Alt-Del to reboot.
8. Repeat steps 2 through 5. The context menu in step 5 should now display the Enable Spooler option.

**Figure 1. Disabling the OS/2 2.0 Print Spooler**

ory (ROM), operates the printer by reading and writing the three registers, shown in Figure 2, that comprise the printer port. The first register sends commands, such as "accept the data," to the printer; the second register provides a path for the data; the third register accepts status reports such as "printer busy" or "need paper."

PIO is an interrupt mechanism for servicing print requests. In PIO mode, data is transferred to the printer port – one byte at a time. When the processor finishes delivering a byte to the printer interface, the printer signals an interrupt to request another byte. Therefore, the processor receives an interrupt for each byte it transfers to the parallel printer port. Each interrupt takes about 200 to 300 instructions, or the software may poll the parallel port interface using as many (or more) instructions, when DOS uses the PC's BIOS[1]. Because there are so many instructions associated with each byte being transferred, and because there are so many bytes to print, the processor can be heavily loaded just servicing printer interrupts. In most PC/XT/AT systems, the processor is additionally burdened with operating the hard disk that is the source of the data being printed. Therefore, the performance of the processor limits the speed of the data-transfer operation. The total

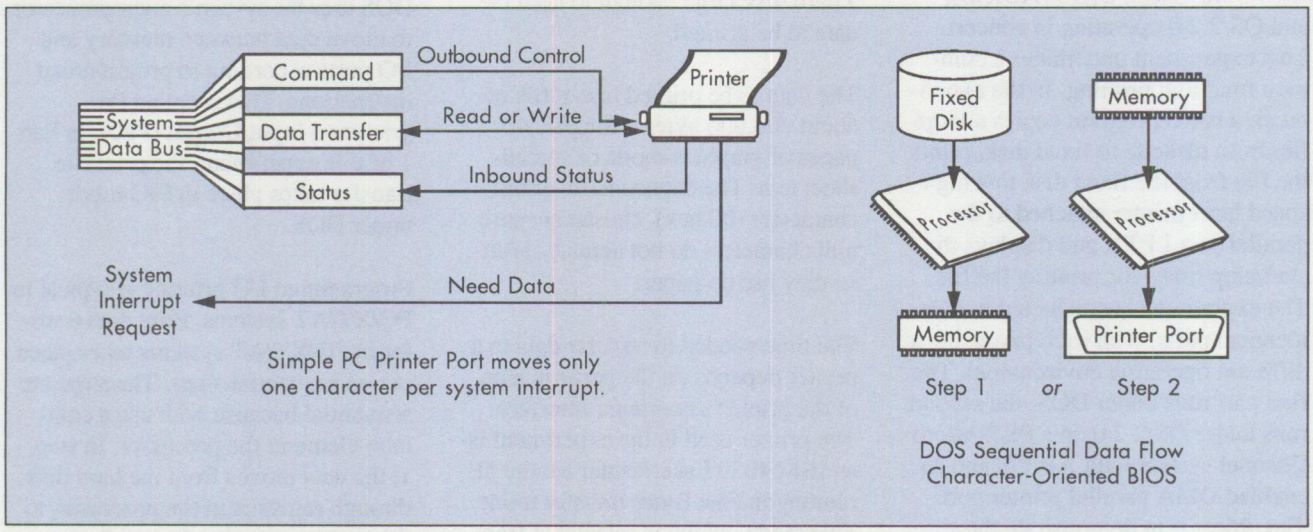



**Figure 2. Simple PC Printer Port Operation**

[1] Even when OEM BIOS poll the printer interface, the character-oriented BIOS of ISA machines still involves a software interrupt per character at the operating system interface.
Running the print operations in the background in a multitasking operating system will demonstrate that ISA systems consume cycles from the foreground process while PS/2 Micro Channel DMA does not. An impressive demo comparing fully installed OS/2 2.0 operating on ISA and PS/2 Micro Channel systems can be made where little residual performance is provided to applications in an ISA system concurrently operating with high-speed background printing, as in a print server. A similarly configured PS/2 system will handle almost any level of background printing with little noticeable degradation of concurrent application performance.

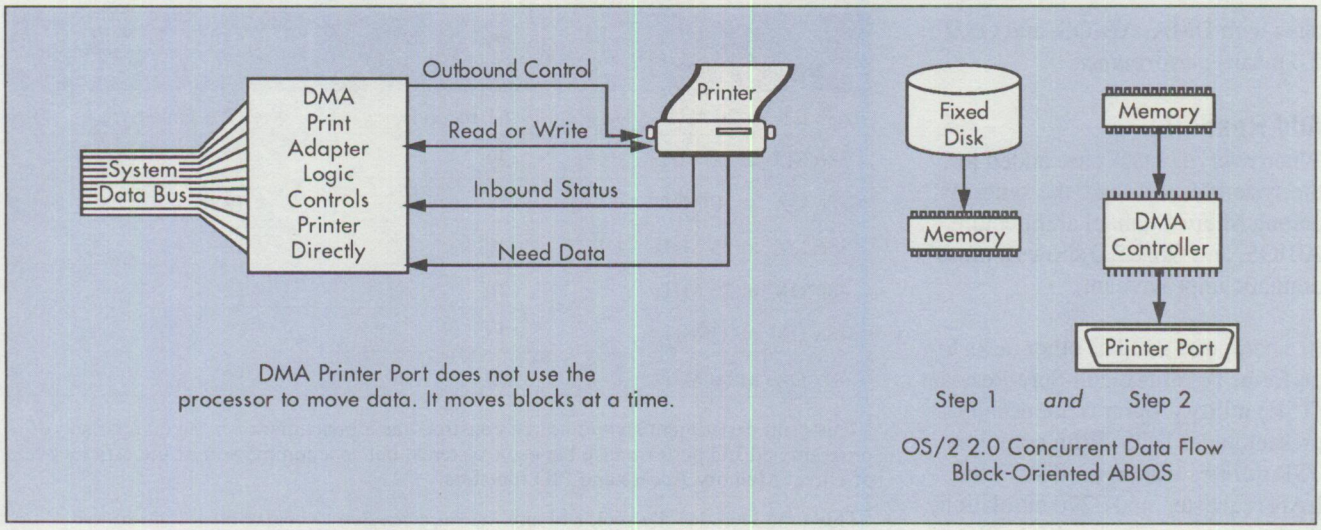



**Figure 3. DMA Printer Port Operation**

time to complete the operation is the sum of all file and print transfer delays, because they are done sequentially.

IBM PS/2 Micro Channel systems provide PIO mode for compatibility with DOS and Microsoft Windows® applications.

## Part 2: OS/2 2.0, Micro Channel, ABIOS, and DMA

Part 2 of the experiment takes place under OS/2 2.0 on an IBM PS/2 Model 57 SX computer with Micro Channel architecture, concurrent ABIOS, and an enabled DMA parallel printer port.

DMA print is an I/O mode that is much more efficient because it supports the concurrency and multitasking of OS/2 2.0. DMA print is included in PS/2 Micro Channel system models recently introduced: PS/2 Models 56, 57, 80-A21, 80-A31, 90, and 95.

DMA uses "intelligent" I/O adapters that off-load much of the processor's burden by using the DMA controller, instead of the processor, to move data (as shown in Figure 3). The DMA parallel adapter moves data from a previously loaded memory buffer, through the DMA controller, to the printer port, bypassing the processor. The SCSI file adapter is a Micro Channel busmaster that controls the fixed disk adapter and moves data from the hard disk directly to a memory buffer in one operation, also bypassing the processor. Because of these intelligent adapters and busmasters, the processor no longer has to schedule and manage the I/O; instead, it can turn its attention to other computing tasks and complete those tasks much sooner.

The combination of Micro Channel architecture (with more available DMA channels and busmaster capability), ABIOS, and OS/2 2.0 enables the two data-transfer steps to take place concurrently. In Micro Channel systems, the two steps can be simultaneous if the operating system defines multiple tasks and has an ABIOS that moves blocks of data concurrently using busmasters and DMA peripheral adapters. OS/2 2.0 has been designed to fulfill these requirements.

If the DMA parallel interface is enabled by Programmable Option Select (POS) during configuration of a Micro Channel system, the DMA interface is then selected by ABIOS and OS/2. However, if the DMA parallel interface is disabled or not present in a Micro Channel system, OS/2 defaults to the DOS-compatible mode, PIO, which has been implemented in Micro Channel systems for compatibility with DOS applications. An IBM Personal Computer AT system and most (if not all) AT clones, even if running OS/2, operate in PIO mode. They cannot have a DMA serial/parallel adapter port without ruling out other devices.

### Comparison of Elapsed Times

When comparing the elapsed time from Part 1 to the elapsed time from Part 2, a dramatic difference becomes evident. That difference is due to the Micro Channel's DMA parallel port and to the way OS/2 2.0 takes advantage of the DMA parallel interface.

Figure 4 shows the approximate results to expect when running this experiment on various Micro Channel computers. As seen in Figure 3, even the most powerful processor available today – a 486 DX running at 50 MHz – takes almost twice as long under DOS and PIO as it takes on any system listed in the figure that is running OS/2 and DMA printing. Clearly, using Micro Channel sys-

tems with DMA, ABIOS, and OS/2 2.0 means *performance*.

## Add Real Life...

When everyday tasks are added to the system's workload, the synergy among Micro Channel architecture, ABIOS, and OS/2 2.0 shows a more dramatic improvement.

In a computer that has other tasks to perform, Terminate-and-Stay-Resident (TSR) utility programs are usually present under DOS. With common TSR utilities loaded for disk cache, LAN requester, and 3270 emulation, the interrupt service times for PIO increase significantly. This is because DOS's primitive ability to support background tasks in TSR programs depends on having each TSR execute for a brief period following every timer tick interrupt. The timer tick has higher priority than both print and serial communications, so it gets first access to the processor's attention. This adds to the delay in servicing the timer tick interrupt and all lower priority interrupts, and places an interrupt latency burden on any system in which I/O performance depends on the processor. In real-life situations, DOS TSRs take more of the processor's capability and give the processor less time to attend to applications such as printing. Further, TSRs reserve part – sometimes as much as half – of the 640 KB of system memory available to DOS.

In contrast, in a PS/2 Micro Channel system running OS/2 2.0, the DMA controller and the DMA print function do not depend on the processor to move data, so they are essentially unaffected by processor activity.

In one test on an unmodified PS/2 Model 57 SX with the common TSRs loaded, Part 1 under DOS and PIO took about four minutes. Part 2, under OS/2 and DMA (print to an IBM 4029 LaserPrinter Model 5E) remained at about 12 seconds. In this real-life situation, the result is a 20:1 improvement.

| Processor | DOS with PIO[1] | Micro Channel with DMA, ABIOS, OS/2 2.0[2] |
|---|---|---|
| 386 SX at 20 MHz | 61 seconds | < 12 seconds |
| 386 SLC at 20 MHz | 39 | < 12 |
| 386 DX at 20 MHz | 58 | < 12 |
| 386 DX at 25 MHz | 49 | < 12 |
| 486 DX at 25 MHz | 38 | < 12 |
| 486 DX at 33 MHz | 29 | < 12 |
| 486 DX at 50 MHz | 22 | < 12 |

[1] This print experiment should not be construed as a benchmark for the comparison of programmed I/O performance between systems, but as a comparison of the efficiency of Direct Memory Access and PIO transfers.

[2] This time includes file seek and any inaccuracies due to coarse resolution of the real-time clock for short intervals.

**Figure 4. Results on Various Micro Channel Systems with a 4029 LaserPrinter**

## Analysis

When both parts of this experiment were performed on a 20 MHz 386 SX-based Micro Channel system, the printing task took 55 seconds longer under DOS and PIO than under OS/2 and DMA. A 20 MHz 386 SX computer executes approximately three million instructions per second (3 MIPS). Therefore, the savings is 3 million instructions/second x 55 seconds = 165 million instructions. By off-loading the print I/O to the DMA controller under OS/2, the processor can do other useful work simultaneously, resulting in higher productivity and throughput.

The 16-bit DMA parallel printer port operates at the same speed on all Micro Channel systems, because its speed is not significantly affected by the efficiency of the processor. The speed of the PIO parallel port, however, depends on the processor's efficiency in executing instructions. Therefore, the performance improvement with DMA print is most significant in systems whose processors have comparatively lower performance and therefore take proportionately longer to transfer the print data. This demonstrates that Micro Channel has value across the whole spectrum of systems from high-end servers to entry-level desktop computers.

### PS/2 Micro Channel Value to the User

It is not unusual to print a spreadsheet or document and then move on to the next task. If the processor moves the data using PIO, the delay before starting the next task can be long enough to impact the user's productivity. Operating systems that use the 32-bit and Virtual 86 modes of Intel 80386 and higher processors (for example, Windows 3.0, OS/2 2.0, and AIX® 386) may use 1,000 or more instructions to respond to an interrupt. Why so many? Recall that PC/XT/AT print adapters can move only one character per interrupt. For each interrupt, the processor must save the state of its registers and other system statuses. The advanced operating systems use many more registers and have far more complex system statuses; these account for the increase from 200 to 1,000 instructions per interrupt.

Let us look at how this affects the printing of very large files. Typically, printed text may contain 2,000 char-

acters per page, including much blank space. A page produced in graphics mode by desktop publishing or spreadsheet software may print in every position on the page; in this case, 5,000 characters per page is not unreasonable. (If we assume that pages have 5,000 characters, the 200,000-byte file transferred to the printer port in this experiment can be considered as 40 pages of graphic or spreadsheet output.)

Therefore, the transfer of the print data to the printer port using PIO and advanced 386 modes consumes 5,000 bytes/page x 1 interrupt/byte x 1,000 instructions/interrupt = 5 million instructions/page. If a printer could print one page per second – a reasonable expectation for laser printers in the foreseeable future – the calculation becomes 5 million instructions/page x 1 page/second = 5 million instructions/second, or 5 MIPS of computer power sapped from the system just to keep up with the printer. This is more computing power than is found in most 386 SX-based systems. As a result, the print speed could be limited by the capabilities of the system processor. Even if the printer were fully buffered, the same number of processor instructions would be necessary to move the same size block of print data. *Net*: The print performance would degrade, or concurrent applications using the processor would be impacted, or both.

