SCB Architecture - Locate Mode

Insert the hardtab labeled "Locate Mode" here and discard this page.

Locate Mode

Figures

Chapter 1. Overview

The Locate mode defines a control structure that allows the coupling of commands, status, and data to represent a request from a client to a server. A client delivers requests, one at a time, in one of two forms: as an immediate command or as a command control block. A client initiates a request by passing either an immediate command or the address of a command control block to a server through a set of I/O ports in I/O address space. The following figure shows an example of the structure of control-block delivery.

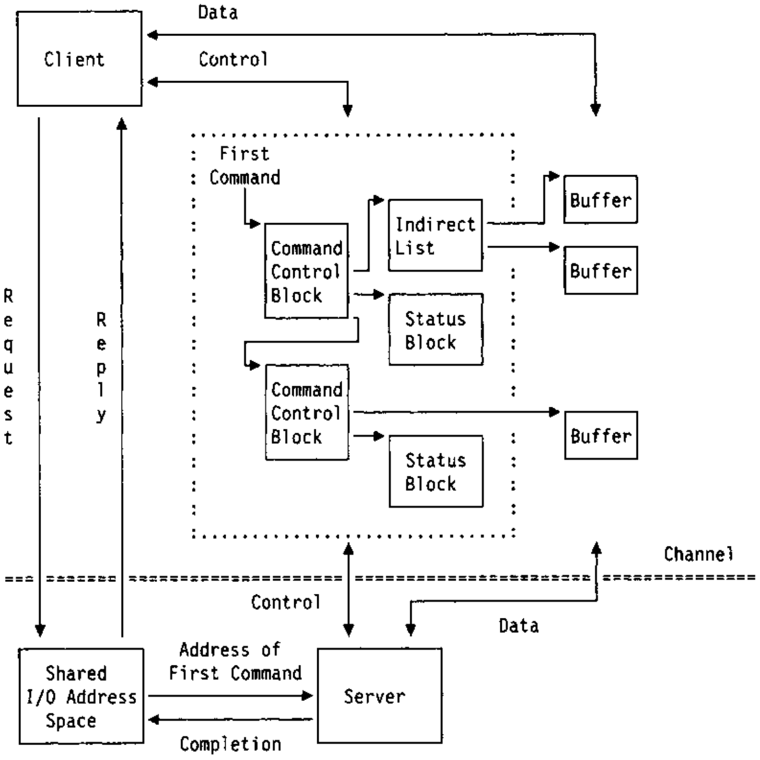


Figure 1-1. Control-Block Delivery Structure Example

A server uses the information passed in the I/O ports to determine whether the request is an immediate command or the address of a control block containing a command.

If the request is in the form of an immediate command, the server uses the information contained in the set of I/O ports directly and returns a reply to the client through the same I/O ports and an interrupt. If the request is in the form of a command-control-block address, the server uses the information contained in the I/O ports to fetch the command control block. Upon completion of the command, the server puts the completion status into the status block and returns a reply to the client, using the I/O ports and an interrupt.

The following information explains the operational characteristics and the flow of the delivery support. This information is necessary to understand the overall operation of the Locate mode.

Operational Characteristics

The operational characteristics of the Locate mode are as follows:

- Unidirectional delivery.

  The flow of requests is from a client in a system master to a server in a subsystem. The server cannot send requests to a client.

- Multiple devices attached to a subsystem.

  A subsystem can support more than one device attached to it. Requests are directed to a specific entity, which supports an attached device. All entities within a subsystem use a single set of delivery resources in the subsystem to communicate with the system master.

- Shared delivery control areas.

  A subsystem has a single set of control areas in I/O space that are used by the system master to send requests to any of its devices and to receive status from any of its devices. The Command port and the Attention port are used to deliver requests. Sending a request to a device on the subsystem makes these shared delivery-control areas busy until the request is accepted by the subsystem. The busy indication is provided in the Command Busy/Status port. A rejected request causes the subsystem delivery space to be put into the reject state. In the reject state, the system-master delivery support needs to issue a special subsystem command to clear it before additional requests can be sent to any device on the subsystem.

  Delivery support in the system manages the subsystem's control areas as a set of serially-reusable resources used for the delivery of one request at a time.

- Device use.

  A device executing a request does not accept a new request until any current request, other than an immediate command such as the Reset command or the Suspend command, has been completed. Once a request has started, a device indicates busy-reject status if it receives another request (unless it has been issued a Suspend or Reset command) and puts the subsystem's delivery space into a reject state. To avoid this, the entity in the system master must manage devices on subsystems as serially-reusable resources. Interrupts sent by a completed (or staged) request can be used to pace requests to a device.

- Signalling of request or command completion through interrupts.

  After a request or a particular command is completed by the device or subsystem, the subsystem optionally signals the system master of the completion by presenting an interrupt request to the system master. Interrupt status from the subsystem is stored in the Interrupt Status port, which supports only one interrupt at a time. When an interrupt is presented to the system master, another interrupt cannot be presented until the Interrupt Status port is cleared with a command from the system master. Other devices must enqueue their interrupts within the subsystem until the port is cleared.

- Handling interrupts from an adapter.

  Request-delivery support in the subsystem services requests to all of the devices attached to the subsystem. Interrupts from the devices are handled by a single shared-interrupt-request handler and serialized through a single set of subsystem interrupt resources.

  The system interrupt-handling logic that receives the interrupt from the subsystem interrupt handler must determine which instance of request-delivery support in the system should be notified of the interrupt. This can be done in the system by using a layered approach to interrupt handling consisting of shared first-level support and instances of second-level support that are unique to request delivery for a specific device.

  First-level support determines which subsystem has an interrupt. If an interrupt is present, it clears the interrupt and notifies the instance of the request delivery, which is waiting for an interrupt from this device. It also disables any further interrupts from this device until the device is enabled by the second-level support. The delivery support uses certain immediate commands to disable and enable device-level interrupts.

  For certain versions of Locate mode operation, the interrupt can indicate multiple device-level interrupts through the Device Interrupt Identifier port. For these, the first-level support notifies each instance that has a device-level interrupt.

The instance of the request delivery (the second level of support) typically has its execution blocked and is waiting for an interrupt from a device to which it has issued a request. When the second-level support has been notified, the first-level support is free to service another interrupt from a subsystem.

Second-level support in a request-delivery instance handles request-completion interrupts from the specific device attached to a subsystem. It determines which command control blocks have been completed and returns this information to the send interface.

Command and Control-Block Delivery

The following figure shows the flow of requests from the system master to a subsystem.

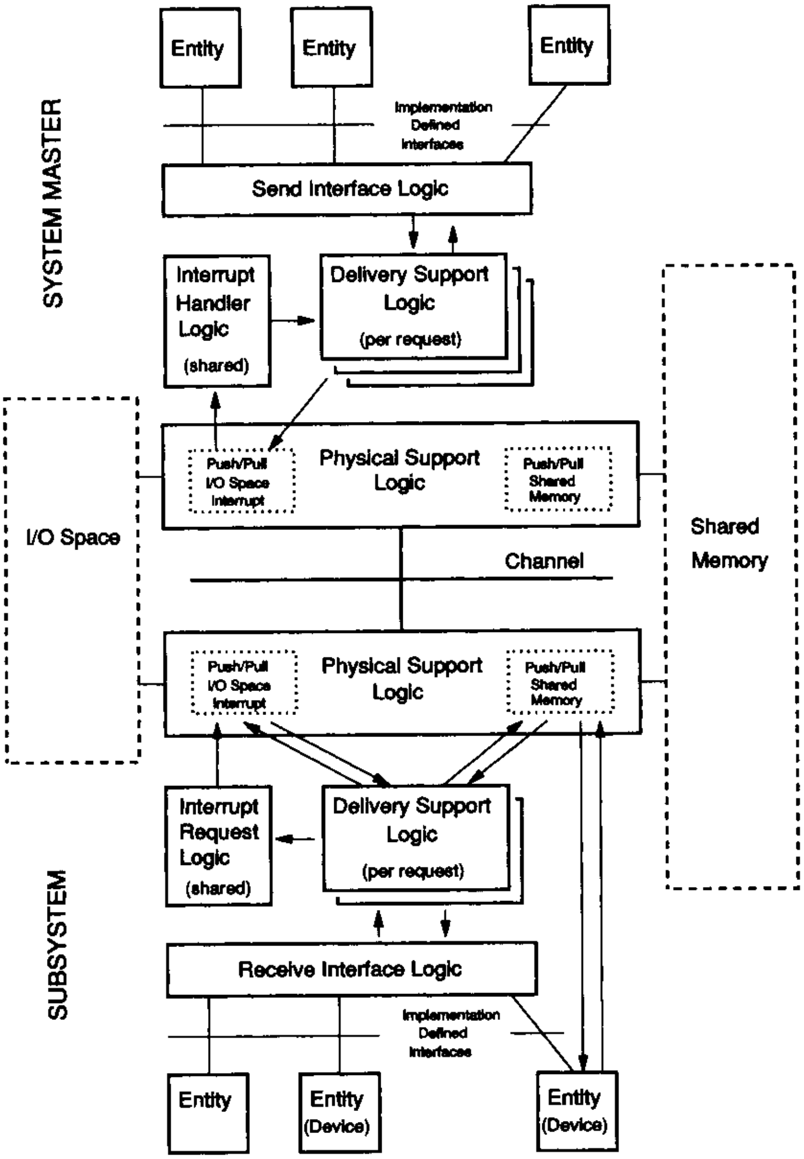

SYSTEM MASTER
Entity
Entity
Entity
Implementation Defined Interfaces
Send Interface Logic
Interrupt Handler Logic (shared)
Delivery Support Logic (per request)
I/O Space
Push/Pull I/O Space Interrupt
Physical Support Logic
Push/Pull Shared Memory
Shared Memory
Channel
Push/Pull I/O Space Interrupt
Physical Support Logic
Push/Pull Shared Memory
Interrupt Request Logic (shared)
Delivery Support Logic (per request)
SUBSYSTEM
Receive Interface Logic
Implementation Defined Interfaces
Entity
Entity (Device)
Entity (Device)

Figure 1-2. Request-Delivery Flow

The following information describes the flow through the various parts of a client in the system master sending a request to a server in a subsystem.

1. The send interface handles the translation of the local form of send, and its parameters, from the client into the request control block, as seen in shared memory.
2. The request delivery has an instance for each entity sending a request to an entity in the subsystem. The following functions are provided by the request-delivery support:
   - Management of access to the control areas in I/O address space shared among all request-delivery flows to the same subsystem
   - Coordination of the request flow with interrupt handling for a particular subsystem or device.
3. After the request delivery obtains access to control areas, it performs the read and write operations to the control areas to pass control parameters to the request delivery in the subsystem.
4. When all parameters have been written to the control areas, the request delivery writes a specific value to the Attention port, causing an interrupt of the subsystem.
5. When this interrupt occurs in the subsystem, the request-delivery support in the subsystem passes the request to the processing-level support, through the receive interface.
6. When the processing level completes the request, it can request that the subsystem schedule an interrupt to the system master.
7. When the system-master interrupt occurs, the interrupt handler in the system master determines which subsystem and device are interrupting and notifies the corresponding request-delivery instance.
8. The request-delivery support determines the completion status of the request and returns that information to the send interface.
9. After being notified that the request was completed, the client can issue the next request or continue to process the notification for other control blocks in a chain.

Because multiple subsystems of the same type or of different types can be used in the system, the base address for the I/O space of each subsystem must be defined during configuration. The I/O ports used by a Locate mode type of subsystem are defined in "I/O Address Space" on page 1-15.

The following are general descriptions of the control areas used by Locate mode delivery. A port is a byte or set of contiguous bytes in I/O space.

- Subsystem Control port

  The Subsystem Control port is used to directly control the subsystem-to-system interface.

- Request ports

  There are five ports that are used in the delivery of requests to a device and the delivery of replies (interrupts) from a device.

  - The Command port is used to pass either the address of a control block or an immediate command.
  - The Attention port contains an attention code and a device identifier. The attention code informs the subsystem that the Command port contains either the address of a control block or an immediate command. The device identifier indicates which device on the subsystem the request is directed to.
  - The Interrupt Status port is used to identify the interrupting device and to indicate any exception that occurred during command processing.
  - The Command Busy/Status port is used in the subsystem to serialize access to the shared logic of the control-block delivery service. This port contains the following indicators:
    - Busy – indicates that the subsystem is busy (using the shared logic) and will therefore ignore submitted commands.
    - Interrupt Valid – indicates that the contents of the Interrupt Status port are valid and that the subsystem has an interrupt pending.
    - Reject – indicates that the subsystem has rejected a request (a Reset Reject command is needed to clear a reject and allow the subsystem to resume accepting requests).
    - Status – indicates the reason for the reject.

- The Device Interrupt Identifier port is an optional method of reporting control-block interrupts. When this port is used, only the interrupt status for immediate commands is reported through the Interrupt Status port. All other interrupts are reported through the Device Interrupt Identifier port.

  This optional port allows the interrupt handler to process multiple control-block interrupts through a single system interrupt. For more information on interrupt handling, refer to "Signalling Protocol" on page 3-13.

Control-Block Structure

The following figure shows the structure and content of a typical command control block. There are two formats for command control blocks: base and extended. Both formats contain all the fields shown in solid lines. The remaining fields (shown in dashed lines) are present only in the extended control-block structure. The device-dependent area is in both formats; however, the actual location of the device-dependent area within each control block depends on whether the base or the extended control block is used.

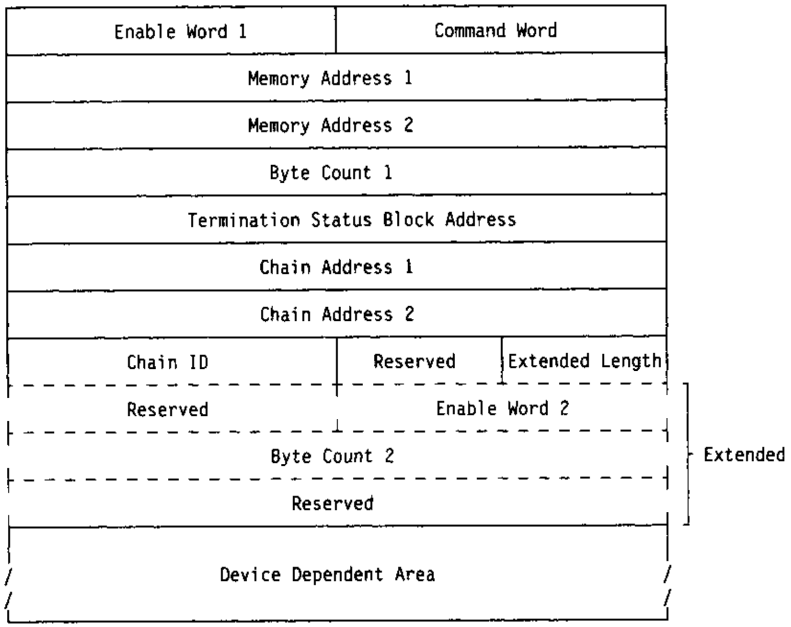


Figure 1-3. Control Block Format

Termination Status Block Structure

In addition to the completion indication in the Interrupt Status port, status information can be reported for each command control block, using the termination status block. A termination status block is always stored if an exception occurs during control-block processing; optionally, it can be stored for each control block processed. A field in the command control block points to the area in shared memory where the termination status block is stored. The format of the termination status block is shown in the following figure.

