R. L. Cormier
R. J. Dugan
R. R. Guyette

System/370 Extended Architecture: The Channel Subsystem

The 370-XA channel subsystem architecture represents a significant extension of the corresponding System/370 architecture. This paper examines the need for these extensions, discusses the important features and facilities of the new architecture, and provides comparisons with its predecessor, the System/370 channel architecture. It also describes, from an operational viewpoint, how these new concepts affect I/O processing and how they relate to the current trend toward using multiple CPUs, increasing CPU execution speed, and increasing the number of I/O attachments.

Introduction

The channel subsystem portion of the System/370 Extended Architecture (370-XA) [1] introduces the most extensive changes in the input/output (I/O) architecture of IBM's large-scale computers since the introduction of System/360 in 1964 [2]. This new I/O architecture provides more capabilities than are provided in System/370 [3] in order to meet the needs of current and future I/O devices and processors. Compatibility with the System/370 I/O architecture has been maintained in two principal areas: (1) *channel-command words* (CCWs) and channel programs and (2) the physical attachment of control units and I/O devices to the system.

The System/370 I/O architecture has evolved from System/360 but maintains a high degree of compatibility with its predecessor [4, 5]. The evolution has occurred at the same time IBM's large-scale computers have evolved into powerful multiprocessor-based systems capable of supporting large, complex configurations of I/O devices, many of which are accessible to the system via multiple paths [6, 7]. The 370-XA channel subsystem architecture has been designed to better meet the needs of these systems and to provide a base for future system evolution. The System/370 I/O structure has been replaced by a new structure that has moved I/O management functions out of CPU programs and by a completely queued interface between the CPU and the channel subsystem. As a result, the new architecture incorporates concepts and uses terminology different from those of the System/370 I/O architecture.

In this paper, we review the more significant features of the 370-XA channel subsystem architecture, including the motivation for these features and their relationships to System/370 structure and terminology. The following sections contain a discussion on the considerations that led to the development of the new architecture, descriptions of the significant facilities included in it, comparisons with System/370, and a description of I/O processing using the 370-XA channel subsystem.

Motivation

The channel subsystem architecture development was initiated early in 1975 at a time when several groups were studying the future technical direction of IBM's large-scale systems. The most serious technical concern was how to achieve higher performance in application program execution. This requirement was being considered from several perspectives, including possible changes to the system structure, circuit technology, and system architecture. With respect to I/O architecture, the problem was viewed as one

that required changing the System/370 I/O architecture to reduce the net number of CPU instructions required for performing I/O functions and to reduce the impact on CPU performance of changes in the design of I/O devices and attachment methods.

• *Objectives of the architecture*

The principal objectives were to increase the performance of the large-scale systems and to structure the I/O architecture to take full advantage of such systems that now consisted of tightly coupled multiprocessors with large main storages and multiple, microcoded microprocessors in lieu of hard-wired logic. An additional and important objective was to provide for better operating efficiency on the I/O interface.

These objectives had to be considered within the constraints of compatibility requirements that had been established relative to System/370 architecture and the Multiple Virtual Storage (MVS) [8] operating system. The first compatibility objective was to support the unmodified attachment of all the I/O control units and devices allowed in MVS/370. This objective was there to protect customer investments in I/O equipment, which is often purchased and replaced on a schedule different from that of the CPU. The second compatibility objective was the support of customer application programs running unmodified. To meet this objective, the architecture requirements for channel programs written for System/370 and for the application program interface (SUPERVISOR CALL) to the operating system could not be changed. With certain minor exceptions, these compatibility objectives have been met in the 370-XA channel subsystem architecture.

• *Development of the architecture*

While the major part of the architecture was developed early in the project, for most of the project's duration the development of the hardware, the software, and the architecture have proceeded in parallel with a high degree of interaction among designers in those three areas. After the basic I/O functions were defined, the channel-subsystem-monitoring facilities, path-management, and dynamic-reconnection facilities were defined, followed by the channel-subsystem-recovery facilities. After these major portions of the architecture were completed, the remaining work consisted mainly in refining the definition to supply the detail necessary for a complete architecture and adding minor functions where needed.

It is important to note that the point of reference for the System/370 I/O architecture discussed in this paper essentially applies to machines released just prior to the IBM 3033 in 1978. At that time, the channel subsystem function was defined, and, with the exception of channel-set switching, the changes made to the System/370 I/O architecture introduced with the IBM 3033 were derived from the channel subsystem definition and retrofitted to System/370. Those functions are the buffering of all status in subchannels, the queuing of I/O requests in subchannels (start-I/O-fast queuing), and the suspend-and-resume facility.

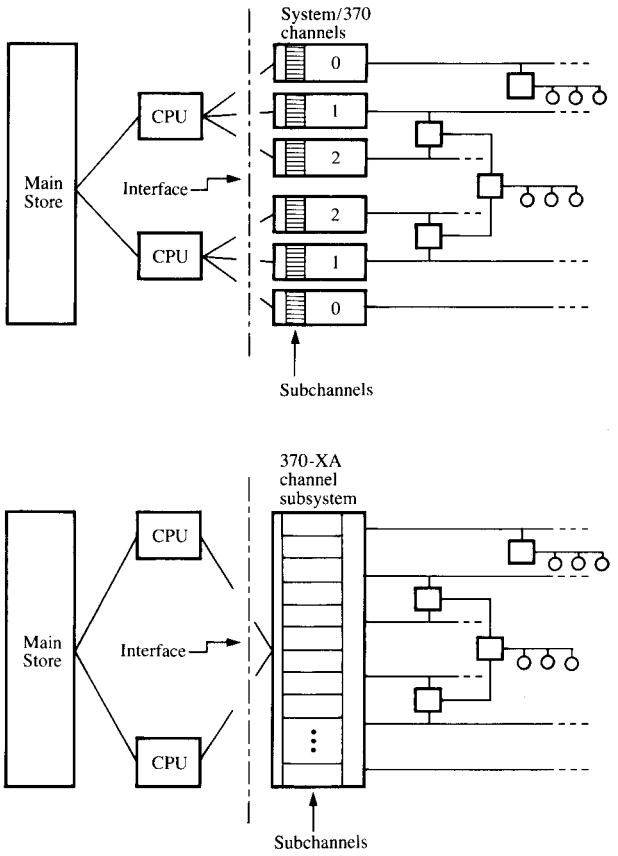

Figure 1 System/370 and 370-XA structures.

It should also be noted that the architecture, in its present form, was influenced by many additional factors. For example, practical considerations of the hardware/software tradeoffs between MVS, the Virtual Machine Facility/370 (VM/370) [9], and the IBM 308X machines influenced the design of the architecture.