As noted above, print performance degrades further when the computer is also running TSR utilities that perform background operations, such as 3270 operation or LAN attachment. In this case, the number of processor cycles available to print operations is further reduced. On a DOS/PIO system burdened by TSRs, the 200 KB file that was printed in this experiment took several minutes to print. (This leads to a strong case for preemptive multitasking, as in OS/2 2.0 – but that is a topic for another article.)

The performance does not degrade, however, in PS/2 Micro Channel systems that can delegate processor tasks to autonomous subsystems such as DMA print. For example, the DMA print function on Micro Channel systems absorbs 96% of the printer management work formerly done by the processor. The SCSI busmaster adapter further relieves the processor. Very little system overhead remains to be done by the processor, so that it can concentrate on executing operating system and application instructions. Therefore, much more of the processor's power is available for program execution.

*The DMA print function on Micro Channel systems absorbs 96% of the printer management work formerly done by the processor.*

In desktop personal systems, to equal the reserve processing power of a 20 MHz, 386 SX-based Micro Channel system with ABIOS and DMA print, a computer that does not have ABIOS and DMA print would perhaps need an additional 5 MIPS of processing capacity with a very high-speed printer. That level of performance is typical of 486 systems operating at 25 MHz or higher.

As a server, a file server system cannot afford to be impacted by adding the print serving function (which includes moving the data among hard disk, memory, and printer port) on top of the file server overhead in the typical PC/XT/AT system. Yet, for economic reasons, high-speed printers are often attached to systems that are shared among several network clients. File server performance degrades noticeably when a fast, fully buffered laser printer is attached to the network's file server system. Consequently, serving data to high-speed printers is usually delegated to a separate print server system.

The DMA print function, on the other hand, operates without significant impact on the processor, so the same server system can be used for both printing and network serving. This eliminates the costs for a second server, its server license, lifetime hardware and software maintenance costs, and several megawatt hours of energy. A cost saving of this magnitude easily justifies investment in an IBM PS/2 Micro Channel system with DMA print capability.

## Other DMA Channels Required

The seven PS/2 Micro Channel systems shown in Figure 4 also contain a DMA serial interface that can operate up to 345K baud – 10 to 20 times the capabilities of today's fastest systems. The DMA serial port uses two more DMA channels, one for read and one for write operations. Thus, three DMA channels are needed to support the DMA serial/parallel adapter. However, PC/XT/AT bus systems do not typically have three spare DMA channels. Only four DMA channels in PC/XT/AT systems are DOS-compatible. Those channels are already assigned to tape backup, Synchronous Data Link Control (SDLC), diskette, and network DMA interfaces by hard-wired connections on typical PC/XT/AT card designs. Consequently, it would be very difficult to retrofit DMA serial/parallel capability into cards designed for most PC/XT/AT systems. An EISA system may be able to configure one DMA serial/parallel combination, but the EISA system does not have enough available DMA channels for additional DMA serial/parallel ports. (For details, see the article "Comparing Architectures: Micro Channel and EISA," parts 1 and 2, in the April

1992 issue and in this issue of *Personal Systems Technical Solutions*.)

The SCSI file adapter is also on the system board and uses another priority level. A total of 15 DMA adapters or busmasters can be added to PS/2 Micro Channel systems, giving the Micro Channel the flexibility to accommodate future growth.

## Making This Experiment Portable

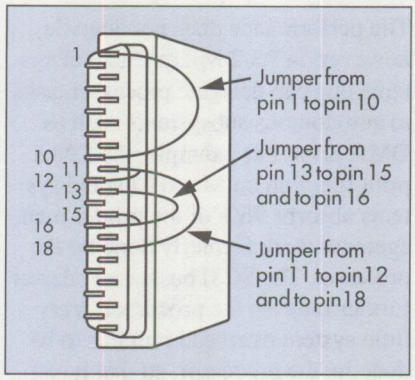
The experiment discussed in this article can be put into a package small enough to carry in your pocket, purse, or briefcase. There are two good reasons to do this: the experiment is then portable and can be performed at other locations, and the difference in the elapsed times will only reflect the system's DMA print performance, with no degradation due to the buffered printer itself, as shown in Figure 4.

The secret is to use a small piece of hardware called a *wrap connector* instead of a real printer. When properly configured, the wrap connector simulates a fully buffered printer. It attaches to the parallel port on the system board. It receives the data intended for the printer, but it discards that data right away and immediately asks for more. The wrap connector therefore removes delays caused by the printer interface, enabling the parallel port to operate at maximum speed. Because there is no overhead from a real printer, the results of the experiment will be a true reflection of the performance improvement.

Figure 5 shows the schematic for wiring the wrap connector to simulate a fully buffered printer.

## Results Using the Wrap Connector

Both parts of the experiment in this article were also performed on Micro Channel systems with the wire wrap connector attached to the parallel port in place of the 4029 printer. Because of the absence of printer interface delays, the performance in both parts of the experiment improved by as much as ten seconds. Figure 6 shows the results using the wrap connector.




The wrap connector can be made by connecting pin 1 to 10, pin 13 to 15 and 16, and pin 11 to 12 and 18 on a standard 25-pin male D-shell (DB-25) connector that mates with the parallel port. Standard D-shell connectors are available at electronics stores. An example is Radio Shack® part number 2761547B (Connector) and part number 2761549B (Cover), or their equivalents.

**Figure 5. Wire Wrap Connector**

In this "pure" environment, the performance advantage of Micro Channel, DMA, ABIOS, and OS/2 2.0 becomes even more striking. Now, comparing the performances of PIO versus DMA on the 386 SX running at 20 MHz, we see that 57 seconds were reduced to less than 2 seconds – a 96.5% improvement. That same 386 SX-20 running OS/2 and DMA was 83% faster (10 seconds saved divided by 12 seconds) than even the fastest 486 DX-50 running DOS and PIO.

## Creating this Experiment from OS/2 2.0 Diskettes

This experiment requires some preliminary creation of diskettes, data files, and a batch program.

### Preliminary Steps

1. Copy the OS/2 2.0 Installation Diskette and label the copy "Installation Utility."
2. Copy OS/2 2.0 diskette #1 and label the copy Diskette 2.
3. On the newly created Diskette 2, modify `CONFIG.SYS` by changing the line `basedev=print01.sys` to `basedev=print02.sys`. This enables the DMA print driver for the listing of Micro Channel systems.

| Processor | DOS with PIO[1] | Micro Channel with DMA, ABIOS, OS/2 2.0[2] |
|---|---|---|
| 386 SX at 20 MHz | 57 seconds | < 2 seconds |
| 386 SLC at 20 MHz | 29 | < 2 |
| 386 DX at 20 MHz | 48 | < 2 |
| 386 DX at 25 MHz | 39 | < 2 |
| 486 DX at 25 MHz | 28 | < 2 |
| 486 DX at 33 MHz | 19 | < 2 |
| 486 DX at 50 MHz | 12 | < 2 |

[1] This print experiment should not be construed as a benchmark for the comparison of programmed I/O performance between systems, but as a comparison of the efficiency of Direct Memory Access and PIO transfers.

[2] This time includes file seek and any inaccuracies due to coarse resolution of the real-time clock for short intervals. Variations from 1.5 to 2.5 seconds are possible.

**Figure 6. Results on Various Micro Channel Systems with a Wrap Connector**

```
echo off
echo ********************************************************************************
echo *                  DMA Parallel Port Performance Test                          *
echo *This test demonstrates the length of time to transmit a file to the    *
echo *parallel port. Running under the DOS operating system, data is         *
echo *transmitted using PIO, resulting in single-byte transfers.  Using the  *
echo *OS/2 2.0 operating system, printing results in large buffer transfers, *
echo *achieving higher performance. OS/2 2.0 takes advantage of a new DMA    *
echo *hardware feature available on PS/2 Micro Channel models 56, 57, 80-A21,*
echo *80-A31, 90, and 95. The improved performance results in higher printer *
echo *utilization by running printers at rated speed while leaving more time *
echo *for the processor to execute applications.                             *
echo *This program is the property of IBM Corporation.                       *
echo *Chet Heath, Barry Noto, and Frank Schroeder, IBM Boca Raton, Florida.  *
echo ********************************************************************************
echo on
pause
echo off
echo ********************************************************************************
echo *The following instructions set up the test to run from the hard disk.  *
echo *Running from the hard disk more closely shows Micro Channel performance*
echo ********************************************************************************
echo on
c:
md c:\prtdemo
cd c:\prtdemo
copy a:\test c:\prtdemo
echo off
echo ********************************************************************************
echo *The following instructions perform the test.                              *
echo ********************************************************************************
echo on
time <CR
copy testprt1 lpt1
time <CR
echo off
echo ********************************************************************************
echo *Calculate the elapsed time, then press any key to continue.               *
echo ********************************************************************************
echo on
pause
echo off
echo ********************************************************************************
echo * The following instructions remove the test from the hard disk.          *
echo ********************************************************************************
echo on
cd c:\
del c:\prtdemo <a:\test\YES
rd c:\prtdemo
a:
********************************************************************************
```

**Figure 7. Batch Program to Run Experiment**

*Note:* For ISA machines, another diskette without this change must be made with the proper driver to enable ISA programmed I/O ports. Otherwise, the version of Diskette 2 would be identical.

*Warning:* If some ISA is operated with the Micro Channel driver, the operation will terminate prematurely with no data transfer and/or other unpredictable results.

4. Create a subdirectory on Diskette 2 named TEST.
5. Create the file TEST.CMD, as shown in Figure 7, in the TEST subdirectory on Diskette 2.
6. Copy the TEST.CMD file and name the copy TEST.BAT in the subdirectory. OS/2 2.0 will execute the

TEST.CMD file in the same manner that DOS executes TEST.BAT.

7. On the hard disk, create a file called TESTPRT1 by copying text files together until the TESTPRT1 file has approximately 200,000 bytes. This is done by listing text file names all on one line, with no spaces around the plus signs, as shown below:

```
COPY
TEXTFILE.one+TEXTFILE.two+...
```

   A file that contains only null characters (00 hex) will not print if you use a real printer instead of a wrap connector for this experiment. After creating TESTPRT1 on the hard disk, copy the completed file to the TEST subdirectory on Diskette 2.

8. Make a three-byte file called CR and fill it with a single carriage-return character. Add this file to the TEST subdirectory. The carriage-return character in the CR file responds to the time prompt, enabling the batch file to continue processing. Most file editors put three characters into this file – the carriage-return character, a line-feed character, and the end-of-file character.

9. Make a four-byte file called YES and fill it with a single letter Y and the carriage-return character. (The file editor will add the line-feed and end-of-file characters.) Add this file to the TEST subdirectory. This file enables the batch file to respond to the "Are you sure?" prompt when deleting the contents of the subdirectory on the hard disk.

10. (optional) The README file is placed on Diskette 2 for convenience, in case these instructions are misplaced.

11. For this experiment to be portable, make the wrap connector shown in Figure 5.

When steps 3 through 10 have been completed, the TEST subdirectory on Diskette 2 will resemble this:

```
.              <DIR>
..             <DIR>
TEST      CMD     3115
TEST      BAT     3115
TESTPRT1        196606
CR                   3
YES                  4
README            4076
```

You are now ready to begin the experiment. Parts 1 and 2 of the experiment are described below.

### Part 1

1. Be sure to purge print spoolers and any other TSR programs before proceeding.
2. Boot DOS from the hard disk or from diskette.
3. Insert Diskette 2 into drive A.
4. If you booted from the hard disk, type `a:`.
5. Type `cd \test`, then press Enter.
6. Type `test`, then press Enter. This starts the batch file TEST from Diskette 2.
7. Follow the prompts in the batch file. Calculate elapsed time from the start and end times that are displayed, and write it down.

### Part 2

1. Insert Diskette 1 into drive A.
2. Press Ctrl-Alt-Del to start OS/2.
3. At the prompt, replace Diskette 1 with Diskette 2, then press Enter.
4. Look for the first screen that allows you to press Esc to cancel.
5. Press Esc. OS/2 will display an [A:\] prompt.
6. Type `cd a:\test`, then press Enter.
7. Type `test`, then press Enter. This starts the batch file TEST from Diskette 2.
8. Follow the prompts in the batch file. Calculate elapsed time from the start and end times that are displayed, and write it down.

After completing Part 2, compare the elapsed times generated in Parts 1 and 2.

## Acknowledgments

I thank Frank Schroeder and Barry Noto of the OS/2 2.0 development team for realizing that if the OS/2 2.0 installation process is canceled, the system is placed in a state in which the DMA print function remains enabled. Without that critical perception, a diskette-based, portable experiment would not have been possible. Frank is also the author of the batch program used in the experiment.

***Chet Heath*** *is a Senior Technical Staff Member and engineer at IBM's Entry Systems Technology Laboratory, in his twenty-second 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 was named one of the ten most influential people on PCs in the '80s by* PC User *magazine in the U. K.*

# High-performance File Server Design

**Gregg McKnight, Avery Lyford, and Jerry Isaac**
**IBM Corporation**
**Boca Raton, Florida**

A growing area of opportunity for the personal systems business is the local area network (LAN) file-server market. Unlike past PC designs, the file server is designed with communications in mind. Instead of being a stand-alone PC, a file server is designed to rapidly move data, such as a Lotus® 1-2-3® spreadsheet or an application program, to client systems attached to the LAN. The design of a system for high-speed communication with a large number of clients is fundamentally different from the design of a stand-alone PC.