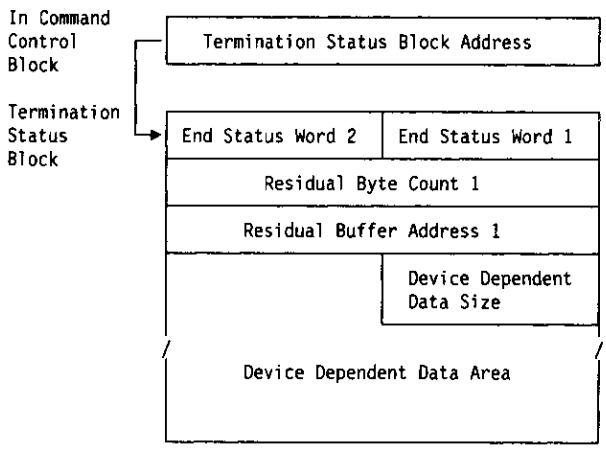

In Command Control Block
Termination Status Block Address
Termination Status Block
End Status Word 2
End Status Word 1
Residual Byte Count 1
Residual Buffer Address 1
Device Dependent Data Size
Device Dependent Data Area

Figure 1-4. Termination Status Block Format

Indirect-List Structure

An indirect list is a variable-length list consisting of the address-count pairs used to support data chaining. Both the location of the indirect list and its length are specified in the control block. Within the indirect list, the Address field and the Count field are each doubleword fields. The format and content of an indirect list is shown in the following figure.

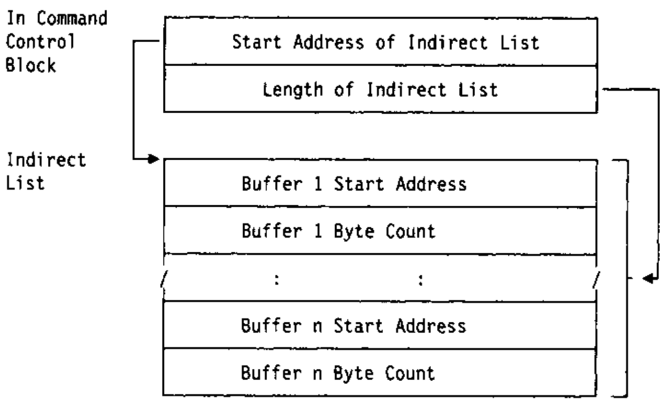


Figure 1-5. Indirect-List Format

Delivery Structure

The following figure shows how the various levels relate to each other and to the control areas in I/O address space and control blocks in shared memory. It also shows the send and receive interfaces between the delivery level and the entity level. There is no unit-to-unit protocol associated with this support.

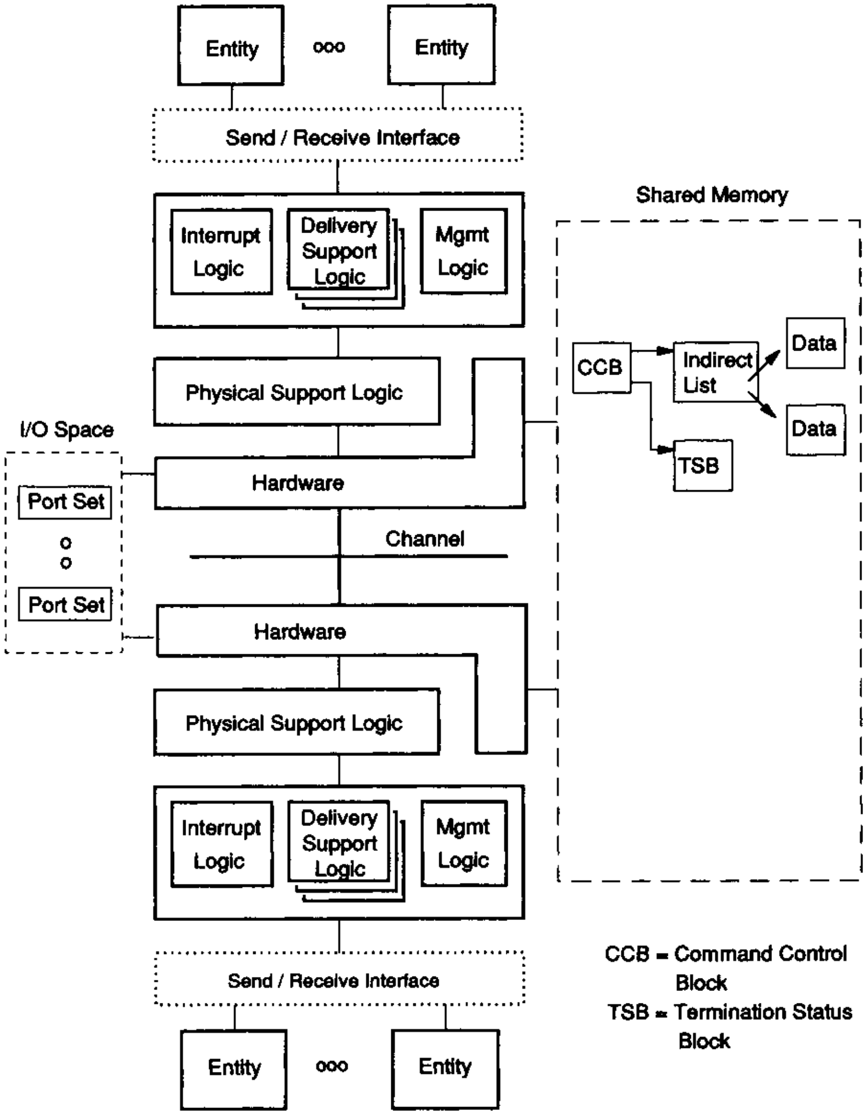


Figure 1-6. Locate Mode Delivery Structure

I/O Address Space

The SCB architecture defines support for two different physical interfaces: Type 1 and Type 2. The registers used in these interfaces perform the same functions; the difference between the interfaces is the location and size of required ports.

The interface used by a subsystem is determined by the number of physical devices that can be attached to it. When 16 or more devices are supported, the Type 2 interface is used.

Physical Interface Description

Both Type 1 and Type 2 physical interfaces use the following control areas in I/O address space:

- Command port
- Attention port
- Subsystem Control port
- Interrupt Status port
- Command Busy/Status port
- Device Interrupt Identifier port.

Because multiple subsystems of the same type can be present on the channel, the addresses within the I/O address space are shown as offsets from the base I/O address. The base I/O address used by each subsystem is determined during system configuration.

The following figure shows the Type 1 interface.

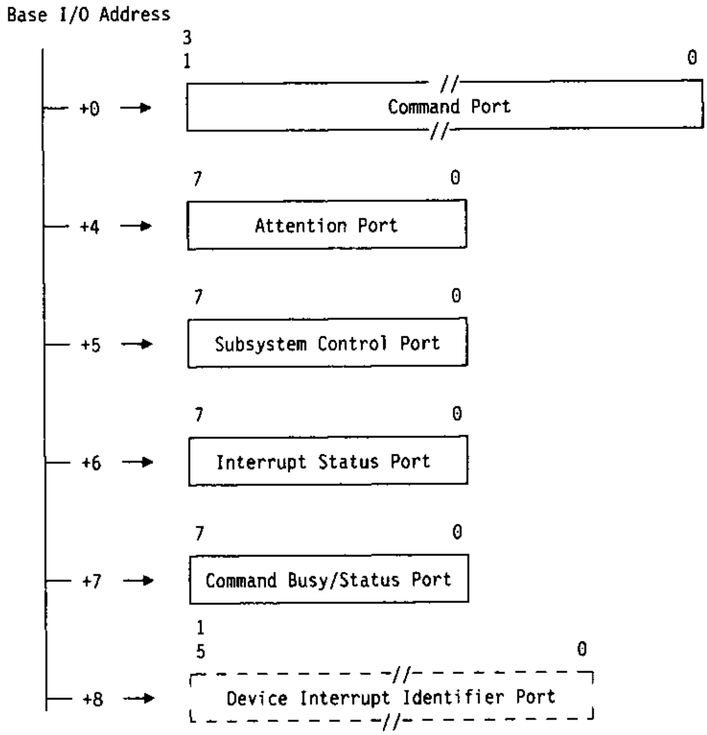


Figure 1-7. I/O Address Map – Type 1

The following figure shows the Type 2 interface.

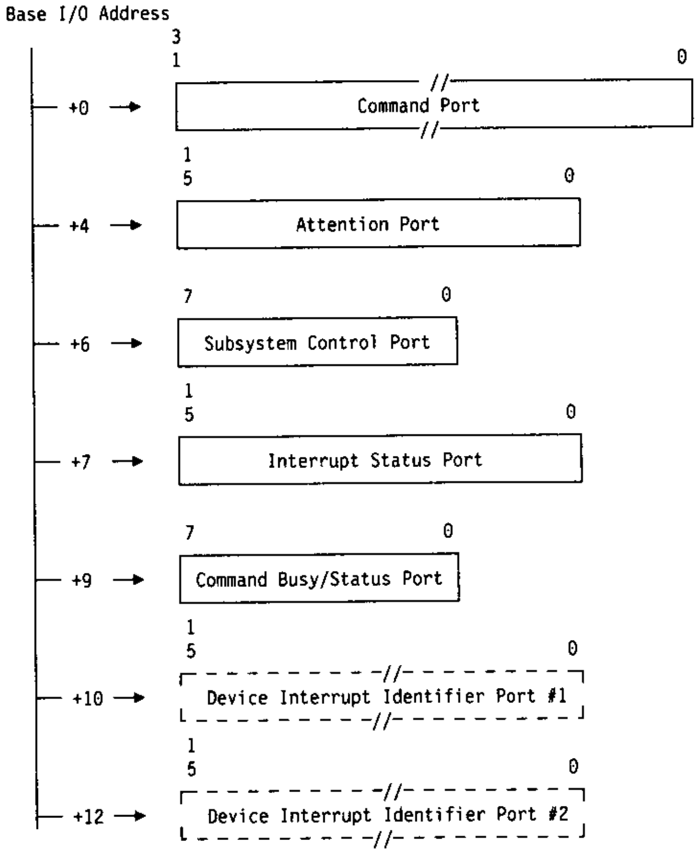


Figure 1-8. I/O Address Map – Type 2

Port Descriptions

The following information describes the ports for the Type 1 and Type 2 interfaces.

Command Port

This 32-bit port is used to deliver a 32-bit immediate command or the physical address of a control block to a subsystem. The immediate command, or the control block pointed to by the address in the Command port, specifies an operation to be performed by the subsystem or the device.

The format and content of the Command port are shown in the following figure.

```
3
1                                                          0
+----------------------------------------------------------+
|        Immediate Command or Address of Control Block     |
+----------------------------------------------------------+
```

Figure 1-9. Command Port

Attention Port

This port signals the subsystem or device associated with the subsystem. The signal can interrupt and notify the subsystem that the Command port contains either an immediate command or the address of a control block. This port is also used to reset an interrupt and can be used to issue a software reset.

The size of the Attention port is determined by the number of devices supported; however, the two fields that make up this port serve the same function, regardless of the number of devices. The Attention port is an 8-bit port if there are fewer than 16 devices supported by the subsystem; it is a 16-bit port if 16 or more devices are supported.

Fewer than 16 Devices

```
   7              4    3               0
  +-----------------+-------------------+
  | Attention Code  | Device Number     |
  +-----------------+-------------------+
```

16 Devices or More

```
   15            12 11                         0
  +-----------------+----------------------------+
  | Attention Code  |      Device Number         |
  +-----------------+----------------------------+
```

Figure 1-10. Attention Port

Attention Code Field: The attention codes identify the specific operations for a device. These codes typically specify an operation that is a single-purpose signal to the device. For example, attention code hex 1 specifies that the Command port is loaded with an immediate command, and attention code hex 3 specifies that the port contains the address of a control block. The following is a summary of the valid attention codes, their names, and a brief description of each.

Code	Name	Description
0	Reset command	Requests the subsystem to perform a device reset for the specified device.
1	Immediate command	Requests the subsystem to execute the command contained in the Command port.
2	Reserved	
3	Start Control Block command	Requests the subsystem to process the control block pointed to by the address in the Command port.
4	Device dependent	See note.
5-C	Reserved	
D	Move mode delivery	Used to signal a request for Move mode command delivery.
E	End of interrupt	Requests the subsystem to perform one of the two interrupt-resetting commands. This is a hybrid command that requires both the subsystem hardware and software to clear the interrupts presented to the system unit.
F	Device dependent	See note.

Note: The device-dependent codes should not duplicate a function that is already defined by the architecture. Although the command or control block associated with these attention codes is device dependent, implementers must not redefine the basic architecture.

Figure 1-11. Attention Codes

Device Number Field: This field specifies which device receives the attention code. The Type 1 interface can select one of 16 devices from 0 to 15; the Type 2 interface can select one of 4096 devices from 0 to 4095. Device 0 is reserved for the subsystem and is used to direct commands or control blocks to the subsystem, rather than to any attached device, allowing the subsystem to act as the focal point for all devices.

Subsystem Control Port

This 8-bit port provides direct control of the subsystem hardware. The following is a description of the Subsystem Control port.

7	6	5	4	3	2	1	0
RST	R	RR	R	DD	DD	DMA	EI

Figure 1-12. Subsystem Control Port

Bit 7	The subsystem-reset (RST) bit provides a hardware-controlled reset of the subsystem and all attached devices. The bit does not affect programmable-option-select data. See "Reset Subsystem (Hardware Controlled)" on page 1-90 for more details on the use of the subsystem-reset bit.
Bit 6	This bit is reserved.
Bit 5	The reset-reject (RR) bit restores the subsystem to a state in which it can receive commands. When this bit is set to 1, the reject state is cleared, and the busy bit and the reject bit in the Command Busy/Status port are reset to 0. If the subsystem is not in the reject state, setting this bit to 1 has no effect. The reject state is always cleared after a reset of the subsystem or after a power-on reset. (See "Command Busy/Status Port" on page 1-24 and "Delivery-Level Protocols" on page 3-10 for more details on the reject state.)
Bit 4	This bit is reserved.
Bits 3, 2	These device-dependent bits can be used by subsystems to meet their specific needs. If the bits are used, they must be defined in the technical documentation for the specific subsystem.

Bit 1 The enable-DMA (DMA) bit controls enabling and disabling of the DMA operations. When this bit is set to 1, DMA operations are enabled. When this bit is set to 0, DMA operations are disabled. DMA operations are always disabled after a reset of the subsystem or after a power-on reset.

Bit 0 The enable-subsystem-interrupts (EI) bit controls physical interrupts to the system master that are generated by the subsystem. When this bit is set to 1, the interrupts are enabled. When this bit is set to 0, the interrupts are disabled. Interrupts are always disabled after a reset of the subsystem or after a power-on reset. (See "Signalling Protocol" on page 3-13 for a description of interrupts.)

Interrupt Status Port

This port is used by a subsystem to present interrupt information for the subsystem or device to the system master. It can be an 8-bit port or a 16-bit port, depending on the number of devices supported by the subsystem.

Because this port can contain interrupt information for only one command at a time, an interrupt for another command cannot be presented to the system master through the Interrupt Status port until the previous interrupt has been reset by the Reset Interrupt Status Port command.

