SCB Architecture

Insert the hardtab labeled "SCB Architecture" here and discard this page.

First Edition (January, 1991)

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM Authorized Dealer or your IBM Marketing Representative.

© Copyright International Business Machines Corporation 1991. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Special Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights or other legally protectible rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, programs, or services, except those expressly designated by IBM, are the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Commercial Relations, IBM Corporation, Purchase, NY 10577.

Notwithstanding the notice of the preceding paragraph regarding rights and licenses, the purchaser of this Technical Reference is granted certain rights as specified on the following page entitled "License Statement."

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States and/or other countries:

IBM Personal System/2 PS/2 Micro Channel

The following terms, denoted by a double asterisk (**) in this publication, are trademarks of other companies as follows:

Intel is a trademark of the Intel Corporation.
Motorola is a trademark of Motorola, Inc.

© Copyright IBM Corp. 1991

LICENSE STATEMENT

By purchase of this Subsystem Control Block I/O Architecture Technical Reference (Technical Reference) you are authorized under IBM's copyrights in the Technical Reference:

1. To reproduce copies (which are not complete copies of the Technical Reference)
2. To prepare derivative works
3. To distribute copies.

Such copies may be in the form of hardware, software or a combination thereof as well as documents. No license is granted, however, to copy any IBM computer code.

You are eligible for a limited license under patents owned by IBM that cover inventions the practice of which is not avoidable in implementing, in device driver computer code, control blocks and elements complying with the Technical Reference at Part 2, Chapter 1, Sections titled "Command Control Blocks," "Termination Status Blocks," "Commands"; Part 2, Chapter 4; Part 3, Chapter 1 Section titled "Control Elements"; and Part 3, Chapter 4. Such patent license is non-exclusive, royalty free and nontransferable and permits you to make, use and sell such device driver computer code. To the extent hardware or software infringes a patent absent device driver computer code licensed hereunder it is not relieved of such infringement by the addition of such code. To qualify for this patent license you must purchase a copy of the Technical Reference, license IBM and its subsidiaries under the same terms and conditions for patents you own that apply to such device driver computer code or improvements thereof and activate the license set forth below.

You need take no action to activate the copyright authorization specified above. It is activated automatically by your purchase of this Technical Reference. You, of course, must make your own independent assessment regarding needs under any patents. Should you wish to activate the patent license specified above you may do so by notifying IBM to that effect in a letter to the IBM Director of Commercial Relations, IBM Corporation, 2000 Purchase Street, Purchase NY 10577.

Preface

The Technical Reference library is intended for those who develop hardware and software products for IBM* Personal Computers and IBM Personal System/2* products. Users should understand computer architecture and programming concepts.

This technical reference provides a description of hardware and software requirements and design considerations about the SCB architecture; it should be used with the following publications:

IBM Personal System/2 Hardware Interface Technical Reference

IBM Personal System/2 and Personal Computer BIOS Interface Technical Reference.

This manual consists of the following sections:

Part 1, "Subsystem Control Block Architecture Overview," lists the basic design consideration of a system using the SCB architecture and describes the overall concepts of the architecture. It includes a brief description of the available modes and lists design requirements to consider in selecting the mode to use.

Part 2 describes the Locate mode.

Part 3 describes the Move mode.

The appendixes contain examples of the data structures used in specific modes.

Warning: The term "Reserved" describes certain bits and data areas that should not be changed. Use of reserved areas can cause compatibility problems. Programs and hardware writing to reserved areas should ensure that the areas are set to 0; programs and hardware reading these areas should ensure that the areas are treated as "don't care."

---

* IBM, Personal System/2, and Micro Channel are trademarks of the International Business Machines Corporation.

© Copyright IBM Corp. 1991

Notes:

Contents

© Copyright IBM Corp. 1991

Figures

© Copyright IBM Corp. 1991

Notes:

Chapter 1. Introduction

In the personal computing and workstation environments, more and more adapters are becoming functional subsystems that provide more than just device attachment.

The bus master in Micro Channel-based systems is capable of transferring data on the system channel independent of the system processor. This allows adapters, such as LAN interfaces and SCSI controllers, to use bus-master capabilities to perform tasks independently, thereby freeing the system processor to perform other tasks and improving the overall throughput within the system.

The Subsystem Control Block (SCB) architecture enhances the potential of the bus master by defining services and conventions that deliver command information and data to the adapter. The architecture provides command chaining, data chaining, signalling, and synchronization of command and command status; it supports system-to-adapter and adapter-to-adapter (peer-to-peer) operations.

This architecture defines the logical protocols and control structures used to transfer command and control information, data, and status information between the system processor and an adapter, or directly between adapters. The architecture separates the delivery of control information and data to increase performance, raise the level of functional capability, and provide design flexibility.

The SCB architecture is based on three levels: the processing level, the delivery level, and the physical level.

Entities at the processing level can be characterized by the roles they perform. *Client* entities are entities that make requests of *server* entities. A server entity performs the work necessary to satisfy the request and return a reply to the client entity. In some cases, an entity might assume the role of both client and server. In this case, the entity is called an *agent*. An agent takes a request from a client entity and requests the services provided by other server entities to complete the request.

The delivery level provides the set of services and protocols used to deliver requests and replies to the specific functional entities for the

© Copyright IBM Corp. 1991

system processor or an adapter. The architecture consists of two modes of delivery: the Locate mode and the Move mode.

The physical level defines the physical interface for accessing the data and control information in locations within the system address space for I/O and memory. The physical level supports the driving and receiving of buses and signals between adapters on the channel and to and from the system processor. The specific implementation of this level is determined primarily by the design of the system hardware.

Because the interaction between levels is at a logical level, enhancements to an adapter or the base system (such as increasing data rates and changing the data-transfer procedures) will not cause changes to the processing-level support.

Architecture Applications

The SCB architecture is designed to have diverse application and to be used in the following areas:

- Traditional I/O-device protocols.

  These protocols typically have a single system processor, which requests an adapter to perform work on its behalf. With these protocols, requests flow from entities in the base system to entities in the adapter. The adapter can support the attachment of one or more devices.

  In this environment, the Locate mode and the Move mode can be used. In the Locate mode, the address of control information is passed to the adapter; the adapter then uses the address to locate the control block and fetch it into its own storage area for execution. The reply to the system must be synchronized with the request.

  In the Move mode, a copy of the control information is passed directly to the adapter. The reply does not have to be synchronized with the request.

- Peer-to-peer protocols.

  In peer-to-peer protocols, requests can flow from entities in one adapter directly to entities in another adapter, as well as from entities in the base system to entities in an adapter.

  For these environments, the Move mode must be used. The control information is physically moved between the requesting and responding entities through two prearranged paths in memory, called pipes. The Move mode is especially useful for controllers that have high arrival rates of work requests or that run in an asynchronous manner. This mode is also useful in supporting the distribution of process-to-process type functions between the base system and adapters.

- Communication protocols.

  Communication protocols require routing to network servers within an adapter, and they require the handling of replies, which can arrive out of sequence and interspersed with requests. They can also require the ability to handle the movement of large amounts of data at very high speeds.

For communication environments, the method of control delivery used would be the Move mode.

- Response-time-critical applications.

  The Move mode is ideal for application environments where response time is critical, because it allows command and data to flow to and from an adapter, can operate with or without interrupts, and can process multiple requests and replies asynchronously.