Structure of the architecture

The structures of System/370 and 370-XA are shown in Fig. 1. In System/370, the channel-to-CPU interface is uniprocessor-oriented. Channels are provided in sets attached to a CPU. A System/370 channel can be addressed only by the single CPU to which it is connected. Also, a System/370 channel can interrupt only that CPU to which it is connected.

In a System/370 multiprocessor (MP) system, each CPU has its own set of channels [10]. This means that the operating system in an MP system has to ensure that the correct CPU is dispatched to perform I/O operations with I/O devices that are not attached to channels on each CPU. While this causes no problems in System/370, it does require extra operating system overhead and delay in performing I/O operations in some MP systems. The 370-XA channel subsystem is designed so that all CPUs in the system have the same access capability for all devices attached to the system. I/O devices are attached to the channel subsystem, rather than to a CPU as in System/370, and the channel subsystem is accessible by any CPU. Therefore, any CPU can initiate an I/O function with any device and can accept an I/O interruption from any device.

The channel-to-I/O-device interface is single-path-oriented in System/370. When the program attempts to initiate an I/O operation, the path over which the operation is to take place is specified as part of the START I/O or START I/O FAST RELEASE instruction. The I/O operation either is initiated on that path or is not initiated. If the operation is not initiated, the program is notified, and the program may attempt to initiate the operation via a different path to the device, if one is available [11].

The channel-to-I/O-device interface is also single-path-oriented in another way. Once a chain of I/O operations is initiated with a device, all data, status, and commands for the chain of operations must use the physical channel path over which the first command was initiated. In particular, if a device disconnects from the channel path during a chain of commands, as when block multiplexing occurs, the device must reconnect to the same channel path to continue executing that chain of commands. If the channel path to which the device must reconnect is in use (for example, is communicating with a different device) when the device is ready to reconnect, the device must wait until the path is free before it can reconnect to continue execution.

In IBM's large-scale systems, devices are often attached to the system by more than one channel path for reasons of availability. It also often happens that when one of these devices is delayed in reconnecting in order to continue execution of a chain of commands, as just described, another path attaching the device to the system is not busy. The channel subsystem architecture makes it possible to design I/O devices that can reconnect to continue execution of a chain of commands on any path by which the device is attached to the system. This mode of operation of the channel subsystem is called *multipath mode*, and it is used for devices having the dynamic-reconnection feature, such as the IBM 3380 Direct Access Storage Facility Model AA4.

System performance

Improvements in system performance relative to System/370 I/O processing have generally been realized in three ways: by redistributing function between CPU programs and the channel subsystem, by reducing CPU program complexity, and by reducing the CPU program overhead.

The use of high-performance microprocessors to perform I/O functions makes possible the further relocation of I/O functions out of CPU programs into the channel subsystem without significantly increasing hardware cost. When these functions are moved, the same functions are performed, but parallelism is increased, which can lead to an increase in system performance.

In 370-XA, all I/O busy conditions are handled by the channel subsystem rather than by the CPU program, as in System/370. The removal of I/O busy and no-longer-busy handling from the CPU program is especially important, given certain system trends for large-scale systems. The trend for control units is to larger numbers of attached devices and the use of microprocessors. Both factors tend to increase the number of control-unit-busy indications. The trend to increased sharing of devices among systems in multicomputer installations leads to an increase in the number of control-unit-busy and device-busy indications.

Another performance benefit in handling I/O busy and no-longer-busy conditions by the channel subsystem is the removal of the overhead introduced when the CPU processes a no-longer-busy indication and encounters another busy condition. This occurs when the delay between the generation of the no-longer-busy indication by the hardware and the subsequent re-initiation of the I/O request by the program has resulted in some shared resource in the path to the I/O device again becoming busy. This situation results in additional busy/no-longer-busy processing cycles in System/370 that have been removed for 370-XA.

In 370-XA, alternate-path selection is handled by the channel subsystem rather than by the CPU program, as in System/370. This means that the functions of testing for path availability and the management of logical channel queues (MVS) and control-unit queues (VM/370) have been removed from the CPU program. I/O requests are queued in software on a device basis and are only dequeued when the associated subchannel is free and the request can be accepted by the channel subsystem.

System performance can also be improved by reducing I/O program complexity. In System/370, channels must be managed by type; that is, there are differences in program action depending on whether a device is attached to a selector channel or a multiplexer channel. In 370-XA, the type of

channel path protocol used is not apparent to the CPU program. Program complexity is also reduced in 370-XA by simplifying I/O interruption handling and status analysis. A program-specified interruption parameter is passed back to the program in the I/O interruption code in order to simplify program handling of the interruption, and additional information is available in the interruption status to simplify analysis.

A third area of system performance improvement is in the reduction of program overhead in handling I/O interruptions. The number of I/O interruptions that must be processed has been reduced because channel-available, control-unit-end, and device-end no-longer-busy-type I/O interruptions are not generated by the channel subsystem. Instead, these are handled directly by the channel subsystem without program intervention. The channel subsystem also allows the merging of device status in certain cases, thus allowing the status to be processed as a single interruption rather than as two interruptions.

In System/370, the I/O device is interrogated during the execution of some I/O instructions and some I/O interruptions. In 370-XA, the I/O device is not interrogated during the execution of any instruction or during an interruption; all I/O instructions are executed by the CPU asynchronously with respect to the channel subsystem, and all I/O interruption status is buffered in the subchannels, where it is immediately available to the CPU program.

The removal of these delays of the CPU is especially important in view of the trend in large-scale systems to higher and higher CPU execution speeds. Consequently, the I/O request rate and the I/O interruption rate also increase. However, as described previously, the CPU delays introduced because of an interrogation of the I/O device do not decrease proportionately. The reason for this is that the delay includes the time it takes for the I/O device to respond to the interrogation. Unfortunately for CPU execution speed, the trend in device response times is toward longer rather than shorter responses. This is because of the use of microprocessors instead of hard-wired logic in control units. This means that the higher the execution speed of the CPU, the more the I/O instruction and interruption times tend to become a proportionately larger factor in CPU performance.

Terminology comparisons with System/370

The provision of new functions in the channel subsystem architecture has resulted in several changes in terminology. Also, some terms used in System/370 to describe the operational aspects of the architecture are not used in 370-XA because the channel subsystem architecture does not require an awareness of these aspects by the control program. In other cases, terminology changes or deletions were necessary because the logical view of the channel subsystem by the control program differs from that of a System/370 channel.