When a subsystem does not support the Device Interrupt Identifier ports, all interrupt information is presented through this port. When the Device Interrupt Identifier port is supported, only the interrupt information for immediate commands or for commands with hardware failures are presented through this port.

When the Device Interrupt Identifier port is supported, interrupt information for control block commands is presented in the Device Interrupt Identifier ports and in the termination status block associated with each control block. See "Device Interrupt Identifier Port" on page 1-28 for more information.

The format and content of the Interrupt Status port are shown in the following figure, followed by a description of the individual fields.

```
Fewer than 16 Devices
           7     4 3     0
          ┌───────┬───────┐
          │  ID   │Device │
          └───────┴───────┘

16 Devices or More
          1   1 1
          5   2 1              0
          ┌────┬───────────────┐
          │ ID │    Device     │
          └────┴───────────────┘
```

Figure 1-13. Interrupt Status Port

Interrupt Identifier Field: The Interrupt Identifier (ID) field is encoded with a value that identifies the cause of the interrupt. If the interrupt is the result of a control-block command, more detailed information on the cause is available in the termination status block for that control block.

The following figure contains a summary of all the valid interrupt-identifier codes.

Hex	Interrupt Definition
0	Reset completed without error
1	Control-block command completed without error
2–4	Reserved
5	Device dependent
6	Inform interrupt
7	Hardware failure – immediate command
8	Hardware failure – control-block command
9	Reserved
A	Immediate command completed without error
B	Reserved
C	Control-block command completed with error
D	Immediate command completed with error
E	Reserved
F	Device dependent

Figure 1-14. Interrupt-Identifier Codes

The following is a description of all the valid interrupt-identifier codes.

Code	Definition
0	The subsystem or device has completed a reset operation without error.
1	The device or subsystem has completed the control-block operation without error. If the specific control block enables storing of the termination status block, additional status is stored in the termination status block. This code is not used when the Device Interrupt Identifier port is supported.
2 – 4	These codes are reserved.
5	This code is device dependent.
6	This code can be used to indicate a condition that is not associated with any specific command.
7	A hardware failure occurred that prohibits the completion of the immediate command or hardware-controlled command.
8	While a control-block command was being processed, a hardware failure caused one of the following: • The subsystem or device attempted to read the control block, and the read operation failed. • The subsystem attempted to store a termination status block, and the store operation failed. If the condition occurs while the device is processing a chained control block, the chain is terminated. If the hardware failure does not prohibit the device from responding to commands, the device can store command-completion status internally. The contents of the termination status block are not valid; however, additional status can be obtained with a Read Completion Status command, depending upon the specific hardware failure.
9	This code is reserved.
A	The subsystem has completed the immediate or hardware-controlled command without an error.
B	This code is reserved.

C A specific control block has been completed, and an error occurred. The subsystem or device has stored the status. See "Termination Status Blocks" on page 1-43 for details on the termination status block and "Signalling Protocol" on page 3-13 for details on interrupt status and completion.

This interrupt-identifier code is not used when the Device Interrupt Identifier port is supported.

D An immediate or hardware-controlled command was completed with a nonhardware failure, or the device failed the diagnostic test. (See "Run Immediate Diagnostic Test" on page 1-87.)

If the device failed the diagnostic test, it can provide additional information about the reason for the failure through a Read Completion Status command or a Run Diagnostic Control Block command.

The subsystem or device might need to be reset before any diagnostic data can be obtained by a Run Immediate Diagnostic Test command.

E This code is reserved.

F This code is device dependent.

Detailed explanations of interrupt handling are provided in "Signalling Protocol" on page 3-13.

Device Number Field: The Device Number (Device) field identifies the device that presented the interrupt status.

Command Busy/Status Port

This 8-bit port has two functions:

- After an immediate command or command-control-block address is submitted, the Command Busy/Status port is read to determine the status of the command or control block. The system must allow the subsystem a predefined period of time before attempting to read this port.
- It also indicates whether the subsystem has a valid interrupt value present in the Interrupt Status port. The Interrupt Status port should not be read unless the Command Busy/Status port indicates that the interrupt is valid.

The Command Busy/Status port is shown in the following figure and is followed by a description of the individual bits.

7 5	4	3	2	1	0
Status	REJ	DD	DD	IV	B

Figure 1-15. Command Busy/Status Port

Bits 7 – 5 The status bits indicate the reason that the command or control block has been rejected. If the reject bit is 0, the status bits are undefined and should be ignored. (See "Delivery-Level Protocols" on page 3-10 for more details on command and control-block submission.)

The encoded values and meanings of the status bits are shown in the following table.

Code	Meaning
0 0 0	**Reserved**
0 0 1	**Device Not Available** The subsystem determined that a device cannot perform the command or process the control block. The condition is cleared by the Reset Device or Reset Subsystem command.
0 1 0	**Invalid Command** The subsystem does not recognize an attention code in the Attention port, or an immediate command is not a valid command.
0 1 1	**Device Busy** A device is busy executing a command or processing a control block and cannot accept a new command or control block.
1 0 0	**Device Control-Block Execution Suspended** The device has been put into a suspended state. It can be reset by the Resume, Reset Device, or Reset Subsystem command.
1 0 1	**Invalid Device Number** The Device Number field of the Attention port contains an invalid value.
1 1 0	**Reserved**
1 1 1	**Device Limits Reached** The device's internal capacity to accept another command has reached its limit. The condition will be cleared as the device completes the execution of commands and the system accepts and resets interrupts from the device. If the condition persists, it can also be cleared by a Reset Device or a Reset Subsystem command. (See "Signalling Protocol" on page 3-13 for details.)

Figure 1-16. Command Busy/Status Port – Status Bit Encoding

Bit 4 The reject (REJ) bit indicates that the subsystem has rejected a command or control block submitted through the Command ports and Attention ports. The reason is encoded in the status bits in the Command Busy/Status port. See Figure 1-17 on page 1-27 for details about this bit.

Bits 3, 2 These bits are device dependent.

Bit 1 The interrupt-valid (IV) bit is set to 1 whenever the subsystem writes an interrupt value into the Interrupt Status port. When this bit is set to 1, the Interrupt Status port contains a valid interrupt identifier and device number. When this bit is set to 0, the Interrupt Status port does not contain a valid interrupt value. (For details on interrupts, see "Signalling Protocol" on page 3-13.)

Bit 0 The busy (B) bit indicates whether the Command port and Attention port are currently being used or whether a hardware-controlled reset is in progress. The following table shows encoding of the busy bit and the reject bit.

Busy Bit	Reject Bit	Description
0	0	Writing to the Command port and Attention port is not blocked.
0	1	Reserved.
1	0	Writing to the Command port and Attention port is blocked. The maximum time for this condition is specified in the configuration data.
1	1	Writing to the Command port and Attention port is blocked because the previous command has been rejected. This condition is cleared by setting the reset-reject bit in the Subsystem Control port to 1.

Figure 1-17. Busy-Bit And Reject-Bit Settings

Device Interrupt Identifier Port

This optional port indicates which devices have control-block logical interrupts pending. When a bit is set to 1, it indicates that the device associated with the bit has at least one control-block logical interrupt waiting to be serviced. The interrupt status associated with the control block has been stored in the termination status block. When a bit is set to 0, it indicates that no control-block logical interrupts have been posted for that device. Bits can be reset with the Reset Control Block Interrupt command.

The following figure shows the format and content of the Device Interrupt Identifier port.

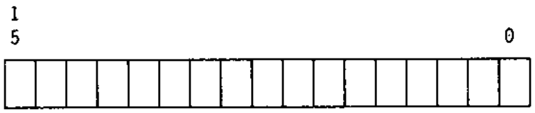


Figure 1-18. Device Interrupt Identifier Port

Each bit, starting with bit 0, has a one-to-one correspondence with a physical device. For example, bit 0 represents the subsystem (device 0); bit 1 represents device 1. When the subsystem supports more than 15 devices, the devices are assigned to additional Device Interrupt Identifier ports consecutively, starting at bit-position 0. For example, a subsystem that has 17 attached devices requires two Device Interrupt Identifier ports; bits 0 and 1 in the second Device Interrupt Identifier port are assigned to devices 16 and 17, respectively.

The Read Configuration command can be used to determine the number of Device Interrupt Identifier ports supported by a subsystem. (See "Read Configuration" on page 1-63.) For more information about the use of the Device Interrupt Identifier ports, see "DIIP Interrupts (Multiple Form)" on page 3-26.

Memory Address Space

In the Locate mode, shared memory is used to hold command control blocks, indirect lists, termination status blocks, and data. Each is described in detail later. The organization of shared memory is described in the System Resource description of the in Part 1, SCB Architecture Overview.

Control Structures

The Locate mode uses the following control structures: the command control block, the termination status block, and the indirect list.

Command Control Blocks

In the Locate mode, command control blocks are used to exchange control and status information between a client and a server. Command control blocks are variable-length areas in the system's memory address space containing the commands and arguments passed to a subsystem. These arguments describe the operation to be performed.

There are two formats for command control blocks: base and extended. The base command control block can use only one memory operand at a time and can increment only the memory address associated with the operation.

The extended command control block can use two memory operands in a single operation and can increment or decrement each memory address independently. The additional fields in the extended command control block used to support the additional functions are Enable Word 2 and Byte Count 2.

Some important characteristics of the command control block are:

- Command control blocks can be chained to other command control blocks.
- After a command control block is sent to a subsystem, it is read-only to a system master and to a device or subsystem. Command control blocks can be modified by the system master before being sent or after the device or subsystem has completed the task.
- Command control blocks must be aligned on a doubleword memory boundary.
- Extended termination status blocks must be used with extended command control blocks, and base termination status blocks must be used with base command control blocks.
- All memory addresses contained within a command control block must be specified as physical addresses.
- A device or subsystem in a feature adapter is required to read only those command-control-block fields that it needs.

The following figure shows the format of the base and extended command control blocks.

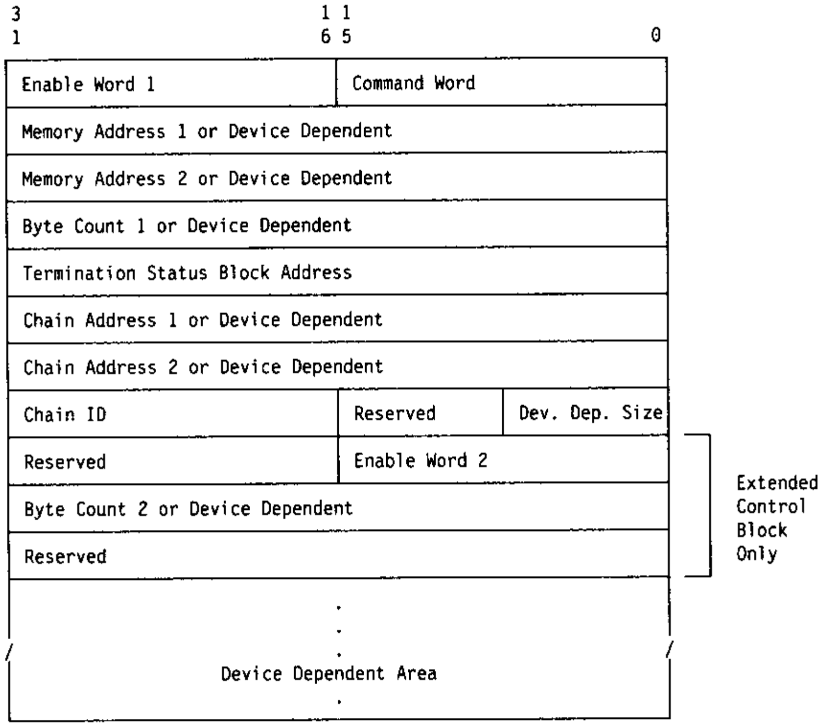

31
16 15
0
Enable Word 1
Command Word
Memory Address 1 or Device Dependent
Memory Address 2 or Device Dependent
Byte Count 1 or Device Dependent
Termination Status Block Address
Chain Address 1 or Device Dependent
Chain Address 2 or Device Dependent
Chain ID
Reserved
Dev. Dep. Size
Reserved
Enable Word 2
Byte Count 2 or Device Dependent
Reserved
Device Dependent Area
Extended Control Block Only

Figure 1-19. Command Control Block Structure

Command Word

The Command Word identifies the operation to be performed by a subsystem or device. The following figure shows the format of the Command Word and a description of each field.

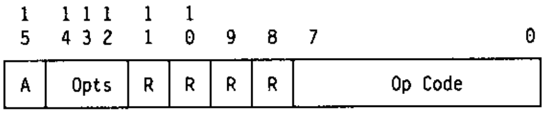


Figure 1-20. Command Word Format

Bit 15	The architected (A) bit indicates whether the command specified in the Op Code field is an architecture-defined command or one that is specific to the device. When this bit is set to 1, the Op Code field contains an operation code defined by the architecture. When this bit is set to 0, the operation code is device dependent. (See "Control-Block Commands" on page 1-55 for definitions of architected commands.)
Bits 14 – 12	The Options (Opts) field is used to modify or qualify the operation performed for each of the architected commands. The meaning of each bit depends on the command (see the specific command for the meaning and use of these bits).
Bits 11 – 8	These bits are reserved.
Bits 7 – 0	The Op Code field is used to identify the specific command to be performed. The value used can be for commands defined by the architecture, or it can be for a command that is defined by and specific to that subsystem or device. Bit 15, the architected bit, indicates whether the op-code value is architecture defined or device dependent.

For commands defined by the architecture, the op-code values are as follows.

Value	**Meaning**
0000 0000	No Operation
0000 0001	Read
0000 0010	Write
0000 0101	Run Diagnostic Test
0000 0110	Initialize Device
0000 0111	Read Completion Status
0000 1010	Read Configuration
All architected values not shown are reserved.	

Figure 1-21. Architected Op Codes

Enable Word 1

Enable Word 1 defines parameters to be used with this command control block. The following is the format of Enable Word 1 and a description of each field. DD indicates that the bit position is device dependent and that the bit can be used by individual subsystems and devices for specific needs.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DD	TSB	DD	IL1	SEL	SES	DD	EXT	DI	MA2	MA1		CCS		CC	CNE

Figure 1-22. Enable Word 1 Format

Bit 15 This bit (DD) can be used for applications that are device dependent.

Bit 14 The conditional termination-status-block (TSB) bit specifies whether status is stored for this command control block. When this bit is set to 0, a termination status block is stored for this command control block. When this bit is set to 1, the termination status block is stored only when an error or exception occurs while the command control block is being executed. (Whenever an error or exception occurs, a termination status block is stored, regardless of whether the condition is suppressed.)

Note: Independent of the setting of this bit, End Status Word 1 in the termination status block is stored whenever the Device Interrupt Identifier port is supported and an interrupt is generated at the completion of the command control block (that is, the disable-interrupt bit is set to 0).

Bit 13 This bit and bits 15 and 9 can be used for device-specific parameters.

Bit 12 The indirect-list-1 (IL1) bit specifies whether an indirect list (used to chain data) is being used with this command control block. If this bit is set to 1, an indirect list is being used (an indirect list must begin on a doubleword boundary). If this bit is set to 0, an indirect list is not being used. Refer to "Data Chaining" on page 4-10 for additional information on using indirect lists.

Figure 1-23 on page 1-36 describes the use of, and restrictions on the use of, the indirect-list-1 bit.