Subsystem Control Block Characteristics

The following are some capabilities of the SCB architecture that enhance system and adapter performance in the Micro Channel and bus-master environments:

- Provides a programming model for use with Micro Channel-based systems
  - Provides greater flexibility of design
  - Provides high levels of function distribution
  - Provides for signalling between bus masters
  - Provides a means of chaining commands and data
  - Defines request and reply protocols
    - Provides a means of correlating requests to replies
    - Provides source and destination identification
    - Allows multiple outstanding requests
    - Allows unsolicited operations
  - Allows data and control information to be delivered together or separately
- Supports bus masters on the channel
  - Provides a processor-independent architecture
  - Supports full-duplex or half-duplex operation
  - Supports adapter-to-adapter operation
  - Supports synchronous and asynchronous operations
- Improves performance and throughput
  - Reduces interrupt overhead
  - Reduces the number of separate memory accesses
  - Reduces overhead in passing control information and data
  - Optimizes burst or streaming data capabilities.

System Overview

Systems in the personal computing and workstation environment have evolved to a point where a user system can have adapters operating as simple I/O slaves or as bus masters. These adapters communicate with the base system, using the channel. Adapters with bus-master capabilities can also communicate directly with each other on this same channel.

These adapters can be a standard part of the system or an option that can be added later to make up a user system. An example of the physical structure for a user system is shown in the following figure.

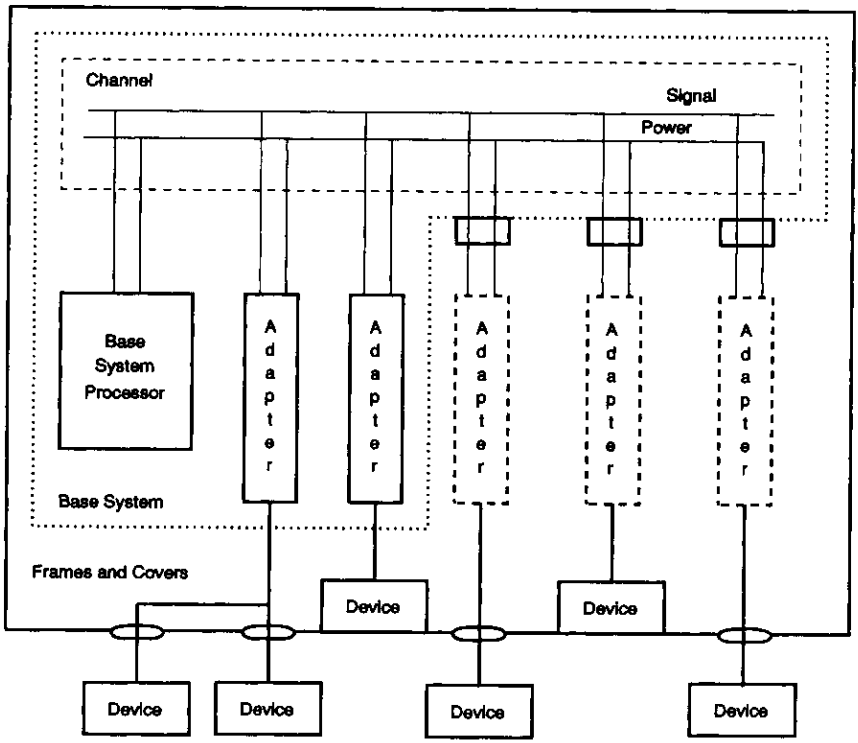

Figure 1-1. User System

In bus-master environments, a high-level, control-block-oriented interface is established between the base system and the adapter or directly between the different adapters.

Bus-Master Characteristics

The following lists some of the characteristics of bus-master adapters:

- An adapter may need to communicate directly with another adapter.
- A single adapter can support the attachment of more than one device. These devices can be of different types and might need to operate asynchronously to the delivery of the control information.
- Adapters with several devices attached can support different forms of function distribution. In one adapter, the function might be distributed using the traditional I/O-device protocol, where the support in the base system manages the devices attached to an adapter.

  In another adapter, the function might be distributed using the peer-to-peer protocol, where the support in the base system and the adapter act as peers, and the adapter itself manages the devices attached to it.

System Resources

This section describes the meaning of the terms byte, word, and doubleword and defines the rules on memory organization. In addition, it describes the use of I/O address space, memory address space, private or local memory, and shared memory.

Because the SCB architecture must work with a variety of system designs, certain aspects of system resources must be defined within the context of the architecture.

Different system hardware designs access I/O address space in different ways. Some designs separate the I/O and memory address space, some have I/O and memory in the same address space, and some designs have separate address spaces but allow I/O to be mapped into the memory address space.

The SCB architecture operates in any of these environments; however, to do that, it places some requirements on adapters and system masters. The architecture requires that all addresses be byte-addressable and defines the I/O addresses used by a given type of adapter.

I/O Address Space

Systems based on Micro Channel architecture have an I/O address space of 64KB.[1] The I/O address space is selected when the signal that selects the address space is in the I/O state. The I/O addresses are mapped into I/O ports or registers, which are assigned to I/O devices, adapters, and system-control functions (such as timers and memory controls). The width of an I/O port defaults to 1 byte; however, it can be a word (2 bytes wide) or a doubleword (4 bytes wide).

Memory Address Space

Systems based on Micro Channel architecture have an address space

[1] KB = 1024 bytes

of 4GB.[2] The memory address space is selected when the signal that selects the address space is in the memory state. The memory address space contains the memory that the system master can access for instructions and data and use for the transfer of data.

Private or Local Memory

Systems based on Micro Channel architecture can have memory that is not part of either the memory or the I/O address space; this memory is called *local memory*. An example of this private or local memory is shown in Figure 1-2 on page 1-11. In the example, all memory on Adapter C and a portion of the memory on Adapter B is outside the 4GB memory address space; this memory is part of a private bus and is local to that adapter.

Shared Memory

Shared memory is that memory in a system that can be directly accessed by all masters on the channel. This includes all memory within the 4GB memory address space, including system board memory and channel memory. In the example, all the memory on Adapter A and a portion of the memory on Adapter B are within the 4GB memory address space and can be shared by all masters on the channel.

---

[2] GB = 1,073,741,824 bytes

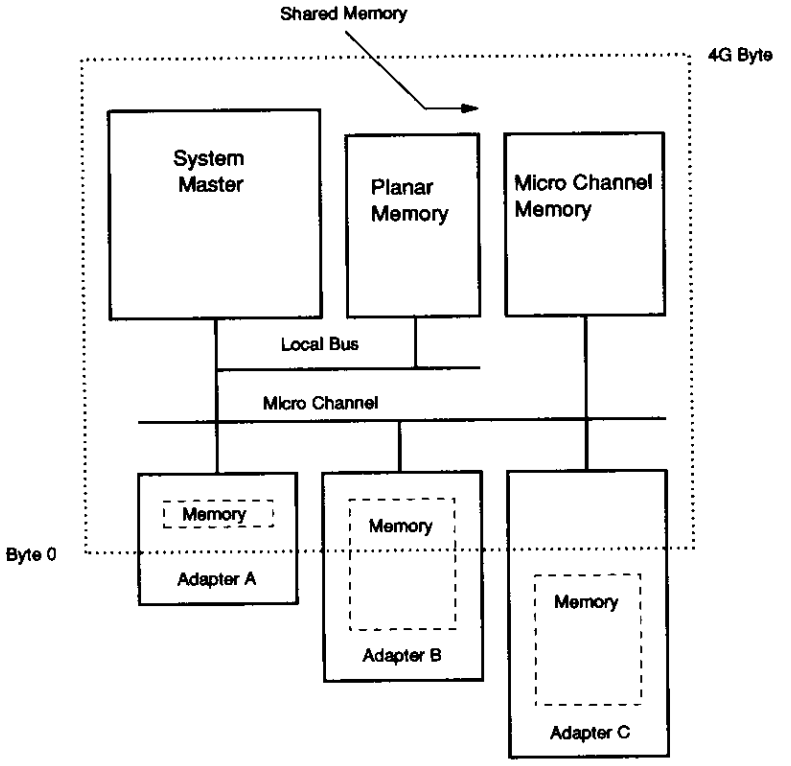

Figure 1-2. Example of Memory

Memory Organization

Since the architecture must work with a variety of system and adapter hardware and software designs in a heterogeneous environment, all data must follow the same mapping and organization conventions to prevent miscommunication between the different components (system units and adapters).

