# The IBM Enterprise Systems Connection (ESCON) Architecture

by J. C. Elliott
M. W. Sachs

**The IBM Enterprise Systems Connection (ESCON™) Architecture™ is the architecture for the new fiber optic serial-I/O channels for the processors in the IBM System/390® family. The architecture is based on message exchanges, which replace the byte-oriented protocols of the predecessor parallel interface architecture. Its interconnection topology employs a dynamic crosspoint switch. This paper describes the major functional components of the architecture and discusses some of the technical problems that were solved during its development.**

## Introduction

The use of fiber optics as the interconnection medium between processors and I/O devices provides a number of benefits. Chief among these is the ability to provide both substantially higher data rates and longer transmission distances compared with the parallel "copper" buses traditionally used for I/O interconnection. The ultimate bandwidth limits of fiber optics are dictated by the frequency of the optical signal that carries the data. The typical wavelength of the infrared radiation used for data transmission is approximately 1 micrometer ($10^{-6}$ meter), corresponding to a carrier frequency of $3 \times 10^{14}$ Hertz. While achieving true optical bandwidths is not yet practicable, serial fiber optic communications systems in use today provide data rates from 45 megabits per second (Mb/s) to several gigabits per second, at distances of several kilometers to several tens of kilometers. In contrast, the IBM System/390® parallel electrical channels [1] are capable of 36 Mb/s, i.e., 4.5 megabytes per second (MB/s), at a maximum distance of 122 meters. [Special-purpose parallel electrical channels, such as the American National Standards Institute X3T9.3 High-Performance Parallel Interface [2] standard, are capable of as much as 800 Mb/s (100 MB/s) but at shorter distances.]

Fiber optic transmission systems have very high noise immunity and low error rates. Error rates of less than one error in $10^{12}$ bits are achievable [3]. This contributes to the ability to transmit over long distances and also permits use of fiber optics in electrically noisy environments, such as the typical manufacturing facility, where long-distance electrical transmission requires special precautions.

Because the very high bandwidths of fiber optics can be achieved with serial transmission on one fiber in each direction, cable bulk is reduced and connector reliability is enhanced, the latter because a bidirectional fiber optic interconnection, or *link*, requires only two fibers, compared to (in the case of the System/390 parallel interface) 48 coaxial cables, with 96 connector contacts at each end of the transmission link. Both cable bulk and connector reliability are significant concerns in a computer system with large numbers of channels and I/O devices.

An I/O interface architecture is a set of rules that govern how information specified by the I/O instructions of the processor is communicated on the transmission medium and how the channel and I/O device cooperate to exchange this information. Some aspects of the architecture are determined by the nature of the transmission medium and its interconnection topology. Thus, it was necessary to design a new I/O interface architecture, in place of the parallel interface architecture, that describes how information is transferred on the serial fiber optic transmission medium. The resulting architecture and related set of IBM products is called Enterprise Systems Connection (ESCON™), and the architecture itself is called the ESCON architecture [4].

Fiber optics was chosen as the ESCON transmission medium in order to meet the requirements for increased bandwidth and distances compared to the predecessor parallel channels. The large increases in processor speed in recent decades have led to large increases in aggregate system I/O bandwidth and in the data transfer rates required of individual I/O devices. Increased distances are needed to permit the high-speed interconnection of multiple computer systems within a site (having dimensions of the order of a few kilometers), to enable printers and terminal controllers to be placed near their users, and to enable critical data-storage devices to be placed in secure locations. The current ESCON optical interconnection provides 200Mb/s duplex point-to-point links using long-wavelength (1300 nm) light-emitting diode emitters and multimode fiber. The maximum transmission distance of a single link is 2 or 3 km, depending on the cable being used. An optional feature called the ESCON Extended Distance Feature provides laser emitters and single-mode fiber, with a maximum transmission distance of 20 km [5]. The fiber optic technology is the subject of a separate paper [3] in this issue.

Another goal for ESCON I/O was a high degree of connectivity between processor channels and I/O devices. In recent years, there has been an increase in the degree of sharing of I/O devices by multiple systems and by multiple channels on the same system. After exploration of a number of interconnection topology alternatives [6], a dynamic crosspoint switch was selected as the basic topology.

The current IBM product that implements the switching topology is called the ESCON Director™ and is described elsewhere in this issue [7]. The ESCON Director can provide 30 simultaneous connections, each capable of transferring data at the 200Mb/s rate of the attached links. It rapidly makes dynamic connections based on addressing information in the transmitted character streams. In addition, static (*dedicated*) connections can be created, and permissible dynamic connections can be specified, by use of an operator console or by host system software [8]. This replaces the static switches that are used for similar configuration-management functions with the parallel channels.

An early decision was made that fiber optics should be introduced into System/390 by changing only the I/O-interface architecture and not changing the I/O architecture of the processor, the system stucture, or existing I/O application software. This enabled most of the performance benefits of fiber optics to be obtained while the cost of the change and the development time were limited. This goal was largely met, although some software changes were eventually made, primarily in support of new functions, including management of the ESCON Director and system-wide link-error reporting and analysis. (While the constraint of compatibility with existing system architecture and software posed many challenges, it also provided benefits by limiting the number of design options.)

The simplest method of introducing fiber optics while preserving system compatibility would have been to preserve as much as possible of the architecture of the parallel interface and simply replace the physical transmission medium. In this approach, the existing parallel-interface protocol is preserved, but the information that is normally placed on the parallel-interface lines is converted to a serial format for transmission on the fiber optic link. There are several ways of doing this; however, none of these methods use the fiber optic bandwidth very efficiently or provide an opportunity to introduce new function. In addition, the parallel-interface protocol does not provide the function needed for establishing dynamic connections through the switch. Therefore, it was decided to design a completely new, message-based interface architecture that directly maps the semantics of the I/O architecture of the processor onto messages on the fiber optic links and provides opportunities for future introduction of new function. The architecture makes efficient use of the link bandwidth by maximizing the number of data bytes transferred per byte of control information and by minimizing the number of "handshakes" (message exchanges) between channel and I/O device. The latter is important, because each handshake results in additional messages to be processed as well as a distance-dependent delay equal to the time for the optical signal to travel from one end of the link to the other and return.

In this paper, we make extensive use of the terminology and concepts of I/O programming as defined in the IBM Enterprise Systems Architecture/390™ (ESA/390™), the I/O architecture of the IBM Enterprise System/9000™ (ES/9000™) processors. An overview of the ESA/390 I/O architecture is provided in the Appendix.

## ESCON structure

• *Architectural levels*

In ESCON architecture, the I/O information flow is naturally divided into basic information-transfer semantics and channel-program semantics. This led to dividing the architecture into two levels of function. The levels of ESCON architecture, called the link level and the device level, are shown in **Figure 1**. The link level provides the actual information transfer across the physical path in the form of transmission frames (described below) containing data and control information. Included in the link level are the functions and protocols necessary for initializing the link, maintaining the link in an operational state, and providing primitive recovery and diagnostic actions. The device level includes the rules for execution of a channel program using the facilities of the link level. It defines the data messages, control messages, and protocol that carry out the intent of the channel program.