Bit 11 The suppress-exception-long (SEL) bit specifies whether a long-length exception is to be treated as a major error. (Refer to "Long-Length Exception" on page 5-11 for more information on long-length exceptions.)

If this bit is set to 1, a long-length exception condition is not treated as a major error. If this bit is set to 0, the exception is treated as a major error.

When a long-length exception occurs, the long-length-exception bit in End Status Word 1 is set to 1. The residual byte count and residual buffer address are stored in the termination status block. A single residual byte count and residual buffer address are stored in the appropriate field if only one of the Memory Address fields is specified in the command control block. When both Memory Address fields are used, two sets of residual byte counts and residual buffer addresses are stored.

Bit 10 The suppress-exception-short (SES) bit specifies whether a short-length exception is treated as a major error. Refer to "Short-Length Exception" on page 5-9 for more details on short-length exceptions.

If the suppress-exception-short bit is set to 1, a short-length-exception condition is suppressed and is not treated as a major error. If the suppress-exception-short bit is set to 0, the exception condition is treated as a major error.

When a short-length exception occurs, the short-length-exception bit in End Status Word 1 is set to 1. The residual byte count and residual buffer address are stored in the termination status block. A single residual byte count and residual buffer address are stored in the appropriate field if only one of the Memory Address fields is specified in the command control block. When both Memory Address fields are used, two sets of residual byte counts and residual buffer addresses are stored.

Bit 9 This bit is device dependent.

Bit 8 The extended-structure (EXT) bit indicates whether this command control block is a base or extended command control block. When this bit is set to 1, the command control block is an extended command control block. When this bit is set to 0, the command control block is a base command control block.

Bit 7 The disable-interrupt (DI) bit specifies whether the subsystem generates an interrupt after the normal processing of this command control block is completed or only when an error or exception occurs.

When this bit is set to 1, no interrupt is generated upon completion of the command control block. When this bit is set to 0, an interrupt is generated upon completion of the command control block.

If an unsuppressible error or exception occurs during control-block processing, an interrupt is generated, regardless of the setting of this bit.

Bit 6 The memory-address-2-selected (MA2) bit indicates whether the Memory Address 2 field contains a physical address to be used in the operation specified. If this bit is set to 1, the Memory Address 2 field contains an address, and the field operates as defined by the architecture. If this bit is set to 0, the Memory Address 2 field can be used as a device-dependent field.

For the base command control block, the Memory Address 1 field and the Memory Address 2 field are mutually exclusive because only one memory operand can be specified. The extended command control block, however, can use operations that specify two memory operands.

Figure 1-23 on page 1-36 describes the use of, and restrictions on the use of, the memory-address-2-selected bit.

Bit 5 The memory-address-1-selected (MA1) bit indicates whether the Memory Address 1 field contains a physical address to be used in the operation specified. If the bit is set to 1, the Memory Address 1 field contains an address, and the field operates as defined by the architecture. If the bit is set to 0, the Memory Address 1 field can be used as a device-dependent field.

For the base command control block, the Memory Address 1 field and the Memory Address 2 field are mutually exclusive because only one memory operand can be specified. The extended command control block can use operations that specify two memory operands.

The following figure describes the use of, and restrictions on the use of, the memory-address-1-selected bit, the memory-address-2-selected bit, the extended-structure bit, and the indirect-list-1 bit. For a description of how to use indirect lists, see "Data Chaining" on page 4-10.

IL1	EXT	MA2	MA1	Description
X	X	0	0	Neither Memory Address nor Byte Count is selected. Both fields are device dependent.
0	0	0	1	Memory Address 1 and Byte Count 1 are selected. The address can only be incremented.
0	0	1	0	Memory Address 2 and Byte Count 1 are selected. The address can only be incremented.
X	0	1	1	Invalid combination. Two memory operands require extended format.
0	1	0	1	Memory Address 1 and Byte Count 1 are selected. The address can be incremented or decremented.
X	1	1	0	Memory Address 2 and Byte Count 2 are selected. The address can be incremented or decremented.
0	1	1	1	Memory Address 1, Byte Count 1, Memory Address 2, and Byte Count 2 are selected. Any of these addresses can be incremented or decremented.
1	0	0	1	Memory Address 1 and Byte Count 1 are selected. The address points to an indirect list; the buffer addresses are incremented.
1	0	1	0	Memory Address 2 and Byte Count 1 are selected. The address points to an indirect list; the buffer addresses are incremented.
1	1	0	1	Memory Address 1 and Byte Count 1 are selected. The address points to an indirect list; the buffer addresses are incremented or decremented.
1	1	1	1	Memory Address 1 and Byte Count 1 are selected. The address points to an indirect list; the buffer addresses are incremented or decremented. Memory Address 2 and Byte Count 2 are selected. The address can be incremented or decremented.

Notes:

1. X means "don't care."
2. If two memory operands are being used, the operation code determines whether one memory operand is the source of the data and one is the target, or whether both operands act as sources or targets.

Figure 1-23. Memory-Addressing-Control Encoding

Bits 4 – 2 The chain-condition-specification (CCS) bits specify the conditions that must exist for chaining to continue. The conditions under which chaining will continue are listed in the following figure.

Value	Condition
000	An error or exception occurred that was not suppressed, other than specification errors, hardware failures while reading a command control block, or hardware failures while storing a termination status block.
001	The command control block is completed, and the associated termination status block indicates that additional device-dependent data is available.
010-111	These condition codes are reserved.

Figure 1-24. Chain-Condition-Specification Encoding

When the conditional-chaining bit is set to 0, these bits are ignored. See Figure 1-25 for the setting of conditional-chaining and other related bits.

Bit 1 The conditional-chaining (CC) bit specifies whether conditional chaining is enabled or disabled. If this bit is set to 1, conditional chaining is enabled. If this bit is set to 0, conditional chaining is disabled.

See Figure 1-25 for the setting of this and other related bits.

Bit 0 The chain-no-error (CNE) bit indicates whether unconditional chaining is enabled. If this bit is set to 1, unconditional chaining is enabled. If this bit is set to 0, unconditional chaining is disabled.

The following figure shows the relationship among the chaining bits: chain-no-error, conditional-chaining, and chain-condition-specifications.

CNE	CC	Met Chain Conditions	Subsystem Chain Action
0	0	X	No chaining active.
0	1	No	Halt chaining.
X	1	Yes	Chain to Chain Address 2.
1	0	X	If no error or error is suppressed, chain to Chain Address 1; otherwise, halt chaining.
1	1	No	If no error or error is suppressed, chain to Chain Address 1; otherwise, halt chaining.
Note: X means "don't care."			

Figure 1-25. Chaining-Bit Settings

Memory Address 1

The Memory Address 1 field is a 32-bit field. If the memory-address-1-selected bit is set to 1, this field contains the physical address of the source or target memory address to be used in this operation. If the memory-address-1-selected bit is set to 0, the field can be used for device-dependent parameters.

When the Memory Address 1 field is selected and an indirect list is being used, this field points to an area in memory that contains a list of addresses and byte counts used as the source or destination of the data transfer. (See "Data Chaining" on page 4-10 for additional information on indirect lists.)

When the Memory Address 1 field is selected and an indirect list is not being used, this field contains the physical address of the buffer in memory that is used as the source or destination of the data transfer.

Memory Address 2

The Memory Address 2 field is a 32-bit field. If the memory-address-2-selected bit is set to 1, this field contains the physical address of the source or target memory address to be used in this operation. If the memory-address-2-selected bit is set to 0, this field can be used for device-dependent parameters.

When the Memory Address 2 field is selected and an indirect list is being used, this field points to an area in memory that contains a list of addresses and byte counts used as the source or destination of the data transfer. (See "Data Chaining" on page 4-10 for additional information on indirect lists.)

When the Memory Address 2 field is selected and an indirect list is not being used, this field contains the physical address of the buffer in memory that is used as the source or destination of the data transfer.

Byte Count 1

The Byte Count 1 field is a 32-bit field that contains the number of bytes to be transferred. The value contained in this field depends on whether an indirect list is used. (See Figure 1-23 on page 1-36 for bit settings that control the use of this field.)

If an indirect list is not being used, this field contains the number of bytes to be transferred. If the count specified is 0, no data is transferred.

If an indirect list is being used, this field contains the number of bytes in the indirect list, not the number of bytes to be transferred. The byte count must be a multiple of 8 because each indirect-list entry consists of 8-byte elements (a 32-bit physical address and a 32-bit byte count). The sum of all byte counts in the indirect list is the number of bytes to be transferred. If the sum of all byte counts is 0, no data is transferred.

This field can be used for device-dependent parameters if neither Memory Address field is used or if, in the extended format, the Memory Address 1 field is not used.

Termination Status Block Address

The Termination Status Block Address field is a 32-bit field. It contains the physical address of a doubleword-aligned memory area that is used by the subsystem to store the termination status block for this command control block. Refer to "Termination Status Blocks" on page 1-43 for a description of the termination status block.

Chain Address 1

The Chain Address 1 field is a 32-bit field that contains the physical address of the next command control block to be processed by a device or subsystem. All command control blocks are aligned on doubleword addresses.

If the chain-no-error bit is set to 0, the Chain Address 1 field can be used for device-dependent parameters. (See Figure 1-25 on page 1-38 for bit settings that control the use of this field.)

Chain Address 2

The Chain Address 2 field is a 32-bit field that contains the physical address of the next command control block to be processed by a subsystem or device when conditional chaining is enabled. The Chain Address 2 field is used when conditional chaining is enabled and the command control block is completed in a manner that matches the condition set in the chain-condition-specification bits.

If the conditional-chaining bit is set to 0, the Chain Address 2 field can be used for device-dependent parameters. (See Figure 1-25 on page 1-38 for bit settings that control the use of this field.)

Device Dependent Size

The Device Dependent Size field is an 8-bit field containing a count of the number of bytes in the device-dependent area of the command control block.

Chain ID

This field is reserved.

Enable Word 2

The Enable Word 2 field is a 16-bit field and is available only in the extended command control block. The field specifies additional parameters to be used when the extended command control block is processed. The following figure describes the Enable Word 2 fields.

15 … 3	2	1	0
Reserved	DEC2	DEC1	IL2

Figure 1-26. Enable Word 2 Format (Extended Command Control Block)

Bits 15 – 3 These bits are reserved.

Bit 2 The decrement-memory-address-2 (DEC2) bit specifies whether the memory address pointed to by the Memory Address 2 field is to be decremented or incremented when Memory Address 2 is selected. When the Memory Address 2 field is not selected, this bit has no effect.

If this bit is set to 1, the buffer addresses are decremented as data is transferred. If this bit is set to 0, the buffer addresses are incremented.

Bit 1 The decrement-memory-address-1 (DEC1) bit specifies whether the memory address pointed to by the Memory Address 1 field is to be decremented or incremented when Memory Address 1 is selected. When the Memory Address 1 field is not selected, this bit has no effect.

If this bit is set to 1, the buffer addresses are decremented as data is transferred. If this bit is set to 0, the buffer addresses are incremented.

Bit 0 The indirect-list-2 (IL2) bit specifies whether an indirect list is being used with the Memory Address 2 field. (Details on indirect lists are described in "Data Chaining" on page 4-10)

Note: This bit is valid only when both the extended-structure and the memory-address-2-selected bits in Enable Word 1 are set to 1.

When this bit is set to 1 and the memory-address-2-selected bit is set to 1, the Memory Address 2 field contains the address of an indirect list. An indirect list must begin on a doubleword boundary.

When the indirect-list-2 bit is set to 0, the Memory Address 2 field does not point to an indirect list.

Byte Count 2

The Byte Count 2 field is a 32-bit field and is available only in the extended command control block. When the Memory Address 2 field is selected, the content of this field depends on whether an indirect list is being used. When the Memory Address 2 field is not selected, this field can be used for device-dependent applications.

When an indirect list is not being used, this field contains the number of bytes in the source or destination buffer to which the Memory Address 2 field points that are to be transferred. If the count specified is 0, no data is transferred.

When an indirect list is being used, this field contains the number of bytes in the indirect list, not the number of bytes to be transferred. The byte count must be a multiple of 8 because each indirect-list entry consists of an 8-byte element (a 32-bit physical address and a 32-bit byte count). The sum of all byte counts in the indirect list is the number of bytes to be transferred. If the sum of all byte counts is 0, no data is transferred.

Termination Status Blocks

Termination status blocks are areas in shared memory that are used by a device or subsystem to report the ending status of a command control block to the software in the system master. The location of a termination status block is contained in the TSB Address field of the command control block in which it is used.

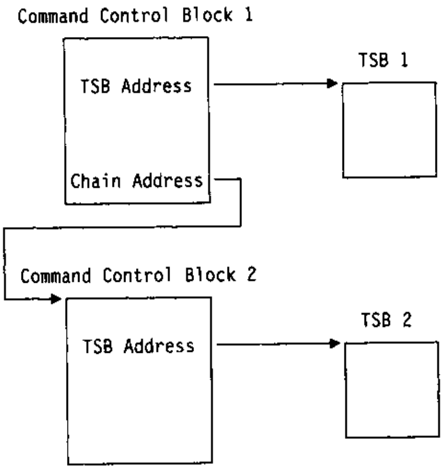


Figure 1-27. Termination Status Blocks

A termination status block is structured so that the status information it contains can be read and interpreted by software in the system master, regardless of the control-block command processed.

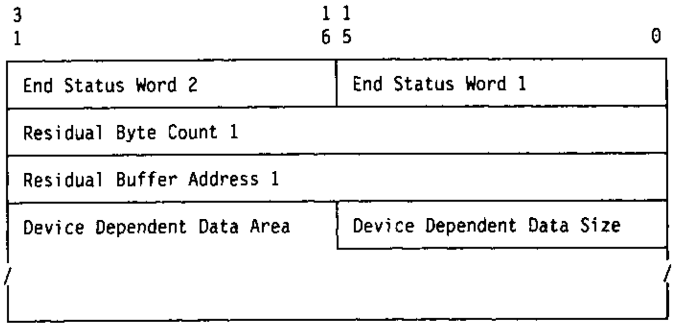
The area allocated to the termination status block must not be modified by the system master after its associated command control block has been delivered to the subsystem. The system software should initialize the End Status Word 1 field in the termination status block to 0 prior to delivering the command control block to the subsystem. This allows the software to do a simple nonzero check of the End Status Word 1 field in the termination status block to determine whether the termination status block was stored upon command-control-block completion.

Any command control block in a chain can suppress the storing of the termination status block for normal completion of the command control block. However, the termination status block is always stored when a command control block is suspended, terminated by an error, reports a suppressed error, or follows a conditional chain. When accompanied by an interrupt to the system master, a termination status block allows a program to monitor a device's progress through a chain of command control blocks.

The Subsystem Control Block architecture supports two formats of termination status blocks: base and extended. The difference between them is the addition of two fields for the extended format: the Residual Byte Count 2 field and the Residual Buffer Address 2 field.

The base format of the termination status block is used by a device or subsystem to report the ending status of a base command control block. A base termination status block is always 32 bytes in length.