To meet the challenges involved, the development team made some fundamental changes in the approach to the requirements and design process. The result of that effort is the new PS/2 Model 95 486 50 MHz Server. This same technology is available to existing owners of PS/2 Model 95 systems as an upgrade option. The new processor complex implements a completely new design for supporting the Intel 50 MHz 486 Memory Interface and IBM's Micro Channel bus master interface. In addition to its functional and performance improvements, the processor complex supports reliability features such as error checking and correcting (ECC) memory technologies.

This article describes the approach taken by the development team to design a reliable, high-performance server that meets customers' needs in mission-critical LAN environments.

## Server Performance Requirements

Developing a system capable of handling a large number of I/O operations while providing high-performance processing and enhanced reliability presented a number of complex and difficult design problems. Servers are increasingly being used in mission-critical applications; therefore, downtime can be very costly. Furthermore, servers require vast amounts of memory for the network operating system, File Allocation Tables (FATs), disk cache and network I/O buffers. These increased memory requirements directly conflicted with the requirement for increasing reliability.

## Bus Master I/O Performance Interference

High-performance servers typically employ bus master network and disk adapters. A Micro Channel bus master adapter is advantageous because of its ability to "master the bus" and move information to and from memory on its own behalf. This is in contrast to a non-bus master adapter that requires the host processor to coordinate the data transfer.

The ability of a bus master adapter to perform unassisted data transfers improves the efficiency of a server and allows it to support an increased workload or additional users. However, as bus master adapters burst larger and larger quantities of information to and from memory, an interesting phenomenon occurs. As the bus master consumes more of the memory bandwidth, the ability of the processor to access memory is impacted.

Bus master memory utilization interferes with the ability of the host processor to access memory during instruction execution. For example, in some systems 50-percent memory utilization by bus masters causes a 50-percent decrease in system processing power. This means that the processor is spending 50 percent of its time waiting to access memory, or that it can sustain only half of its original instruction processing performance. Thus, there are two factors that contribute to increased processor utilization with increased file-server workload. First, an increase in I/O operations is typically coupled with an increase in CPU load since each I/O operation requires some processing. Furthermore, the bus master I/O itself causes interference with the host processor's ability to execute instructions. The greater the amount of information or the number of I/O operations sustained by the bus master adapters, the greater the amount of interference encountered by the processor as it tries to access the same memory space (see Figure 1). This vicious cycle results in decreased server capacity in poorly designed systems.

## Solution: The 486 50 MHz Server

The 50 MHz server conquers the I/O bandwidth interference problem with a unique system design, which provides a "dual bus" or independent memory paths for the processor and bus master devices. This means that performance of the file server – compared to that of conventional PC designs – is significantly improved (see Figure 2).

The ability of the processor to access memory and execute instructions is insulated from the interference of bus

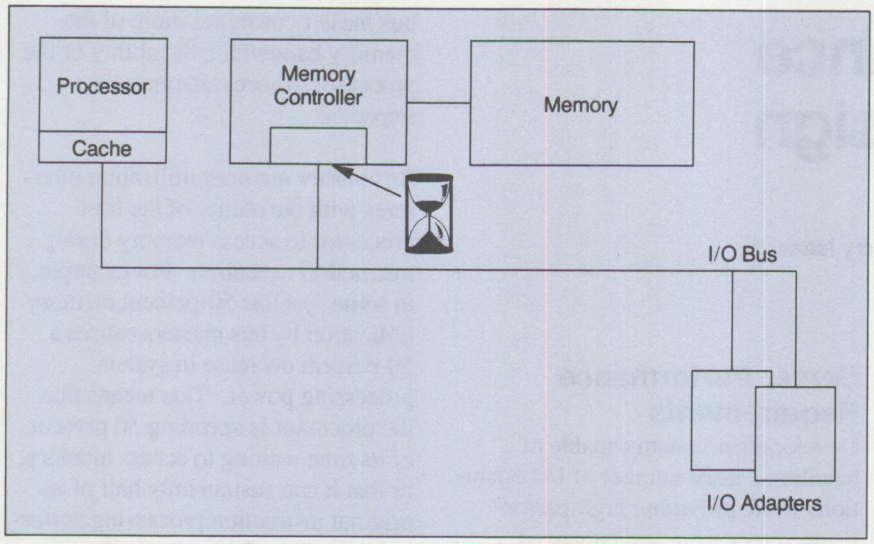



**Figure 1. Logical View of the Conventional PC Design and Resulting Contention of Processor and I/O Adapters for Memory Access**

master adapter accesses to memory. This translates into more consistent processor performance as bus master I/O load or access to memory increases. In contrast, the typical PC processing performance is degraded more rapidly as bus master I/O load increases (see Figure 3).

Beyond the dual-bus capabilities, the 50 MHz server offers additional performance by supporting the Micro Channel 40MB streaming data protocol. This means that bus master adapters designed to support the 40MB streaming data protocol can now be used in a 50 MHz server system to significantly improve bus master data throughput. For example, older Micro Channel machines, such as the PS/2 Model 80-311, are capable of supporting 300-nanosecond Micro Channel cycles. The IBM 16/4 Token-Ring Bus Master Adapter/A installed in a Model 80-311 is capable of bursting data over the Micro Channel at up to about 6.6MB per second. This same card in the 50 MHz server is able to burst data at 20MB per second, and a 32-bit token-ring adapter with streaming data support would be able to burst at 40MB per second.

*Beyond the dual-bus capabilities, the 50 MHz server offers additional performance by supporting the Micro Channel 40MB streaming data protocol.*

## Increased Server Reliability

Market research shows that customers are demanding increased server reliability. Examining a history of server failures highlights that many server outages can be attributed to memory failures. Furthermore, servers employ significantly more memory than most desktop systems. Since the potential for memory-related outages increases with the amount of available memory, a solution had to be found.

The solution was to design additional reliability features into the 50 MHz server processor complex. These features include support for both parity and error checking and correcting (ECC) memory. ECC memory offers the desirable characteristic of detecting and correcting single-bit memory errors. In addition, ECC memory technology can detect very rare double-bit, and some triple- and quadruple-bit errors. This is a great improvement over the current state of the art for OS/2® LAN Server, Novell® NetWare® and Banyan® Vines® based file servers.

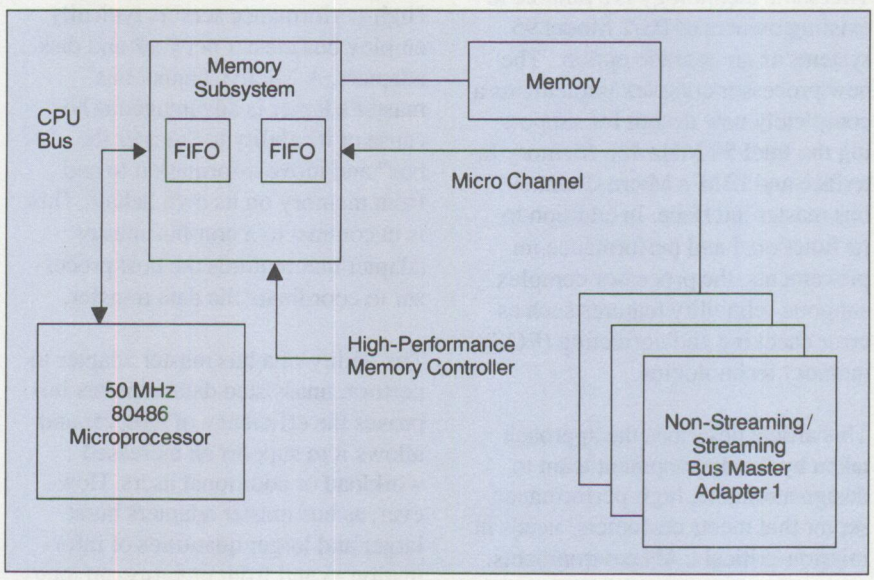



**Figure 2. Improved I/O Performance with Dual Bus Technology**

Conventional PC-type servers employ parity memory. Parity memory will detect only single-bit errors and stop all processing when an error is detected. When a single-bit error occurs in the 50 MHz server, the ECC circuitry detects and corrects the error, allowing processing to continue uninterrupted. Of course, the system logs the address of the failing memory location, allowing the server manager to know an error condition has occurred. The server manager then has the option of replacing the failing memory module at some later time. In addition to correcting single-bit memory errors, the 50 MHz server checks for errors during transmission of data over the Micro Channel. This additional data checking significantly improves data reliability. Conventional PC systems, such as the Compaq® Systempro® and the Dell® 486/50, do not implement reliability checking. In fact, no AT-bus or Extended Industry Standard Architecture (EISA) bus-based system offers this capability since it is not part of the architectural definition for those buses.

The development of the 50 MHz server employed unprecedented performance modeling and analysis early in the design cycle. Several new tools were used in an effort to better understand how various design alternatives would impact customer requirements. For example, a processor simulation model was developed to explore the results of changes to the processor-memory subsystem. This included a simulation and analysis to understand and virtually eliminate any performance impact from the additional function required to support error checking and correcting memory.

The chip design process for this system was significantly different from the process used for previous PS/2 systems. Given the complexity of the logic required to support these new

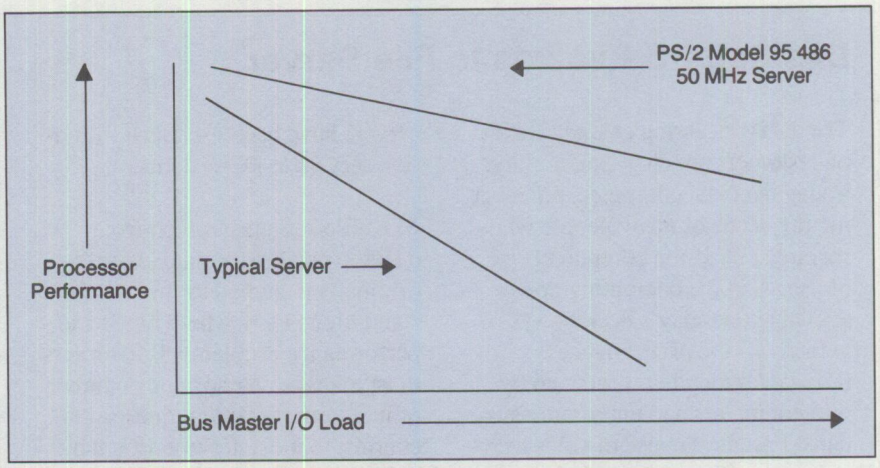



**Figure 3. Performance of 50 MHz Server vs. Conventional PC**

functions and the importance of reducing cycle time, a new design and development process was adopted. Developing the integrated circuits that embody the logic that determines how a system works is among the most arduous and time-consuming steps in developing a new system.

## Design Verification

### Use of Integrated Circuits

Integrated circuits, which contain many transistor-equivalents on a single piece of semiconductor material, can be considered the fundamental building blocks of personal computers. The level of integration achieved by silicon foundries over the past decade has increased by two orders of magnitude, from tens of thousands to more than a million circuits on a single microchip.

Personal computers have been leveraging this technology trend as evidenced by the proliferation of very large scale integration (VLSI) chips interconnected on system boards and cards. Reflecting the trend to achieve higher levels of integration, the two-card 50 MHz server processor complex contains nine unique application-specific integrated circuits (ASICs). Only through the pervasive use of ASIC technology could a system such as the 50 MHz server be built.

Developing an ASIC-based system can be a challenging task. Having such high levels of integration makes it more difficult to troubleshoot problems that arise during the bring-up of the system because there are fewer test points that can be probed. Likewise, a definitive correction to a design error may not be possible without incurring the expense and delay of recycling an ASIC. Therefore, to minimize the number of design passes, the system development team exercised a very thorough verification process prior to fabricating any of the integrated circuits.

The 50 MHz server processor complex development team focused on three particular areas: digital logic verification, analog and electrical verification, and timing analysis.

### Digital Logic Verification

Digital logic verification is the process of simulating the functional behavior of the design. The logic verification was run at both the unit level (single ASIC) and the system level (interconnection of ASICs). The unit level of simulation was run against each ASIC using an IBM simulator capable of modeling the

## Desktop PC Evolves to File Server

The data processing characteristics of customers are changing rapidly. Today personal computers are meeting the needs of many people who require application computing power. And the computing power available on today's desktop systems rivals the performance provided only a few years ago by minicomputers and mainframe systems. Further, the use of LAN technology to connect these desktop PCs gives users access to centralized storage and the ability to share information and system resources. LAN file-serving technology offers today's users potential improvements in cost, productivity and system availability.

Typically, the low-end, 80286- and 80386-based PCs provide sufficient processing capability to handle the basic applications required by most users. High-performance 80386 and 80486 systems handle those applications that require greater amounts of processing resource. Low cost and the ability to meet the wide range of users' processing requirements are two fundamental advantages of employing PCs. A major disadvantage, however, is the inherent inability of PCs to directly share data.

LAN technology enables PCs, as well as minicomputers and mainframes, to share information via a network. In a LAN environment, a powerful machine – designated as a file server – provides data storage and resource-sharing to LAN-attached PCs, referred to as clients. A typical PC server is a high-performance 80386- or 80486-based system with large disk capacity, a high-performance network adapter, and a network operating system. This approach is not new; in fact, for many years, file servers have been used as information servers to PC systems.

Most file servers evolved from desktop machines; many were originally designed to function as a stand-alone PC – which has led to performance problems. File servers must manage an enormous number of input/output (I/O) requests – in contrast to most desktop PCs that run computing- and graphics-oriented applications. The evolution in processor speed from the 8086-based PC to the 80486-based system has accounted for more than a tenfold increase in microprocessor performance. However, disk, network adapter, and I/O bus performance has failed to keep pace with improvements in microprocessor performance. As a result, most desktop systems are not well-suited for file-serving applications.

In addition to file serving, PC systems are used as print, database, E-mail and communications servers. (Typically, file serving is generalized to include print, file and E-mail serving.) File-serving applications treat the server as a shared disk drive or printer. Entire files are exchanged between the network-attached client workstations and the server. This is in sharp contrast to database or client/server environments in which application processing is divided between the client workstation and the server. For example, rather than request an entire file to update one record, a database client will request only the required record. The database server interprets this request and retrieves the corresponding record for the client. In this scheme, a single record is passed from the server to the client. The net effect of this distributed application is that the server is now responsible for record searching, increasing both disk and microprocessor resource utilization and, at the same time, decreasing network utilization.