Whether in memory or I/O address space, a *byte* is 8 contiguous bits and must be considered as a single object; the bits are numbered from 0 to 7. Bit 0 is the least-significant bit.

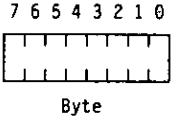

Figure 1-3. Byte Notation

Whether in memory or I/O address space, a *word* is always 16 bits and must be considered as a single object; the bits are numbered from 0 to 15. Bit 0 is the least-significant bit and is in byte 0, which is the least-significant byte (low byte).

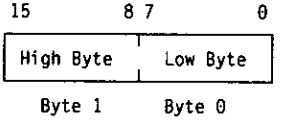

Figure 1-4. Word Notation

Whether in memory or I/O address space, a *doubleword* is always 32 bits and must be considered as a single object; the bits are numbered from 0 to 31. Bit 0 is the least-significant bit and is in byte 0, which is the least-significant byte (low byte).

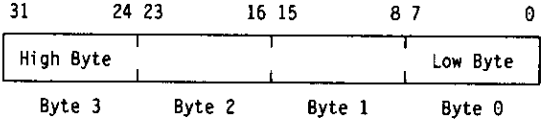

Figure 1-5. Doubleword Notation

When stored in shared memory, all data must follow the same storage conventions. The following shows the mapping and organization of shared memory.

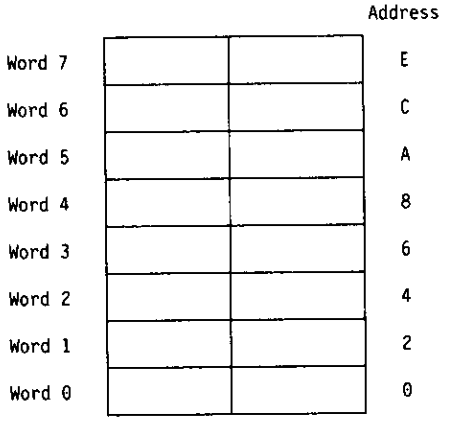

Figure 1-6. Memory Organization

In shared memory, a word must be stored and read as a single object and must be aligned on even address boundaries with the high byte (the byte containing the most-significant bit) in the low-byte position, as shown in the following figure.

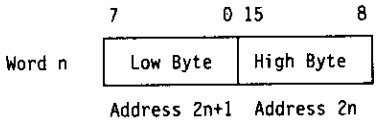

Figure 1-7. Word Storage Format

In shared memory, a doubleword is stored and read in the following format.

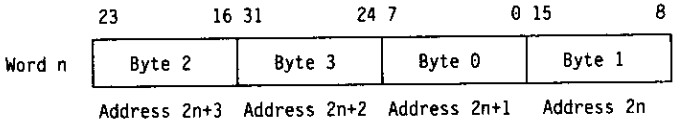

Figure 1-8. Doubleword Storage Format

Notes:

Chapter 2. SCB Architecture Overview

The SCB architecture defines a method of transferring control-related information and data to an adapter that maximizes the throughput and minimizes the time that adapters and the system processor spend managing the transfer. The control-related information is in the form of requests to an adapter and replies from an adapter and includes the asynchronous notification from or to an adapter.

By packaging control-related information in blocks (including commands, options, and data pointers), the architecture allows the individual adapter to control and manage the transfer with minimal or no intervention from the system processor. Each adapter can manage several tasks concurrently, allowing several devices to be operated at the same time by one adapter.

© Copyright IBM Corp. 1991

Logical Levels

Each adapter and system unit is logically structured into three levels: the physical level, the delivery level, and the processing level.

The *physical level* defines a set of services and protocols for accessing control areas in I/O address space, for transferring data and control information into and out of shared memory, and for signalling between the system master and adapters.

The *delivery level* defines a set of services and protocols for managing the delivery of one or more control blocks or elements between the system master and an adapter, or between adapters.

The *processing level* directly supports the exchange of requests and replies between a pair of cooperating entities. One entity, the client, sends requests to the other entity, the server, and receives replies in response.

Physical Level

Services at the physical level are used to pass data and control information at locations within the I/O and memory address space and to support driving and receiving signals between adapters on the channel.

The services provided depend on the mode of delivery.

Push The Push service moves data or control information from a location in the caller's memory to a location in shared memory or I/O address space.

Pull The Pull service moves information from a location in shared memory or I/O address space to a location in the caller's memory.

Signal The Signal service alerts the logic in the system master or adapter that some information needs to be processed.

These services are used by support logic at the delivery level in the base system and adapters to access data and control information in memory or I/O address space and to signal the logic in the base system or another adapter.

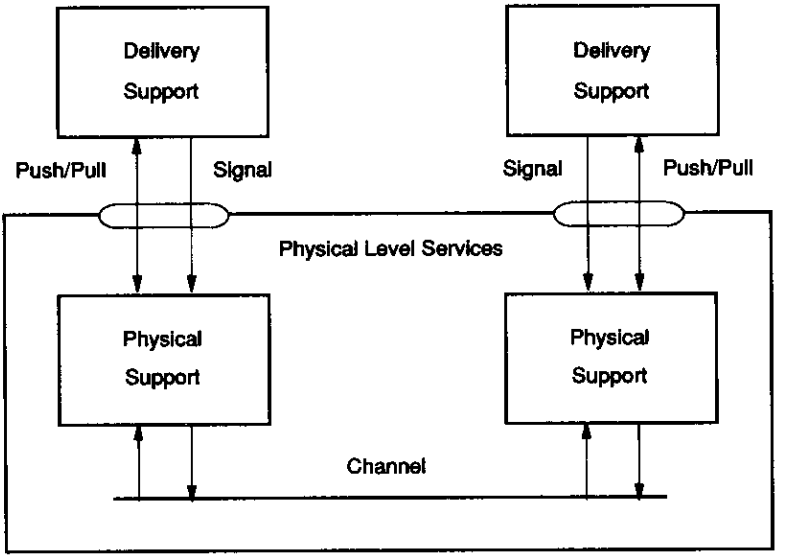

Figure 2-1. Physical-Level Structure

Delivery Level

The delivery level provides a set of services and protocols used to support the delivery of requests from the client to the server and replies from the server to the client. The following figure shows the delivery-level structure.

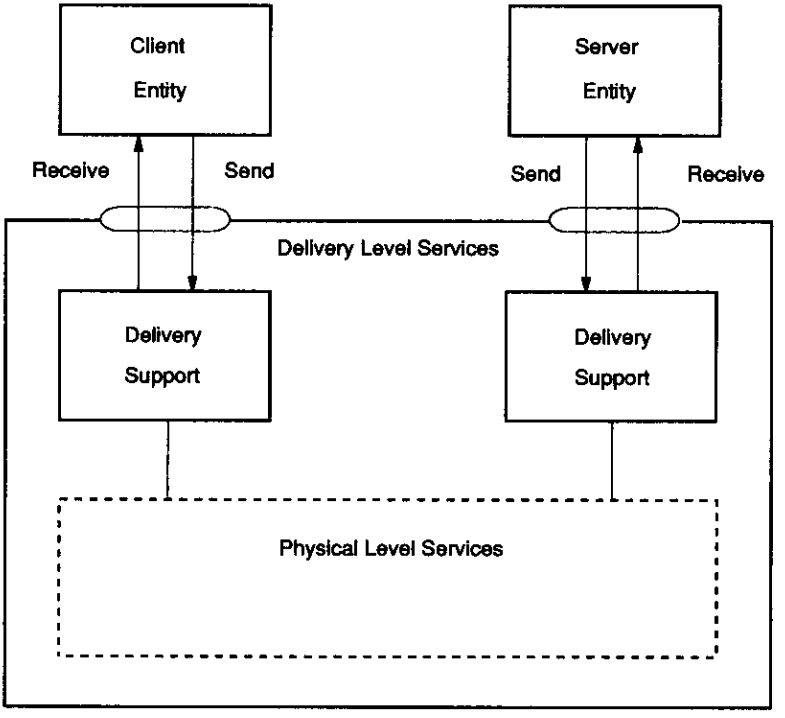

Figure 2-2. Delivery-Level Structure

The delivery level provides the interface between the processing level and the physical level. The interface between the delivery level and the processing level is in the form of a set of primitives for exchanging control-related information, status information, and data. The interface between the delivery level and the physical level is in the form of a set of primitives for accessing the control areas in I/O and memory address space and for signalling between units.