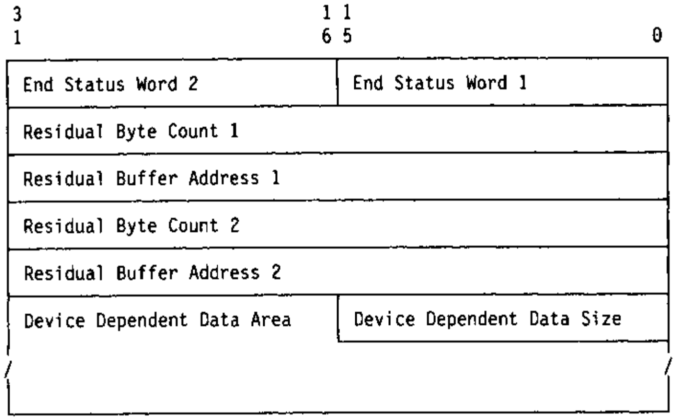
The structure of the base termination status block is shown in the following figure.



3 1
1 6
1 5
0
End Status Word 2
End Status Word 1
Residual Byte Count 1
Residual Buffer Address 1
Device Dependent Data Area
Device Dependent Data Size

Figure 1-28. Base Termination Status Block

The extended format of the termination status block is used to report the ending status of an extended command control block. An extended termination status block is always 40 bytes in length. The following figure shows the structure of the extended termination status block and a description of each of the individual fields.



3 1
1 6
1 5
0
End Status Word 2
End Status Word 1
Residual Byte Count 1
Residual Buffer Address 1
Residual Byte Count 2
Residual Buffer Address 2
Device Dependent Data Area
Device Dependent Data Size

Figure 1-29. Extended Termination Status Block

End Status Word 1

End Status Word 1 is a 16-bit structured field that indicates how a command in the associated command control block was processed. Information is placed in this field by the device or a subsystem when:

- The command control block was completed with an error (including suppressed errors).
- The command control block was completed without error but had requested that a termination status block be stored unconditionally (the TSB bit in Enable Word 1 is 0).
- The command control block was suspended by a Suspend command.
- The device addressed in the Attention port supports the Device Interrupt Identifier port (refer to "Signalling Protocol" on page 3-13 for a description), and an interrupt was requested upon normal completion of the command control block.

When the termination status block is written, End Status Word 1 must be the last portion of the status information stored; the update is done prior to requesting a system interrupt. The system should reset this field to 0 prior to delivering a command control block to a device or subsystem. The system can then test this field for a nonzero value to determine whether the termination status block has been stored.

The format of End Status Word 1 is shown in the following figure. It is followed by a description of the content, meaning, and use of each of its individual bits.

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ES	SUS	CD	ME	INI	DO	TSA		INT	R	LLE	SC	DD	DD	SLE	D

Figure 1-30. End Status Word 1 Format

Bit 15 The extended-status (ES) bit indicates how the End Status Word 2 field is being used. When this bit is 1, the End Status Word 2 field contains status information as defined by the architecture. When this bit is 0, End Status Word 2 can be used as a device-dependent field.

Bit 14 The suspended (SUS) bit indicates that a Suspend command has been received during the execution of this command control block. The device has entered the suspend state after completing the execution of this control block.

When this bit is 1, the processing of the associated command control block has been completed; however, the device is in the suspended state and will not allow execution of any more command control blocks. When this bit is 0, the processing of control blocks has not been suspended.

Bit 13 The chain-direction (CD) bit indicates whether the address in the Chain Address 2 field of the current command control block has been used to obtain the next command control block in the chain.

When this bit is 1, the Chain Address 2 field has been used as the pointer to the next command control block in the chain. When this bit is 0, the Chain Address 2 field has not been used.

Bit 12 The major-error/exception (ME) bit is used to indicate whether a major error or exception has occurred in the processing of the associated command control block.

When this bit is 1, the command in the associated command control block has been completed unsuccessfully. The interrupt-requested bit is also set to 1 because an interrupt must be requested when a major error or exception is detected.

When this bit is set to 0, the command in the command control block has been completed without major errors or exceptions.

Bit 11 The device-not-initialized (INI) bit indicates whether a device has completed initialization.

When this bit is 1, the device has not completed the initialization sequence. When this bit is 0, the device has completed the initialization sequence.

Bit 10 The device-overrun (DO) bit indicates that a device overrun has been detected.

When this bit is 1, the device has lost data because of an overrun. The device also sets the major-error/exception and interrupt-requested bits to 1. When this bit is set to 0, the device has not detected an overrun.

See "Device-Overrun Exception" on page 5-14 for a definition of the condition and the actions taken.

Bits 9, 8 The TSB-available (TSA) bits contain an encoded value that indicates the amount of status information stored in the termination status block and whether more status information is available. The value and meaning of these bits are as shown in the following figure.

Value	**Meaning**
00	Status information has been placed in End Status Word 1.
01	All fields, except for the Device Dependent Data Size and the Device Dependent Area, contain status information.
10	All fields, including the Device Dependent Area, contain status information. The Device Dependent Data Size field specifies the number of bytes of device-dependent data stored. Up to 18 bytes of device-dependent data can be stored in a termination status block. If the value in the Device Dependent Data Size field exceeds 18, the Read Completion Status command must be used to retrieve this additional data. See "Read Completion Status" on page 1-60 for more details on Read Completion Status.
11	All fields of the termination status block, except for the Residual Buffer Address and Residual Byte Count fields, contain status information. The value of the Device Dependent Data Size field specifies the number of bytes of device-dependent data stored.

Figure 1-31. TSB-Available Bit Encoding

Bit 7	The interrupt-requested (INT) bit indicates whether an interrupt has been requested for the associated command control block. When this bit is set to 0, an interrupt has not been requested. When this bit is 1, an interrupt has been requested for the completed command control block.
Bit 6	This bit is reserved.
Bit 5	The long-length-exception (LLE) bit indicates whether a long-length exception has been detected. See "Long-Length Exception" on page 5-11 for the definition of the condition and the actions taken when the condition occurs. When this bit is set to 1, a long-length exception has occurred. When this bit is set to 0, a long-length exception has not occurred.
Bit 4	The specification-check (SC) bit indicates that an invalid field has been detected in the command control block. Checking the command control block for valid fields is optional. When this bit is 1, checking has been performed, and an invalid field has been found. The operation is terminated, and no data is transferred. When this bit is 0, either the fields have not been checked or an invalid field was not detected. See "Specification Exception" on page 5-13 for definitions of the conditions and the actions to be taken when a condition is detected.
Bits 3, 2	These bits (DD) are device dependent and contain information meaningful only to the specific device implementation.
Bit 1	The short-length-exception (SLE) bit indicates whether a short-length exception has been detected. When this bit is set to 1, a short-length exception has occurred. When this bit is set to 0, a short-length exception has not occurred. See "Short-Length Exception" on page 5-9 for the definition of the condition and the actions taken when the condition is detected.

Bit 0 The done (D) bit indicates whether the command in the associated command control block was completed without error.

When this bit is 1, the associated command control block has been completed without an error or exception. When this bit is 0, other bits defined in End Status Word 1 must be checked to determine the status of the associated command control block.

End Status Word 2

The End Status Word 2 field is a 16-bit structured field used to hold additional status information. This field can also be used as a device-dependent field, depending on the setting of the extended-status bit in End Status Word 1.

The architected format of End Status Word 2 is shown in the following figure and is followed by a description of the individual bits.

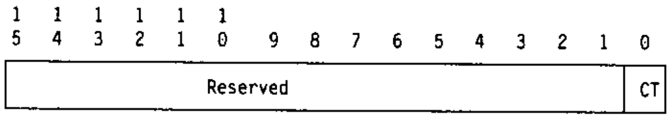


Figure 1-32. End Status Word 2 Format

Bits 15 – 1 These bits are reserved.

Bit 0 The command-type (CT) bit is always set to 0.

Residual Byte Count 1

The Residual Byte Count 1 field is a 32-bit field that is aligned on a doubleword boundary and contains the number of bytes that remain to be read into, or written from, the buffer identified by the address in the Residual Buffer Address 1 field.

This field must be provided when either a short-length or long-length exception condition is indicated. If neither the short-length-exception bit nor the long-length-exception bit in End Status Word 1 is used, this field is optional. For more detail see "Residual Byte Count" on page 5-19.

Residual Buffer Address 1

The Residual Buffer Address 1 field is a 32-bit field that is aligned on a doubleword boundary and identifies the buffer last read from, or written to, in the associated command control block.

If the associated command control block buffer is not a data chain element in an indirect list, this field contains the address of the buffer (Memory Address 1 or Memory Address 2) as indicated by bits in Enable Word 1 of the command control block.

If the associated command control block buffer is a data chain element in an indirect list, this field contains the address of the buffer in the data-chain element that was most recently used to transfer data.

This field must be provided when either a short-length or long-length exception condition is indicated. If neither the short-length-exception bit nor the long-length-exception bit in End Status Word 1 is used, this field is optional. For more detail see "Residual Buffer Address" on page 5-20.

In the extended termination status block, this field pertains only to the Memory Address 1 field of the command-control-block structure. It is used to identify the data transfer buffer that was located through Memory Address 1.

Note: Software in the system master can use the Residual Buffer Address 1 and Residual Buffer Count 1 fields to determine how far a command has progressed toward completion. The encoded value of the TSB Available (TSA) field in End Status Word 1 indicates whether these two fields have been stored by the device. For more details, see "Number of Bytes Transferred" on page 5-21.

Residual Byte Count 2

The Residual Byte Count 2 field is present only in an extended termination status block. It is a 32-bit field that is aligned on a doubleword boundary and contains the number of bytes that remain to be read into, or written from, the buffer identified in the Residual Buffer Address 2 field.

This field must be provided when either a short-length-exception or long-length-exception condition is indicated by the short-length-exception (SLE) bit or the long-length-exception (LLE) bit in End Status Word 1. If neither the short-length-exception (SLE) bit nor the long-length-exception (LLE) bit is used, this field is optional and does not need to be reported. Refer to "Residual Byte Count" on page 5-19 for a detailed description of this field.

Residual Buffer Address 2

The Residual Buffer Address 2 field is present only in an extended termination status block. It is a 32-bit doubleword-aligned field identifying the buffer last written to or read from in the associated command control block.

If the associated command-control-block buffer is not a data-chain element in an indirect list, this field contains the address of the buffer (Memory Address 2).

If the associated command-control-block buffer is a data-chain element in an indirect list, this field contains the address of the buffer in the data-chain element that was most recently used to transfer data.

This field must be provided when either a short-length-exception or long-length-exception condition is indicated by the short-length-exception (SLE) bit or the long-length-exception (LLE) bit in End Status Word 1. If neither the short-length-exception (SLE) bit nor the long-length-exception (LLE) bit is used, this field is optional and does not need to be reported. Refer to "Residual Buffer Address" on page 5-20 for a detailed description of this field.

Note: Software in the system master can use the Residual Buffer Address 1, Residual Buffer Count 1, Residual Buffer Address 2, and Residual Buffer Count 2 fields to determine how far a command has progressed toward completion. The encoded value of the TSB Available (TSA) field in End Status Word 1 indicates whether these two fields have been stored by the device. For more details, see "Number of Bytes Transferred" on page 5-21.

Device Dependent Data Size

The Device Dependent Data Size field is a 16-bit field that is used to indicate the number of bytes of device-dependent data stored by a device or subsystem as a result of the associated command control block. If the value of this field exceeds 18 bytes, a Read Completion Status control-block command must be issued to retrieve the data beyond byte 18. Refer to "Read Completion Status" on page 1-60 for details on the Read Completion Status command.

The validity of this field is determined by examining the encoded value of the TSB-available bit in End Status Word 1.

Device Dependent Data Area

The Device Dependent Data Area is an 18-byte field used to hold a maximum of 18 bytes of device-dependent data. The amount of device-dependent data contained within the field is indicated by the value in the Device Dependent Size field.

Indirect-List Structure

An indirect list is a variable-length list consisting of the address-count pairs used to support data chaining. Both the location of the indirect list and its length are specified in the control block. Within the indirect list, the Address field and the Count field are each doubleword fields. The format and content of an indirect list is shown in the following figure.

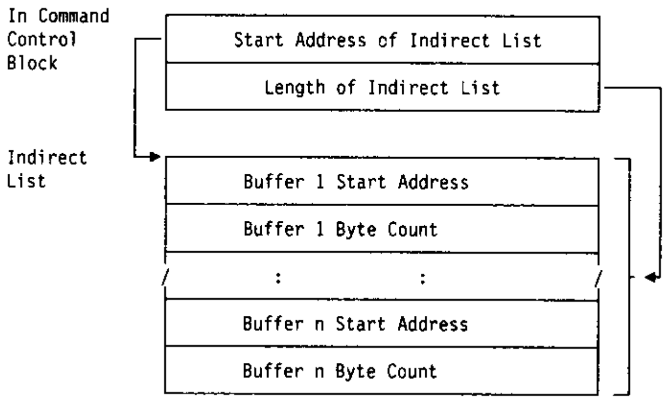


Figure 1-33. Indirect-List Format

Commands

In the Locate mode, there are two types of commands: control-block commands and immediate commands.

Control-Block Commands

The Locate mode architecture defines a number of control-block commands that have common uses. This section describes these commands and their associated command control blocks. These commands are defined in alphabetic order. The architected control-block commands are:

- Initialize Device
- No Operation
- Read
- Read Completion Status
- Read Configuration
- Run Diagnostic Test
- Write.

Control-block commands are primarily concerned with the transfer of data. Therefore, the semantics of commands that transfer data follow those of the Read and Write commands.

All control-block commands can be in the form of the base or extended command control blocks and can use indirect lists. The semantics of commands transferring indirect data in read and write operations are defined in "Reading with a Base Control Block and Indirect Lists" on page 4-12 and "Writing with a Base Control Block and Indirect Lists" on page 4-14. The semantics of interrupts used in control-block commands follow those described in sections "Signalling Protocol" and "Exception Conditions" on page 5-9.

Initialize Device

The Initialize Device command is used to write initialization data to a device. Prior to receiving initialization data, a device might not be able to execute its entire command set. However, prior to initialization, the device must be able to execute the Read Configuration command and the Initialize Device command.

The Memory Address 2 field points to initialization data, and the Byte Count 1 field defines the amount of initialization data to be read by the device. Data is transferred to the address in a device's memory, using the same semantics as the Write command.

The Initialize Device command should be directed only to a device that requires initialization and that has been put into a state in which it can be initialized. Attempting to initialize a device that does not require initialization can cause a specification exception. Software in the system master can determine whether a device needs to be initialized by issuing the Read Configuration command.

The Initialize Device command has the option to request an interrupt when the command is completed without an error or with a suppressed error.