Database and client/server applications usually involve entirely different system requirements. Thus, file-server and database-server system requirements are evaluated differently. The reason is that each application has unique characteristics that exercise system resources differently. For file servers, the ability to move large amounts of data rapidly is critical, whereas in a database server, it is important that the system be able to do processing concurrently.

Market research by Dataquest and Infonetics suggests that PC file-server systems are likely to repeat the usage trends of minicomputer and mainframe server systems, which depict an evolution from file-server applications to more sophisticated database or client/server applications. Research indicates that in mature mainframe environments, database usage is approximately equal to that of file serving. Currently the overwhelming use of PC server machines is for file serving. However, it is very plausible that, over time, the number of database installations will increase to equal the number of installations currently employing file serving. At the same time, PS/2 Micro Channel systems are increasingly being used to provide solutions to business problems that a few years ago would have required a minicomputer or mainframe. This results in much more stringent requirements from customers for the reliability and availability to complement the increased system performance.

Boolean logic states of zero and one, as well as additional states entitled "uninitialized," "indeterminate" and "high impedance." This simulator also incorporated estimated timing delays into the logical paths in order to model the actual ASIC characteristics as closely as possible.

Once each ASIC design team felt satisfied that their chip performed to specification at the unit level, a system-level model was built to verify that the ASICs communicated correctly. In addition to the ASICs, models were developed for the i486, Micro Channel bus I/O adapters, system memory and any miscellaneous logic residing within the processor complex. The interconnection of all these elements of the system model was specified by the same database used to wire the two 50 MHz server cards, thereby ensuring that the cards were wired correctly. Two IBM-developed simulation tools were used for system-level verification. One tool was a dedicated hardware machine designed specifically for the high-speed simulation of integrated circuits, and the other was a software simulator that ran on an IBM mainframe. These two simulators are different from the one (described above) used during unit-level simulation.

The hardware simulator was used to simulate more complex and elaborate scenarios, such as simultaneous processor and Micro Channel bus I/O activity to shared memory. The software simulator was quite beneficial during the initial model bring-up since the process of editing the design of an ASIC, followed by a compilation of a new system model, took only minutes.

The system-level simulation proved beneficial by:

- Resolving errors both within and between ASICs
- Serving as a microcode development tool since the Power-On Self-Test (POST) code was run and partially developed on the system model
- Providing a vehicle for problem resolution during hardware bring-up
- Providing a means for training engineers in the operation of the system, which served them well during bring-up of the system

### Analog and Electrical Verification

Analog and electrical verification of the 50 MHz server processor complex was performed to ensure that the electrical specifications as well as proper signal quality were maintained throughout the design. The primary tool used was an IBM-developed analog simulator that models the resistive, capacitive and inductive characteristics of a physical node. The analog operation of every interchip signal was individually modeled under best-, nominal- and worst-case conditions to ensure that acceptable margins were maintained in such areas as overshoot, undershoot, and slope reversal. This type of analysis is required to meet the operational and reliability specifications of the system. In addition to signal quality, this analog simulator yields signal delay data that is used in the third area of verification, timing analysis.

### Timing Analysis

Timing analysis must be performed to guarantee that a signal, both within and between ASICs, arrives at its destination within a specified time. The timing analysis conducted on the 50 MHz server was done at two different stages. First, each ASIC was individually analyzed. This unit-level timing verification was implicitly done during unit-level simulation as well as separately using an IBM-developed timing analysis tool that checks for internal ASIC timing violations under best-, nominal- and worst-case conditions. External signal timings were also verified at this stage.

Once an ASIC designer felt comfortable with the timing margins at the unit level, the external timing information derived from this level, as well as the timing data from the analog simulator, was used to analyze the timings on a system level. Every signal interconnecting all elements of the processor complex was analyzed to ensure that the correct timing margins were maintained under both best- and worst-case conditions.

Thorough execution of digital logic verification, analog and electrical verification, and timing analysis prior to fabricating any of the ASICs or cards comprising the processor complex yielded tremendous benefits. Following the insertion of the two-card processor complex into the PS/2 Model 95 system board, the engineering development team was able to boot three different operating systems within five days (DOS, OS/2 and AIX). Also, the total number of problems detected and, consequently, the number of ASIC design passes required, was held to unprecedented minimums.

Most important, the verification process ensures that the customer receives a file-server system that provides outstanding quality and reliability.

## Benefits to the Customer

Technical jargon such as improved bus master throughput, CPU performance, ECC memory, and Micro Channel data checking describe attributes of the system, but what do they really mean to the customer?

In today's – and tomorrow's – LAN file-server environment, the impor-

tance of overall system design and a balanced system becomes paramount. File-server subsystems should be combined to maximize value to the customer. The whole can be greater or less than the sum of its parts. Often file-server performance is defined by the synergy of the corresponding subsystems. The PS/2 Model 95 486 50 MHz Server is a leadership server system because of its high-speed I/O bus, enhanced reliability memory subsystem, and dual-bus memory architecture. Combined with optimized 32-bit I/O adapters, this system will be faster than any currently shipping PC server. Furthermore, high-performance disk subsystems can be coupled with advanced system-management features and the 50 MHz server to effectively increase capacity to more than several hundred simultaneous users running popular applications such as Lotus 1-2-3, WordPerfect® and cc:Mail.® And the system is designed to handle even higher bandwidth applications such as multimedia. But even more important, this performance improvement is complemented by increased system reliability. Users now have server solutions available to support large amounts of memory and avoid downtime resulting from frequent memory failures. Perhaps most exciting is a view of the future – the 50 MHz server technology could be swept into lower-cost modules employed in entry-level through ultra-high-end server systems.

*ABOUT THE AUTHORS*

*Gregg McKnight is the lead engineer of the LAN Systems Performance department in Entry Systems Technology. Avery Lyford is the manager of LAN Systems Performance in Entry Systems Technology. Jerry Isaac is an advisory engineer in the VLSI Development department in Entry Systems Technology.*

# *International Catalog of Micro Channel Adapter Cards*

The ninth edition (June 1992) of the *International Catalog of Micro Channel Adapter Cards* includes information on over 1,000 expansion adapters from 382 developers in twenty-two countries worldwide for the following IBM Micro Channel architecture-based systems:

- PS/2 Computers
- RISC System/6000® POWERstations and POWERservers
- Micro Channel 370 Models

The information included in this edition was certified as accurate and up to date by an authorized representative of each developer prior to being included in this edition. The adapter listings are presented in two parts – those available from IBM and those available from other developers – and each adapter card is listed alphabetically by developer under one of the 46 category headings that best describes its function. The table of contents in the front will assist you in quickly locating available adapter cards to perform certain functions.

Each listing provides the developer name, address, phone and FAX number (if available), adapter name, Programmable Option Select Identifier (POS ID), bus master designation, date of availability, and a brief product description.

An index of developers in the back of the catalog assists you in locating information about all of the listed products from a given developer.

Part 3 includes information and phone numbers to help potential developers of Micro Channel bus expansion cards obtain the necessary documentation and assistance to support the Micro Channel architecture-based systems from IBM.

The retail price of the ninth edition of the catalog is $13.95 U.S.

IBMLink customers with PUBORDER access may order directly from Mechanicsburg by specifying IBM Form G360-2824-08.

IBMLink customers without PUBORDER access may order directly from the publisher, Tecnicom Printing & Publishing Corp. The cost is $13.95 each plus $2.00 per copy for shipping and handling. Total price is $15.95 U.S. Ordering instructions are as follows:

- Send an electronic note on IBMLink to Tecnicom's Catalog Order Department (HONE83 at LNK2212) requesting a copy of ISBN #0-9632894-0-3. Include your MasterCard™ or VISA™ credit card number, expiration date, and complete authorized signature as it appears on your credit card. Please include your name, complete business address, and phone number in your electronic note.
- Call Tecnicom's Catalog Order Department at 1-800-362-6134 or send a FAX to 1-216-349-4771. (When ordering volume quantities, please place your order by phone or FAX.)

  or

- Copy and complete the order form on the next page and mail it to:

Tecnicom Printing & Publishing Corporation
Attn: Catalog Order Department
6810 Cochran Road
Solon, OH 44139
U.S.A.

Since 1988 a listing of available adapters for IBM Micro Channel bus-based systems has been made available twice a year. The 10th edition of the catalog will be published by Tecnicom in December. Should you wish to place a subscription to automatically receive the next edition of this publication, you may contact your nearest IBM Branch Office and ask to place an order for IBM Form G360-2824 under the System Library Subscription Service (SLSS).

**Order Form for Adapter Catalog on following page**

## MAIL ORDER FORM

Please send me one copy of the ninth edition of the ***International Catalog of Micro Channel Adapter Cards*** at $13.95 U.S. plus $2.00 for shipping and handling. Total cost is $15.95 U.S.

[ ] Check Enclosed [ ] Money Order Enclosed
(Payable to "Tecnicom Corporation".)

*Please print*

Name_____

Title_____

Company_____

Address_____

City_____

State/Province_____

Country_____

ZIP/Postal Code_____

PHONE_____ FAX_____
(If out of the U.S., please include country & city code.)

# Bus Masters and OS/2

*Bob Williams*
*IBM Corporation*
*Boca Raton, Florida*

*Editor's note: This article was written in 1989. While the basic concepts are the same for all versions of OS/2, the article was written specifically for versions of OS/2 prior to 2.0. References to 286 architecture apply equally to 386 or 486.*

**This article focuses on the bus master feature of Micro Channel architecture as it relates to the OS/2 multitasking operating system. The bus master feature allows full exploitation of the benefits of the overlapped I/O characteristic of a multitasking environment. These benefits are directly related to the concurrent programming paradigm (model). A multitasking application will generally isolate I/O logic to one or more separate I/O threads of processor execution. A system exploiting a bus master I/O subsystem can take advantage of this model and reduce the percentage of the system processor needed to drive the I/O. Examples discussed include an intelligent file I/O subsystem and support for an intelligent graphics processor. The system configuration requirements and operating system services necessary to support such Micro Channel bus masters are also summarized.**

## New Opportunity for Overlapped I/O

The multitasking environment of the OS/2 operating system provides the user with many benefits, such as the ability to support the "messy desk" concept where multiple applications can be in various stages of execution. But executing multiple applications is just the beginning of the benefits of a true multitasking system. The ability to support reliable background execution of communications and data base applications and subsystems is where the power of OS/2 begins to be felt. What may not be obvious is that, as software developers begin exploiting the concepts of the multithreaded programming model, the underlying system architectural requirements change. This is one of the areas in which Micro Channel architecture is a major breakthrough in the PC-compatible arena. The synergy between the bus master feature of Micro Channel architecture and the support structures and capabilities of the OS/2 multitasking programming environment is the subject of this article.

## The Multithreaded Programming Model

The multithreaded programming model allows the isolated (asynchronous) processing requirements of software to conveniently execute independently using the concept of separate "threads of execution." A thread of execution can be thought of as a processor state – a snapshot of the values for the instruction pointer and other registers, along with a stack buffer. Multitasking can be thought of as the mechanism for overlapping the execution of more than one thread of execution (ignoring for the moment the various definitions for task, process, and thread).

Some examples of the independent processing that can be done by separate threads of execution are receiving user keyboard input, updating a video display, and performing file I/O.

In many ways this programming model simplifies the structure of the highly interactive software that is characteristic of the popular programs found in personal computer systems. Furthermore, software structured in this fashion is well-positioned to exploit the distribution of system processing. Moving I/O execution responsibility out to intelligent I/O subsystems reduces the processing burden on the system processor. This is precisely the opportunity presented by the multiple bus master capability of Micro Channel architecture.

There are many publications available on both the fundamentals of concurrent programming and the specifics of the OS/2 multitasking environment. Instead of covering this same ground, this article looks beneath the application programming level. We use examples to highlight the system structures and services available to build intelligent I/O subsystems exploiting the bus master capability of Micro Channel architecture. To better understand the examples, let's look first at some important points about the system extension mechanisms and structures provided by OS/2.

## System Structures - Dynamic Link Libraries and Device Drivers

OS/2 application programs generally take advantage of bus master devices through the use of a subsystem. Examples of two OS/2 subsystems are the file system and the Presentation Manager™ windowing and graphics services. These subsystems support extension mechanisms that allow customization of much of the device-specific support.

Primarily for performance reasons, the Presentation Manager is designed as a dynamic link library (dynamically bound subroutine) subsystem that executes within the most protected and least privileged application privilege domain (Intel 80286 ring 3). Device-specific routines that need to do IN/OUT instructions to hardware registers can be designed to execute within the less protected I/O-privilege domain (Intel 80286 ring 2). A mechanism is needed to allow Presentation Manager to adapt to the unique programming interface of a specific graphics device. Presentation Manager is structured to support a replaceable device-specific module called a presentation driver. Such a presentation driver (also called a presentation device driver) is implemented as an installable dynamic link library module.

On the other hand, to preserve data integrity and for other reasons, the file subsystem is designed as a privileged extension of the operating system within the least protected and most privileged kernel-privilege domain (Intel 80286 ring 0). A customized set of routines allowing the file system to drive a specific device is contained in a structure called a device driver (also called kernel device driver or physical device driver). Device drivers are the components of the operating system that contain the hardware interrupt handlers.

Figure 1 shows the relationship of dynamic link libraries and device drivers to the rest of the system.

With this brief description of two of the important OS/2 system extension mechanisms as background, let's look at some examples of intelligent I/O subsystems that exploit bus masters.