The term "channel" in System/370 and the terms "channel path" and "channel subsystem" in 370-XA describe different architectural concepts. In System/370, a single physical path usually exists between a channel and the attached control units (the IBM 2870 channel is an exception, having five paths to attached control units). The physical path is referred to (and still is in 370-XA) as the System/360 and System/370 I/O interface [12]. In System/370, communication between the channel and I/O device occurs by using a subchannel, the physical path, and the control unit. Communication between another channel and the I/O device requires a different subchannel, physical path, and control unit (or the same control unit when a two-channel switch is used). The need for separately identifying the physical path is unnecessary; therefore, the term "channel" also implies the physical path.

In 370-XA, the use of any of up to eight different physical paths is permitted when communication takes place between a single subchannel and an I/O device. Frequently it is necessary to refer to a function executed on a specific physical path apart from the channel subsystem. For example, in System/370, a system reset is performed on a physical path as a result of executing CLEAR CHANNEL [3]. The reset function causes all attached I/O devices, subchannels, and the entire channel to be reset. When the same function is performed on a physical path in 370-XA, only those I/O devices attached to that path, as well as conditions in the channel subsystem associated with that path, are reset. The subchannels, other ongoing operations in the channel subsystem, or operations involving other physical paths remain unaffected. Here there is a need for identifying the physical path because the reset function applies only to that physical path and not to the entire channel subsystem.

In System/370, a channel is distinguishable from another channel by three characteristics: a unique address, a separate set of subchannels for attaching up to 256 I/O devices, and a connected CPU. The channel is only available to the CPU to which it is connected. In 370-XA, the channel subsystem is available to any CPU (if more than one CPU exists), is not identified by an address, and can permit attachment of enough I/O devices to correspond with 256 System/370 channels. However, all of the I/O devices attached to the channel subsystem are represented by a single set of subchannels. These characteristics, in addition to the increased functional handling capabilities discussed previously, present the appearance of a collection of channels, or what is termed a "channel subsystem."

Table 1 Corresponding I/O instructions in System/370 and 370-XA.

System/370	*370-XA*
CLEAR CHANNEL	RESET CHANNEL PATH
CLEAR I/O	CLEAR SUBCHANNEL[1]
HALT DEVICE	HALT SUBCHANNEL
HALT I/O	—[2]
RESUME I/O	RESUME SUBCHANNEL
START I/O	—[2]
START I/O FAST RELEASE	START SUBCHANNEL
STORE CHANNEL ID	—[2]
TEST CHANNEL	STORE CHANNEL PATH STATUS
TEST I/O	TEST SUBCHANNEL[3]
—[2]	MODIFY SUBCHANNEL
—[2]	STORE SUBCHANNEL
—[2]	SET CHANNEL MONITOR
—[2]	SET ADDRESS LIMIT
—[2]	TEST PENDING INTERRUPTION
—[2]	STORE CHANNEL REPORT WORD

[1]The I/O device is issued a selective reset in 370-XA.
[2]The function is not provided.
[3]The I/O device is not interrogated as in System/370.

Table 2 Format comparisons of System/370 and 370-XA.

System/370	*370-XA*
Content (length in bytes)	*Content* (length in bytes)
Channel-address word (4)	Operation-request block (12)
Channel-command word (8)	Format-0 channel-command word[1] (8)
—	Format-1 channel-command word (8)
Indirect-data-address word (4)	Indirect-data-address word[1] (4)
Channel-status word (8)	Subchannel-status word (12)
Channel status (1)	Subchannel status[1,2] (1)
Device status (1)	Device status[1] (1)
Limited channel logout (4)	Extended-status word (4)
Full channel logout[3] (–)	Extended-control word[4] (32)

[1]Format in 370-XA unchanged from System/370.
[2]Channel status has been renamed in 370-XA.
[3]Length of logout is model-dependent.
[4]Words 8–15 of the extended-control word are used for logout.

In System/370, the channel architecture refers to *nonshared* and *shared* subchannels. Such references are necessary since the effects of executing certain I/O instructions can vary as a function of the type of subchannel involved. For example, if a subchannel is designated to be shared, and HALT I/O [3] is executed to halt an I/O operation involving I/O device *A*, I/O device *B* can be halted if the subchannel was currently executing an I/O operation with I/O device *B*. With 370-XA, all subchannels appear to the control program as though they were nonshared, and control over the execution of I/O operations is independent of subchannel type; therefore, a discussion of this concept is not applicable in the 370-XA architecture.

System/370 channels are defined in the architecture as either selector, byte-multiplexer, or block-multiplexer channels. Depending upon the channel design or type, some I/O instructions are executed differently. For example, termination of a burst operation with an I/O device by HALT I/O (HIO) on a selector channel causes the channel and subchannel to be immediately placed in the interruption-pending state without their having received status from the I/O device. When HIO is executed on a byte-multiplexer channel, the interruption-pending state of the subchannel is deferred until the I/O device provides ending status. When HIO is executed on a block-multiplexer channel, the function may be executed as for a selector channel or byte multiplexer, depending upon the model. In 370-XA, attachments of devices designed to operate in either selector, byte-multiplexer, or block-multiplexer modes are allowed for reasons of compatibility; however, a single view is presented to the control program since execution of operations directed to any of the attached devices is determined by the I/O instruction and not by the mode of operation being performed on the channel path. Further, an I/O instruction is executed in the same way, independently of the channel subsystem, subchannel, channel path, and I/O device involved.

Tables 1 and 2 provide further comparisons in terms of nomenclature and correspondence of functions provided in System/370 and 370-XA.

Significant facilities

Several key facilities provided by the channel subsystem architecture allow increased functional capabilities to exist in the channel subsystem and, in some cases, provide a reduction in the CPU instruction load supporting I/O processing compared with System/370. In other cases, facilities are provided which ensure that programs written to run under the architectures of System/360 and System/370 can also run compatibly under 370-XA.

- *Path-independent I/O addressing*

In 370-XA, the architecture is structured and defined in such a way that any I/O function can be initiated with any I/O device regardless of the CPU executing the I/O instruction or the physical attachment of the I/O device to the channel subsystem. This is referred to as *path-independent I/O addressing.*

Another significant feature of 370-XA is that any CPU in the system enabled for I/O interruptions can accept an interruption request from any subchannel. Initially, with

System/370, if a CPU failed, pending interruptions for that CPU, the use of its attached channel or channels, and potentially much of the status of attached I/O devices, could be unrecoverable. Improvements that would provide increased CPU availability for System/370 were examined. For example, increasing the size of the channel address and making all channels addressable by all CPUs in the system could have provided the needed improvement. However, this potential extension proved to have too great an impact on the control program; several internal control blocks that used the current address format would have had to be modified.