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
┌───────────────────────────────┬─┬─────┬───────┬───────────────┐
│ Enable Word 1                 │A│R L C│  Res  │0 0 0 0 0 1 1 0│
├───────────────────────────────┴─┴─────┴───────┴───────────────┤
│ Start Address or Device Dependent                             │
├───────────────────────────────────────────────────────────────┤
│ Memory Address of Source                                      │
├───────────────────────────────────────────────────────────────┤
│ Byte Count 1                                                  │
├───────────────────────────────────────────────────────────────┤
│ Termination Status Block Address                              │
├───────────────────────────────────────────────────────────────┤
│ Chain Address 1 or Device Dependent                           │
├───────────────────────────────────────────────────────────────┤
│ Chain Address 2 or Device Dependent                           │
├───────────────────────────────┬───────────────┬───────────────┤
│ Chain ID                      │ Reserved      │ Dev. Dep. Size│
├───────────────────────────────┴───────────────┴───────────────┤
/                               .                               /
│                    Device Dependent Area                      │
│                               .                               │
└───────────────────────────────────────────────────────────────┘
```

Figure 1-34. Initialize Device Control Block

Options

The following options are used with the Initialize Device command; if the device or subsystem does not support these options, a specification error can result.

Bit 14	The restart (R) bit specifies whether a device restarts initialization or continues from a previous point. If this bit is set to 1, the initialization sequence starts at the beginning. If this bit is set to 0, the sequence continues from a prior point.
Bit 13	The location (L) bit specifies whether the Start Address field, used as a data load point, is a destination address that is internal to the device. If this bit is set to 1, the Start Address field is an internal load point. If this bit is set to 0, the Start Address field is available for device-dependent applications.
Bit 12	The complete (C) bit specifies whether the device should complete initialization with this command. If this bit is set to 1, the device should complete initialization. If this bit is set to 0, the device should not complete initialization. If this bit is set to 1 and the device can complete its initialization, the device-not-initialized bit in End Status Word 1 is set to 0. If initialization cannot be completed when this bit is selected, the device-not-initialized bit is set to 1.

No Operation

The No Operation (Noop) command initiates a branching structure within command chains or requests an interrupt that can signal a progress point in a command chain.

No data transfer to buffer areas in shared memory is performed. Memory Address 1 and Memory Address 2 fields do not specify buffer areas.

This command has the option to request an interrupt when the command is completed.

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-------------------------------+-+-----+-------+---------------+
| Enable Word 1                 |A| Opts|  Res  |0 0 0 0 0 0 0 0|
+-------------------------------+-+-----+-------+---------------+
| Device Dependent                                              |
+---------------------------------------------------------------+
| Device Dependent                                              |
+---------------------------------------------------------------+
| Byte Count 1                                                  |
+---------------------------------------------------------------+
| Termination Status Block Address                              |
+---------------------------------------------------------------+
| Chain Address 1 or Device Dependent                           |
+---------------------------------------------------------------+
| Chain Address 2 or Device Dependent                           |
+-------------------------------+---------------+---------------+
| Chain ID                      | Reserved      | Dev. Dep. Size|
+-------------------------------+---------------+---------------+
/                               .                               /
|                     Device Dependent Area                     |
|                               .                               |
+---------------------------------------------------------------+
```

Figure 1-35. No-Operation Control Block

Read

The Read command transfers data from the device specified in the Attention port (source) to a location in shared memory that is pointed to by the Memory Address 2 field (destination).

This command has the option to request an interrupt when the command is completed without error or with a suppressed error.

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Enable Word 1	A	Opts	Res	0 0 0 0 0 0 0 1
Device Dependent				
Memory Address of Destination				
Byte Count 1				
Termination Status Block Address				
Chain Address 1 or Device Dependent				
Chain Address 2 or Device Dependent				
Chain ID	Reserved			Dev. Dep. Size
. Device Dependent Area .				

Figure 1-36. Read Control Block

If an indirect list is specified in this control block (the indirect-list-1 bit of Enable Word 1 is set to 1), semantics for data movement are defined in "Reading with a Base Control Block and Indirect Lists" on page 4-12.

If an indirect list is not specified (the indirect-list bit of Enable Word 1 is set to 0), the Memory Address 2 field contains the starting address of the destination area. The Byte Count 1 field contains the length, in bytes, of the data to be transferred. If the Byte Count 1 field is 0, no data is transferred.

A running count of the number of bytes transferred is maintained within the device (residual byte count). This count is developed from the initial value in the Byte Count 1 field. As each byte is transferred from the device, the Residual Byte Count field is decreased by 1. The operation is completed when the Residual Byte Count field is 0 or the device has no more data to transfer. The address in the Memory Address 2 field is copied to a Residual Buffer Address field in the termination status block, and the residual byte count is transferred to the Residual Byte Count field in the termination status block.

When a command control block is completed, a program can obtain the residual byte count and residual buffer address from the associated termination status block to determine the progress of the operation. Storing these fields in the termination status block is optional, except when short-length or long-length exception conditions are detected.

Note: Refer to "Exception Conditions" on page 5-9 for a description of short-length and long-length exceptions.

If the residual byte count and residual buffer address are not stored for a read operation that is completed with an error, the entire operation should be performed again.

Data transfers during a read operation can also be terminated because of irrecoverable errors at the device (other than short-length and long-length exceptions). The major-error/exception and interrupt-requested bits in the End Status Word 1 are set to 1 to indicate that a major error occurred and that an interrupt was requested.

Read Completion Status

The Read Completion Status command allows a program to retrieve the completion status of the previous immediate or control-block command from a device. The data is moved from the device to an area that is pointed to by the Memory Address 2 field.

Completion status data is transferred to the system master, using the same semantics as the Read command.

This command must be implemented if a device has the potential to report more status than can be presented in the termination status block, or if the device has the potential to report status in response to immediate commands.

The Read Completion Status command has the option to request an interrupt when the command is completed without error or with a suppressed error.

The maximum size of the completion status information depends on the device and can be determined with the Read Configuration command.

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+--------------------------------+-+-----+-------+---------------+
| Enable Word 1                  |A| CFCS|  Res  |0 0 0 0 0 1 1 1|
+--------------------------------+-+-----+-------+---------------+
| Device Dependent                                               |
+----------------------------------------------------------------+
| Memory Address of Completion Status Data                       |
+----------------------------------------------------------------+
| Byte Count 1                                                   |
+----------------------------------------------------------------+
| Termination Status Block Address                               |
+----------------------------------------------------------------+
| Chain Address 1 or Device Dependent                            |
+----------------------------------------------------------------+
| Chain Address 2 or Device Dependent                            |
+--------------------------------+---------------+---------------+
| Chain ID                       | Reserved      |Dev. Dep. Size |
+--------------------------------+---------------+---------------+
/                                .                               /
|                       Device Dependent Area                    |
|                                .                               |
+----------------------------------------------------------------+
```

Figure 1-37. Read Completion-Status Control Block

Byte Count 1 specifies the number of bytes of completion status data that the software will read from the device.

The completion status data returned to the system is dependent on the encoding of the Command for Completion Status field, as described in the following table. An invalid or unsupported encoding of this field might raise a specification error.

Value	Description
000	The completion-status data to be returned is for the last command (either immediate or control block) that was executed by the device prior to execution of this Read Completion Status command.
001	The completion-status data to be returned is for the last control-block command that was executed by the device prior to execution of this Read Completion Status command.
010	The completion-status data to be returned is for the last immediate command that was executed by the device prior to execution of this Read Completion Status command.
All other values are reserved.	

Figure 1-38. Command for Completion Status Field Encoding

If the device does not have completion status to report that matches the specification given in the Command for Completion Status field, a short-length exception condition can be returned to the system master.

The format of the data returned in response to the Read Completion Status command is shown in the following figures. The data block returned depends on whether the status corresponds to a control-block command or to an immediate command. The following figure shows the format of completion-status data when the previously executed command was an immediate command. Figure 1-40 on page 1-63 shows the format of completion-status data when the previously executed command was a control-block command.

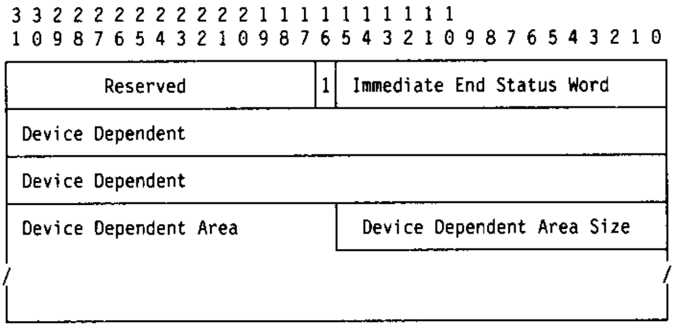


Figure 1-39. Completion-Status Data Block for Immediate Commands

The format, content, and description of the Immediate End Status Word are the same as those of End Status Word 1 of the termination status block defined in "End Status Word 1" on page 1-46.

Note: It is suggested that the device return (as device-dependent data) the immediate command or the address of the control block with the completion status.

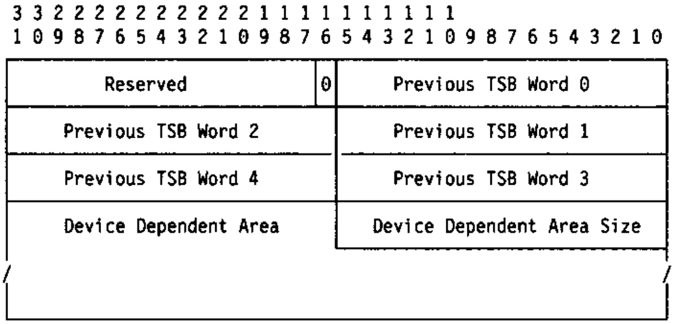


Figure 1-40. Completion-Status Data Block for Base Control Blocks

Read Configuration

The Read Configuration control-block command is used to obtain parametric data about the subsystem or the devices attached to the subsystem. The device must always be capable of executing this command, even if the device requires initialization using the Initialize Device command.

Configuration data is transferred to the system master, using the same semantics as the Read command. Data is delivered to the system master in the location pointed to by the Memory Address 2 field.

The Read Configuration command has the option to request an interrupt when the command is completed without error or with a suppressed error.

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-------------------------------+-+-----+-------+---------------+
| Enable Word 1                 |A| Opts|  Res  |0 0 0 0 1 0 1 0|
+-------------------------------+-+-----+-------+---------------+
| Device Dependent                                              |
+---------------------------------------------------------------+
| Memory Address of Read Configuration Data                     |
+---------------------------------------------------------------+
| Byte Count 1                                                  |
+---------------------------------------------------------------+
| Termination Status Block Address                              |
+---------------------------------------------------------------+
| Chain Address 1 or Device Dependent                           |
+---------------------------------------------------------------+
| Chain Address 2 or Device Dependent                           |
+-------------------------------+---------------+---------------+
| Chain ID                      | Reserved      |Dev. Dep. Size |
+-------------------------------+---------------+---------------+
/                               .                               /
|                     Device Dependent Area                     |
|                               .                               |
+---------------------------------------------------------------+
```

Figure 1-41. Read Configuration Control Block

Byte Count 1 specifies the number of bytes of configuration data that software will read from the device. Byte Count 1 should contain a value that is equal to at least 44 bytes. Data read beyond the first 44 bytes is device dependent.

If a long-length-exception condition is detected, the system master can determine the maximum amount of configuration data by checking the device-dependent-area size and reissuing the command with the appropriate byte count.

If a short-length-exception condition is detected, the number of valid bytes returned depends upon the value specified for Byte Count 1.

The following figure shows the format of the read-configuration data block returned to the system. It is followed by a description of each of the configuration data fields.

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+---------------+---------------+-------------------------------+
|POS Register 3 |POS Register 2 |         Attachment ID         |
+---------------+---------------+---------------+---------------+
|POS Register 7 |POS Register 6 |POS Register 5 |POS Register 4 |
+---------------+---------------+---------------+---------------+
| Attachment Revision Level     |   Reserved    |   Int Level   |
+-------------------------------+---------------+---------------+
|           Reserved            | Number Of Devices Supported   |
+-----------------------+-+-+-+-+-----------------------+-+-+-+-+
| Max Device Reset Time |n|u|m|s| Max Subs Reset Time   |n|u|m|s|
+-----------------------+-+-+-+-+---------------+-------+-+-+-+-+
|                  Reserved                     |     Flags     |
+-----------------------+-+-+-+-+---------------+-------+-+-+-+-+
| Max Reset ISP Time    |n|u|m|s| Max Reset CB Int Time |n|u|m|s|
+-----------------------+-+-+-+-+-----------------------+-+-+-+-+
|           Reserved            | Min Cmd Status Time   |n|u|m|s|
+-------------------------------+-----------------------+-+-+-+-+
|  DIIP Register Address        | Max Number Of Queued Ints     |
+-------------------------------+-------------------------------+
| Max Size Completion Status    |  DIIP Bit Position            |
+-------------------------------+-------------------------------+
| Device Dependent Area Size    | Max Size Diagnostic Status    |
+-------------------------------+-------------------------------+
|                    Device Dependent Area                      |
/                                                               /
|                                                               |
+---------------------------------------------------------------+
```

Figure 1-42. Read Configuration Data Block

Attachment ID
The Attachment Identification field is a 16-bit field that contains the ID of the subsystem.

POS Register 2
The POS Register 2 field is an 8-bit field that reflects the contents of POS Register 2.

POS Register 3
The POS Register 3 field is an 8-bit field that reflects the contents of POS Register 3.

POS Register 4
The POS Register 4 field is an 8-bit field that reflects the contents of POS Register 4.

POS Register 5
The POS Register 5 field is an 8-bit field that reflects the contents of POS Register 5.

POS Register 6

The POS Register 6 field is an 8-bit field that reflects the contents of POS Register 6.

POS Register 7

The POS Register 7 field is an 8-bit field that reflects the contents of POS Register 7.

Interrupt Level (Int Level)

The Interrupt Level field is an 8-bit field that contains the interrupt level assigned to the specified device at the end of system initialization.

Reserved

These areas are reserved.

Attachment Revision Level

The Attachment Revision Level field is a 16-bit field that indicates the revision level of the subsystem.

Number of Devices Supported

The Number of Devices Supported on the Subsystem field is a 16-bit field that indicates the maximum number of devices that can be attached to the subsystem.

Maximum Subsystem Reset Time

The Maximum Subsystem Reset Time field is a 16-bit field that contains the maximum time for the subsystem to reset. It can be specified in seconds, milliseconds, microseconds, or nanoseconds, depending on the setting of the 4 lower-order bits. The bits are defined in the following figure.

Bit Name	Description
s	The seconds bit, when set to 1, indicates that the field is specified in seconds.
m	The milliseconds bit, when set to 1, indicates that the field is specified in milliseconds.
u	The microseconds bit, when set to 1, indicates that the field is specified in microseconds.
n	The nanoseconds bit, when set to 1, indicates that the field is specified in nanoseconds.
Note: One and only one bit must be set to 1. All other combinations are reserved.	

Figure 1-43. Time-Field Specifier

Maximum Device Reset Time

The Maximum Device Reset Time field is a 16-bit field that contains the maximum time for the device to reset. It can be specified in seconds, milliseconds, microseconds, or nanoseconds, depending on the setting of the 4 lower-order bits as indicated in Figure 1-43.

Flags

The Flags field is an 8-bit field that consists of the following indicators.

Bits 7 – 4 These bits are reserved.

Bit 3 The device-dependent-data-available bit (bit 3) indicates whether the specified device supports the device-dependent data in the termination status block. See Figure 1-31 on page 1-48 for a description of the TSB-available bits. The maximum amount of device-dependent data is specified in the Maximum Size for Completion Status field. When the bit is set to 1, the device supports device-dependent data available in the termination status block. When the bit is set to 0, the device does not support device-dependent data available in the termination status block.

Bit 2 The interrupt-support bit (bit 2) indicates whether the device uses the Device Interrupt Identifier port. When the bit is set to 1, the device uses the Device Interrupt Identifier port. When the bit is set to 0, the device does not support the Device Interrupt Identifier port.

Bit 1 The reset bit (bit 1) indicates whether the Reset Subsystem command resets all devices. When the Reset bit is set to 1, the Reset Subsystem command resets all devices. When the Reset bit is set to 0, the Reset Subsystem command does not reset all devices attached to the subsystem. In this case, individual Reset Device commands are required.

Bit 0 The load bit (bit 0) indicates whether the specified device requires a program load using the Initialize Device command. If this bit is set to 1, a program load is required. If this bit is set to 0, the device never requires a program load.

Reserved
This field is reserved.

Maximum Reset Control Block Interrupt Time
The Maximum Reset Control Block Interrupt Time field is a 32-bit field that indicates the maximum time the addressed device takes to perform a Reset Control Block Interrupt command. It can be specified in seconds, milliseconds, microseconds, or nanoseconds, depending on the setting of the 4 lower-order bits as indicated in Figure 1-43 on page 1-67.

Maximum Reset Interrupt Status Port Time
The Maximum Reset Interrupt Status Port Time field is a 32-bit field that indicates the maximum time the specified device takes to perform a Reset Interrupt Status Port command. It can be specified in seconds, milliseconds, microseconds, or nanoseconds, depending on the setting of the 4 lower-order bits as indicated in Figure 1-43 on page 1-67.

Maximum Command/Busy Status Time

The Maximum Command/Busy Status Time field is a 32-bit field that indicates the maximum time required to present status for an incoming command to the Command Busy/Status port. It can be specified in seconds, milliseconds, microseconds, or nanoseconds, depending on the setting of the 4 lower-order bits as indicated in Figure 1-43 on page 1-67.

Maximum Number of Queued Interrupts

The Maximum Number of Queued Interrupts field is a 32-bit field that indicates the maximum number of interrupts that can be internally queued by the specified device.

DIIP Address

The DIIP Address field is a 16-bit field that indicates the I/O address of the Device Interrupt Identifier port for the specified device. This field is valid only when the interrupt-support bit in the Flags field is set to 1.

DIIP Bit Position

The DIIP Bit Position field is a 16-bit field that indicates the bit position assigned to the specified device in the Device Interrupt Identifier port. This field is valid only when the interrupt-support bit in the Flags field is set to 1.

Maximum Size for Completion Status

The Maximum Size for Completion Status field is a 16-bit field that indicates the maximum number of bytes needed to hold the completion-status data block for the specified device. See "Read Completion Status" on page 1-60 for a complete description of this command.

Maximum Size for Diagnostic Status

The Maximum Size for Diagnostic Status field is a 16-bit field that indicates the maximum number of bytes needed to hold diagnostic-status data as a result of the Run Diagnostic Test command. See "Run Diagnostic Test" on page 1-70 for a complete description of this command.

Device Dependent Area Size

The Device Dependent Area Size field is a 16-bit field that indicates the number of bytes in the Device Dependent Configuration Data Area field.

Device Dependent Configuration Data Area

The Device Dependent Configuration Data Area field is a variable-length field containing device-dependent configuration data. The use of this field is device dependent and can be determined by reading the device specifications.

Run Diagnostic Test

The Run Diagnostic Test command causes the specified device to run diagnostic tests and to return the results to the location pointed to in the Memory Address 2 field. The data transfer uses the same semantics as the Read command.

The maximum size of the data area needed by the device to successfully store its diagnostic status is obtained by the Read Configuration command. The format, meaning, content, and uses of the data returned by the command are device dependent.

The Run Diagnostic Test command has the option to request an interrupt when the command is completed without error or with a suppressed error.