Processing Level

The entity represents the basic functional element supported by the architecture. The client and server operate as an entity pair. Usually, the client is located in one unit and the server in another, although both can be in the same unit.

The following figure shows the processing-level structure. The protocol between entity pairs is concerned with the format and content of the requests and replies. Requests are always from the client, and replies are always from the server. The processing level uses the services provided by the delivery level to manage the exchange of requests and replies between the client and server.

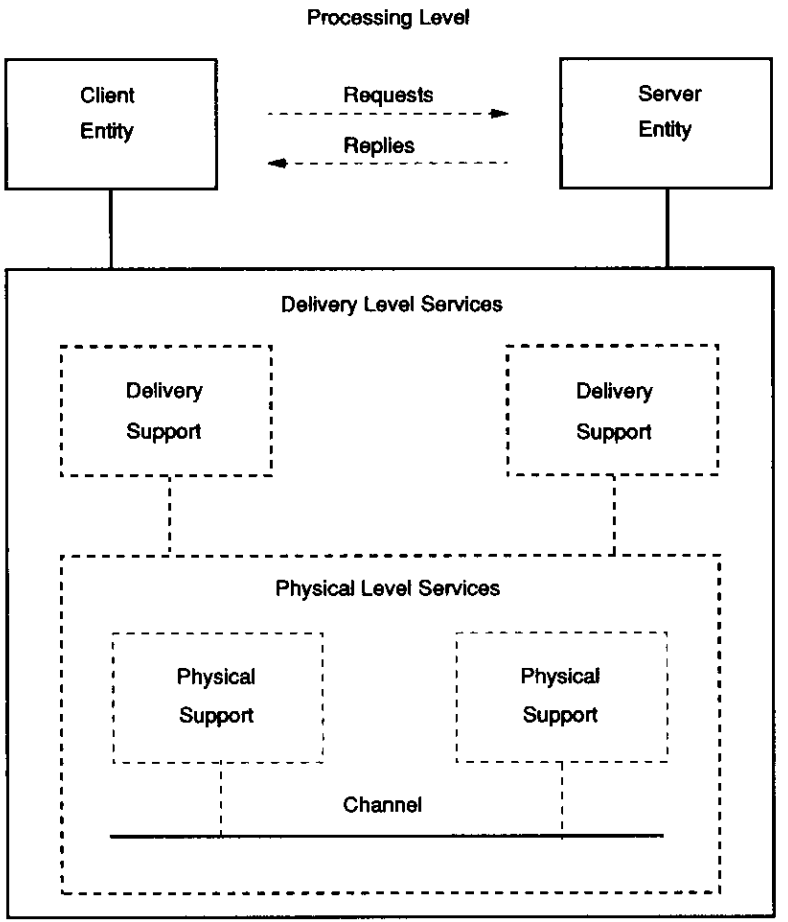

Figure 2-3. Processing-Level Structure

Client

The client entity is the entity that uses or requests services provided by another entity. An example of a client is a program that is used to initiate the reading or writing of data to or from a disk drive.

Server

The server is the entity that provides services to another entity. An example of a server is the SCSI adapter.

Management Entity

The SCB architecture requires that an entity in the base system and in each adapter provide delivery-management support for that unit. This management entity is shared by all entities in that unit. The following figure shows the unit-level management entity relative to the delivery and physical levels.

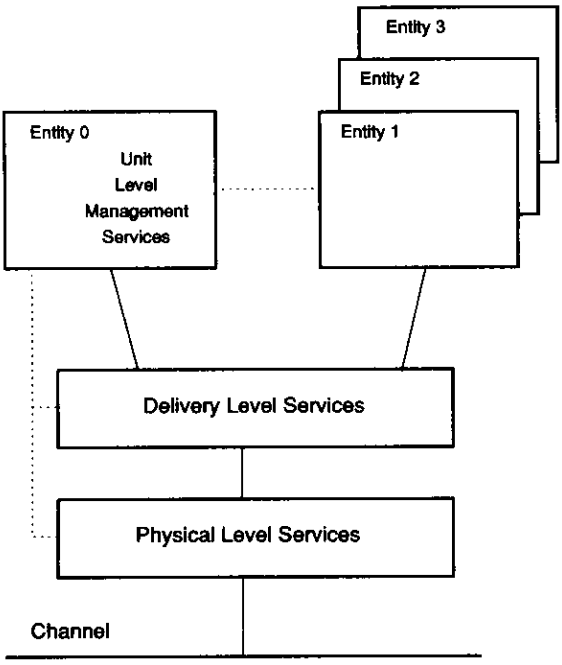

Figure 2-4. Delivery-Management Services

The SCB architecture requires that one entity in the system act as the focal point for all the unit-level management entities in the individual units. This entity is called the *system-level management entity*. The system-level management entity can physically reside in any unit (see Figure 2-5 on page 2-8).

The unit-level management entities in the individual units (system or adapter) communicate with the system-level management entity. The services provided by the delivery level are used to exchange requests and replies.

This management structure supports management services of various types and is not limited to the delivery service. In other words, the management entity can be used for entity-level reporting and testing, as well as for delivery-level and physical-level management.

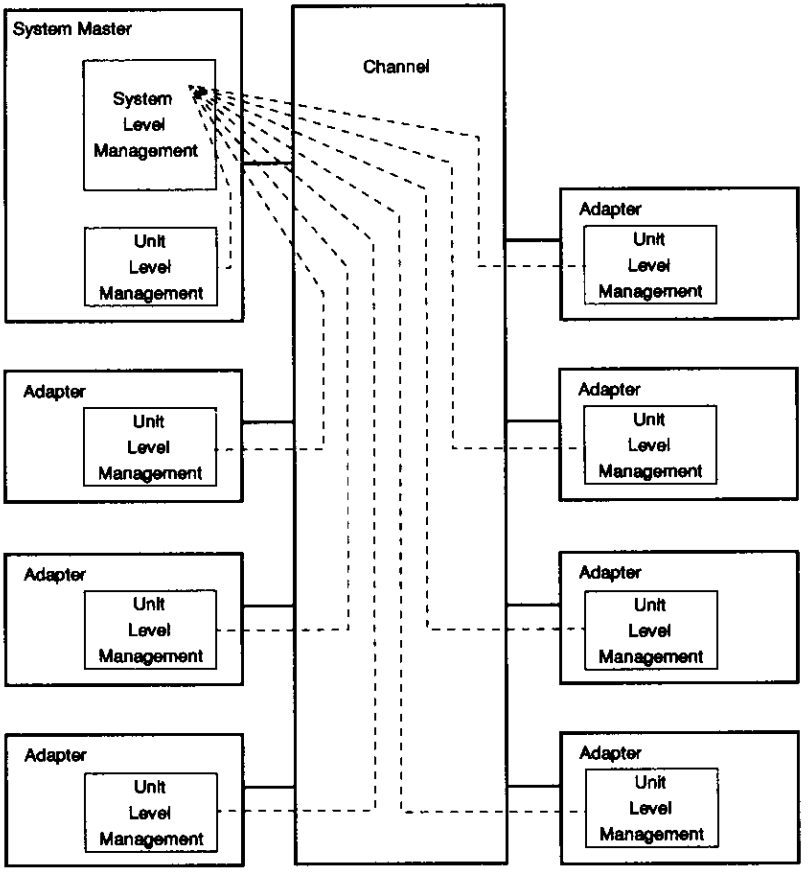

Figure 2-5. System-Management Services Structure

The management structure represents a form of client-server support between the system-level management entity and other unit-level management entities. The management services for a system are hierarchical, even though the support for entity-to-entity delivery supports peer-to-peer protocols.