## Intelligent I/O Subsystems

**A DASD Device:** One of the most obvious opportunities for exploiting the Micro Channel bus master capability is to support one of the most important subsystems in the operating environment – the file system. File I/O is often optimized within a file system using caching and asynchronous prefetch techniques. These techniques can help speed up individual file I/O operations from an application-program perspective. Unfortunately, they also tend to add additional system overhead to an already slow and expensive process. This overhead results regardless of whether direct memory access (DMA) or programmed I/O is used, because extra data is moved between memory and secondary storage. The effect of this additional overhead can be minimized by using a bus master-based intelligent I/O subsystem to reduce the burden on the system processor.

In any computer system, the file system generally maps a user file I/O function call (read/write) into a series of block I/O requests to a direct access (secondary) storage device (DASD). On the Personal Computer AT class of machines, it was up the disk/diskette device driver to translate these individual I/O requests into the low-level disk controller commands. Assume for this example that the file system is extended to support intelligent DASD subsystems such as those that are now practical with the bus master capability of the Micro Channel architecture. This intelligent DASD subsystem is supported by borrowing a concept common among mini and mainframe systems.

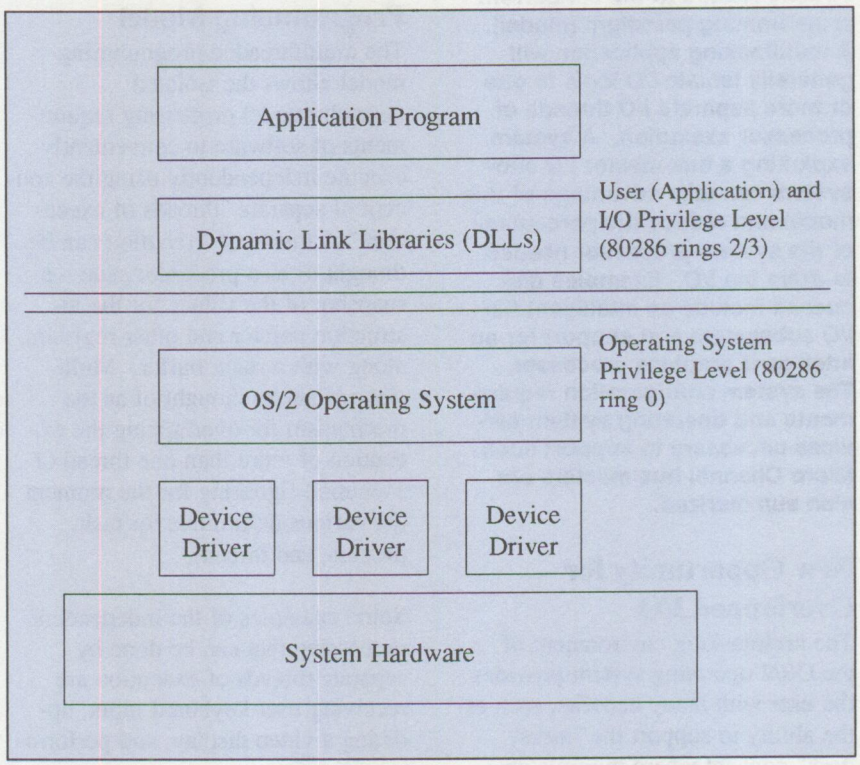



**Figure 1. Relationship of Device Drivers and DLLs to System**

The file system no longer sends single-block I/O requests individually to the disk device driver. Instead, it can build a list data structure that describes the entire series of block I/O requests corresponding to a user I/O function call (similar to System/370 channel programs). This data structure includes the appropriate read/write commands, pointers to user I/O buffers, and other relevant DASD management information. The device driver for the intelligent I/O adapter is then only responsible for initiating the I/O operation. This can be as simple as performing any necessary preparation of the user's buffers for DMA, and then passing the command data structure to the DASD bus master. At this point the device driver yields the system processor and allows the bus master adapter to run totally asynchronously to the rest of the system.

The thread in the device driver remains blocked (asleep) on some synchronization primitive (that is, semaphore) until the bus master completes the list of operations. While this I/O thread is blocked, the system processor is free to execute other threads in the system. The bus master signals completion of the command list by issuing a hardware interrupt. The device driver, having previously registered an interrupt handler, gets control and clears the semaphore to wake up the blocked I/O thread.

Figure 2 shows a representation of the file system and the DASD device driver and their relationship to the rest of the operating system.

The overall performance of multithreaded software in this scenario increases as it receives additional system processor time for the non-I/O-bound threads. This is because the bus master enables the system to overlap the I/O with other operations driven by the system processor. We will look in more detail at the services the device driver uses to manage a bus master adapter.

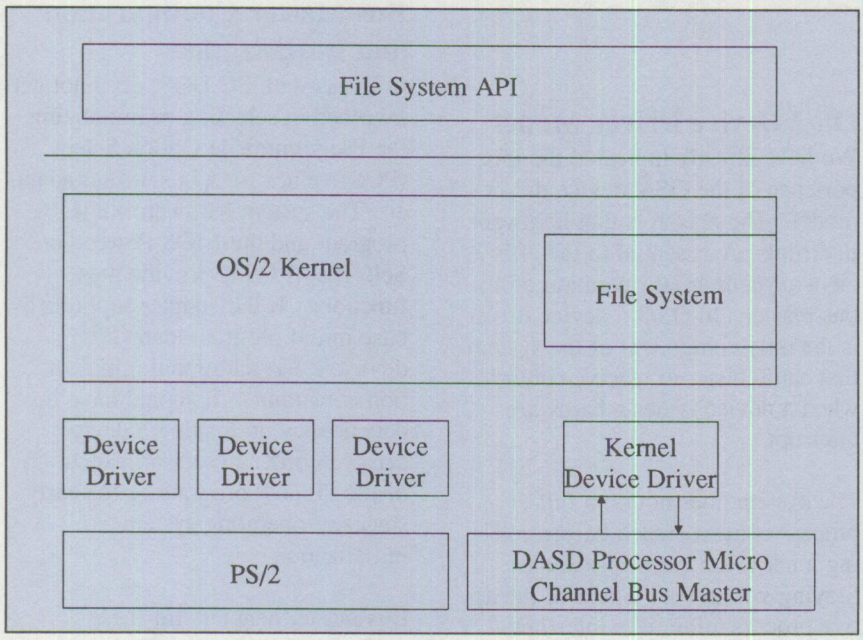



**Figure 2. OS/2 Bus Master DASD Processor Subsystem**

But before looking at these services, let's briefly consider another type of intelligent I/O subsystem.

**A Graphics Processing Device:** The same bus master concepts discussed in the previous DASD example can be applied to an intelligent graphics subsystem. Complex graphics operations can be system-processor-intensive, requiring manipulation and movement of large quantities of data. A thread within a graphics application (or text application using graphics fonts) can spend many system-processor time slices down in the video subsystem (that is, OS/2 Presentation Manager). Micro Channel architecture enables the development of an intelligent graphics I/O processor that can offload much of the work from the system processor. Such graphics processors can be optimized for very fast raster- and/or vector-based graphics operations.

In the OS/2 Presentation Manager (PM) environment, a bus master subsystem more than likely includes at least two subcomponents. One PM presentation driver (video display or printer/plotter driver) component exploits the PM open architecture to register to receive control on certain graphics operations. These graphics operations can be offloaded to the bus master instead of executing the corresponding function in the system-processor-driven Presentation Manager graphics engine software.

The second component of the bus master subsystem is an OS/2 device driver that registers a hardware interrupt handler, as in the previous DASD example. This component is needed to receive asynchronous notification from the bus master of graphics operation completion. As longer and more complex drawing

instructions are passed to the bus master video adapter, increased concurrency is achieved between graphics processing and other functions of the system. We should note that the designer of bus master graphics-processor subsystems will have to make important trade-offs of video throughput versus user-input responsiveness.

Graphics operations that do not directly affect the console display are more easily optimized. For example, a single large and/or complex operation can be issued to the graphics processor to prepare an off-screen bit map for printing. Other considerations come into play in this case. One consideration is making sure that such an operation is interruptible and preemptable by a higher priority request to the graphics processor to update the console. Another consideration is balancing the resident memory requirements of a large off-screen bit map against the needs of the rest of the system.

Figure 3 shows a representation of the operating system structures discussed in this graphics processing example.

## Intelligent I/O Subsystem Possibilities

There are many other examples of bus master subsystems that we could have used. The DASD subsystem could be generalized to support a Micro Channel bus master adapter that attaches a small computer system interface (SCSI) bus. The bus master feature is also well-suited for intelligent data communications adapters. Such an adapter could be designed to support multiple protocols. Similarly, it is now quite practical to build a multiple-port asynchronous communications adapter with onboard character and modem control signal protocol processing. Such an asynchronous adapter would significantly reduce the system overhead presently associated with serial port (RS-232C/ RS-422) support.

The important point in these examples is to show how the bus master capability of Micro Channel architecture opens many new possibilities for achieving parallelism between I/O-driven operations and other system functions in the multitasking environment. With these examples as background, let's look now at the OS/2 services-enabling bus master support.

*Graphics operations can be offloaded to the bus master instead of executing in the system processor.*

## OS/2 Device Driver Model

We have already indicated the importance of the OS/2 device driver model. The reason is that hardware interrupts are essential to the efficient asynchronous operation of a bus master. In OS/2 a device driver is the only component of the system that can register to receive control when a device issues a hardware interrupt.

The system does not do a full-process context switch before calling a hardware interrupt handler. Staying within the context of the active process minimizes interrupt latency (the time it takes to get to the correct interrupt handler when a hardware interrupt occurs). This also reduces the overall system interrupt-handling overhead. On the other hand, running in an unknown process context imposes significant restrictions on the interrupt handler. Performing device I/O outside the context of the requesting thread is one of the primary programming considerations addressed within the OS/2 device driver model.

Besides containing interrupt handlers, device drivers are the most privileged user-installable components of the system. The OS/2 architecture treats a device driver as an extension of the operating system kernel. This includes making available many system services that are not available to normal application software. Several of these services are particularly necessary to support a bus master device. From a structural and system service standpoint, a subsystem supporting a bus master in OS/2 must incorporate a device driver component.

## Bus Master Configuration and Initialization

OS/2, as with PC DOS, does not get involved in selecting or initializing the Programmable Option Select (POS) values for a bus master adapter. The system hardware SETUP program and the BIOS Power-On Self-Test (POST) provide these functions. If the adapter supports a base initial program load (IPL) device, it has additional initialization constraints. It should have no dependency on loading a device driver in order to support execution of the SETUP program or the early stages of operating system initialization.

Bus master adapters for these devices should have sufficient logic

and/or some form of non-volatile memory (that is, ROM) to operate in a minimum function mode until a device driver can be loaded. For example, a DASD device should power-up with support for the BIOS real mode INT 13 functions if it is to contain a bootable volume. Similarly, the primary console video adapter should power-up in some default VGA-compatible mode. Once system initialization progresses far enough to load an installable device driver, a bus master adapter subsystem can enable more advanced function modes.

Upon initialization, the device driver for a bus master adapter is responsible for determining any configurable values that are set by the POS mechanisms to initialize the adapter. There are several techniques that can be used to accomplish this. One approach is to find the POS registers for the adapter and then read and interpret the POS register values. A device driver may have little use for some of the POS information, such as the bus arbitration level. Other parameters that must be understood by the device driver include the physical addresses for any memory-mapped buffers, any I/O port addresses, and the hardware interrupt level(s) used by the adapter.

## OS/2 System Services Important to Bus Master Device Drivers

Two volumes of the *IBM Operating System/2 Technical Reference*, titled *I/O Subsystems and Device Support*, contain a complete description of the programming model and services for OS/2 device drivers. Volume 1 addresses the privileged device drivers that effectively extend the operating system kernel. Volume 2 addresses presentation

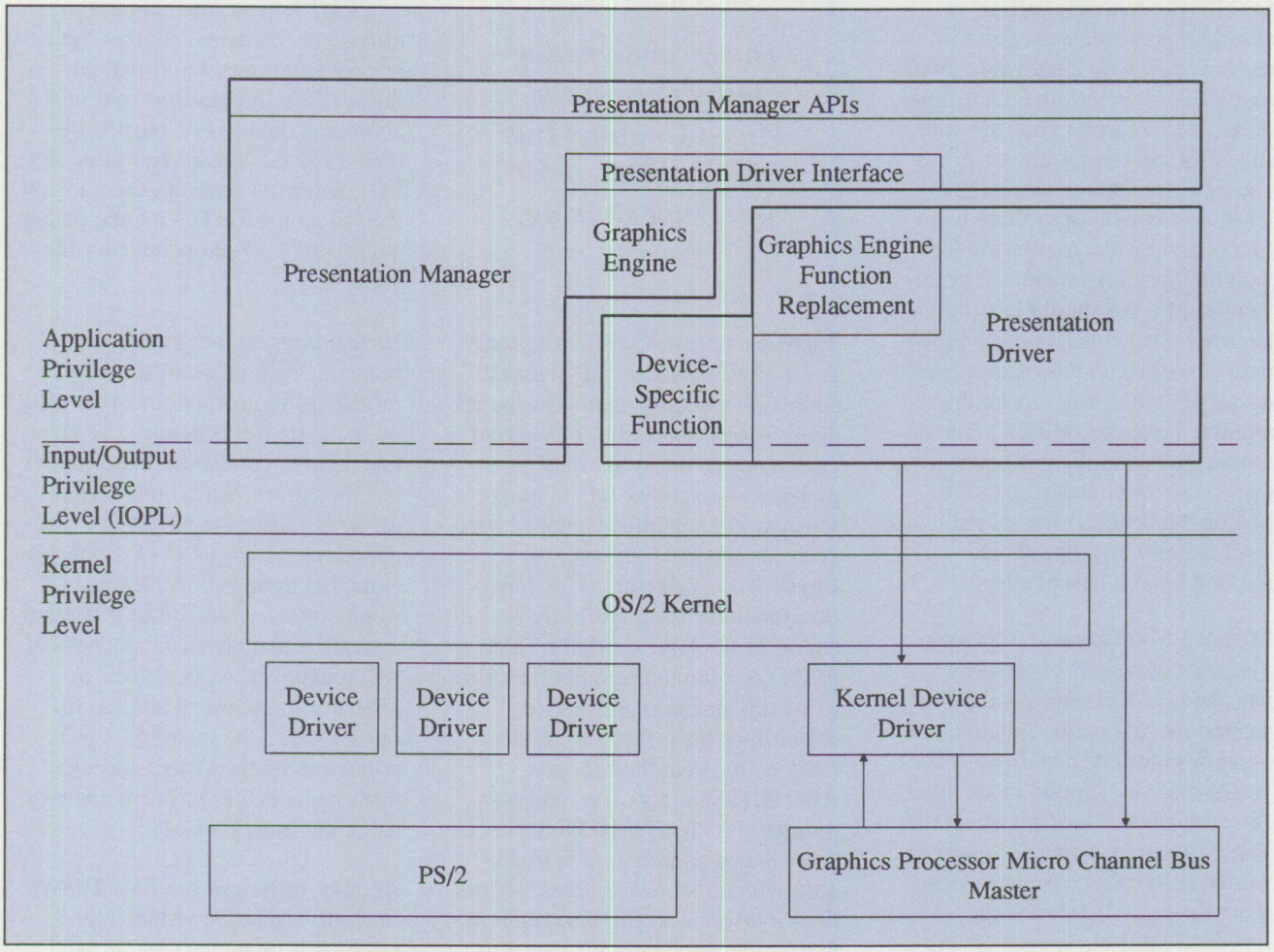



**Figure 3. OS/2 Bus Master Graphics Processor Subsystem**

drivers that support the graphics capabilities of Presentation Manager.