```
 3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
+-------------------------------+-+-----+-------+---------------+
| Enable Word 1                 |A| Opts|  Res  |0 0 0 0 0 1 0 1|
+-------------------------------+-+-----+-------+---------------+
| Device Dependent                                              |
+---------------------------------------------------------------+
| Memory Address of Test Results                                |
+---------------------------------------------------------------+
| Byte Count 1                                                  |
+---------------------------------------------------------------+
| Termination Status Block Address                              |
+---------------------------------------------------------------+
| Chain Address 1 or Device Dependent                           |
+---------------------------------------------------------------+
| Chain Address 2 or Device Dependent                           |
+-------------------------------+---------------+---------------+
| Chain ID                      | Reserved      | Dev. Dep. Size|
+-------------------------------+---------------+---------------+
/                               .                               /
|                     Device Dependent Area                     |
+---------------------------------------------------------------+
```

Figure 1-44. Run-Diagnostic-Test Control Block

The Byte Count 1 field specifies the number of bytes of diagnostic data to be returned.

Write

The Write command transfers data from the location in shared memory pointed to by the Memory Address 2 field to the device specified in the Attention port.

The Write command has the option to request an interrupt when the command is completed without error or with a suppressed error.

```
3 3 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1
1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
```

Enable Word 1	A	Opts	Res	0 0 0 0 0 0 1 0
Device Dependent				
Memory Address of Source				
Byte Count 1				
Termination Status Block Address				
Chain Address 1 or Device Dependent				
Chain Address 2 or Device Dependent				
Chain Id	Reserved			Dev. Dep. Size
. Device Dependent Area .				

Figure 1-45. Write Control Block

If an indirect list is specified, the semantics for data movement are defined in "Writing with a Base Control Block and Indirect Lists" on page 4-14.

If an indirect list is not specified, the Memory Address 2 field is the starting address of the source data in shared memory, and the Byte Count 1 field contains the number of bytes to be transferred.

The Byte Count 1 field is copied to a residual byte count, which is used as a counter during the transfer. As each byte is transferred from shared memory to the device, the residual byte count is decreased by 1. The operation is completed normally when the residual byte count is 0 and all bytes are received by the device. The address in the Memory Address 2 field is copied to a residual buffer address, which is used as the current pointer during the transfer.

These two residual values can be stored in the termination status block. After the values are stored, a program can determine the progress of the operation by obtaining the residual byte count and residual buffer address from the associated termination status block. When the residual byte count and residual buffer address are not stored in the termination status block and an error occurs, the entire operation should be performed again.

Transfer of data might be terminated because of an irrecoverable error at the device. When this happens, the major-error/exception and interrupt-requested bits in the End Status Word 1 are set to 1.

Immediate Commands

This section contains general descriptions of the immediate commands defined in the Locate mode. These commands are control oriented and use the same delivery interface as control-block commands (that is, delivery using the Command Interface and Attention ports). The immediate commands are:

- No Operation (Noop)
- Reset Device
- Reset Interrupt Status Port
- Reset Control Block Interrupt
- Reset Subsystem (software-controlled)
- Resume
- Run Immediate Diagnostic Test
- Suspend
- Reset Subsystem (hardware-controlled).

An immediate command uses one of two formats, depending on the value of the format identifier (bit 8) in the immediate command. Format 0 allows a command to use bits 31-16 for operation-code-dependent purposes. This gives the implementer additional flexibility. Format 1 reserves bits 31-16 of the immediate command. This allows architecture-defined commands the flexibility to grow. All commands described in this document are Format 1.

Immediate Command – Format 0

31–16	15	14	13–9	8	7–0
OP Code Dependent	DCI	DDI	OP Code Dependent	0	Operation Code

Immediate Command – Format 1

31–16	15	14	13–9	8	7–0
Reserved	DCI	DDI	OP Code Dependent	1	Operation Code

Figure 1-46. Immediate-Command Formats

The following is a description of the fields of an immediate command and their meanings:

Bits 31 – 16 Refer to bit 8.

Bit 15 The disable-command-interrupt (DCI) bit specifies whether the device requests an interrupt after successfully completing this command. If this bit is set to 1, the device does not request a system interrupt. If this bit is set to 0, the device requests an interrupt after completing the command and reports the status to the Interrupt Status port. For some commands, this bit is ignored.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is enabled to generate interrupts to the system. If this bit is set to 1, interrupts are disabled for the device specified. If this bit is set to 0, interrupts are enabled for the device specified. When interrupts are disabled, the device must be able to internally store interrupts until the system enables the interrupts.

Bits 13 – 9 The Op Code Dependent field is defined for each command.

Bit 8 The format-identifier bit determines the format used by the immediate command. If this bit is set to 0, Format 0 is used, allowing the immediate command to use bits 31-16 for operation-code-dependent purposes. If this bit is set to 1, Format 1 is used, causing bits 31-16 of the immediate command to be reserved.

Bits 7 – 0 The Operation Code field identifies the specific operation to be performed by the immediate command. Format 0 operation codes are implementation dependent. Format 1 operation codes are defined in the following figure.

Operation-Code Bits	Description
7 6 5 4 3 2 1 0	
0 0 0 0 0 0 0 0	Reset device
0 0 0 0 0 0 0 1	Reserved
0 0 0 0 0 0 1 0	Noop
0 0 0 0 0 0 1 1	Reserved
0 0 0 0 0 1 0 0	Reset Interrupt Status port
0 0 0 0 0 1 0 1	Reserved
0 0 0 0 0 1 1 0	Reserved
0 0 0 0 0 1 1 1	Reserved
0 0 0 0 1 0 0 0	Reset Control Block interrupt
0 0 0 0 1 0 0 1	Reserved
0 0 0 0 1 0 1 0	Reserved
0 0 0 0 1 0 1 1	Reserved
0 0 0 0 1 1 0 0	Device dependent
0 0 0 0 1 1 0 1	Device dependent
0 0 0 0 1 1 1 0	Device dependent
0 0 0 0 1 1 1 1	Device dependent
0 0 0 1 0 0 0 0	Reserved
0 0 0 1 0 0 0 1	Reserved
0 0 0 1 0 0 1 0	Run Immediate Diagnostics
0 0 0 1 0 0 1 1	Device dependent
0 0 0 1 0 1 0 0	Reserved
0 0 0 1 0 1 0 1	Reserved
0 0 0 1 0 1 1 0	Device dependent
0 0 0 1 0 1 1 1	Device dependent
0 0 0 1 1 0 0 0	Reserved
0 0 0 1 1 0 0 1	Reserved
0 0 0 1 1 0 1 0	Reserved
0 0 0 1 1 0 1 1	Reserved
0 0 0 1 1 1 0 0	Reserved
0 0 0 1 1 1 0 1	Reserved
0 0 0 1 1 1 1 0	Reserved
0 0 0 1 1 1 1 1	Suspend
0 0 1 0 0 0 0 0	Resume
All combinations of bits 7-0 not shown are reserved.	

Figure 1-47. Operation-Code Bits – Format 1

Some immediate commands use interrupts to reply to the system master. The semantics of interrupts are defined in "Signalling Protocol" on page 3-13. Details on exception handling are found in "Exception and Error Handling" on page 5-4.

No Operation (Noop)

The No Operation command can change the interrupt-enablement status of the device specified in the Attention port without causing any other effect.

This command can request an interrupt, upon successful completion of the command, by setting the DCI bit to 0.

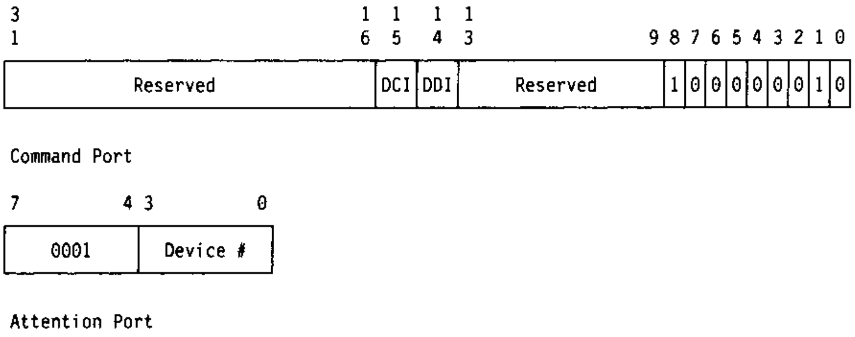


Figure 1-48. No-Operation Command Structure

Bit 15 The disable-command-interrupt (DCI) bit specifies whether the device requests an interrupt after successfully completing this command. If this bit is set to 1, the device does not request a system interrupt. If this bit is set to 0, the device requests an interrupt after completing the command.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.

Bits 13 – 9 These bits are reserved.

Programming Note: A Noop command will be rejected by a device that is currently executing a control block.

Reset Device

The Reset Device command is used to put the specified device into a known state. A device number of 0 directs the command to the subsystem. The known device state is defined as follows:

- All pending interrupts for the device are cleared. If the device supports Device Interrupt Identifier ports, the internal count of logical interrupts is set to 0, and the DIIP bit is set to 0.
- Any control block or control-block chain that is currently being executed is purged.
- Any control-block chain that is suspended at the device is purged.
- The device is restored to its default setting (device interrupts are enabled or disabled depending on the value of the DDI bit in the Immediate Reset Device command).
- The device executes any device-dependent reset activities.

A Reset Device command always requests an interrupt upon successful completion.

31	16	15	14	13	12	11	9	8	7	6	5	4	3	2	1	0
Reserved		DCI	DDI	OP2	OP1	Reserved		1	0	0	0	0	0	0	0	0

Command Port

7	4	3	0
0000		Device #	

Attention Port

Figure 1-49. Reset Device Command Structure

Bit 15 The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is always requested on completion of this command.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.

Bit 13 The option-2 (OP2) bit specifies whether the subsystem and all its attached devices are reset, or whether just the subsystem is reset. If the device number is not 0, the option-2 bit is ignored. If the device number is 0 and this bit is set to 0, the subsystem and all its attached devices are reset. If the device number is 0 and this bit is set to 1, the subsystem is reset, but all attached devices are not reset.

Bit 12 The option-1 (OP1) bit specifies whether a partial or complete reset of the device is performed. When this bit is set to 0, the device is reset completely so that the device must be reinitialized. The device's internal memory is reset. If this bit is set to 1 and the device has already been initialized, the device is partially reset and does not need to be reinitialized. The device's internal memory is not reset.

Programming Notes:

1. The Reset Device command should be used to force a device into a known state. It should, however, be used with care in error-recovery situations because all pending interrupts are lost when the device is reset.

2. When a Reset Device command is executed to device 0, with the option-2 bit set to 1, only the internal states of commands directed to device 0 are affected. Other devices attached to the subsystem are not reset.

3. An interrupt from a prior command for a device might be present in the Interrupt Status port when a Reset Device command is issued. A program must receive and ignore this interrupt by reading and resetting the Interrupt Status port without processing the interrupt. The program should test for an interrupt identifier of 0 because a Reset Device command always enqueues an

interrupt with an identifier of 0 when the command is completed without error.

For details on error handling see "Subsystem Level" on page 5-6.

Reset Interrupt Status Port

This command resets an interrupt in the Interrupt Status port. It clears the Interrupt Status port only if the device specified in the Attention port is the same as the device associated with the interrupt in the Interrupt Status port. If the Interrupt Status port is cleared, the interrupt-valid bit in the Command Busy/Status port is set to 0. If the device specified does not match the device in the Interrupt Status port, the Interrupt Status port is not cleared, and the disable-device-interrupt bit is the only bit checked by the device.

When the Interrupt Status port is cleared by this command, the physical interrupt to the system master is reset unless the Device Interrupt Identifier port is used. When the Device Interrupt Identifier port is used, the interrupt is reset only when all bits in the Device Interrupt Identifier port are 0.

This command does not request an interrupt upon completion.

31 … 16	15	14	13 … 9	8	7	6	5	4	3	2	1	0
Reserved	DCI	DDI	Reserved	1	0	0	0	0	0	1	0	0

Command Port

7 … 4	3 … 0
1110	Device #

Attention Port

Figure 1-50. Reset Interrupt Status Port Command Structure

Bit 15	The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is never requested on completion of this command.
Bit 14	The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.

Programming Notes:

1. A device that is currently executing a command must accept the Reset interrupt Status Port command. This command allows a program to reset interrupts that occur within a chain of control blocks without having to issue a Suspend command to the device.
2. Refer to "Signalling Protocol" on page 3-13 for details on interrupt handling.

Reset Control Block Interrupt

The Reset Control Block Interrupt command resets logical interrupts received for control-block commands when the Device Interrupt Identifier port is used. It causes the internal logical-interrupt count of the device specified to be decreased by the number specified in the Count field.

When a count of 0 is specified, the logical-interrupt count is not modified; the disable-device-interrupt bit is the only bit that effects a change.

This command never requests an interrupt of the system master.

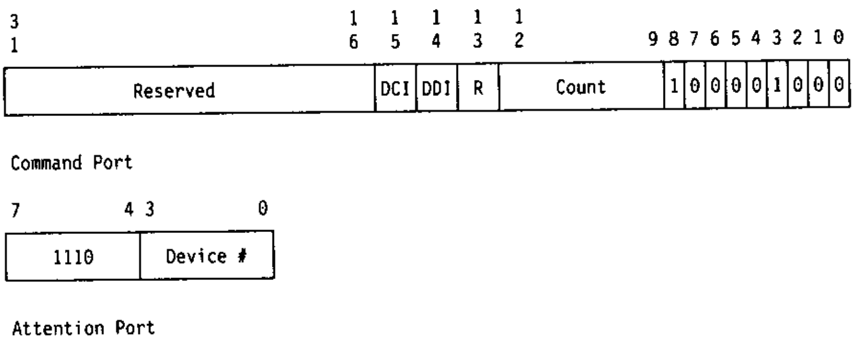


Figure 1-51. Reset Control Block Interrupt Command Structure

Bit 15 The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is never requested on completion of this command.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.

Bits 12 – 9 The Count field is a 4-bit field that specifies the number of logical interrupts to be reset for the device specified. A value of 0 does not affect the logical-interrupt count.

When the count of logical interrupts becomes less than or equal to 0, the bit in the Device Interrupt Identifier port that is assigned to the device is set to 0. When all bits in the Device Interrupt Identifier port are set to 0 and the Interrupt Status port does not indicate an interrupt, the physical interrupt to the system master is reset.

When the Device Interrupt Identifier port is being used, a program can reset logical interrupts for control blocks singly or in multiples by using this command. The program does not have to wait for a physical interrupt in order to reset logical interrupts.

Programming Notes:

1. A device that is currently executing a control-block command must accept the Reset Control Block Interrupt command. This allows a program to reset interrupts that occur within a chain of control blocks without having to issue a Suspend command to the device.
2. When specifying the count value in this command, do not exceed the number of termination status blocks that have the interrupt-requested bits in the End Status Word 1 set to 1; otherwise, loss of physical interrupts can result.

   If the logical-interrupt count for a device becomes less than or equal to 0, the bit in the Device Interrupt Identifier port associated with the device is set to 0, and the subsystem does not generate a physical interrupt for logical interrupts from the device. While in this state, the logical interrupts continue to be posted in the associated termination status blocks, and the logical-interrupt count is increased. A logical-interrupt count greater than 0 causes the device to set the associated bit in the Device Interrupt Identifier port to 1 and to raise a physical interrupt if the interrupt is enabled.
3. When the Device Interrupt Identifier port is being used, the Reset Control Block Interrupt command is used to clear interrupts for control blocks. However, when a control block cannot be read or the termination status block cannot be stored, the device puts an interrupt into the Interrupt Status port. This interrupt must be cleared using the Reset Interrupt Status Port command.
4. Refer to "DIIP Interrupts (Multiple Form)" on page 3-26 for details on interrupt handling.

Reset Subsystem (Software Controlled)

The Reset Subsystem command has the same format as the Reset Device command, except that it must be issued to device 0.

The Reset Subsystem command can be used to perform a software-controlled reset (soft reset) of the subsystem and all attached devices. The option-2 bit specifies whether the subsystem and all attached devices are reset or whether the subsystem alone is reset.

When the command is initiated, all device activity for that subsystem is stopped, and the subsystem is reset to a known state.

After the command is completed, the Interrupt Status port is set to 0, and the interrupt-valid bit in the Command Busy/Status port is set to 1, indicating that a reset of the subsystem has been completed and an interrupt has been requested. The results of the Reset Subsystem command are:

- The enable-interrupts, DMA, and reset-reject bits in the Subsystem Control port are set to 0.
- Any previous physical interrupt to the system is cleared.
- All activity on the subsystem not related to resetting the subsystem ends.
- If the subsystem supports the Device Interrupt Identifier port, all bits in this port are set to 0.
- The Command and Attention ports are set to 0.
- When the option-2 bit is set 0, all devices attached to the subsystem perform the following actions:
  - All pending interrupts are cleared.
  - If the device uses the Device Interrupt Identifier port, the logical-interrupt count is set to 0.
  - Any suspended chain operation is purged.
  - The device interrupts are enabled.
  - The device is put into a state enabling execution of control-block commands.
  - The device performs any device-dependent reset activities.
- If the option-1 bit is set to 0 (indicating a complete reset), the subsystem and attached devices perform a power-on self-test and might require reinitialization.