Delivery-Service Structure

The SCB architecture defines protocols that support the physical delivery of command- and control-related information between a pair of entities. These delivery services are used with Micro Channel-based systems to enhance the operation of adapters operating as bus masters and to separate the physical implementations of Micro Channel procedures from the logical operations on the adapter.

The terms used in describing this architecture are similar to those in the Micro Channel architecture. The various adapters that have bus-master or I/O-slave functions are referred to as adapters, regardless of whether they are packaged as adapters that plug into a channel connector or are physically located on the system board. The system processor and its supporting logic are referred to as the system unit.

The architecture separates the delivery of control information from the delivery of data and separates the delivery of control information from the processing of control information (see Figure 2-6 on page 2-10).

The delivery service provides delivery of the control information and is used by an entity in either the system unit or an adapter to communicate the information needed to perform an operation to an entity in another adapter. These entities build and process the control information in the form of control blocks. The delivery service is responsible for delivering control blocks between cooperating entities.

The information in a control block identifies whether the operation specified in the control block also involves the transfer of data between the two entities. The delivery of data is independent of the delivery of command and control information, and the individual entities are responsible for initiating and completing its delivery.

Also, configuration information is required during system initialization to tailor the control and data-delivery services to support a particular operational environment.

The following figure shows an overview of the control and data-delivery support. It also shows that configuration information is needed to support the operations for both control and data delivery.

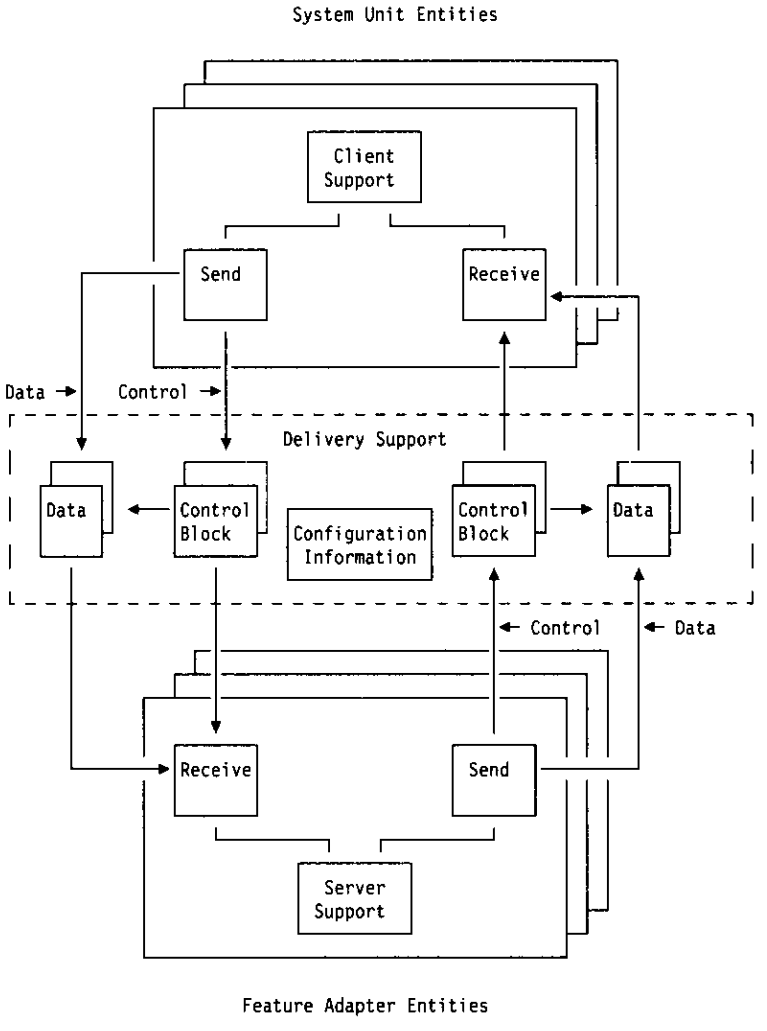

Figure 2-6. Overview of the Delivery Support

Delivery-Service Levels

The following figure shows how the delivery services in different units interact to provide the overall architecture.

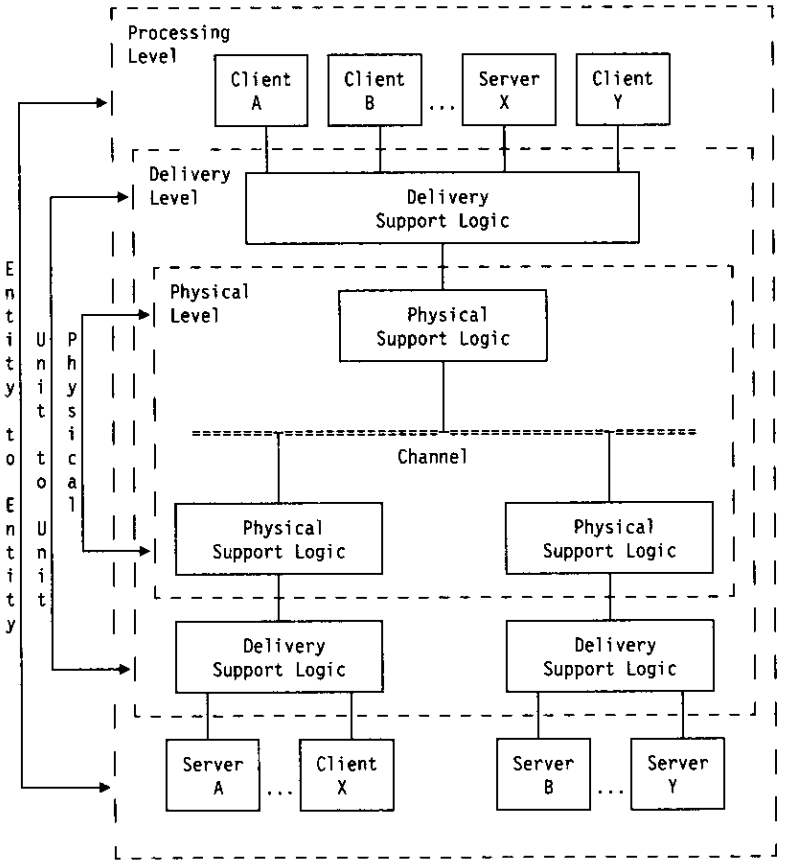

Figure 2-7. Delivery-Service Structure

Because of the structure of the delivery service, the various entities at the processing level can share a common delivery-level support. Additionally, a common delivery-level support can be mapped to different physical-level support if an adapter can be used with different system hardware.

The delivery-level protocols use the I/O address space and shared memory to pass command and control information between units.

The processing-level protocols can also use shared memory to pass data (pointed to by control blocks).

Delivery Services

In a simple form of control delivery, the control information is passed between a client in the system unit and a server in an adapter. In more sophisticated forms, the client or server can be either in the system unit or in the adapter (the client in one location always makes requests of the server in the other location).

The delivery service itself is distributed among all adapters and the system master. The part of the delivery service that is local to each is the delivery agent. The delivery agents in the various adapters communicate with each other using delivery protocols that are built on top of physical-level services, which interface with the channel. (The channel provides access to memory and I/O space in the system unit and the adapters, using various Micro Channel procedures.) A simple view of the delivery service is shown in the following figure.