A complete description of all the services available to an OS/2 device driver is beyond the scope of this article. Let's concentrate on pointing out some of the specific services that can be used to support the bus master scenarios we have discussed.

The special system services available to OS/2 device drivers are called Device Helper Services, or DevHelps. These DevHelp services are the functions described in the following paragraphs.

**Registering for an Interrupt Level:** The function **SetIRQ** is used to register an entry point to receive control on a hardware interrupt. The level-sensitive interrupts of the Micro Channel bus are well suited for interrupt sharing. A bus master device driver can specify whether it is willing to share its interrupt level when registering for it with the operating system. The advantage of a bus master sharing its interrupt level is that it could potentially coexist with more interrupting devices in the system, including multiple instances of itself. The disadvantage is that there is a performance overhead associated with polling the devices on a shared level to see which one should respond to an interrupt event.

**Memory Management Services:** There are a number of memory management functions available to support the addressing requirements of a bus master device driver. The virtual memory support of the Intel 286 architecture does not affect the single I/O address space. In other words, all processes in the system share the same I/O port address space. Therefore, no services are necessary to support any type of I/O port address translation. The main memory address space, on the other hand, has a number of considerations. First, the virtual addresses used in software for execution by the system processor are different from the physical addresses used across the Micro Channel bus. Second, these addresses are process-context-sensitive and are not usable outside the context of the I/O thread such as at interrupt time. Finally, the addresses are specific to the real and protected addressing modes of the 286 processor.

*The level-sensitive interrupts of the Micro Channel bus are well suited for interrupt sharing.*

There are different services to translate a physical address to a virtual address, depending upon how the address is to be used. If a bus master device driver wants to establish global addressability (not context-sensitive) to a memory-mapped buffer, it can use the function **PhysToGDTSelector.** This function should be used sparingly because the Global Descriptor Table (GDT) is a limited resource. Before using this function, the device driver must have first allocated a GDT entry using the function **AllocGDTSelector.** The address returned by **PhysToGDTSelector** is a privileged address that is usable only within the device driver. If the device driver wants to pass a bus master buffer address back to some user (application privilege level) software for its use, it would use the **PhysToUVirt** function. The addresses created by this function are within the user privilege domain, and only valid within the context of the process owning the thread that issued the **PhysToUVirt** function call.

The device driver may want to convert a virtual address passed from a user-level program to a physical address. A physical address must be used in a Micro Channel DMA operation (first-party bus master or third-party DMA). The function for this type of address conversion is called **VirtToPhys.** Because there is only one physical address space in a PS/2 system, this function is also used to obtain a context-free normalized form of a virtual address. This "normalized" physical address is then usable outside the context of the thread requesting the I/O, such as at interrupt time, to call the function **PhysToVirt** and obtain a temporary virtual pointer for the address.

Virtual pointers, not physical pointers, must be used for memory references in instructions executed by the system processor. The virtual address created by **PhysToVirt** is valid for use within the device driver because it is allocated at the operating-system level of privilege. Note that there is one necessary step before a virtual address can be converted to a physical address that will remain valid over time. In order for the physical address to remain valid, the memory object pointed to by the virtual address must be locked in physical memory using the function **Lock.**

Another feature of the **PhysToVirt** function is that the virtual address returned is guaranteed to be valid for the current addressing mode

(real mode versus protected mode) of the 286 processor. This bimodal support is a significant part of the device driver I/O support of the OS/2 PC DOS application compatibility environment. OS/2 device drivers can be written so that regions of code can execute whether the processor is in real or protected mode.

On the other hand, a device driver may choose not to execute in real mode. For example, if a bus master device driver finds that its interrupt handler has gained control while the processor is in real mode, it can use the functions **RealToProt** and **ProtToReal** to get to and then return from protected mode. This could be critical if the interrupt handler must address some memory object located above the 1 MB address boundary of real mode.

**Synchronizing Primitives and Process Management:** There are two basic approaches to thread synchronization that a device driver can use. One is to use the explicit **Block** and **Run** functions, which employ an arbitrary and yet global 32-bit event ID value. There is no registration for this value. The convention used to avoid conflicts is to use the physical address of the I/O request packet associated with the blocked thread for the event ID value. A slightly safer synchronizing approach is to use a system semaphore and its associated functions **SemHandle, SemRequest** and **SemClear.** Although there is slightly more overhead associated with their use, system semaphores are guaranteed to be unique. A system semaphore will also be released if the owner of the semaphore terminates.

Interrupt Management
- SetIRQ
- UnSetIRQ

Memory Mangement
- AllocGDTSelector
- PhysToGDTSelector
- PhysToUVirt
- PhysToVirt (UnPhysToVirt)
- VirtToPhys
- Lock (Unlock)

Addressing Mode Management
- RealToProt
- ProtToReal

Synchronization Primitives and Process Management
- Block
- Run
- SemHandle
- SemRequest
- SemClear
- Yield
- TCYield

**Figure 4. Services Important for Bus Masters**

One of the characteristics of the OS/2 device driver model is that a thread executing down in a device driver will not be preempted (except by a hardware interrupt). Therefore, it is the device driver's responsibility not to execute for extended periods without yielding the system processor. It is particularly important for the device driver to yield to any higher priority, time-critical thread that may have transitioned to the ready-to-run state. There is a system data structure (yield flag) that a device driver can check to see whether some other thread is scheduled to run. The functions available to a device driver to release the system processor are **Yield** and **TCYield** (yield only to a time-critical thread).

These are just some of the OS/2 system services available that should be of particular interest to developers of bus master device drivers. Most of the services described have corresponding services for undoing or deregistering the respective service. There are many other services not discussed in this article that will be useful to the bus master subsystem developer. These services include functions for allocating physical system memory for device-driver data structures and buffers, managing I/O request packet queues, and requesting timer services.

The services discussed in this article are summarized in Figure 4.

## Summary

We have discussed how the multithreaded programming model enabled by a multitasking operating system such as OS/2 creates an environment that the bus master capability of the Micro Channel architecture is ideally positioned to

support. We have reviewed some of the considerations for exploiting the structures and services of OS/2 to build a bus master-based, intelligent I/O subsystem. It is clear that Micro Channel architecture is an excellent platform for the future which is, in fact, already here for multitasking subsystem developers. Opportunities for a new generation of performance and function in the PC-compatible market are limited only by the imagination of Micro Channel bus master subsystem builders.

*ABOUT THE AUTHOR*

*Robert L. Williams is a software development manager in the OS/2 Control Program development organization, and was previously involved with the design of OS/2 as a member of the OS/2 Systems Architecture and Design team. He joined IBM's Entry Systems Division in 1983 and has worked on several personal computer office system software projects in areas including telephony, LAN communications, and document retrieval. Bob is coauthor of a book titled* OS/2 Features, Functions and Applications *(John Wiley & Sons, Inc.). He holds bachelor's and master's degrees in computer science and engineering from the University of South Florida.*

# Micro Channel Issues in AIX PS/2

*Robert Mercier, Richard Bass, and Jill Dinse*
*Locus Computing Corporation*
*Inglewood, California*

**This article describes the features of the Micro Channel system as they relate to the implementation of Advanced Interactive Executive (AIX™) on the PS/2 computer. The purpose of any operating system is to bridge the gap between the user and the hardware. While most of the AIX operating system is portable, there are portions that are hardware-specific. Handlers for Micro Channel devices, device drivers, are examples of such machine-specific code. It is these device drivers that communicate with devices and, depending on the device, may take advantage of the facilities of the Micro Channel system.**

**Micro Channel architecture contains several features that allow efficient utilization of the resources of the PS/2 system. We will discuss several of these features as they relate to the implementation of AIX on the PS/2. Topics include the Programmable Option Select, DMA, and interrupt chaining. Authors of device drivers for AIX may find this information particularly useful.**

The advent of Micro Channel architecture has brought about a new level of technology for the personal computer user. The facilities it provides, along with the capabilities of the Intel® 80386™ microprocessor, create a strong platform on which to support advanced operating systems such as AIX. This article describes the Micro Channel-related aspects of AIX on the 80386-based PS/2.

It is the duty of the operating system to maximize the use of the hardware by users of the machine. Recently, it has become generally accepted that the operating system should mask the details of the hardware from the user and provide a general, consistent interface to devices. This philosophy is most strongly embodied in the UNIX® operating system, on which AIX is based. The ease with which this task can be accomplished is a characteristic of the machine architecture. The PS/2 line of computers, using the 80386 microprocessor and the Micro Channel architecture, provides a good foundation on which to implement multiprocessing, multiuser operating systems.

## AIX Device Interface

In this section, we give a general overview of the kernel/device interface in the AIX kernel. Readers familiar with these aspects of AIX or UNIX may continue to the next section.

Figure 1 provides a view of the relationship between the user, the kernel, and the devices. The device interface between applications and the kernel is through a special file system entry. These files are referred to as device special files and are commonly grouped together in the /dev directory. These files possess two attributes, major and minor numbers, that establish their relationship with specific device drivers. The major number ties the device special file to a specific driver and the minor number denotes a subunit of a particular device.

Common I/O operations on device special files such as open, close, read, write, and so on, are routed through a table of indirect function pointers stored in the kernel (devsw in Figure 1). The major number is used as the index to this table. The

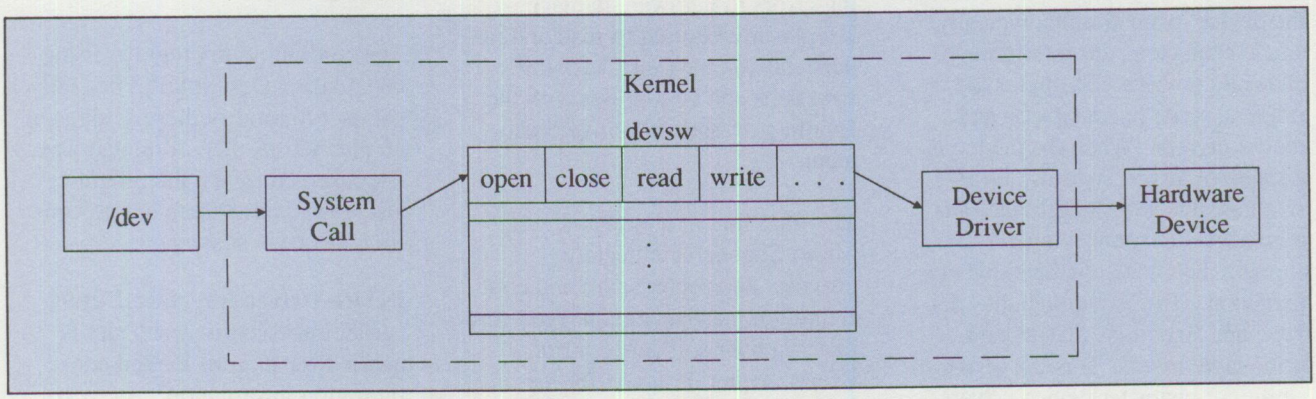



**Figure 1. Relationship Between User, Kernel, and Devices**

```
#define TKRPSID    0xE000   /* PS/2 pos cardid */
/* return the next slot for the token ring */
for (slot = 0; (slot = devexist(TKRPSID, maj, 1, slot)) = 0; slot++) {          /*
initiate adapter setup in slot "slot" */
.
.
.
}
```

**Figure 2. Driver Initialization Routine**

minor number is passed as an argument to the appropriate routine.

When drivers initialize, they typically advertise a routine that they want to have called when an interrupt occurs at a specified interrupt request (IRQ) level. When a user requests I/O to a device, and the operation cannot be completed immediately, the operation is initiated and the process puts itself to sleep, enabling other processes to run. When the operation completes, the device interrupts the system unit, and the device-driver interrupt routine is called, the interrupt routine marks the operation as complete and the kernel wakes up the sleeping process. If the interrupt routine finds that there is more work to do, another operation is initiated and the entire process is repeated.

**Driver Types:**There are essentially two types of drivers. Those which transfer data between the kernel and the device in small units, typically single characters, are referred to as character devices. Examples are printers, serial line adapters, and mouse drivers. When the device is capable of larger, typically fixed-size transfers and can provide random access to some external storage, then it fits the block device paradigm. This group includes diskette and fixed-disk drivers and some tape drivers. It is the responsibility of character device drivers to make it appear to higher levels of the kernel that data received from a device resemble a linear stream of characters. For block device drivers, the data should appear to be a randomly accessible collection of fixed-size blocks numbered consecutively beginning with zero to end of the media.