• *Architecturally defined entities*

The physical information transfer facilities defined by ESCON architecture consist of ESCON interfaces and links. An ESCON *interface* consists of an optical emitter and receiver and the associated electronics as well as the connector for the fiber optic cable. The term *link* refers to a single fiber optic point-to-point connection between two ESCON interfaces.

The architecture is embodied in channels, control units and their attached I/O devices, and ESCON Directors. A *channel* directs the transfer of information between I/O devices and main storage and provides the common controls for the attachment of different types of I/O devices. Each channel has one ESCON interface. An ES/9000 processor may be divided into several logical partitions, each of which functions as a separate processor. A single ESCON channel may be shared by multiple partitions. Architecturally, each of the sharing partitions has a separate channel "image," which represents the shared channel.

A *control unit* provides the logical capability necessary to operate and control one or more *I/O devices* and adapts the characteristics of each I/O device to the requirements of the channel. A control unit has one or more ESCON interfaces. A single control unit may consist of multiple logical entities called *control-unit images*. Architecturally, each control-unit image is treated as an independent

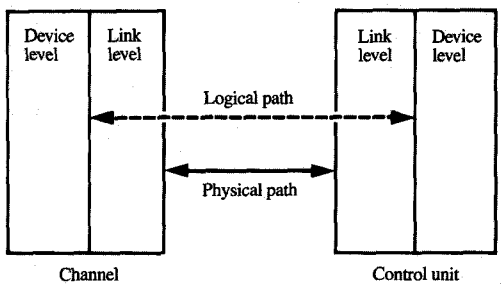



**Figure 1**

Connection between channel and control unit. The relationships among link level, device level, physical path, and logical path are shown.

control unit, with its own complement of I/O devices. Each ESCON interface on a control unit provides communication with multiple images. The terms *control unit* and *control-unit image* are used interchangeably in the remainder of this paper.

The ESCON Director provides the capability to interconnect any two links that are attached to it. The link attachment point on the ESCON Director is called a dynamic-switch port, or simply port. A *port* consists of an ESCON interface and the electronics that implements the port function defined by the architecture.

• *Topology*

The architecture supports two topologies, point-to-point and switched point-to-point. The link-level and device-level functions and protocols are identical for both topologies. The point-to-point topology, illustrated in **Figure 2(a)**, consists of a single link between the ESCON interface of a channel and an ESCON interface of a control unit. The switched point-to-point topology, illustrated in **Figure 2(b)**, consists of a number of channels and control units with their ESCON interfaces each connected by a point-to-point link to a port on an ESCON Director. The ESCON Director permits any channel to communicate with any device attached to any control unit; however, the system configuration definition, which is beyond the scope of this paper, generally restricts which channels can communicate with which devices in any particular installation.

The basic function of the ESCON Director is to create a connection between two ports, thereby enabling the channel and control unit attached to those ports to communicate with each other. All ports can be simultaneously participating in connections, and each

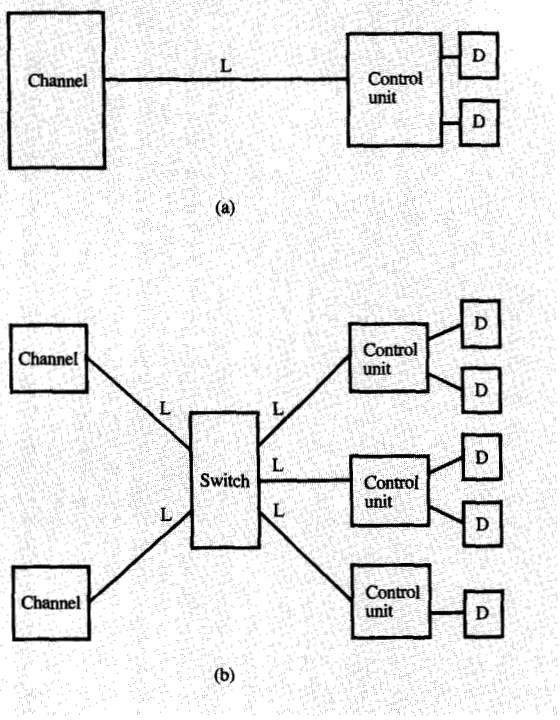



**Figure 2**

Interconnection topologies: (a) Point-to-point. (b) Switched point-to-point. D = I/O device; L = link.

connection can be transferring data at the maximum data rate of the individual links. Because each channel or control unit is connected directly to the ESCON Director by means of a separate link, the ESCON Director provides a degree of isolation among the channels and control units, so that failures or maintenance operations on one do not affect the others. Although a failure of an ESCON Director affects all attached channels and control units, it has optional fault-tolerance capabilities. In addition, redundant paths between processors and devices can be provided through separate ESCON Directors. This protects the system against individual link failures as well as against ESCON Director failures.

- *Information format*

Information is transferred in a synchronous bit stream. The bit stream is divided into transmission characters, which represent data bytes or perform control functions. There are two information structures: frames and sequences. The frame is the primary unit of information transfer. It consists of a group of transmission characters organized according to a defined format that includes address information, data, control information, and frame delimiters. The frame also includes a cyclic redundancy check (CRC) field, which assists in detection of transmission errors in the frame (error correction is performed using retransmission). A sequence is a special stream of transmission characters used for certain primitive signaling functions which, because of unusual conditions, cannot be performed reliably using frames.

- *Addressing*

The ESCON Director makes it possible for one channel to communicate with multiple control units and for one control unit to communicate with multiple channels. (A channel can communicate with only one control unit at a time; similarly, each control unit ESCON interface can communicate with only one channel at a time.) Each channel may have multiple channel images. Each control unit may contain multiple control-unit images, and each control-unit image may control multiple I/O devices.

Addressing information in each frame causes the frame to be directed to a single destination out of many possible destinations. There are three components to the address:

- Link address—selects a particular ESCON interface, to which the ESCON Director then makes a connection. (The link-address information in each frame also contains a source address. The recipient of a frame uses the source address in the received frame as the destination address in a response frame.)
- Logical address—selects a particular channel image or control-unit image of those sharing the selected ESCON interface.
- Device address—selects a particular I/O device of those controlled by the selected control-unit image.

Between the link address, logical address, and device address, 28 bits of information are specified by the ESCON architecture for addressing the I/O devices. This far exceeds the total device-addressing capacity of the ESA/390 architecture. However, segmenting the addressing information into link, logical, and device addresses simplifies the process of defining the I/O configuration of a system. Configuration definition is simplified because the device addresses and logical addresses on a particular control unit can be assigned at the time of manufacture. This is possible because the unique link address assigned to each control unit ESCON interface attached to an ESCON Director guarantees that the total address of each device will be unique. Preassigning the logical and device addresses eliminates the need for them to be manually set into the control units during installation of a system. In addition, the ESCON architecture provides a mechanism for automatically assigning a unique link address to each ESCON interface, as is described below. In contrast, on

the parallel interface, only eight bits of addressing are provided. The 8-bit address of each device must be manually set into the control unit when the device is installed.

Because in the ESA/390 architecture a program refers to a device by a subchannel number, software need not be concerned with the difference between the addressing structures on the parallel interface and ESCON architecture. The effect of the difference is confined to the channel itself. On the parallel interface, the channel uses the 8-bit device address to identify the subchannel for an operation. In the ESCON architecture, the channel must use the combination of link address, logical address, and device address to identify the subchannel.