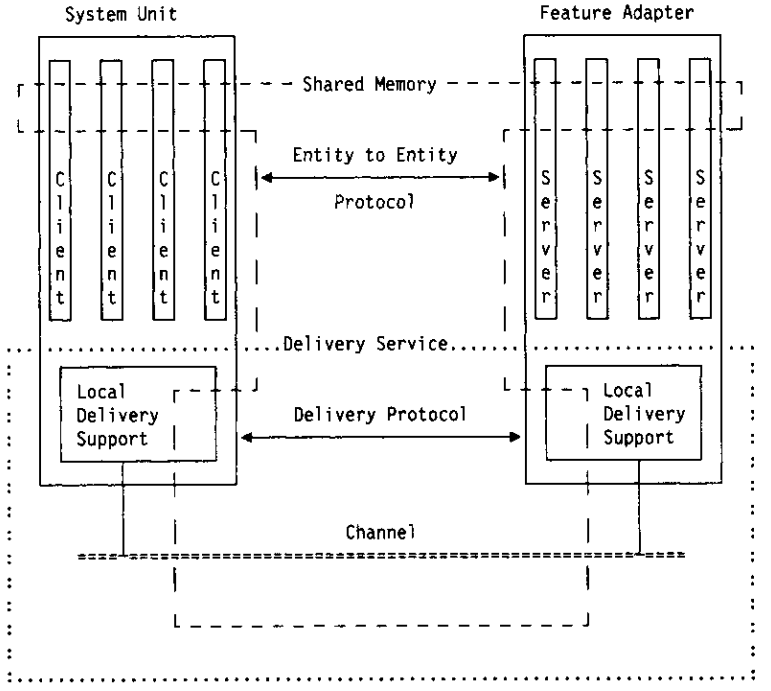

Figure 2-8. Generic Delivery Service

The delivery service supports the delivery of control blocks between pairs of entities (a client and a server). These entities build and interpret the control blocks. The entities themselves determine the actual protocol used between client and server; the overall operation at the processing level determines the services needed from the delivery level and the protocols needed to support the distribution of these services.

The delivery service is logically structured into processing-level and physical-level support. The processing-level support provides services for delivering control blocks between entity pairs. The physical-level support provides access to I/O and memory address space and provides a means for signalling between the adapters and the system unit. The physical-level support contains both support logic and Micro Channel-related hardware.

Delivery Modes

The SCB architecture defines two methods of delivering command and control information: the Locate mode and the Move mode. The following provides a brief description of the operations of each mode and the underlying control structures.

Locate Mode Delivery

In the Locate mode form of control-block delivery, the control structure is a relatively fixed format. The structure allows command and control information, status information, and pointers to data blocks to be passed from a client in the system unit to a server in the adapter. The data block pointed to by the control block can contain data or a list of data blocks (indirect list).

To send a request using control blocks, the system unit passes the physical address of the control block and a device identifier to the adapter through the control areas in I/O address space.

The operation of the Locate mode is serial and requires that the entity pairs complete one request before another request is sent to that entity. A request can be a single control block or a group of control blocks that have been chained together. The operation of the Locate mode is shown in the following figure.

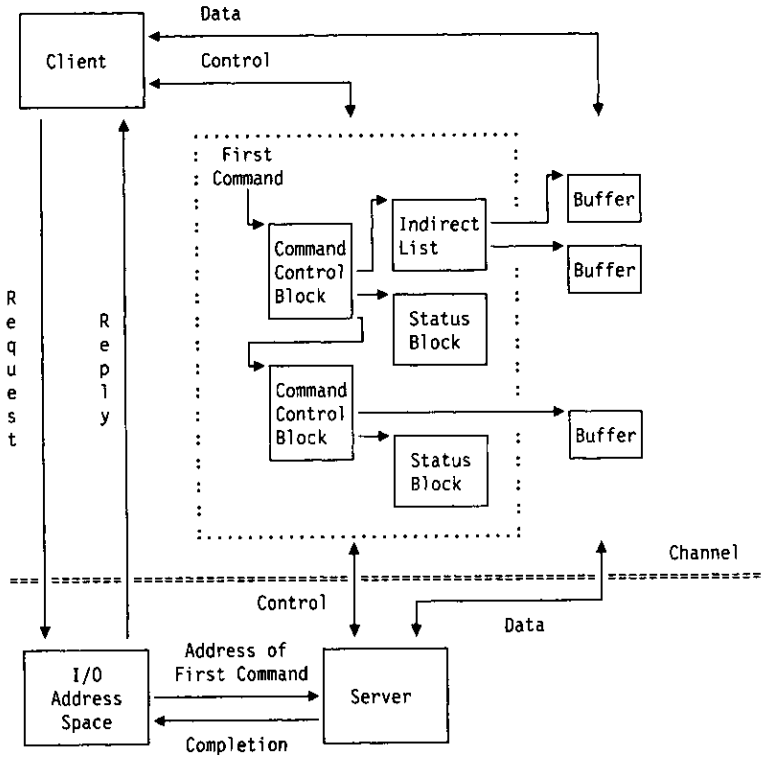

Figure 2-9. Control-Block Delivery Example — Locate Mode

Move Mode Delivery

The Move mode supports a control delivery using control elements to deliver control-related information between the client and the server. The control elements are variable in length and can contain requests, replies, error notifications, or event notifications for a specific entity or for different entities.

The entity-level operation of the Move mode is similar to that of the Locate mode (the client builds requests, requests are delivered to the server, the server builds replies, and replies are delivered to the client). However, in the Move mode, clients can be in the system unit or in adapters, which means a client in an adapter can send requests directly to a server in another adapter without having to go through the system unit. Additionally, the server can be in the system unit, which means that the system unit can receive requests from an

adapter. These capabilities provide a peer-to-peer relationship between adapters and the system unit.

The Move mode uses control elements instead of control blocks. The control element is used by the client to deliver the request, and it is used by the server to deliver the reply.

The control elements are moved between entity pairs using a pair of delivery pipes; each pipe behaves as a first-in-first-out queue. Each pipe allows for the delivery of control elements in only one direction; together, the two pipes provide full-duplex operation and allow multiple control elements to be delivered and processed asynchronously by each entity.

The operation of the Move mode is shown in the following figure.

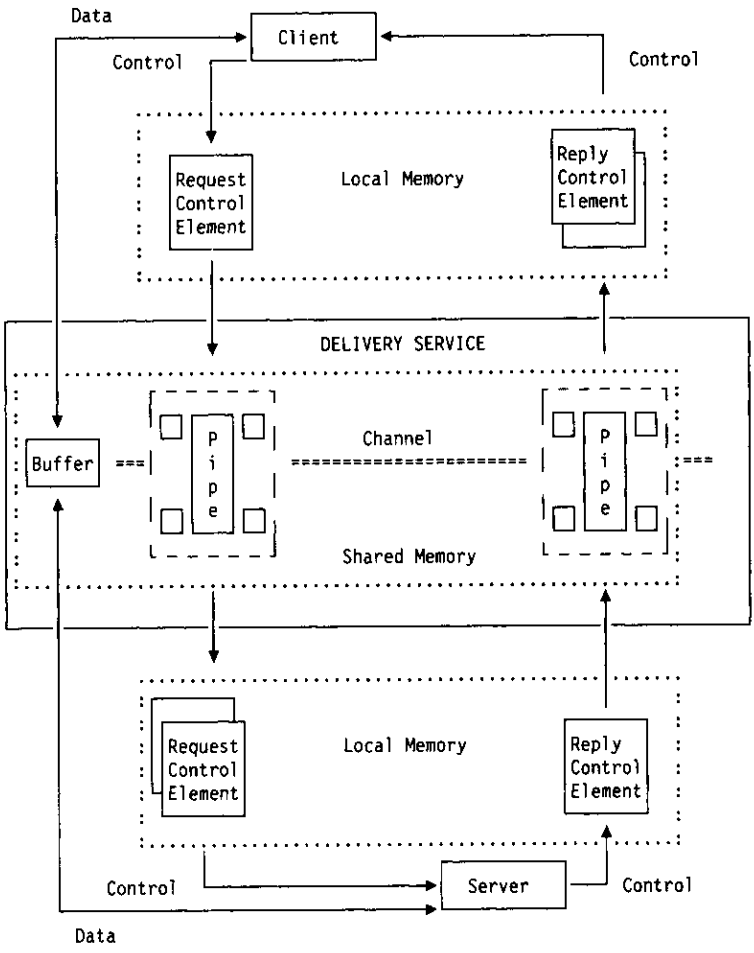

Figure 2-10. Control-Element Delivery Example — Move Mode

Control Areas

The SCB architecture defines and identifies specific control areas that are used when interfacing with the adapter or system unit.