An alternative to increasing the channel-addressing capability was to switch sets of channels between CPUs when a failure occurred. Channel-set switching did not require a change to the channel-address format and was subsequently introduced into System/370 [3]. When a CPU fails, channel-set switching allows the control program to connect the set of channels of the failing CPU to another CPU. The function is invokable only during unusual circumstances. A more generalized approach of allowing channel sets to be equally accessible to CPUs during normal I/O processing was not provided in this extension because the impact on the control program would have been significant. For example, a new class of interruption would have been required that would cause the control program to connect a channel set to a CPU whenever an I/O interruption condition was recognized. Also, it appeared that a performance penalty would have to be paid if a channel set had to be connected each time before executing an I/O instruction. Finally, additional complexity would have been introduced into the control program in order to handle the connection and disconnection of channel sets whenever I/O instruction execution and I/O interruption handling were required simultaneously.

Because of the improved accessibility in 370-XA and the buffering of status in the subchannel prior to an I/O interruption, provisions had to be made to prevent potential data integrity exposures. Assume that status contained in a subchannel indicated that a media change had occurred at the I/O device. If a procedure for handling I/O interruptions similar to that of System/370 were in use, one CPU could clear the status from the subchannel during the associated I/O interruption while another CPU attempted to initiate a new I/O request with the same I/O device. As a result, the new I/O request could be executed without recognition of the media change at the I/O device.

This condition is avoided by (1) employing an I/O instruction (TEST SUBCHANNEL) that explicitly retrieves status from the subchannel, (2) requiring that I/O interruptions cause the storing *only* of information that identifies the subchannel having available status, and (3) preventing the acceptance of a new I/O request while the subchannel has pending status.

This solution enables the control program to perform interruption handling and status processing using the following conventions: (1) any CPU is allowed to handle the I/O interruption, (2) the subchannel number stored during the interruption identifies the subchannel having status, and (3) TEST SUBCHANNEL is executed only by the CPU whose currently running program maintains ownership (locking) of the control block representing the interrupting subchannel.

In System/370, the situation is handled by requiring the I/O device to present an indication of its status to all attached channels. As each channel receives the indication, either an I/O interruption occurs or the indication is presented if execution of an I/O instruction is currently in process. Such a procedure ensures that the control program receives the indication before any CPU can initiate a new I/O request with the I/O device, but it has the disadvantage of causing multiple interruptions, all describing the occurrence of a single event at the I/O device.

The procedure adopted for handling I/O interruptions and status presentations in 370-XA is different in that, for System/370, the status is stored during the I/O interruption. However, application programs are unaffected by this change and there is no difference in system performance when compared with System/370. Such a comparison reveals that the status stored during an I/O interruption in System/370 needs to be moved from the fixed location in main storage before the CPU is enabled again for I/O interruptions or executes the next I/O instruction; otherwise, that status may be overlaid. In 370-XA, the control program can move the status directly from the subchannel to the area in main storage where status processing is to be performed.

- *I/O unit designations*

Communication between the control program and the channel subsystem regarding an I/O device depends upon the use of a *subchannel number*. Communication between the system operator and the control program regarding an I/O device depends in turn upon the use of a *device number*. The channel subsystem and the I/O device communicate by using a *device address*. The subchannel number identifies the target subchannel during the execution of I/O instructions and during the handling of I/O interruptions. The device number identifies the physical device to the system operator during initiation of the system initial-program-load (IPL) procedure and when the I/O device is enabled and disabled [13]. These parameters are assigned during installation of the I/O device, and they bear no relationship to the physical attachment of that I/O device. Compatibility of addressing

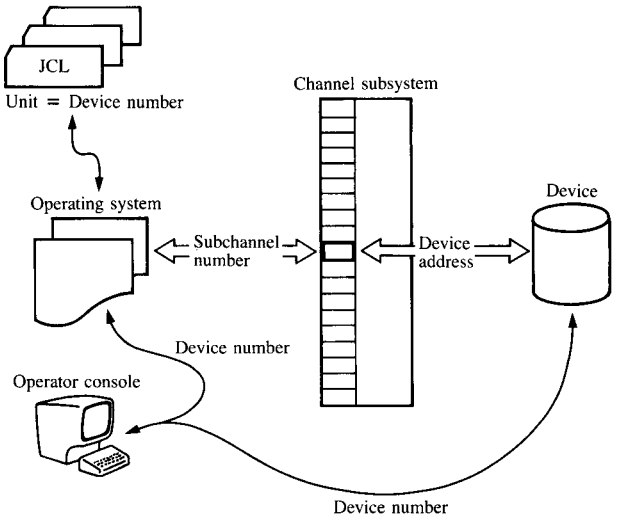

Figure 2 I/O unit designations.

between the channel subsystem and the I/O device has been maintained from System/370 to 370-XA since the device address is still used in 370-XA; however, this address is not visible to the control program. The designation of these parameters and their use are identified in Fig. 2.

Two objectives were achieved by defining a device number: (1) I/O device uniqueness, and (2) a decoupling of physical device addressing from the control program. In System/370, the I/O address is used whenever the control program and the system operator need to identify the physical I/O device. The I/O address is also used when logging information describing a system malfunction involving an I/O device. However, in some cases, actual identification of the I/O device becomes difficult, especially in multicomputer installations where more than one I/O device is identified by using the same address. In other cases, when a maintenance or diagnostic function is initiated subsequent to the reporting of a system malfunction, the I/O address previously logged during the system incident may no longer be assigned to the same physical I/O device.

A unique device number can be assigned, in any arrangement, to each I/O device (up to 64K devices) in the installation. This allows the IBM customer complete freedom in device number assignments. For visual identification of the I/O device, the number assigned is physically affixed to the outside cover of the I/O device (or placed at another equally suitable location) in the same manner previously used.

Architecturally, the device number has no relationship to the device address used in the communication between the channel subsystem and the I/O device or the channel path to which the I/O device is physically attached. Since there is no architectural relationship between the device number and the device address, the second objective previously mentioned is achieved. Consequently, physical addressing changes can be made between the channel subsystem and the attached I/O device, as a result of either configuration changes or technological advances in the attachment of I/O, without impacting the control program.

A question is frequently raised as to the necessity of both a subchannel number and a device number, since the previously discussed objectives were achieved by defining the device number. Initially, only the device number was defined for communications among the control program, the channel subsystem, and the system operator, and the device number was used to identify the target subchannel during the execution of I/O instructions and I/O interruptions. However, as implementation of the architecture progressed, some constraints were placed upon the assignment of device numbers. For example, the channel subsystem implementation required that all device numbers be contiguous. Further, device numbers had to be assigned starting with zero.