• *Logical path*

On the parallel interface, each channel interface on a control unit is associated with and connected to one and only one channel. In the ESA/390 architecture, this association is called a "channel path." Because of the switching topology, an ESCON interface on a control unit may be shared by more than one channel, each of which may request dynamic connections to that interface. To make ESCON architecture compatible with the channel path as defined in the ESA/390 architecture, it was necessary to define an association, equivalent to a channel path, between a channel image, the ESCON interface on the channel, a single ESCON interface on a control unit, and a single control-unit image. This construct is called a logical channel path or, for short, a logical path.

With the parallel interface, each channel path is established at the moment the control unit is physically connected to the channel. With the ESCON architecture, each logical path must be established by the channel with an exchange of messages in which the channel informs the control unit of the channel link and logical addresses. The link and the logical addresses become the unique identifier of a channel path from the viewpoint of the control unit.

Until a channel establishes a logical path to a control unit, the control unit cannot perform device-level operations with the channel. Only link-level functions can be performed.

## Link level of the architecture

The link level contains two distinct kinds of functions, which we term low-level signaling and logical functions. Low-level signaling includes the definition of the transmission code (described below) and certain primitive control functions. The primitive control functions (described below) are performed using defined sequences of transmission characters. The logical functions are performed with exchanges of frames. They include various link-level control functions and transport of data and control information for the device level.

• *Low-level signaling*

*Transmission code*

The transmission code [9] maps each 8-bit data byte into a 10-bit transmission character for transmission on the link. It also defines a number of control characters (also having ten bits) that are outside the data alphabet. One of the control characters is the idle character, which is transmitted continuously between frames. Control characters are also the basis of ordered sets, which are used for various control functions. An ordered set is a defined combination of control characters or of control and data characters. The ordered sets are used for frame delimiters and sequences. An ordered set used as a frame delimiter consists of two or three control characters. An ordered set used as part of a sequence consists of the idle character and a different data character for each type of sequence.

*Frame delimiters*

Frame delimiters identify the beginning and end of each frame and are also used for control functions, primarily related to the ESCON Director, in which the control operation must be identified quickly, independently of the contents of the frame, by hardware. The delimiter-ordered sets were designed to provide maximum immunity against link errors. This is important because the delimiters are not included in the CRC for the frame. No single-bit error can cause a false delimiter to be detected within the data stream. Also, no single-bit error can convert one delimiter into another, which is important in ensuring that the state of the connection through the ESCON Director is immune to alteration by link errors. A single-bit error can cause a delimiter not to be recognized, in which case higher-level functions are invoked for error recovery.

The following delimiters are defined: (a) start of frame (SOF), either connect or passive; and (b) end of frame (EOF): disconnect, passive, or abort. The connect, disconnect, and passive delimiters control the ESCON Director switching operations, as described below. The abort EOF delimiter indicates that the destination is to ignore the frame. It is used to terminate a frame when an internal condition at the sender of the frame prevents complete transmission of the frame.

*Delimiter-controlled circuit switching*

The ESCON Director is a circuit switch. In order to communicate with another unit, a channel or control unit requests that a connection be created to the required destination. The request for a connection consists of the connect SOF delimiter followed by the link address in the first information frame sent to the other end. The two units communicate, using the connection that was created, by exchanging one or more frames. When the communication

is ended, one of the two units explicitly requests that the connection be removed.

Minimizing the delay associated with the protocols used for creating and removing dynamic connections was considered key to providing efficient channel utilization over a wide range of uses. If the request for a connection had been defined as part of the information content of the frame, it would have been necessary for the switch to store the whole frame and check its CRC before determining whether a connection was being requested. To avoid the resulting delays, it was decided to use unique frame delimiters to control the connection process. The connect SOF delimiter alerts the ESCON Director that a connection is to be made and that the Director can make the connection as soon as the destination link address has arrived, while the rest of the frame is still arriving at the switch. Although this process uses the destination link address without checking the CRC of the frame, the destination ESCON interface checks both the CRC and the destination address, thus detecting any routing errors that might result.

The other frame delimiters are used as follows: The passive SOF delimiter indicates to the ESCON Director that this frame is to be sent through the existing connection. The disconnect EOF delimiter alerts the ESCON Director control facility to break the existing connection after the frame is sent through it to the destination. The passive EOF delimiter indicates that the frame is to be sent to its destination without breaking the connection.

*Sequences*

Under certain conditions, the use of frames for communication is either unreliable or inappropriate. (For example, if the error rate on a link is much higher than expected, frames are likely to fail the CRC test.) Instead, sequences are used for communication.

Each sequence consists of the continuous repetition of a particular ordered set until some event, defined for the particular sequence, occurs. Events that terminate sequence transmission are the receipt of a sequence in response, expiration of a defined time period, and several others. Continuous repetition ensures that the sequence will be correctly recognized by the receiver—even in the presence of a high link-error rate.

The following sequences are defined:

- Not operational (NOS)—The sender is not receiving a signal or cannot synchronize with the signal it is receiving.
- Off-line (OLS)—The sender is off-line.
- Unconditional disconnect (UD)—The sender does not know whether it is connected to another ESCON interface through the switch and is attempting to ensure that there is no connection.
- Unconditional disconnect response (UDR)—response to UD.

*Transmission errors*

Transmission errors are caused by transient noise or malfunctions in the channel, control unit, or ESCON Director, including a failed or failing link. The architecture requires the receiver to detect the following types of transmission errors:

- Link-signal error: The amplitude or power of the received signal is below the value required for reliable communication, or the receiver has determined that it has lost synchronization with the incoming character stream.
- Code-violation error: The receiver has detected an invalid transmission character.
- CRC error: A received frame has failed the cyclic redundancy check.

*Character synchronization*

The transmission code requires that the receiver be synchronized with the correct transmission character boundaries as well as with the individual bit boundaries in the character stream. To ensure that receivers remain synchronized with the character boundaries, a stream of idle characters is transmitted whenever no frames or sequences are being sent. In addition, as described previously, each sequence consists of a continuous alternation of the idle character and a data character. The idle character has the property that no combination of adjacent error-free data characters can result in that 10-bit pattern. This fact enables a receiver to synchronize with the transmission character boundaries in the incoming character stream.

An indication that a receiver is out of synchronization is the detection of invalid transmission characters. However, the detection of a single invalid transmission character does not necessarily mean that the receiver is out of synchronization. The clock-recovery system in the receiver has sufficient "inertia" to prevent a single transmission error from causing it to go out of synchronization. Also, a single-bit error can cause an apparent misaligned idle character to appear in a stream of data characters. Therefore, a receiver does not simply align its character boundaries with any detected idle character. In general, if loss of synchronization is declared too quickly and resynchronization occurs too quickly, the possibility exists that synchronization will be unstable. To ensure stability, the architecture includes rules for determining when the receiver is out of synchronization and for acquiring synchronization.

- *Link-level logical functions*

The link level of the ESCON architecture defines the functions and protocols necessary for link initialization,

sequence transmission and reception, frame transmission and reception, and link-error recovery.