Ideally, there should be only one physical interface used to interface with any adapter on the channel; however, conflicting factors (such as differences in expected performance, implementation cost, complexity of implementation, and ever-improving technology) lead to the need for the architecture to support alternative interfaces.

In the Locate mode, the control areas are a set of ports in the I/O address space that are used to control and report the status of the physical interface and to deliver the control and status information between entities.

The memory address-space address of the control block containing the command is passed from the entity in the system unit to the entity in the adapter, using a port in the I/O address space. The entity in the adapter uses this memory address-space address to fetch the control block and perform the indicated operation. Upon completion, the status is put into the status block, and the entity in the system unit is notified, using a port in the I/O address space.

In the Move mode, the control areas are a set of ports in the I/O address space and a pair of delivery pipes in shared memory. Shared memory is located in the memory address space. The I/O ports are used to control and report the status of the physical interface; the delivery pipes are used to deliver the control and status information.

The control elements carrying the requests and replies are moved between entity pairs, using a pair of delivery pipes maintained in shared memory. The delivery pipe behaves as a first-in-first-out queue, which makes the control elements appear to be contiguous. The pipe allows multiple control elements to be grouped into a single request.

Using delivery pipes allows multiple control elements to be delivered and processed asynchronously by each entity.

Shared Memory

The SCB architecture defines and uses areas in shared memory to maintain control structures used to support the delivery of command control information, and data between the system unit and adapters or directly between peer adapters.

The mapping and organization of shared-memory space is described in "System Resources" on page 1-9. This memory space is directly accessible by the system unit and adapters with bus-master capabilities.

In the Locate mode, shared memory is used to hold control blocks, indirect lists, termination status blocks, and data.

In the Move mode, shared memory is used as an area to deliver control elements (request, reply, error, and event) through the two delivery pipes, to manage the delivery through these pipes, and as a signalling area between the system unit and adapters. It is also used to hold data.

Structures

The Locate mode uses the command control block, the termination status block, and indirect lists; the Move mode uses the control element. All of these structures are variable-length structures in shared memory that are passed between entity pairs.

Control Blocks

Control blocks are used to pass commands and control information from a client to a server. Control blocks contain the commands and parameters to be passed to the server in an adapter. The parameters describe the particular options for the operation to be performed.

Some guidelines and rules for control blocks are:

- Control blocks must be aligned on doubleword boundaries.
- Control blocks must be placed in shared memory in the format defined for shared memory.
- Addresses in the control blocks must be physical addresses.
- Control blocks can be chained together.
- Entities need to read only those fields in the control blocks that they use.

Termination Status Blocks

Termination status blocks are used to report status information for each control block processed. The termination status block is used to report error and exception status that occurs during control-block processing. A field in the control block points to the termination status block.

Indirect Lists

An indirect list is a variable-length list that is used in the Locate mode and consists of a series of address and count fields used for data chaining. The address fields and count fields are 4-byte fields that specify the location and size of the data blocks. Both the location of the indirect list and its length are specified in the control block.

Control Elements

The Move mode uses control elements to exchange control information between a client and a server. Control elements are like control blocks; however, they differ in that:

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

The following are some rules for control elements:

- Control elements are aligned on doubleword boundaries.
- Control elements must be placed in shared memory in the format defined for shared memory.
- Addresses in the control elements must be physical addresses.

Chapter 3. Subsystem Control Block Capabilities

This section provides some criteria to help in deciding which form of the Subsystem Control Block Logical I/O architecture an implementation chooses to support.

Locate Mode

The Locate mode provides support for:

- Multiple devices per adapter.

  The adapters can support multiple entities (devices and other resources). Some examples of entities are small computer system interface (SCSI) devices, LAN connections, X.25 virtual circuits, Integrated Services Digital Network (ISDN) channels, communication lines, and processes. The Locate mode supports the delivery of requests to specific entities through a device identification number (ID).

- Adapter management.

  The adapter can perform operations that involve the function of the entire subsystem and its attached devices. The management entity is assigned device identification number 0 and receives all adapter management-type information.

- Requests to devices.

  To use a device or resource, a program (client) sends requests in the form of control blocks to the device ID that represents the device or resource (server) and receives replies for those requests.

© Copyright IBM Corp. 1991

- Command and data chaining and detailed status.

  The control structure defined for the Locate mode provides for an immediate-command type of request, a request that has one command control block and status block and a request made up of multiple command control blocks chained in a specific order and treated as one logical request. Command control blocks can point to data directly or via an indirect-list control block.

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

- Use of direct memory access (DMA).

  The Locate mode defines the interface to transfer both the control structure for a request and the data associated with the request between the system unit and an adapter. The transfer uses DMA operations managed from the feature adapter's delivery support and enity support.

- Interrupts.

  The Locate mode allows each request to define when and under what conditions interrupts are generated. During normal, error-free operation, an interrupt is optionally generated after each request is completed. Additional interrupts can be requested to synchronize the delivery of a chain of control blocks at intermediate points within the request.

  The flow of the interrupt processing in a system is from the hardware to the operating-system kernel to the delivery-interrupt handler to the delivery-request delivery support. The interrupt processing uses status information in the I/O space, in the local state control areas, and in the termination status blocks to determine the cause of the interrupt.

Move Mode

The Move mode provides support for:

- Request and reply extensions.

  Request and reply extensions to the control element provide for delivery of event and error information, in addition to delivery of request and reply information. Unlike the control elements for requests, errors, and replies, the control element for events can flow in either direction.

- Shared memory.

  Shared memory allows for the use of memory in adapters as well as memory in the system unit. This allows the control elements to be in either the system unit or the adapter and allows for more flexibility in how the control structure is built and moved to the destination entity. (The destination entity is the entity receiving the control element. The server is the destination entity for requests, and the client is the destination entity for replies.)

- Variable-length requests and replies.

  The Move mode provides a control structure that allows a request to be made up of a set of variable-length control elements. At the interface to the entity, these control elements appear to be contiguous. This allows for chaining of control elements within a request. It also allows for a minimum-size control element for passing simple requests and replies.

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

- Lists of requests and replies.

  Figure 2-10 on page 2-17 shows an example of control elements flowing from a client to a server and another flowing from a server to a client. Both flow on the shared-memory delivery pipes.

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

- Full duplex.
- Multiplexing of entity pairs.

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

  The control structure supports the intermixing of requests and replies as well as other control-element types (error and event) in the same pipe.
- Continuous running.

  The control structure permits the delivery of control elements in a continuous flow. Mechanisms are defined to provide a common way for entities to suspend the delivery of control elements at the entity-to-entity level, to notify the destination entity that one or more control elements are available, and to provide for synchronization between the entities in the source and the destination units.
- Peer-to-peer support.