These restrictions were necessary in order to reduce the microcode overhead when locating the appropriate subchannel during the execution of I/O instructions. As a result, because some of the flexibility in device number assignment was being reduced, a new parameter was added to the architecture. This parameter, the subchannel number, is used to identify the target subchannel, just as the early-version device number did; however, the implementation restrictions were applied to this parameter instead, thus preserving the flexibility in the assignment of device numbers.

- *Channel path management*

Channel path management is a functional capability whereby the channel subsystem, during initiation of I/O functions, performs tests on the availability of channel paths to the associated I/O device. On the basis of information provided by the control program, the availability testing results in one or more channel paths recognized by the channel subsystem as being available for selection. One of these paths is selected during initiation of an I/O function and, if a busy condition is encountered, an alternate path from the set of paths, if any, is chosen. If yet another busy condition is encountered, another path is selected if one is available. This function is performed without any direct interaction with the control program.

Provision in the architecture of a channel-path-management facility proved to be particularly challenging. Initially, the path-management function entailed only the handling of

control-unit and device-busy conditions that would cause alternate path selection to be attempted. However, additional aspects had to be considered. For example, multidevice control units already attaching to System/370 channels and designed to execute I/O operations with *shared subchannels* could not be handled in a manner similar to that for control units designed to execute with *nonshared subchannels* without compromising data integrity [3]. In particular, only a single I/O operation at a time could be attempted with these control units because the attached I/O devices shared the single subchannel. This restriction was part of the basic design in these control units. If initiation of more than a single operation was attempted by the channel, the control unit could lose control of the ongoing I/O operation. Since, in 370-XA, each I/O device is assigned to a different subchannel, special path-management-handling procedures were defined for these control units to ensure that data integrity was not lost. These procedures took into account the kind of I/O function being initiated, the conditions existing at the control unit, and the type of control unit involved.

It was determined that each control unit could be classified as one of three types, with each type requiring a particular path-management algorithm. The types established were based upon the ability of the control unit to sustain concurrency of execution with multiple I/O devices. If concurrency could not be tolerated but the control unit was unable to preclude initiation of an I/O operation with more than one device at a time, the control unit was classified as Type 1. If the control unit was capable of handling concurrent execution of operations with multiple I/O devices and could preclude initiation of new operations when necessary (by signaling a busy condition to the channel subsystem), the control unit was classified as Type 2. If the control unit had capabilities similar to that of a Type-2 control unit except that, when an error condition was encountered, it was unable to preclude initiation of new operations (characteristic of a Type-1 control unit), the control unit was classified as Type 3.

Also to be taken into account in path-management handling was the consideration that, by means of a command, I/O devices can be reserved for use on a single channel path. That is, if a device is reserved to a channel path, it responds busy when interrogated via other channel paths, with the busy condition persisting until the reservation is cleared at the device. Therefore, a special approach was required for taking this condition into account in path selection. The approaches considered were (1) to require the channel subsystem to decode all commands transferred to devices and to associate reserve commands with channel paths, or (2) to require the channel subsystem to perform alternate-path selection whenever a device-busy condition was encountered during path selection until all paths available for selection had been tried. The latter approach was chosen because it created less operating overhead in the channel subsystem.

- *Programmable interruption subclasses*

Interruption subclasses are assigned to I/O devices attached to the channel subsystem. That is, I/O interruption requests from individual I/O devices can be assigned to any one of eight maskable interruption subclasses. Masking of these subclasses is provided by the use of a control register in each CPU. Subclass assignments are made to each subchannel during the execution of MODIFY SUBCHANNEL. This feature is a refinement of a capability provided in System/370 in which interruption requests are provided on a channel basis. Interruptions from each System/370 channel are controlled by a mask bit in a control register. However, masking interruption requests at the channel level had the disadvantage that *all* of the attached I/O devices (up to 256) were likewise being masked. Alternatively, in 370-XA, assignment of subclasses to subchannels provides greater flexibility in controlling I/O interruptions from I/O devices. For example, a software-controlled priority-interruption methodology can be employed whereby only the lowest-priority programs are executed with all subclasses enabled, and successively higher-priority programs are executed with fewer subclasses enabled. As a result, low-priority programs can be interrupted by all of the I/O devices, but high-priority programs can be interrupted by only a few devices.

- *Address-limit checking*

Address-limit checking is a storage-protection facility that augments the key-controlled protection mechanism provided in System/370 and 370-XA. In a virtual-machine environment, it is common to have one machine operating as a *preferred virtual machine* (PVM). Typically, the PVM is used in scheduling batch-type processing that has resulted from supporting on-line terminal applications. In order to meet performance requirements, the PVM is executed by using an addressing methodology referred to as *virtual-equals-real* (V = R). That is, the PVM guest storage has its pages mapped one-to-one with the host real main storage. As a result of this mapping, a performance advantage is realized by the PVM in that the addresses used in the channel programs need not be translated by the host. However, a channel program written for the guest PVM may, because of a programming error, attempt to cause the channel subsystem to access a storage location outside of the storage range assigned to the PVM. Normally, such an attempted access would not be allowed because the program error would be recognized during the address translation process by the host program. However, since the PVM is being executed without host translation of its channel programs, it is possible that access to the improper storage location would be allowed if the storage key assigned by the host matched the key used by the channel program being executed in the guest PVM. To

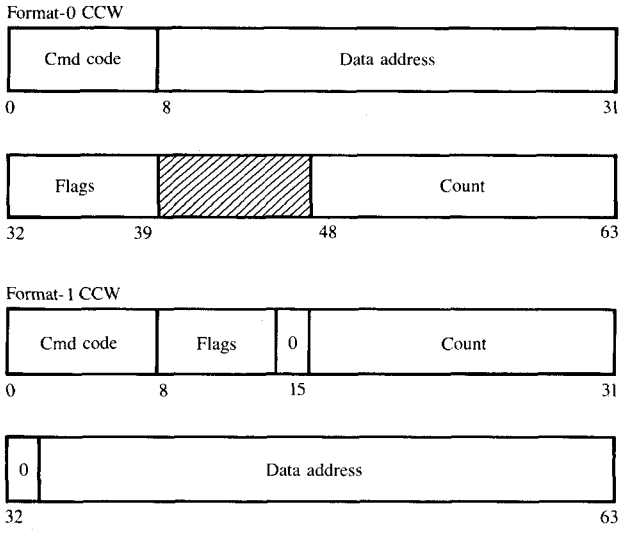

Figure 3 CCW formats. Format-0 CCWs can be located anywhere in the first 16 777 216 bytes of main storage. Format-1 CCWs can be located anywhere in main storage.

avoid the potential integrity exposure, data accesses to storage locations above or below a specified absolute address can be prevented in 370-XA by setting an absolute-address limit value in the channel subsystem. This limit is set by the I/O instruction SET ADDRESS LIMIT. Control-bit settings provided by the control program and placed in the subchannel during the execution of MODIFY SUBCHANNEL specify that data accesses are allowed only at or above, or only below, the limit address.