*Frames*

Frames are classified as link-control frames and device frames. Link-control frames are used for various link-control and management functions. Device frames are used to transport data and for control functions associated with performing an I/O operation. Every frame has delimiters, addressing information, and a CRC field. Most frames also have information fields. The information field of a link-control frame contains parameters that further describe the link-control function being performed. The information field of a device frame contains data or device-level control information.

*Dynamic-connection rules*

As previously described, a channel or control unit requests the ESCON Director to make a connection to a specified destination by sending to the destination a frame that is headed by a connect SOF delimiter and contains the link address of the desired destination. This frame, in general, includes information related to the function being requested. As this request frame is passing through the ESCON Director, the connection is created. However, the requester may consider the connection to exist only after it receives a response frame from the destination. Until then, the requester is not permitted to send any further frames, since the request frame may have been corrupted by a link error and never caused the connection to be created. In addition, queuing delays at the ESCON Director may cause the request frame to be delayed.

When a channel or control unit requests a dynamic connection, there is no guarantee that the required destination will be available. The destination may already be involved in another dynamic connection, or it may be engaged in some activity that prevents it from accepting the connection request. A busy condition is signaled by responding to the connection request with either a link-level-busy (LBY) frame or a port-busy (PBY) frame. LBY indicates that the busy condition was detected by the destination ESCON interface. PBY indicates that the busy condition was detected by the ESCON Director. The information field of each of these frames contains a reason code, which describes the nature of the busy condition.

*Automatic link-address acquisition*

In order for communication to take place, each channel and each control unit must recognize its own link address in the destination-address field of each frame. Each channel and control unit ESCON interface must thus be provided with a unique address to use. Manually setting the link address into each control unit would require considerable travel, time, and complexity in a topology that can be spread out over a large site. Therefore, the architecture includes protocols that allow each channel and control unit ESCON interface to acquire its link address automatically from the ESCON Director.

To acquire a link address, a channel or control unit sends a frame called "acquire-link-address," which contains link-level addressing information that indicates that the sender is unidentified (has no link address). If the channel path is configured switched-point-to-point, the ESCON Director determines the link address associated with the port at which the frame was received and puts this link address in the destination address field of a response frame that it sends to the requester. When the channel path is configured point-to-point, the protocols permit a channel to assign its own link address and a control unit to acquire its link address from the channel. A control unit performs this protocol on each of its ESCON interfaces.

*Initialization procedures*

To bring the links to an operational state, certain initialization procedures must be performed in a prescribed order. Link initialization is performed by the use of sequences. After link initialization is complete, further initialization steps are performed using exchanges of frames.

To perform link initialization, each ESCON interface transmits a prescribed sequence and simultaneously attempts to acquire bit and character synchronization from the received signal. Each interface indicates that it has acquired character synchronization by transmitting a prescribed response sequence.

After becoming identified by the link-address-acquisition process described above, a channel is permitted to communicate with control units, and a control unit is permitted to communicate with channels. The next initialization step is the exchange-ID procedure. In this step, each ESCON interface obtains from the ESCON interface at the other end of the link the latter's unique identifier. The identifier includes information such as the type of product and its serial number. It is used for verifying the configuration and for problem determination.

Using system configuration information, the channel determines those control units with which it will communicate. The channel uses this configuration information to establish logical paths. Logical-path establishment provides each control unit with the link and logical addresses that have been assigned to the channel image. The control unit uses the link and logical addresses in future communication with that channel image. When a logical path is being established, the channel and control unit are essentially agreeing on the configuration and agreeing that all of the necessary initialization procedures required to support device-level communication have been

successfully completed. After this, either the channel or control unit can initiate device-level communication.

*State-change notification*

When an ESCON Director lies between a channel and a control unit and no dynamic connection exists between the channel and the control unit, events on the link between the ESCON Director and the control unit are invisible to the channel. Similarly, events on the link from the channel to the ESCON Director are invisible to the control unit. However, the ESCON Director is capable of detecting these events and reporting them to the channels and control units that need the information. The mechanism used is the state-change-notification (SCN) link-control function. Changes of state at a port caused by events that can affect logical paths or affect the ability of channels and control units to communicate result in the ESCON Director sending an SCN frame. Examples of events for which an SCN frame is sent include completion of link-address acquisition (a link is now available) and link failures. The SCN frame is sent on all ports that are permitted to make dynamic connections to the port undergoing the state change. It contains, in its information field, the link address that the ESCON Director associates with the port undergoing the state change.

When a channel or control unit receives an SCN frame, it determines whether it should establish logical paths or whether it already has any logical paths to the ESCON interface with the link address given in the information field. If it should establish logical paths, it proceeds to do so; if it already has logical paths to that link address, it is necessary to test whether the logical paths still exist. The test is performed with the test-initialization (TIN) link-control function. If the response indicates that a logical path no longer exists, recovery action is started to re-establish the logical path.

*Switch-contention management*

The protocols for sending frames that initiate dynamic connections provide for the case in which a channel attempts to initiate a dynamic connection to a control unit at the same time the control unit attempts to initiate a dynamic connection to that channel. This is commonly referred to as a "frames-passing scenario." For example, when an I/O device disconnects between commands in a chain and then attempts to reconnect in order to continue the chain, it is possible that, at the same moment, the channel is attempting to initiate a new I/O operation with a different I/O device on the same control unit. Only one of the two operations—the channel initiative to start a new operation with a second I/O device or the control unit request to continue the channel program already started with the first I/O device—may be permitted to continue if compatibility with the ESA/390 architecture and operating system is to be maintained.

It was decided that it is best to manage contention for this case at the end points, because the correct outcome depends on the function being performed. The end points have the information necessary to determine the appropriate "winner" in each case. This decision resulted in a new type of dynamic connection, called a "dialog-2 connection," that is created in the ESCON Director by two frames, each frame received by a different port but each frame requesting the same dynamic connection between the two ports. When a dialog-2 connection is created, the end points must resolve the contention and send the appropriate responses, so that the dynamic connection is "owned" by only one of the end points. It will be noted that having the end points resolve the contention is exactly what happens in the point-to-point topology. The result is in keeping with the goal of making the end-to-end protocol the same for the switched point-to-point topology and the point-to-point topology.

In another type of contention, some unit attempts to initiate a dynamic connection with a second unit at the same time a third unit attempts to initiate a connection with the first unit. This is commonly referred to as a "three-party scenario." Consider, for example, a channel attempting to connect with control unit A while control unit B is attempting to initiate a connection with the channel. If the control unit B request for a dynamic connection is serviced first, the channel request for a dynamic connection with control unit A cannot be allowed, because the channel port is already engaged in a dynamic connection with another port (control unit B "won"). In this case the ESCON Director sends a port-busy frame to the channel with a reason code that indicates that the channel port is busy because it is already connected to a source different from the destination of the channel request.

*Link-level recovery*

Certain errors leave the state of a dynamic connection in doubt. For example, if a channel or control unit sends a connection request and does not receive a response, either the request frame or the response frame may have been lost. (One cause of loss of a frame is a transmission error which makes a delimiter invalid.) Therefore, the sender of the connection request does not know whether the connection was made. The first concern in performing link-level recovery is establishing a known connection state at the ESCON Director. The channel or control unit that is in doubt as to whether it is connected to some other unit initiates the connection-recovery procedure. If a connection exists, the connection-recovery procedure removes it. The connection-recovery procedure involves the use of the UD and UDR sequences, mentioned above, in interlocked fashion.