  An entity in a system unit or feature adapter can send requests directly to, and receive replies directly from, any other system unit or feature adapter attached to the channel without having to go through a third party. This means an entity in a feature adapter can send requests and receive replies directly from an entity in another feature adapter without having to go through a system unit.

Selecting the Appropriate Mode

Three architecture designs are available:

- Locate mode.

  This refers to both commands and delivery as defined in the Locate mode.

- Move mode.

  This refers to control elements and delivery as defined in the Move mode. It excludes control elements used to deliver commands in the form of control blocks.

- Combined modes.

  This refers to using some of the features defined for the Locate mode in combination with features defined for the Move mode.

When to Use the Locate Mode

Locate mode commands and delivery are most useful when the following conditions exist:

- Command flow is from a single system unit.

  In the single system-unit configuration, command flow is from the system unit to the subsystem. Full-duplex operation is not needed.

- Command requests and replies do not require high speed.

  The rate at which requests are sent to the subsystem does not require high-speed transfer to maintain a timely response. The rate at which interrupts or replies are generated by devices attached to the subsystem, per command or command chain, is low to moderate.

- The command and reply format is fixed in size.

  There is no special need to deal with a wide range of command and reply sizes. Control information comes in a smaller range of sizes.

- Command chains are not modified dynamically.

  Command chains do not need to be dynamically modified as they are executed by a device, such as with the Notify and Wait facilities of the Move mode.

- The number of devices attached is small.

  The number of devices attached to the subsystem is small, typically in the range of 1 to 15 devices.

- Peer-to-peer operations are not needed.

  Requests and replies are not sent directly to another subsystem or adapter on the channel. All work requests and replies flow from the system unit to a particular subsystem.

When to Use the Move Mode

The Move mode should be selected over the Locate mode, unless compatibility with the Locate mode command set is required. This recommendation is based upon the following considerations, which were developed by contrasting the Move mode and the Locate mode:

- The Move mode is more flexible.

  The Move mode allows greater flexibility in the following areas:

  - Variable-length control elements can be defined.

    A Length field is included in control-element headers at a standard location to simplify the handling of variable-length control elements.

  - Control elements can contain immediate data.

    Control elements are defined to allow reading and writing of data from or to control elements. This is useful when small amounts of data need to be intermixed with control flows, because a separate data operation is not needed.

  - Implementations can develop their own control elements.

    Control elements have a type code, which allows an implementation to define its own control elements. By defining their own control elements, designers can match their designs to specific implementation needs.

- Servers can send multiple replies from the same request.

  A server in the Move mode can send back multiple replies or events from a single request, which is useful for operations that are performed in stages.

- Location of delivery space is flexible.

  Move mode architecture has the ability to locate the memory for delivery pipes in either the destination or source unit. This gives greater flexibility in adapter design.

- The Move mode offers better potential performance.

  Move mode architecture offers better potential performance than does Locate mode architecture, for the following reasons:

  - More parallel execution between units is allowed.

    Enqueue and dequeue functions are defined that allow requests to be queued for processing. This allows for more parallel execution between units.

  - Client serialization is not required.

    In the Move mode, a client thread does not need to serialize all access to a subsystem after placing a control element in the delivery pipe. The pipe is free flowing, and as soon as the element is placed in the pipe, the client is free to perform other work. This allows more parallel processing to occur.

  - Multiple control elements can be processed on an interrupt.

    In the Move mode, clients and servers can process control elements without requiring individual interrupts for each one. This is an important performance benefit because processing an interrupt usually carries a heavy penalty.

  - Several requests can be handled at the same time.

    A Move mode server can be defined to handle multiple work requests at the same time. The client does not have to wait for the server to complete a request before submitting other requests. Requests can be queued at the server.

  - Less data is required per control element.

    Generally, Move mode architecture requires less data to be transferred to process a unit of work. This gives better performance.

- Correlation of commands is supported.

  The Move mode provides better support for programs to correlate a reply to a specific command and program thread. This is done through a Correlation ID field, which is included in all control elements, and which can be used for programming-defined purposes. Fewer instructions are used per reply.

- Handling is expedited.

  The Move mode provides an expedited-handling flag in control elements. This allows an implementation to get performance- or response-critical elements to the attention of the server.

- Control-element execution is optimized.

  A Move mode server is allowed to optimize the processing order of control elements outside of command chains. This gives a potential for improved performance.

- Full-duplex operation is allowed.

  The Move mode entities can process inbound and outbound requests independently. This gives a greater potential for parallel execution.

- Real-time command chains are allowed.

  The Move mode use of notify and wait flags allows the programming of command chains that are more real-time responsive. This improves the response time of a system using the Move mode.

• The Move mode provides improved peer support.

  Move mode architecture supports improved peer capability when compared to the Locate mode architecture, for the following reasons:

- The Move mode supports replies to peer sources.

  Move mode control elements contain the source address of the peer unit in the control-element header. This allows a reply to be sent directly to the peer unit that originated the request.

- The Move mode supports requests and replies on a single pipe.

  In the Move mode, a single delivery pipe can contain request and reply control elements. This is needed in a peer-to-peer environment where work flows depend on transaction type. That is, a unit can be a server for one form of work request, and it can be a client for others.

When to Use the Combined Modes

No detailed description of combined modes is presented in this book. Each implementation must define an appropriate combination of Locate mode and Move mode features.

Modes can be combined, for example, for upward compatibility from adapters that implement Locate mode commands and wish to provide an implementation bridge so that Locate mode commands can be used in a better, more responsive way. This is done using Move mode command delivery with the Locate mode command set. In this case, the following improvements are realized:

- Performance is improved over the Locate mode, for the following reasons:
  - Less serialization is needed at the client.

    Less synchronization is needed at the client to send a request to a subsystem than is required in the pure Locate mode. The busy time in the sending unit is reduced using Move mode delivery in the combined modes.
  - Less serialization occurs on replies.

    The use of a Move mode delivery pipe instead of a single Interrupt Status port means that less serialization of reply processing occurs at both the client and the server. This means more parallel execution is possible.

- More data is given with replies.

  The use of Move mode delivery means that a correlation ID, as well as the physical address of a control-block command can be supplied with each reply. The additional data makes it easier to determine which specific command and execution thread should handle the reply. Fewer instructions per reply means better performance.

- Control-element execution is optimized.

  A server using Move mode delivery is allowed to optimize the processing order of control elements outside of command chains. This gives a potential for improved performance.

- Dequeue synchronization with enqueue is minimized.

  An entity using Move mode delivery dequeues control elements from a delivery pipe with a minimum amount of synchronization between itself and entities placing control elements in the pipe. This allows for more parallel execution between units.

- Multiple control elements can be processed on an interrupt.

  In Move mode delivery, clients and servers can process control elements without requiring individual interrupts for each one. This is an important performance benefit because processing an interrupt usually carries a heavy penalty, especially in processors that are pipelined.

- Handling is expedited.

  Move mode delivery provides an expedited-handling flag in control elements. This allows an implementation to get performance- or response-critical elements to the attention of the server.

- Full-duplex operation is allowed.

  Entities using Move mode delivery can process inbound and outbound requests independently. This gives greater potential for parallel execution.

- Real-time command chains are allowed.

  The use of notify and wait flags in control elements allows the programming of command chains that are more real-time responsive. This improves the response times of a system using Move mode delivery.

- Functionality might be improved in the following areas:
  - Peer support is better.

    An adapter is able to serve multiple clients without going through a single system unit because of the peer nature of the Move mode delivery service. This can be a performance boost in a LAN bridge or multimedia system where DASD data, for example, needs to be processed by a peer adapter and does not need to be processed by a system unit.

  - Expansion is easier.

    It would be easier to support the command queuing proposed by the SCSI II standard, given the free-running nature and additional data provided by the Move mode delivery service.

Notes:

Index

© Copyright IBM Corp. 1991