- *Channel subsystem monitoring*

The monitoring facility of the channel subsystem provides measured parameters in main storage as I/O-resource-usage data. This information is made available to the *resource management facility* (RMF), which assists in managing the performance of the system. The RMF also performed this function in System/370; however, in that case most of the information was obtained through sampling techniques that examined the delays or busy conditions encountered by the control program while attempting to initiate I/O operations. A monitoring facility in 370-XA was needed because of the many changes made to the internal interfaces of the control program and because the I/O queue management and busy-handling functions were moved into the channel subsystem. The monitoring facility provides measured elapsed-time parameters in main storage that describe the extent of I/O resource usage, delay time, and I/O contentions encountered during execution of I/O operations. These data are accumulated on a subchannel basis and are made available as each operation concludes at the respective subchannel.

The instruction SET CHANNEL MONITOR places the channel subsystem in the monitoring mode and identifies the starting location in main storage where the measured data are accumulated. Additionally, control bits provided by the control program and placed in the subchannel during execution of MODIFY SUBCHANNEL selectively enable or disable a subchannel for monitoring.

- *CCW data addressing extension*

A 31-bit data address is provided in a new CCW format that allows direct use of 31-bit addresses in channel programs. In System/370, 31-bit addressing of I/O data can only be accomplished by use of the *indirect data address word* (IDAW), and all CCWs and IDAWs must reside in the first 16M bytes of storage. In 370-XA, two modes of operation are provided: a compatible 24-bit addressing mode for executing old CCWs (Format-0 CCWs) and a 31-bit addressing mode for executing the new format CCW (Format 1). When Format-1 CCWs are specified, the CCWs and IDAWs may reside anywhere in storage. The mode is controlled by a bit that is passed to the channel subsystem during the execution of START SUBCHANNEL. In 370-XA, depending upon the setting of the control bit, direct addressing in either the 24-bit or 31-bit mode is applicable for the entire channel program being executed. Mixed CCW formats within a channel program are not allowed. The two CCW formats are shown in Fig. 3.

- *Dynamic reconnection*

The dynamic reconnection facility allows an I/O device to reconnect to any available channel path in order to continue execution of a chain of commands in a channel program. The instruction MODIFY SUBCHANNEL (MSCH) describes to the channel subsystem, by means of mask bytes, the set of available channel paths for which reconnection is permitted. Use of this facility is controlled by a mode setting (multipath mode) in each subchannel. The MSCH instruction is also used to selectively allow or disallow use of the feature at each subchannel by setting the mode bit (multipath mode). The set of available channel paths for which reconnection is permitted is communicated to the I/O device through command execution by the control program. This capability, together with the channel path management capability, allows the channel subsystem and I/O device to choose the first available channel path for the purpose of initiating or resuming execution of a chain of I/O operations.

I/O processing in 370-XA

- *Channel program structure*

In order to perform an I/O operation, it is necessary to construct a channel program consisting of one or more *channel-command words* (CCWs). These CCWs control operations at the device (for example, seek, read, etc.) and the actions that the channel subsystem performs in executing the I/O operation. The 370-XA channel subsystem architecture was defined to allow execution of the CCWs that were defined for System/370 as well as the new Format-1 CCWs.

Regardless of the format of the CCWs, the same channel and control-unit operations can be executed. While the ability to compatibly execute channel programs is maintained, the portions of the architecture dealing with the initiation of I/O operations and with the I/O interruption mechanism have been changed extensively, requiring new programming at this level.

• *Operation initiation*

In System/370, I/O operations can be initiated by the START I/O instruction, which identifies the channel and the device address on the specified channel. In addition, START I/O causes the channel to fetch the *channel-address word* (CAW) from a fixed location in real storage. The CAW contains the subchannel key and designates the location in storage from which the channel subsequently fetches the first CCW. If the specified channel is busy at the time START I/O is executed, the operation is not initiated and the program is notified. If the specified channel is available, the CPU is delayed while the channel attempts to initiate the operation at the device. The length of time required is determined by the I/O device and, in some cases, may be more than 100 microseconds. While attempting to initiate the operation at the I/O device, the channel may receive a control-unit-busy indication, in which case the operation is not initiated and the program is notified. If other channels in the configuration exist that are connected to the device, the program can repeat the procedure, specifying a different channel in the instruction. In this instance, multiple control-unit-busy indications are possible, with a resulting CPU delay.

The start-I/O-fast function was introduced with System/370 to reduce CPU delay in initiating the operation at the device. This function basically allows the CPU to proceed with execution of the next instruction as soon as the channel availability is determined. Thus, processing continues while the channel, in parallel with the CPU, attempts to initiate the operation at the device. With this approach, on encountering a control-unit-busy condition, the channel notifies the program by means of an I/O interruption so that operation initiation can be attempted on an alternate path. It was soon determined that in some configurations the additional processing required to handle the interruptions reporting control-unit busy more than offset the gain from the start-I/O-fast function. As a result, the function was modified to cause the CPU to wait until the channel could determine if the control unit was busy before allowing the CPU to proceed with the execution of the next instruction. Since the time required to determine a control-unit-busy condition is less than the time to initiate an I/O operation, the start-I/O-fast function remains faster than the original start-I/O function. However, this change to the start-I/O-fast function resulted in the loss of most of the performance improvement originally projected for it.

Subsequently, start-I/O-fast queuing was introduced for System/370. With this function, the channel would return an I/O request to the program only if the desired subchannel was busy executing an operation. If any other busy conditions were encountered, the channel would wait for the busy condition to end and then initiate the operation. While this approach offers performance improvement in some cases, it suffers the deficiency that an I/O request can be queued in one channel because of a busy condition while other channels with paths to the desired device are idle.

In 370-XA, operations are initiated by the START SUBCHANNEL instruction which, unlike START I/O or START I/O FAST RELEASE, does not specify the channel path. Since there is only one subchannel for each device in the system regardless of the number of paths that exist, the program specifies the subchannel corresponding to the desired I/O device; it does this by loading a register with the subchannel number. START SUBCHANNEL also specifies the address of the *operation-request block* (ORB) which contains the address of the first CCW to be executed. Except for the case of a busy subchannel, the I/O request is accepted for subsequent execution regardless of busy conditions existing at the time the instruction is executed. However, unlike start-I/O-fast queuing in System/370, the I/O operation is not queued for a specific channel path. Rather, the channel subsystem selects an available path and attempts to initiate the operation. If a busy condition is encountered, attempts are made by the channel subsystem on other available channel paths. If busy conditions are encountered on all paths, the I/O request remains queued in the subchannel until the operation is initiated on one of the channel paths.