Other errors do not affect the state of the dynamic connection and therefore require other recovery actions.

Typically, these errors are associated with abnormal conditions such as malfunctions. When the recipient of a connection-request frame detects one of these errors in the frame, it sends a response which breaks the connection that was created by the connection-request frame and indicates the unsuccessful delivery of the frame and the reason why. For this purpose, two link-control frames are defined: link-level reject (LRJ) and port reject (PRJ). The information fields of these frames contain reason codes which describe the error that was detected. The reason code is used by the recipient of the reject frame to determine the recovery action and as part of problem analysis. Retransmission of the frame is a common recovery action for those errors that are typically transient in nature.

The most drastic recovery action is the removal of a logical path, which is done only after a reasonable number of retries have confirmed with a high level of certainty that communication over the logical path is no longer possible.

## Device level of the architecture

The function of the device level is to provide the mapping between the I/O architecture as seen by the programmer [10] and the information-transfer facilities of the link level. The device-level architecture consists of frame formats and rules that convey the semantics of the channel program to the I/O device and provide for data transfer between the I/O device and channel as well as for various control functions. One of the design goals for the device level, as noted earlier, was that the replacement of the parallel interface by the ESCON architecture should not require changes to the I/O architecture as seen by the programmer and therefore should be transparent to software. Channel programs originally written for I/O devices on the parallel interface can be used unchanged, with the same I/O devices connected (through their control units) to ESCON channels. Another goal was to minimize the dependence of performance on distance; this was accomplished by having as few handshakes as possible. In almost all cases, an I/O operation in the ESCON architecture has fewer handshakes than the equivalent operation on the parallel interface.

Existing I/O software depends on the assumption that the states of certain indicators associated with subchannels reflect the actual state of the I/O operations at the I/O devices. To meet the goal of transparency to existing software, it was necessary to include protocols that preserve tracking the state of the I/O operation by the subchannel indicators.

In this section, we use the terms *I/O device* and *control unit*. In general, we use *I/O device* when we discuss device-level functions that always concern a specific I/O device. We use the term *control unit* when we discuss functions that may involve more than one I/O device or an I/O device to be determined by the control unit at a later time.

**Table 1** Device information block contents.

| *Device-frame type* | *Device information block contents* |
|---|---|
| Data | Data |
| Command | Flags, CCW command, count |
| Status | Flags, status, count, supplemental status |
| Control | Control function and parameters |

• *Basic device-level architectural constructs*

The primary purpose of the device level is to define the protocol for executing an I/O operation. An I/O operation is performed by a specific I/O device designated by the addressing information in the frames exchanged between the channel and the control unit. The basic divisions of the protocol, which are the same as on the parallel interface, are initiation, data transfer, and ending. Protocols are also provided for command chaining, data chaining, and various control functions. Some of these protocols are described below. (A complete description can be found in [4].)

The information used to execute the device-level protocol is divided into four categories: command, data, status, and control, each of which is carried by a similarly named subtype of the device frame. Command frames contain the contents of those fields of the channel command word (CCW) that the channel sends to the I/O device. Data frames contain the contents of the buffer referenced by the CCW, either read from the I/O device or written to the I/O device. Status frames contain the device-status byte and other information that the I/O device sends to the channel at the end of the I/O operation. Control frames contain various types of control information.

The information field of a device frame comprises 1) a device header, containing an information-field identifier (IFI), a device address, and device-header flags (DHF), and 2) a device-information block (DIB). The IFI and DHF contain control bits. Two of the bits in the IFI denote the device-frame type (command, data, status, or control). The device-information block (DIB) contains information appropriate to each frame type. This information is summarized in **Table 1**.

• *Initiation of first command of chain*

**Figure 3** shows the protocol for initiating execution of the first command of a chain. Each arrow represents one frame. The direction of the arrow indicates whether the frame is sent from the channel to the I/O device or from the I/O device to the channel.

The channel initiates the I/O operation by sending a command (CMD) frame, which contains the link and

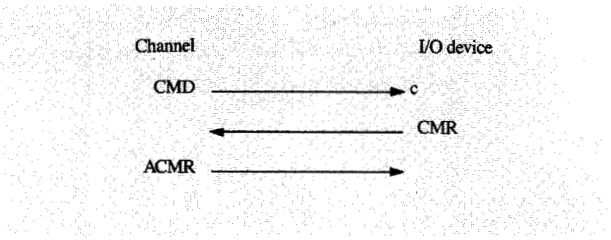



**Figure 3**

Protocol for initiating first command of chain. CMD = command frame; CMR = command-response frame; ACMR = accept-command-response frame; c = frame begins with connect SOF delimiter.

logical addresses of the control-unit image (in the link header), the device address, the command code and flags from the CCW, and a count field. For a write command, the count field contains the byte count from the CCW; for a read command, the count field contains the byte count for the first data request of the channel (see the section on data transfer, below). The CMD frame begins with a connect SOF delimiter and ends with a passive EOF delimiter, thus requesting a connection through the ESCON Director. If the device address is valid and the I/O device is able to perform the command, the I/O device responds with a command-response (CMR) frame. The I/O device does not, however, commence performing the command at this point. Instead, it waits for the channel to respond to the CMR frame with an accept-command-response (ACMR) frame. When the I/O device receives the ACMR frame, it begins performing the command.

Frames can be corrupted by transmission errors. For example, the I/O device may send the CMR frame, but the frame may be corrupted and thus not be received as a valid frame by the channel. If the protocol had not included the ACMR frame, the I/O device would have commenced execution of the I/O operation as soon as it sent the CMR. If the channel did not receive a CMR frame, it would not know whether the original CMD frame was received successfully, the CMD frame was lost, the CMR frame was lost, or something else abnormal happened at the I/O device. Thus, it would not know whether the operation had been initiated. In this situation, a recovery action could cause undetected corruption of data. For example, if the command were to backspace a magnetic tape, loss of the CMR frame would mean that the channel would not know the position of the tape. If it were to retry the backspace, the result might be to backspace the tape twice. If the backspace were followed by a write command, data would be lost.

Inclusion of the ACMR frame in the initiation protocol ensures that the I/O device does not start the operation unless the channel has received the CMR frame. If the channel does not receive a valid CMR frame, it does not send the ACMR frame, and it is guaranteed that the I/O device has not started and will not start the operation. If the ACMR frame is corrupted, the I/O device will not start the operation, but the channel will assume that the operation has begun when, in fact, it has not. Since the operation has actually not started, the recovery action will not cause corruption of data.

Whenever a channel or I/O device is expecting a response or other action, it starts a timer. If any of the three frames involved in command initiation is corrupted, the timer in either the channel or the I/O device will eventually exceed the time limit and initiate a recovery action. Depending on the circumstances, this will cause either the channel to retry the command or an error to be signaled to software.