In the following sections, we will discuss the features specific to the Micro Channel system and their relationship to the implementation of AIX device drivers on the PS/2.

## Programmable Option Select

One of the major frustrations experienced by users of many machine architectures is the installation and setup of device adapters. Previous designs typically required that users set a variety of non-intuitive switches and jumpers on the adapter before installing it in the machine. Results of incorrect switch settings range from mystifying errors to hardware damage. Users were required to read and understand complex installation instructions and be familiar with the handling of static-sensitive equipment.

The PS/2 line of computers, with Micro Channel architecture, provides an adapter setup and configuration mechanism managed completely under software control. Every adapter designed for the Micro Channel system must adhere to this mechanism. PS/2 computers are shipped with a configuration program, and adapters are accompanied by Adapter Description Files describing the resources used by the adapter. This facility is collectively referred to as the Programmable Option Select (POS).

**User Setup:**Each adapter is required to supply a 16-bit card identification (ID) number when queried by the system unit. This number indicates the function of the adapter. For example 0xEEFF has been reserved for serial adapters. In all, 8 bytes are available for each adapter to communicate option settings to and from the system unit. Certain fields are reserved for specific functions such as DMA arbitration level, device ROM segment address, device RAM segment address, and device I/O address. Four of the bytes are completely free-form, and their use is defined by the adapter manufacturer.

The configuration program, using the Adapter Description File, allows the user to modify these settings in an interactive, menu-oriented way. For reserved fields, the program also ensures that there are no conflicts between adapters.

**Device-Driver Interface:**During kernel initialization, each slot is queried for its card ID and configuration information. As each driver initializes, it calls a utility

```
struct devdata {
  unsigned char    pd_pos0; /* Card ID low */
    unsigned char    pd_pos1; /* Card ID high */
    unsigned char    pd_pos2; /* option select data byte 1 */
    unsigned char    pd_pos3; /* option select data byte 2 */
    unsigned char    pd_pos4; /* option select data byte 3 */
    unsigned char    pd_pos5; /* option select data byte 4 */
    unsigned char    pd_pos6; /* subaddress extension LSB */
    unsigned char    pd_pos7; /* subaddress extension MSB */
    unsigned short   pd_major; /* major dev */
    unsigned short   pd_flags; /* flags */
 };
struct devdata devdata[NSLOTS];
```

**Figure 3. Global Array Describing Adapter Slots**

routine to determine whether its corresponding adapter is present. The example in Figure 2 contains an excerpt from the IBM Token-Ring driver initialization routine. The function **devexist** takes four parameters: the adapter identification number, the device major number, flags, and a slot number. The adapter ID is used to locate the adapter by type. If an adapter with this ID is located, the major number and flags are stored in the kernel structure that describes the slot containing the adapter. The slot number indicates the position at which **devexist** should begin searching. This allows **devexist** to locate multiple cards with the same ID.

**devexist** returns an index to a global array that describes the slot containing the adapter (Figure 3).

The driver can use the information in this structure to determine how to communicate with the adapter. For example, the Ethernet™ adapter supports a configurable read-only memory (ROM) address window. The Adapter Description File indicates which bits in the POS information correspond to which ROM addresses, and the piece of code in Figure 4, taken from the driver, decodes the information.

The Adapter Description File must advertise the meaning of these bits so that the configuration program can prevent conflicts.

Additional information about the Programmable Option Select, including the format of the Adapter Description Files, is available in the *IBM Personal System/2 Hardware Interface Technical Reference* manual.

## Shared Interrupts

A significant feature of Micro Channel architecture is the use of a level-sensitive interrupt mechanism. This feature allows multiple adapters to interrupt the system unit at the same level simultaneously. Adapters are required to provide an interrupt-pending latch readable by the system unit so that drivers can determine whether their device is responsible for the interrupt.

```
seg = (unsigned int) devdata[slot].pd_pos4 & 0x0f;
off = 0x4000 * (unsigned int)
((devdata[slot].pd_pos5 > 2) & 03);
rom_addr = (unsigned int) (seg < 16) | off;
```

**Figure 4. Decoding Adapter Description File Information**

```
    intrattach(saintr, 3, SPL_HIGH);
    intrattach(saintr, 4, SPL_HIGH);
```

**Figure 5. Example of intrattach Routine**

```
struct intrshare {
  int (*intr_handler)(); /* The interrupt handler */
      /* Pointer to the next handler */
      /*  for this interrupt level. */
    struct intrshare *intr_next;
   int intr_plevel;       /* Requested spl level */
};
                /* Interrupt vector switch */
struct intrshare *intvecsw[NUM_INTVECS];
```

**Figure 6. Format of Array of List of Drivers**

In AIX, a driver "signs up" for the interrupt levels it is interested in examining. For devices supporting configurable interrupt levels, this information is typically supplied as part of the Programmable Option Select. During device initialization, each driver registers an interrupt service routine by calling the **intrattach** routine. Figure 5 illustrates this.

The arguments indicate the routine to be called, the interrupt level, and the priority-level mask, respectively. This mask is used to selectively disable interrupts when the interrupt handler is called. All drivers for devices of a common type use the same mask value. The example above is from the serial device driver. All such drivers use **SPL_HIGH.** Similarly, block device drivers, such as fixed-disk drivers, use **SPL_BLKIO.** This mechanism helps provide consistent system processor utilization and priority to devices requiring prompt attention.

When the **intrattach** routine is called, the service routine and mask are placed on a list of drivers interested in the specified interrupt level. The kernel maintains an array of such lists, indexed by interrupt-level number. The format of the array is shown in Figure 6.

At kernel initialization time, all hardware-interrupt vectors are pointed at a single global interrupt-handling routine. When an interrupt occurs, the global handler resets the interrupt controller (End Of Interrupt), indexes into **intvecsw** by interrupt-level number, traverses the **intr_next** list, masks out all interrupts below **intr_plevel,** and then calls **intr_handler.**

Each interrupt handler is responsible for determining whether its device caused the interrupt. An interrupt-pending latch is required of each adapter for this purpose. If a driver's interrupt routine is called, and the driver finds that its adapter is not asserting an interrupt, then it simply returns. Figure 7 contains the prologue of a representative device interrupt handler (the 6157 Streaming Tape Adapter).

The **inb** function reads a byte from an I/O address, in this case, the Status register for the tape adapter. This is an active low flag on the tape adapter, so the above example is checking for a 0 bit. If several devices were sharing the same interrupt level and an interrupt occurred, all but one would return immediately when their interrupt handlers were called.

The interrupt-level sharing, combined with the facilities provided by Programmable Option Select, results in an extremely flexible configuration mechanism easily manipulated by users with modest technical expertise.

## DMA Facilities

The Micro Channel DMA controller contains several features that

```
if ((inb(MTFBP1R) & INTTR) || (! mtintrenabled))
return;          /* not for us */
/* handle interrupt */
.
.
.
```

**Figure 7. Prolog of a Device Interrupt Handler**

```
struct dmaralloc {
   /* Driver initialized elements */
   char *dma_devicename;      /* Name of the device (as an ASCII string) */
   int (*dma_availfunc)();    /* Function to call when resource available */
   short dma_priority;        /* Relative priority */
   short dma_arblevel;        /* Requested arbitration level */


   /* dma.c initialized elements */
   short dma_channel;         /* Channel assigned by dmachanalloc() */
   short dma_flags;         /* Flags */
   struct dmaralloc *dma_next;    /* Next pointer of linked list */
};
```

**Figure 8. DMA Resource Allocation Structure**

facilitate easy and efficient support of a wide range of device controllers. These include register compatibility with the AT bus 8237 controller, eight independent channels, two programmable channels, and selectable byte or word transfer modes.

The most interesting feature of the Micro Channel DMA controller is the two programmable DMA channels. These channels can be set to any arbitration level. With a little careful programming, it is possible to support multiple devices at the same arbitration level, a feature normally disallowed by the Micro Channel system. While devices sharing the same arbitration level experience some performance loss, the ability to support them is important.

DMA devices communicate with the system unit DMA controller over one of 15 arbitration levels. For levels 1 through 3 and 5 through 7, the device uses the corresponding DMA channel, because these channels are hardwired to the specified arbitration level. By this we mean that a device set to arbitration level 2 would use DMA channel 2 because it is permanently set to arbitration level 2. Devices set to levels 0 or 4 or levels 8 through 14 all share programmable channels 0 and 4.

While these restrictions may seem to limit the AIX kernel, there is an important positive aspect. DMA channel-allocation routines allow drivers to allocate channels at the arbitration levels of their devices without concern that another adapter might be set to the same arbitration level.

**DMA Channel Allocation:** Each driver that intends to use the DMA controller must provide a DMA resource allocation structure (Figure 8).

The driver initializes the first four fields before calling **dmachanalloc**. The fragment in Figure 9 is taken from the enhanced small device interface (ESDI) disk driver.

The **dma_devicename** field is informational. The **dma_availfunc** is part of a call-back mechanism for managing multiple, simultaneous requests for the same arbitration level. If the driver attempts to allocate a channel for some arbitration level, and the request cannot be satisfied because the resource is busy, the allocation fails. When a suitable channel is later freed, the **dma_availfunc** is called.

```
struct dmaralloc ehddma = {
   "ESDI hard disk",   /* device name */
    ehdstart,    /* dma_avail func (called when chan becomes avail) */
    DMA_AVEPRI,   /* priority */
    0     /* arb level to be filled in by ehdinita() */
};
```

**Figure 9. Fragment from ESDI Disk Driver**

```
/*
 * Try to allocate  a DMA channel. If we can't, dmachanalloc()
 * will set up to call us (ehdstart()) again when the right
 * channel becomes available.
 */
if (( !(ehddma.dma_flags & DMA_HAVECHAN)) &&
   ( !dmachanalloc(&ehddma)))
      return;
```

**Figure 10. Example from ESDI Disk Driver Start Routine**

Many devices manage interrupt-driven I/O through two cooperating functions commonly known as **start** and **intr.** When an I/O operation is requested, it is typically placed on a queue of pending operations. If the driver is idle, the **start** routine is called to remove the first item from the queue and initiate the operation. The device will generate an interrupt when the I/O operation completes and **intr** will be called. Before it returns, it will check the pending queue and call **start** again if there is more work to do.

By carefully coding **start,** it is possible to arrange for the DMA allocation routine to use **start** as the **dma_availfunc** or call-back routine. The example in Figure 10, from the ESDI disk driver start routine, illustrates this.

If the driver does not already have the channel and is unable to allocate it, then return. When the **dmachanalloc** failed, the **dmaralloc** structure was placed on a list of drivers waiting for this channel.

The channel lists are kept sorted by **dma_priority**. All requests requiring programmable channels are placed on the list for channel 0. When a channel is freed, the head of the list for that channel is removed and its **dma_availfunc** is called. Care is taken so that when either of the programmable channels is freed, the list for channel 0 is used. In the example above, if the channel was initially busy, **ehdstart** would be called by the DMA channel free routine when a suitable channel became available.

**DMA Usage:** A function called **dmasetup** is provided to set up a variety of DMA transfers. It can initiate transfers through I/O space or by using arbitration levels, and it allows the selection of 8- or 16-bit transfer sizes. Figure 11 contains an example from the ESDI disk driver:

The arguments are the physical address of the buffer, the function to perform (**B_READ** if read operation, otherwise write is assumed), the length of the transfer, a pointer to the **dmaralloc** structure, the I/O space address to program into the controller, and the transfer size in bytes. The **outb** statement instructs the ESDI controller to initiate the operation.

When the DMA operation completes, the channel should be freed even if there is more work pending. Even though the interrupt routine will call **start** if there is more work, and it seems as if the channel will become available immediately, there are cases in which this will not happen. It is **dmachanfree** that calls the **dma_availfunc** functions so that if another driver is waiting to use this channel, it will be allocated as soon as it is free. Drivers waiting to use channels are serviced round-robin when priority levels are equal.

The DMA channel-management facilities and drivers written to use them help to ensure that nearly every possible combination of arbitration-level requests can be satisfied. This design is in keeping with

```
dmasetup(bp-b_physaddr, bp-b_flags & B_READ, bp-b_bcount,
   &ehddma, EHDDMAIO, sizeof(unsigned short));
outb(BCR, BC_DMAENABLE | BC_INTRENABLE);
```

**Figure 11. Example of Transfers from ESDI Disk Driver**

the philosophy of Micro Channel technology: increased flexibility and function.

## Conclusion

The Micro Channel system provides a rich environment for supporting a wide variety of devices in an efficient and flexible manner. The Programmable Option Select feature combines well with the programmability of the DMA controller and the ability of the interrupt controller to share levels. When adapters are constructed and drivers are written to take advantage of these features, machine setup and configuration become easy enough for even the novice user.

The AIX interface to these resources was designed to facilitate the easy addition of new drivers to the operating system while maintaining efficient utilization of the hardware. In particular, the ability of the DMA allocation routines to support multiple adapters at the same arbitration level allows the use of devices where it might be otherwise impossible.

Additional information about these topics, as well as a comprehensive guide to writing device drivers for AIX can be found in "Writing Device Drivers," *AIX PS2 Technical Reference*, Volume 2, Appendix C.