• *Operation execution*

In 370-XA, the functions performed by the channel subsystem while it is executing a channel program (addressing storage, counting data bytes, command and data chaining, etc.) are compatible with those performed in System/370. However, additional functions can be invoked by the program or device to modify certain aspects of channel program execution.

As mentioned earlier, CCWs may be either Format-0 (24-bit data address) or Format-1 (31-bit data address), the latter allowing for expanded storage. Even if the new Format-1 CCWs are specified, the same chains of commands with devices are possible, and no changes are required to devices.

Address-limit checking, if used by the program, can also affect the execution of channel programs. When this is used, the channel subsystem compares the data address being used to access storage with the boundary address previously established by SET ADDRESS LIMIT. If an address-limit

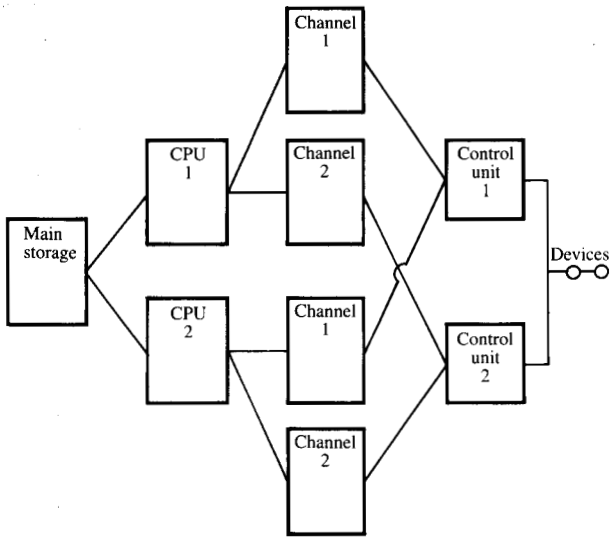

Figure 4 Multiprocessing system configuration.

violation is detected, a program check is indicated even though the CCWs may appear valid according to System/370 rules. Thus, error-analysis programs must be aware of this new checking feature.

Another part of the 370-XA channel subsystem architecture that affects program execution is the dynamic reconnection facility. In System/370, when a channel program is initiated with a device, the path that is used to initiate the operation is the path that must be used to execute the entire channel program. This often causes users to attempt to keep the usage of channel paths very low by attaching few devices to a channel, hoping to ensure availability of the channel path when it is needed by a device for a time-dependent reconnection. Rotational devices are examples of devices that experience a long time delay (one revolution) to be added to the length of time to execute the operation if a channel path is not available when needed. The dynamic reconnection facility allows a device that disconnects from a channel path during an operation to use any path to the system in reconnecting. Thus, if multiple paths exist, a higher activity on each interface is possible while at the same time a high probability exists that an interface to the system is available when required.

• *I/O interruption procedure*

I/O interruptions allow the channel to inform the program of the status of I/O operations, as well as external events at devices. An I/O device will transfer to the subchannel a status indication (channel end) when an operation with the channel is ended, and a different indication (device end) when the operation is completed at the device. In System/370, the first status indication is accepted by the subchannel and then presented to the program by means of an I/O interruption. In this case, the I/O interruption is completed in only a few microseconds since it is not necessary to contact the device. However, the second status indication, if it is presented as a separate sequence, is held pending at the device, and the channel must select the device to retrieve the status as part of the I/O interruption procedure. In some cases these I/O interruptions may take in excess of 100 microseconds because of device delay. In 370-XA, both types of status are accepted by the subchannel, thus reducing the time required for an I/O interruption.

The I/O interruption procedure is initiated in 370-XA when an interruption is pending in a subchannel and the interruption subclass assigned to that subchannel is allowed in any processor in the configuration. The channel subsystem interrupts an enabled processor and stores the interruption code, leaving the subchannel in the status-pending state. The interruption code provides the subchannel number to the program receiving the interruption and allows that program to gain control of the appropriate control blocks prior to clearing the status with the TEST SUBCHANNEL instruction. In a multiprocessor configuration, this process prevents one processor from initiating an operation with a device at the same time that a second processor may be handling status from the device.

Another new aspect of interruption processing in 370-XA is the TEST PENDING INTERRUPTION instruction. In System/370, after one I/O interruption is handled, it is customary to enable the CPU for I/O interruptions again to see if any other interruptions are pending, and if so, the interruption procedure is repeated, with all the programming overhead associated with the required state switching. In 370-XA, this is not necessary since the program can determine if another interruption is pending by means of the TEST PENDING INTERRUPTION instruction, and if there is one, can clear the interruption request and determine which subchannel caused it. It is then possible to clear the status information from the subchannel by issuing the TEST SUBCHANNEL instruction, using the subchannel number provided by TEST PENDING INTERRUPTION. This procedure can be repeated until all pending interruptions are cleared and without the intervening saving of machine state descriptions. Since I/O interruption conditions are made available to all CPUs in a multiprocessing system, the program has the flexibility of allowing all CPUs to handle I/O interruptions or of specifying that a single CPU process all interruptions.

• *Illustration of advantages of 370-XA*

The benefits of the most significant features of the 370-XA channel subsystem architecture are apparent if one considers the I/O activity that can occur in a multiprocessing system

with multiple paths to the I/O devices from each CPU. Figure 4 shows an example of this type of configuration, which has become more commonplace in recent years. The efficiency of the I/O operation in this type of configuration is especially important in the overall system operation.

In the configuration illustrated in Fig. 4, with System/370 it is possible for a program running on CPU 1 to attempt to initiate an operation by using Channel 1. If this attempted initiation encounters a control-unit-busy condition, the program could attempt the request on Channel 2. If this attempt encounters a channel-busy condition, the program in CPU 1 could issue a signal-CPU instruction to interrupt CPU 2 in order to initiate the request on that processor. At this time, CPU 2 could try the request on Channel 1 and encounter a control-unit busy prior to initiating the request on Channel 2. Although the request would now be initiated, additional overhead would be encountered since Channel 2 on CPU 1 would generate a channel-available interruption when the channel-busy condition ended and since Control Unit 1 would generate a control-unit-end interruption for both channel paths. Thus, an attempt to initiate one I/O request in a heavily loaded large system could result in up to

- Four START I/O attempts,
- One SIGNAL PROCESSOR instruction,
- One external interruption,
- Two I/O interruptions with control-unit-end status, and
- One I/O interruption signaling channel available.

The exact sequence of events described may occur infrequently, but it does illustrate the problems that can occur in attempting to initiate an I/O operation. In addition, it is obvious that, as the loading on the system increases, the number of busy conditions and the associated overhead increase.