- *Data transfer*

The rate of data transfer on any transmission medium must be regulated by the throughput capabilities of the sender and receiver. On the original IBM System/360™ parallel interface, the transfer of each byte of data was interlocked by control-signal handshakes. As data-transfer rate requirements increased with the increase in CPU speeds in later generations of processors, and transmission distance requirements increased, this interlock protocol became inadequate because the time consumed by the handshakes (due to signal propagation delays) at the longer distances limited the data-transmission rate. Eventually, the parallel interface was extended to incorporate the data-streaming protocol, which allows a stream of synchronizing pulses to be sent by the I/O device, each of which results in transferring a byte. This protocol enables the I/O device to regulate the data-transfer rate without the performance degradation from a handshake with every byte. The ESCON architecture further extends this type of protocol.

There are two components to ESCON flow control: rate pacing and data-request pacing. Rate pacing controls the minimum time (expressed as the minimum number of idle characters) between successive data frames. Data-request pacing allows the recipient of data to regulate the rate at which the sender sends data frames, without requiring a handshake for every frame. A stream of data-request frames is sent by the data recipient to control the flow of data. A data request may be for any amount of data, from one byte to the entire count of the CCW. Each data request causes the sender to transmit the requested amount of data. The data recipient sends a data request when it is able to receive the requested amount of data. A single data request may ask for so much data that the data sender must send multiple data frames; rate pacing applies within this group of data frames. If the recipient's buffer space for receiving data frames is nearly used up, it stops

sending data requests until it has transferred the buffer contents to memory. If this delay is long enough, it results in a handshake, since the sender stops sending until it receives another data request. Since each handshake results in a distance-dependent delay, as much data as possible should be sent without requiring a handshake, data requests should specify large amounts of data, and several outstanding data requests (see below) should be permitted. The optimum combinations of frame size, data request size, and number of outstanding data requests depend on the performance requirements of each implementation.

When a logical path is established, the channel specifies the minimum number of idles (for each data-frame size it permits) that the control unit must supply between data frames sent to the channel, and the number of data requests it is willing to receive at any time (outstanding data requests). When each I/O operation is initiated, the control unit informs the channel, using fields in the command-response frame, of the maximum data-frame size it can receive and, for a read operation, the number of data requests it is willing to receive at any time and the minimum number of idles it requires between successive data frames. The data-frame size it specifies must be one of those permitted by the channel, as indicated when the logical path was established.

• *Ending last command of chain*

The process of ending the execution of the last command of the chain consists of sending the device-status information to the channel, checking the number of bytes of data transferred, and breaking the connection through the ESCON Director. The I/O device always initiates the ending process, whether the command was a read or a write. The ending protocol is illustrated, for a write command, in **Figure 4**.

When the I/O device receives the final data frame, it sends a status (STA) frame to the channel. The STA frame contains information that describes the success or failure of the operation. The channel then prepares to present the status information to the program and sends an accept-status (ACC) frame to the I/O device. When the I/O device receives the ACC frame, it responds with a device-level-acknowledgment (DACK) frame ending with a disconnect EOF delimiter, which breaks the connection through the ESCON Director. When the channel receives the DACK, it makes the status information available to the program and sets the appropriate state information associated with the subchannel.

The reason for the three-frame handshake in the ending procedure is to ensure that the channel and I/O device agree as to whether the channel did or did not accept the status information. In some situations, the channel is unable to accept the status information and signals the I/O

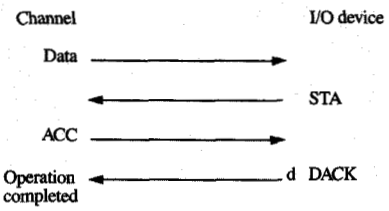



**Figure 4**

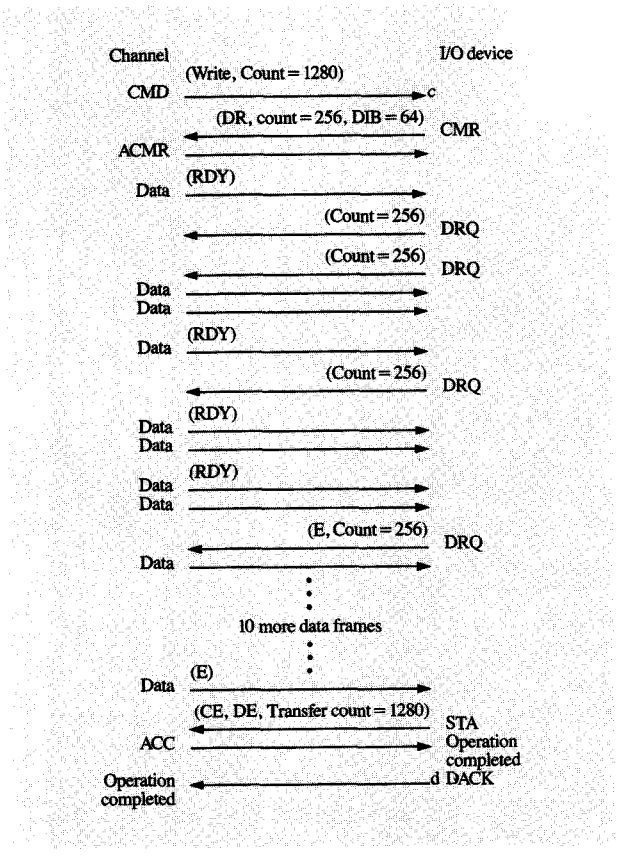
Protocol for ending last command of chain for write command. Data = final data frame; STA = status frame; ACC = accept-status frame; DACK = device-level-acknowledgment frame; d = frame ends with disconnect EOF delimiter.

device to retain the status information for presentation at a later time. Unless the I/O device receives a response from the channel, it does not know whether or not the channel accepted the status information. If the STA frame or the channel response frame is corrupted by a link error, the I/O device does not receive a response and does not know whether or not to retain the status information.

The final DACK frame resolves the ambiguity. If the channel receives the DACK, it knows that the I/O device received its response. If the channel does not receive the DACK, either the DACK or the channel response to the status frame has been corrupted. In either case, the channel can proceed as if the I/O device has not received the channel response. If the I/O device has received and acted on the channel response to the STA frame, there is no ambiguity in the recovery process. If the channel accepted the status the first time, the I/O device has no status information to present in the recovery action. If the channel did not accept the status the first time, the I/O device will re-present it in the recovery action.

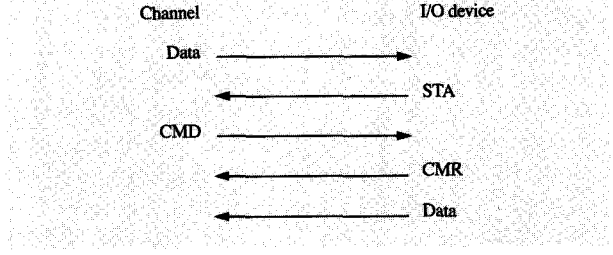
• *Complete protocol for single CCW*

**Figure 5** illustrates how the protocols described in the previous sections are combined to execute a single write command. When the system was initialized, the channel established the logical path and permitted the I/O device to have two outstanding data requests. The program issues a write CCW with a count of 1280 bytes. The channel starts the I/O operation by sending the I/O device a command (CMD) frame with a CCW count of 1280 bytes. The I/O device accepts the command and responds with a command-response (CMR) frame specifying a device-information block (DIB) of 64 bytes, meaning that it will accept data frames, each holding 64 bytes of data. The CMR includes a data request (DR bit set) for 256 bytes, allowing the channel to send 256 bytes (four data frames)




## Figure 5

Protocol for single write CCW. Labels above arrows denote fields in frames; CMD = command frame; CMR = command-response frame; DR = data-request bit; DIB = number of data bytes per frame; ACMR = accept-command-response frame; Data = data frame; RDY = ready bit; E = end bit; CE = channel-end bit; DE = device-end bit; DRQ = data request frame; ACC = accept-status frame; DACK = device-level acknowledgment; c = frame begins with connect SOF delimiter; d = frame ends with disconnect EOF delimiter.