*ABOUT THE AUTHORS*

*Robert Mercier received his bachelor of science degree in computer science from Indiana State University and has been at Locus Computing Corporation for two years. He was part of the kernel development team for AIX PS/2 release 1, concentrating especially in the Micro Channel aspects of the operating system. He is currently working on an X-windows development project.*

*Richard Bass received his bachelor of science degree in information and computer science from the University of California at Irvine and has been at Locus Computing Corporation for four years. He has worked on various projects at Locus, including operating system software for personal computers. Mr. Bass was a key technical lead in the AIX PS/2 release 1 project, particularly in the Micro Channel area. Currently, he works as a technical lead in the AIX PS/2 release 1 Level 4 support team.*

*Jill Dinse received her bachelor of arts degree in professional writing from Carnegie-Mellon University and has been at Locus Computing Corporation for two years. She has worked on various projects at Locus, including documentation for AIX PS/2 release 1. Currently, she is manager of the Custom Division Technical Publications Department and is working on documentation for AIX PS/2 release 2 and AIX/370 release 1.*

# The Onion

**Jeff Westerinen**
**IBM Corporation**
**Boca Raton, Florida**

## A View of Compatibility

A wise professor told me one day that all computers could be viewed as onions. At first this was difficult to accept – *ONIONS?* – I was accustomed to thinking of computers in terms of bits and bytes . . . silicon and solder . . . diskettes and displays . . . but never *ONIONS!* Well, I've recently been wrestling with a particularly difficult problem. In order to describe the problem I needed a conceptual model that could be used to portray the various layers and interfaces inside a Micro Channel specification-compatible computer. None of the conventional schematic representations of computers seemed to be able to account for all of the interfaces. Every "picture" I would draw of the problem would very quickly evolve into a confused maze of circles, boxes, arrows, and lines. Late one night after staring at my work product for hours, one of the diagrams started to look like – that's right – an *ONION!* Perhaps my professor was right.

The problem I struggled with that night was finding a way to describe the various compatibility points in an IBM PS/2 to my colleagues. Indeed, the description of the various interfaces inside a microcomputer is especially important in a marketplace based on open standards. The challenge in such a marketplace is to properly "architect" these boundaries. If done correctly, the resulting architecture will facilitate the interoperability of hardware and software from different developers. However, pinning down such interfaces improperly can result in poor performance. An additional consideration in an open architecture is the anticipated need of future products. It's important that these interfaces have built-in headroom for greater function and performance.

The Micro Channel architecture interface meets these open standard challenges. A computer with Micro Channel architecture as its foundation can be designed with the assurance of complete compatibility with both hardware and software. Advanced implementations of Micro Channel provide the necessary performance and features to carry a product line well into the future. In this article, I will use the conceptual model of an onion to describe several of the interfaces in a Micro Channel machine which are important when addressing compatibility and open standards.

## Wedges of the Onion

### System Board, Micro Channel, and Cards

If you refer to Figure 1, you will see that the onion can be sliced into three wedges. The bottom wedge represents the system board – and includes the system's processor, memory, support circuitry, and logic. The top wedge represents a Micro Channel adapter card. The middle wedge, which connects the system board and the adapter card, is the Micro Channel Interface. In contrast to the other two wedges, which both contain system elements like chips and software, the Micro Channel wedge is a logical wedge and is specified by the Micro Channel Technical Reference. System compatibility will address all three wedges of the onion.

System and card diagnostics are shown in Figure 1 as a separate small sliver above the system board wedge. Each manufacturer may implement system or card diagnostic interfaces specific to the individual design; compatibility should not be seen to cover these areas. However, diagnostic programs should exercise compatible interfaces properly.

## Layers of the Onion

### Layer 1 – Physical

The innermost layer of the onion is involved with the physical layout of the various system components. It specifies how the system board is mechanically organized. An important compatibility consideration of layer 1 is the design and alignment of card and system connectors.

### Layer 2 – Signal

Layer 2 is composed of the electrical parameters of the system board and adapter cards. It includes the specifications of the drivers and receivers and other components necessary for signal transition. For example, the Micro Channel specification has requirements for 24 mA drivers to connect most signals on cards and system boards to the channel. Much of the system board's electrical design is basically hidden and not a direct consideration in compatibility. The effects of power distribution, capacitance, resistance etc., will, however, have an impact on system timings and can lead to compatibility problems.

### Layer 3 – Hardware Protocol

This critical layer involves the specifications for operation of the hardware registers in the system. Principles of operation of system elements such as bus controllers, timers, clocks, interrupt controllers, disk/diskette controllers, memory controllers, etc., are included in the hardware protocol layer of the system board. The system board I/O address map-

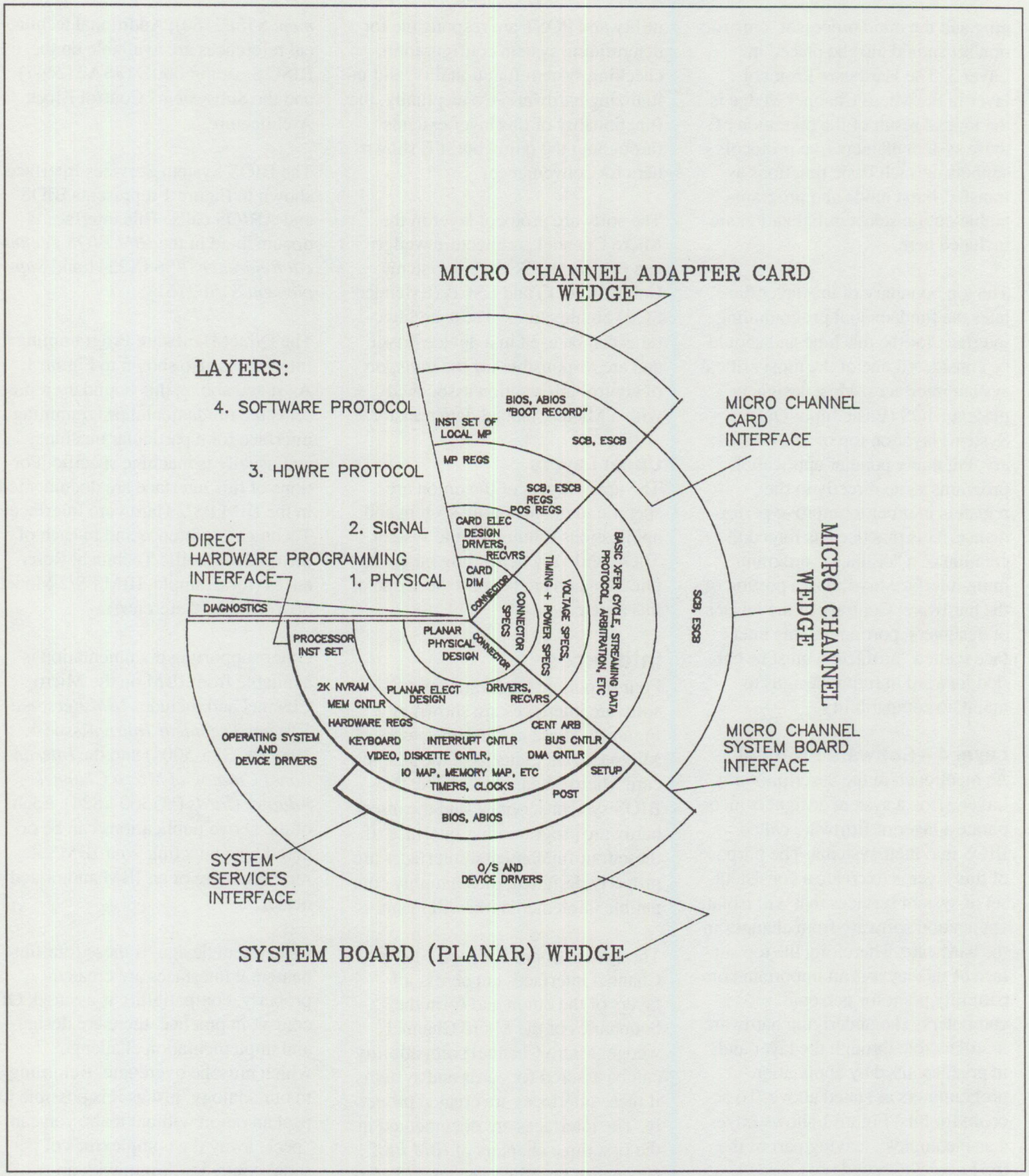



**Figure 1. The Onion – A View of a Compatible System**

"The Onion" reprinted from *The Micro Channel Architecture Handbook*
Brady Books, Simon & Schuster, Inc.

ping and the main processor's instruction set should also be placed in Layer 3. The Hardware Protocol layer in the Micro Channel wedge is the logical result of the operation of these system elements, so protocols supporting such basic functions as transfer, burst mode and programmable option select initialization are included here.

The top boundary of this layer dictates the fundamental programming interface for the machine and should be considered one of the most critical system interfaces. Most designers place a BIOS (Basic Input Output System) layer on top of this boundary, but many popular application programs write directly to the registers in order to improve performance. This practice has forced the creation of a "de-facto" programming interface involving a portion of the hardware. Contrary to the wishes of designers, portions of this interface (called "artifacts") must be carried forward in future designs to maintain compatibility.

### Layer 4 – Software Protocol

As mentioned in the description for Layer 3, most system designers incorporate a layer of firmware called BIOS into their systems. The purpose of this layer is to create a consistent set of system services that will isolate application software from changes in the hardware. Therefore, the top surface of this layer is an important compatibility point for personal computers. The underlying hardware is still visible through the layer and, in practice, used by application programmers as stated above. To account for this, Figure 1 shows Layer 4 as incomplete, leaving part of the top Layer 3 exposed.

Also shown as part of the layer is the system's setup and POST (Power On Self Test) architectures. The setup utility and POST are responsible for determining system configuration, checking system functionality, and initializing hardware. Conceptually, the functionality of this layer extends throughout the onion but it is shown here for convenience.

The software protocol layer in the Micro Channel architecture wedge can contain the SCB (Subsystems Control Block) and ESCB (Extended SCB) architectures. These architectures can be used in a device driver and are responsible for the transport of control information associated with a Micro Channel data transfer.

### Upper Layers

The upper layers of the onion are specific to the operating system and applications running on the system. These will vary depending on the application and are beyond the scope of this article.

## Interfaces

Four important hardware and software interfaces are shown in Figure 1. These are the system board Micro Channel interface, the adapter card Micro Channel interface, the BIOS system services, and the direct hardware programming interface. If these four fundamental interfaces are properly designed, then a highly compatible implementation will result.

The system board and card Micro Channel interfaces cut across all layers of the onion and form the boundaries of the Micro Channel wedge. Micro Channel compatibility can be assured for cards and systems if these interfaces are created properly. The interfaces are documented in the first three chapters of *IBM PS/2 Hardware Interface Technical Reference*, commonly called the Micro Channel Technical Reference (IBM Publication #S68X2330 and *Supplement* S15F2160). Additional technical references are available on the RISC System/6000® (#SA232647) and the Subsystems Control Block Architecture.

The BIOS System Services Interface shown in Figure 1 represents BIOS and ABIOS calls. This interface is documented in the *IBM BIOS Technical Reference* (#S68X2341 and *Supplement* S15F2161).

The Direct Hardware Programming Interface is also shown in Figure 1. As stated above, this boundary represents the fundamental programming interface for a particular machine, and usually is machine specific. Portions of this interface are documented in the IBM PS/2 Hardware Interface Technical Reference and in each of the model specific Technical References (for example, IBM PS/2 Model 80 Technical Reference).

Other supporting documentation is available from IBM on the Micro Channel and includes *IBM Personal Systems Technical Journal*Issue 4, 1989 (#G325-5004) and the *International Catalog of Micro Channel Adapter Cards* (#G360-2824). Each of the above publications can be ordered by contacting your IBM sales representative or an IBM authorized dealer.

In a system design, if these four fundamental interfaces are created properly, compatibility is assured. Of course, in practice, there are design and implementation challenges which must be overcome. Returning to our analogy, just as it is possible to peel an onion without tears, you can "peel" away the complexities of personal computer design with appropriate understanding and documentation.

# Further Reading

The Micro Channel Architecture Handbook
by Chet Heath and Winn L. Rosch
Brady, Division of Simon and Schuster
ISBN 0-13-583493-7

Micro Channel Architecture: Revolution in Personal Computing
by Dr. Patsy Bowlds
Van Nostrand Reinhold
ISBN 0-442-00433-8

The Winn Rosch Hardware Bible
by Winn L. Rosch
Brady, Division of Simon and Schuster
ISBN 0-13-160979-3

Insie the IBM PC and PS/2 – Fourth Edition
by Peter Norton
Brady, Division of Simon and Schuster
ISBN 0-13-465634-2

IBM PS/2 - A Business Perspective – Fifth Edition
by Jim Hoskins
John Wiley & Sons, Inc.
ISBN 0-471-55195-3

Personal System/2 Hardware Interface Technical Reference – Common Interfaces
IBM Form # S84F-9807

Personal System/2 Hardware Interface Technical Reference – System Specific Information
IBM Form # S84F-9807

Personal System/2 Hardware Interface Technical Reference – Architectures
IBM Form # S84F-9808

**Trademarks**

IBM, AIX, Micro Channel, Operating System/2, OS/2, Personal Computer AT, AT, Personal System/2, PS/2, Presentation Manager, RT, Systems Application Architecture, and SAA are registered trademarks of International Business Machines Corporation.

ES/9000, MVS/ESA, MVS/XA, MVS/SP, Personal Computer XT, PC XT, RISC System/6000, System/360, VM/XA, and 3090 are trademarks of International Business Machines Corporation.

Apple is a registered trademark of Apple Computer, Inc.
Banyon and Vines are registered trademarks of Banyon Systems, Inc.
COMDEX is a registered trademark of the Interface Group, Inc.
Compaq and Systempro are registered trademarks of Compaq Computer Corporation.
Dell is a registered trademark of Dell Computer Corporation.
Ethernet is a trademark of Xerox Corporation.
Intel is a registered trademark of Intel Corporation.
Lotus and 1-2-3 are a registered trademarks of Lotus Development Corporation.
Microsoft and Windows are registered trademarks of Microsoft Corporation.
NetWare and Novell are registered trademark of Novell, Inc.
UNIX is a registered trademark of AT&T Bell Laboratories.
i860, 386, 486, 80286, 80386, 80386SX, and 80486 are trademarks of Intel Corporation.


Printed in U.S.A.

*Printed on recycled paper and may be recycled.*

G325-5018-00