In 370-XA, an attempt to initiate an operation, given the same busy conditions as in System/370, would result in only the execution of a single START SUBCHANNEL. This is true regardless of the level of activity in the system. Thus, in a heavily loaded system, when efficiency is most important, the 370-XA channel subsystem architecture provides the greatest improvement.

Conclusion

System/370 Extended Architecture has been introduced to provide a better match with the evolutionary trends taking place in IBM's large-scale systems structure. By redistributing I/O functions between CPU programs and the channel subsystem, more parallelism in I/O processing has been introduced, and the I/O processing functions performed by CPU programs have been streamlined.

The channel subsystem architecture provides a new structure on which to base further advances in machine, software, and system-design technologies while maintaining essential compatibility with System/370 architecture for those interfaces intended for application development or for the attachment of current I/O equipment.

Acknowledgments

The authors wish to recognize the significant technical contributions made by M. J. Halma, A. S. Meritt, P. J. Wanish, L. W. Wyman, and C. Zeitler, who helped establish many of the key concepts of this architecture. Their persistence in helping to resolve key technical issues affecting the development of the architecture is also appreciated. The authors also wish to recognize R. E. Wright for his many hours spent reviewing and editing the architecture documentation. Also recognized is J. D. Evangelista for her work in preparing the numerous presentations, changes, and extensions of the architecture.

References and notes

1. *IBM 370-XA Principles of Operation*, Order No. SA22-7085, available through IBM branch offices.
2. G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM J. Res. Develop.* **8,** 87–101 (1964).
3. *IBM System/370 Principles of Operation*, Order No. GA22-7000, available through IBM branch offices.
4. A. Padegs, "System/360 and Beyond," *IBM J. Res. Develop.* **25,** 377–390 (1981).
5. R. P. Case and A. Padegs, "Architecture of the IBM System/370," *Commun. ACM* **21,** 73–96 (1978).
6. M. S. Pittler, D. M. Powers, and D. L. Schnabel, "System Development and Technology Aspects of the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26,** 2–11 (1982).
7. R. N. Gustafson and F. J. Sparacio, "IBM 3081 Processor Unit: Design Considerations and Design Process," *IBM J. Res. Develop.* **26,** 12–21 (1982).
8. M. A. Auslander, D. C. Larkin, and A. L. Scherr, "The Evolution of the MVS Operating System," *IBM J. Res. Develop.* **25,** 471–482 (1981).
9. R. J. Creasy, "The Origin of the VM/370 Time-Sharing System," *IBM J. Res. Develop.* **25,** 483–490 (1981).
10. While this is true in the sense of System/370 architecture, machines have been implemented in the past which may seem to violate this statement. For example, in the *attached processor* (AP) configurations of System/370 Models 158, 168, 3031, and 3033, one of the two processors is not provided with channels. Architecturally, such systems are considered to be MP systems which have no channels configured to one of the processors. An earlier example is found in the System/360 Model 67 operating in the extended PSW mode. In this system, each of the two processors can access all channels provided in the system and can accept I/O interruptions from any of the channels. These facilities are extensions of the System/360 architecture which were not carried forward into the System/370 architecture. They are, however, provided as part of the 370-XA channel subsystem architecture.
11. With start-I/O-fast queuing, channel, control-unit, and device-busy conditions may not cause the program to be notified.
12. *IBM System/360 and System/370 I/O Interface: Channel to Control Unit, Original Equipment Manufacturers' Information*, Order No. GA22-6974, available through IBM branch offices.

13. *OS/VS2 MVS and Stand-Alone Versions: Input/Output Configuration Program User's Guide and Reference*, Order No. GC28-1027, available through IBM branch offices.

Received August 12, 1982; revised November 24, 1982

R. L. Cormier *IBM Information Systems and Technology Group, P.O. Box 390, Poughkeepsie, New York 12602.* Mr. Cormier is a senior engineer and manager of input/output architecture with responsibility for IBM System/370 and 370-XA channel and input/output interface architecture. He joined IBM in Poughkeepsie in 1960 and worked on the design of the 7909 data channel for the 7090 system. He subsequently worked on the design of a plotter control system and the 2870 multiplex channel. He joined architecture in 1968 and became I/O architecture manager in 1978. Mr. Cormier received a B.S. in electrical engineering from Worcester Polytechnic Institute, Massachusetts, in 1960, an M.S. in information sciences from Syracuse University, New York, in 1970, and an M.B.A. from Marist College, Poughkeepsie, New York, in 1982. Mr. Cormier received an IBM Outstanding Invention Award for work on the IBM System/370 I/O interface control in 1972 and a Second Level Invention Achievement Award in 1980.

Robert J. Dugan *IBM Information Systems and Technology Group, P.O. Box 390, Poughkeepsie, New York 12602.* Mr. Dugan is a senior engineer in the Central Systems Architecture Department. He received his B.S. and M.S. in electrical engineering from Auburn University, Alabama, in 1968 and 1969. In 1969, he joined IBM at the Poughkeepsie laboratory, where he has worked on various assignments in the architecture department involving the development and extensions to the IBM System/370 I/O architecture. In 1975, Mr. Dugan started the development of the 370-XA channel subsystem architecture. The development of this architecture continues to be his primary responsibility. Mr. Dugan has received a Second Level IBM Invention Achievement Award and is a member of Eta Kappa Nu.

Richard R. Guyette *IBM Information Systems and Technology Group, P.O. Box 390, Poughkeepsie, New York 12602.* Mr. Guyette has been a member of the Central Systems Architecture Department at the Poughkeepsie development laboratory since 1974. His primary responsibility since early 1975 has been the development of the 370-XA channel subsystem architecture. Mr. Guyette was also responsible for developing the architecture for the suspend-and-resume and start-I/O-fast-queuing facilities which are part of the 3033 extensions feature on the IBM 3033 and 308X processors. He joined IBM in 1965 as a systems engineer in the Rochester, New York, branch office. In 1968 he became a regional data acquisition and control systems (DACS) engineer on the eastern region staff in New York City. In 1969 he joined a programming architecture group in the Endicott, New York, development laboratory to work on architecture and programming extensions for real-time processing. It was as a result of this work that he ultimately joined the Central Systems Architecture Department in Poughkeepsie in 1974. Mr. Guyette received a B.A. from the University of Vermont in 1961 and an M.S. from Cornell University, Ithaca, New York, in 1965, both in physics. He received an IBM Eastern Regional Manager's Award in 1968, a First Level IBM Invention Achievement Award in 1980, a Division Award in 1981 (for his work on the architecture for the 3033 extension feature), and a Second Level IBM Invention Achievement Award in 1982.