## Figure 6

Protocol for command chaining of a write to a read. Data = data frame; STA = status frame; CMD = command frame; CMR = command-response frame.

before it must wait for another data request. The channel responds with an accept-command-response (ACMR) frame and immediately follows it with a data frame containing 64 bytes of data and having the ready (RDY) bit set, giving the I/O device permission to send more data requests (DRQ frames). Since the I/O device is allowed to have two DRQs outstanding at any time, it sends two more. (It should be understood that although the figure shows DRQs and data frames interspersed, DRQs are not interlocked with data frames; DRQs may be sent at any time at which they are permitted by RDY bits in data frames.) The channel then sends two more data frames and a third data frame with the RDY bit set, giving the I/O device permission to send one more DRQ, which it does. The channel sends another data frame with the RDY bit set. At this point the I/O device could send another DRQ, but it does not do so until later. After three more data frames, the I/O device sends its last DRQ, signifying that it is the last by setting the end (E) bit in the DRQ. The channel now sends the remaining data frames, setting the E bit in the twentieth data frame. The I/O device then responds to the E bit with a status (STA) frame containing a transfer count indicating that it received 1280 bytes. The device-status byte has only the channel-end (CE) and device-end (DE) bits set, indicating that no errors occurred and the operation is complete. The channel accepts the status, verifies that the transfer count is equal to the number of data bytes it sent, and responds with an accept-status (ACC) frame. When the I/O device receives the ACC, it considers the operation completed and responds with a device-level acknowledgment (DACK) frame containing a disconnect EOF delimiter, which breaks the connection through the ESCON Director. When the channel receives the DACK, it considers the operation completed and makes the status information available to the program.

- *Command-chaining protocol*

Command chaining provides sequential execution of multiple I/O operations in the same channel program. To preserve compatibility with existing software, each I/O operation must be concluded with the presentation of the device-status information. However, since the chain continues, there is no need to go through the full ending and initiation protocols described previously. Instead, the channel responds to the status frame of the first command by sending the command frame for the second command. Further, during command chaining, there is no need for special precautions regarding synchronizing the states of the subchannel and device. Therefore, there is no need for the accept-command-response frame. The command-chaining protocol is illustrated in **Figure 6.**

The ESCON architecture also includes a protocol that removes the connection through the ESCON Director

between commands in a chain. This protocol is used when the device has long mechanical delays between commands. For details, see [1, 4].

## Error handling

The ESCON architecture contains extensive facilities for detecting, reporting, and recovering from various kinds of errors. In this section, we provide an overview of some of the key device-level facilities.

*Transmission-error handling* The selection of fiber optics permits the design of links that are orders of magnitude less sensitive to noise than links using electrical transmission at similar distances. The resulting low transmission-error rate means that retransmission of a data frame containing an error does not significantly improve performance compared to repeating (at the device level) the I/O operation from the beginning. Therefore, the ESCON architecture relies on the same command-retry procedures as on the parallel interface for recovery from errors during I/O operations.

*Unit check status and sense data* When the I/O device detects certain types of errors, such as data errors on its storage medium and link errors in received information, it signals these errors by terminating the I/O operation and presenting status with the unit-check bit set. It makes available sense data which further describe the error or indicate a recovery action that is to be performed. On the parallel interface, the program responds to the unit-check status by issuing a sense command to obtain the sense data. This can also be done in the ESCON architecture; however, the ESCON architecture also provides means for the I/O device to automatically send the sense data to the channel, thus avoiding the additional program activity required to issue the sense command. When control bits indicate that both the channel and the I/O device are capable of performing the automatic-sense operation, the I/O device includes the sense data in the status frame with the unit-check bit, in a field called supplemental status. A corresponding facility in the processor I/O architecture, called concurrent sense, makes the sense data available to the program with the status information.

*Command retry* The ESCON architecture provides the same channel-command-retry capability as the parallel interface. If the I/O device is capable of automatically retrying a command to recover from an error, the device includes bits in the status frame which request that the channel retry the command. The channel responds by re-initiating the latest command. The ESCON architecture provides an additional facility, called channel-initiated command retry, that permits automatic recovery from channel-detected errors. When the channel detects an error for which automatic retry is appropriate, it sends a frame to the I/O device that gives the device permission to request command retry. If the I/O device does request command retry, the channel reissues the command instead of reporting the error to software. This increases the proportion of errors that can be retried without invoking software and thus potentially improves system performance. It is particularly valuable in the case of elevated link-error rates, because it allows the link to continue operating, without severe performance degradation, until a repair action can be scheduled at the convenience of the user.

## Summary

The IBM Enterprise Systems Connection Architecture is the message-based architecture of the fiber optic I/O interconnect system for the IBM System/390 computer family. It replaces the parallel electrical bus system used since the first System/360 computers appeared in 1964. The architecture permits exploitation of the high data-transfer rate and long-distance capabilities of fiber optics. Its interconnection topology is based on a dynamic crosspoint switch that provides a high degree of connectivity among multiple systems and their shared I/O devices. One of the key constraints on the design was compatibility with channel programs written for the parallel interface and with most other I/O software. Other goals included high reliability and availability and, in particular, robustness to enable operation under conditions of degraded error rates. This paper has described some of the key features of the ESCON architecture.

## Appendix: System/390 I/O programming

To define some of the terminology used in this paper, this appendix provides a brief overview of System/390 I/O programming as defined by the ESA/390 Principles of Operation Manual [10].

**Figure 7** shows the data structure for I/O programming. To perform data transfer or other operations with an I/O device, the programmer writes a "channel program" which specifies the operations the device is to perform and which contains the addresses of data buffers in the processor main storage. In order to execute the channel program, the processor program issues the Start-Subchannel (SSCH) instruction, whose parameters include the address in main storage of a control block called the Operation Request Block (ORB), and the identification number of a subchannel, which is the representation of the I/O device to the program. The ORB contains, among other things, the address of the channel program. Information associated with the subchannel includes the address of the I/O device. For devices attached to ESCON channels, this address includes the link address, logical address, and device address, as discussed in this paper.

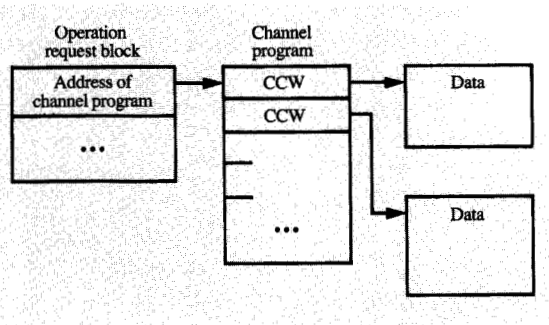



**Figure 7**

ESA/390 architecture I/O-programming data structure.

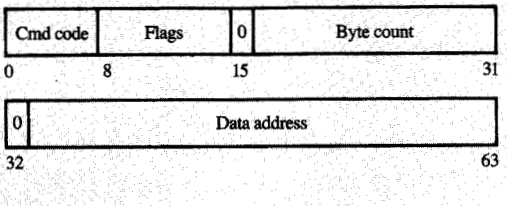



**Figure 8**

Channel command word format. The numbers identify bit positions in the 64-bit CCW.

A channel program consists of one or more channel command words (CCWs). Each CCW identifies a buffer in processor memory. The buffer contains either data to be sent to the device or space for receiving data from the device. As described further below, each CCW may also contain a command to the device. **Figure 8** shows the structure of a CCW as defined in the ESA/390 architecture. The command-code field contains the command to the I/O device. Examples of commands are read, write, and seek. The byte-count field specifies the number of bytes of data to be transferred to or from the I/O device. The data-address field contains the address, in System/390 main storage, of the data buffer. The flags field contains a number of control flags, most of which are outside the scope of this paper.

When the channel program consists of more than one CCW, the CCWs are said to be chained together under control of the chain-data (CD) and chain-command (CC) flags. Chained CCWs are usually executed in the order in which they appear in the program; however, certain actions of the device may modify this order. The CD flag in a CCW indicates that the CCW that follows in storage contains the address of a new buffer to be used to continue data transfer for the current command. Data chaining thus permits "scatter-gather," the use of a set of noncontiguous buffers for the data associated with one command. The CC flag in a CCW indicates that the next CCW contains a new I/O device command and a new buffer address.

The term *I/O operation* denotes the execution of a CCW containing an I/O device command; this includes any further data-chained CCWs. A channel program consists of one I/O operation or several I/O operations that are command-chained together. When discussing interface protocol, we refer to a command-chained channel program as a *chain*. Since an I/O operation involves a single command, we frequently use the term *command* in place of *I/O operation* when discussing ESCON architecture.

Typically, completion of a chain causes an interruption of the currently executing program and causes a program to be executed that processes the interruption. The processor makes available to the interrupt-processing program information from which the program can determine the success or failure of the chain and the number of bytes of data transferred to or from the I/O device. Included in the interruption information are a byte of device status, provided by the I/O device as part of the interface protocol, and a byte of channel status. These status bytes indicate success or failure of the I/O operation and provide some information about the nature of any failure that might have occurred. In the case of error conditions, the I/O device provides one or more bytes of sense data, which more fully describe the error condition. The sense data can be transferred to the channel subsequently using a sense CCW. For certain devices on ESCON channels, the sense data are automatically transferred to the channel at the end of the I/O operation.

## Acknowledgments

Many people have contributed to the development of the ESCON architecture. In particular, we express our appreciation to Paul Brown, John R. Flanagan, Karl Hoppe, Allan Meritt, John Sorg, and Peter Tallman. We also thank Robert Dugan, manager of IBM Enterprise Systems Central I/O Architecture, for his support during the development of the ESCON architecture.



## References

1. *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.
2. *High Performance Parallel Interface Mechanical, Electrical, and Signalling Interface Protocol Requirements*, American National Standard ANSI X3.183-1991, American National Standards Institute, New York, 1991.
3. N. R. Aulet, D. W. Boerstler, G. DeMario, F. D. Ferraiolo, C. E. Hayward, C. D. Heath, A. L. Huffman, W. R. Kelly, G. W. Peterson, and D. J. Stigliani, Jr., "IBM Enterprise Systems Multimode Fiber Optic Technology," *IBM J. Res. Develop.* **36**, 553–576 (1992, this issue).
4. *IBM Enterprise Systems Architecture/390 ESCON I/O Interface*, Order No. SA22-7202; available through IBM branch offices.
5. *IBM Enterprise Systems Architecture/390 ESCON I/O Interface Physical Layer*, Order No. SA23-0394; available through IBM branch offices.
6. S. A. Calta, J. A. deVeer, E. Loizides, and R. N. Strangwayes, "Enterprise Systems Connection (ESCON) Architecture—System Overview," *IBM J. Res. Develop.* **36**, 535–551 (1992, this issue).
7. C. J. Georgiou, T. A. Larsen, P. W. Oakhill, and B. Salimi, "The IBM Enterprise Systems Connection (ESCON) Director: A Dynamic Switch for 200Mb/s Fiber Optic Links," *IBM J. Res. Develop.* **36**, 593–616 (1992, this issue).
8. *Introducing the Enterprise Systems Connection Manager*, Order No. GC23-0422; available through IBM branch offices.
9. A. X. Widmer and P. A. Franaszek, "A DC-Balanced, Partitioned-Block, 8B/10B Transmission Code," *IBM J. Res. Develop.* **27**, 440–451 (1983).
10. *IBM ESA/390 Principles of Operation*, Order No. SA22-7201; available through IBM branch offices.

*Received January 30, 1991; accepted for publication June 19, 1991*

**Joseph C. Elliott** *IBM Enterprise Systems, P.O. Box 390, Poughkeepsie, New York 12602 (ELLIOTT at TDCSYS2).* Mr. Elliott is a senior engineer in the Enterprise Systems Central System Architecture Department. His primary responsibility since joining the department in 1980 has been the development of new I/O architecture, and he was the lead architect in the development of the ESCON architecture. In 1967 Mr. Elliott joined IBM in East Fishkill, New York, where he worked on the design and development of process control systems, the application of liquid crystal devices, and communication systems. He received his B.S. degree in electrical engineering from Drexel University in 1967 and his M.S. degree in electrical engineering from Syracuse University in 1974. Mr. Elliott has received an IBM Outstanding Innovation Award for his work on the ESCON architecture; he has also received the IBM Third-Level Invention Achievement Award.

**Martin W. Sachs** *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 704, Yorktown Heights, New York 10598 (SACHS at YKTVMH, sachs@watson.ibm.com).* Dr. Sachs is a research staff member in the Department of Computer Sciences at the IBM Thomas J. Watson Research Center. He received the A.B. degree in physics from Harvard University in 1959 and the M.S. and Ph.D. degrees in physics, specializing in nuclear physics, from Yale University in 1960 and 1964, respectively. From 1964 to 1966 (including one year as a NATO postdoctoral fellow), he performed nuclear physics research at the Weizmann Institute of Science in Israel. From 1967 to 1976, he was a member of the Yale University Wright Nuclear Structure Laboratory. There, he continued research in nuclear reactions and was responsible for the Yale half of an IBM–Yale joint study in nuclear physics data acquisition and for the laboratory's subsequent activities in computer-based data acquisition. In 1976, he joined the IBM Thomas J. Watson Research Center, where he initially worked in the computer systems aspects of signal processing and then began his current work on the I/O architecture of general-purpose computers. He has received IBM Outstanding Innovation Awards for his research on serial I/O architecture and for his contributions to ESCON architecture. He has also received three IBM Invention Achievement Awards. Dr. Sachs is a Senior Member of the Institute for Electrical and Electronics Engineers, and a member of Sigma Xi, the Association for Computing Machinery, and the American Physical Society.