```
3                                 1   1   1   1   1   1
1                                 6   5   4   3   2   1          9 8 7 6 5 4 3 2 1 0
+--------------------------------+---+---+---+---+------------+-+-+-+-+-+-+-+-+-+
|            Reserved            |DCI|DDI|OP2|OP1|  Reserved  |1|0|0|0|0|0|0|0|0|
+--------------------------------+---+---+---+---+------------+-+-+-+-+-+-+-+-+-+

Command Port

7          4 3          0
+-----------+-----------+
|   0000    |   0000    |
+-----------+-----------+

Attention Port
```

Figure 1-52. Reset Subsystem Command Structure

Bit 15 The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is always requested on completion of this command.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the subsystem is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the subsystem. If this bit is set to 0, interrupts are enabled from the subsystem.

Bit 13 The option-2 (OP2) bit specifies whether the subsystem and all its attached devices are reset or whether the subsystem alone is reset. If this bit is set to 0, the subsystem and all its attached devices are reset. If this bit is set to 1, the subsystem is reset, but all attached devices are not reset.

Bit 12 The option-1 (OP1) bit specifies whether a partial or complete reset of the subsystem is performed. When this bit is set to 0, the subsystem is reset completely so that the subsystem must be reinitialized; the subsystem's internal memory is reset. If this bit is set to 1, the subsystem is partially reset and does not need to be reinitialized; the subsystem's internal memory is not reset.

Programming Notes:

This command differs from the hardware-controlled reset (hard reset) in that it does not cause a reset of the internal logic that controls the subsystem. Additionally, the soft reset depends on the availability of the Command and Attention ports for delivery, whereas the hardware-controlled Reset Subsystem command does not.

1. Programmable-option-select data is not reset by this command.
2. After a subsystem reset begins, the Command and Attention ports are not available until the reset has been completed.
3. The system software should use the software-controlled reset (soft reset) when it is unable to recover a device through the Reset Device command. The Reset Subsystem command must be used carefully because it ends all commands in progress on the subsystem. It also causes any interrupts queued by attached devices to be lost.
4. If the subsystem fails to complete a software-controlled reset within a specified time, the software in the system master should use a hardware-controlled reset.
5. The software-controlled reset normally is completed by setting the enable-DMA (DMA) and enable-interrupt bits in the Subsystem Control port to 0. These bits remain in that state unless they are changed after the reset starts. If the enable-interrupt bit is not changed, the subsystem will not generate an interrupt.
6. A subsystem reset causes the busy bit in the Command Busy/Status port to be set to 1 and causes the reject bit in the Command Busy/Status port to be set to 0. Commands submitted normally through the Attention port have the same effect on the busy and reject bits. Programs cannot use the busy and reject bits to distinguish between commands submitted through the Attention port and commands that cause a subsystem reset (which take longer). Therefore, to avoid resource conflicts, programs should maintain an external indication of any subsystem-reset activity.

Resume

Resume is an immediate command that puts the device that is specified in the Attention port into a state that enables it to execute all control-block commands. The Resume command is used to restart an active control-block chain that has been halted by a Suspend command. The device might not be able to restart if it is suspended because of internal conflicts or conditions. If the interrupt queue is full, the device remains in a suspended state until its queue is no longer full.

If the Resume command requests an interrupt on completion, the interrupt is requested before execution of a suspended control-block chain is resumed.

31–16	15	14	13–9	8	7	6	5	4	3	2	1	0
Reserved	DCI	DDI	Reserved	1	0	0	1	0	0	0	0	0

Command Port

7–4	3–0
0001	Device #

Attention Port

Figure 1-53. Resume Command Structure

Bit 15 The disable-command-interrupt (DCI) bit specifies whether the device requests an interrupt after successfully completing this command. If this bit is set to 1, the device does not request a system interrupt. If this bit is set to 0, the device requests an interrupt after completing the command and reports the status to the Interrupt Status port.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is enabled to generate interrupts to the system. If this bit is set to 1, interrupts are disabled for the device specified. If this bit is set to 0, interrupts are enabled for the device specified.

Run Immediate Diagnostic Test

The Run Immediate Diagnostic Test command is used to initiate a diagnostic test to the specified device. The results of the test are returned as an interrupt in the Interrupt Status port upon command completion. The interrupt code returned indicates whether the command was completed with or without errors. The command always requests an interrupt upon completion.

A more comprehensive diagnostic report can be obtained with the Run Diagnostic Test control-block command.

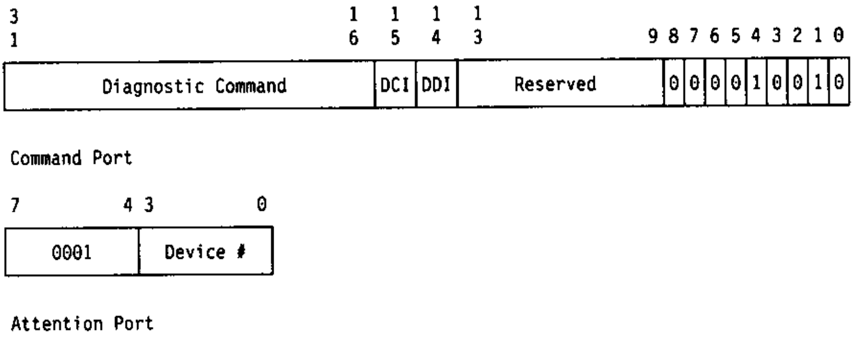


Figure 1-54. Run Immediate Diagnostic Test Command Structure

Bits 31 – 16 The Diagnostic Command field is a 16-bit field that indicates the diagnostic test to be performed. The code used is device dependent.

Bit 15 The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is always requested on completion of this command.

Bit 14 The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.

Bits 13 – 9 These bits are reserved.

Suspend

The Suspend command is used to put the specified device in the Attention port into a state that no longer executes control-block commands. A Resume, Reset Device, or Reset Subsystem command is needed to restore the device to a state in which it can execute control-block commands again.

When the command is accepted by a device that is not executing a command, the internal state is set to show that the control-block execution is disabled.

When the command is accepted by a device that is executing a control block or a control-block chain, the execution of the Suspend command is deferred until the current control block or chain element has completed its execution. The completed command always stores the termination status block. The suspend bit in End Status Word 1 of the termination status block is set to 1 to indicate that the associated control block was the last one completed before the suspend request was completed. If this command is issued while a control block in a chain element, other than the ending chain element, is being run, the next element in the chain will be executed only after a subsequent Resume command is received.

If the Suspend command is issued while a control block that is not a chain element, or is the ending chain element, is being executed, the current control block will be completed, and all subsequent control blocks will be rejected until a Resume command is received.

When the Suspend command is completed, the device will reject any control-block commands directed to it until either a Resume, Reset Device, or Reset Subsystem command has been successfully executed. The device will accept any non-control-block command or direct hardware command directed to it.

A Suspend command always requests an interrupt upon completion. If a control block is being executed at the device, the interrupt will be a control-block interrupt. If a control block is not being executed, the interrupt returned will be an immediate command completed, without error.

31–16	15	14	13–9	8	7	6	5	4	3	2	1	0
Reserved	DCI	DDI	Reserved	1	0	0	0	1	1	1	1	1

Command Port

7–4	3–0
0001	Device #

Attention Port

Figure 1-55. Suspend Command Structure

Bits 31 – 16	These bits are reserved.
Bit 15	The disable-command-interrupt (DCI) bit is ignored by the device. An interrupt is always requested on completion of this command.
Bit 14	The disable-device-interrupt (DDI) bit specifies whether the device is allowed to generate interrupts to the system. If this bit is set to 1, interrupts are disabled from the device specified. If this bit is set to 0, interrupts are enabled from the device specified.
Bits 13 – 9	These bits are reserved.

Programming Notes:

1. If a chain operation is suspended, the termination status block can be examined to determine the next control block to be executed when the chain is resumed. For more information, refer to "Signalling Protocol" on page 3-13.
2. A Suspend command is not necessary before a Reset Control Block Interrupt or Reset Interrupt Status Port command is issued to a device executing a control-block chain because these commands are always accepted by the device.
3. If a control block cannot be read, or the termination status block cannot be stored, and a Suspend command is issued, the program can determine that the Resume command will not be able to restart the chain.

4. The Suspend command always requests an interrupt upon completion. By examining the interrupt value returned as a result of a Suspend command, a program can determine the device state at the time the command was executed. If the device is not executing a control block when the Suspend command is accepted, the interrupt returned indicates that the immediate command was completed without error. If the device is executing a control block when the Suspend command is accepted, the command requests a control-block interrupt for the device. If the interrupt returned indicates that a hardware failure occurred while executing a control block, the program determines that the device was executing a control block and that the chain cannot be resumed. If the interrupt returned is not for a hardware failure, the program might have to examine the completion status of the termination status blocks associated with the device to determine whether the chain can be restarted.

Reset Subsystem (Hardware Controlled)

A program uses the hardware-controlled Reset Subsystem command to stop the activity of all devices attached to the subsystem and to restore the subsystem and attached devices to a known state. The state of each device after the reset is defined in the description of the Reset Device command.

This command is initiated by the reset bit in the Subsystem Control port, which controls a reset signal internal to the subsystem. If the subsystem and its attached devices require an initial program load, they are put into a state in which they can accept an Initialize Device command to receive their program loads. Upon completion, the command stores an interrupt in the Interrupt Status port. POS data is not reset by this command. If the Device Interrupt Identifier ports are supported, these ports are set to 0.

A Reset Subsystem command is originated by the program. This is done by writing a 1 to the reset (RST) bit of the Subsystem Control port, followed by writing a 0 value to the reset bit in an implementation-defined amount of time. In other words, the reset bit is switched from 0 to 1 to 0.

Setting the reset-control bit of the Subsystem Control port to 1 clears the physical interrupt line from the subsystem and causes all device activities on the subsystem to abend. The following values are set in the Command Busy/Status port:

- Busy (B) = 1
- Interrupt valid (IV) = 0
- Reject (REJ) = 0.

The enable-subsystem interrupt (EI), enable-DMA (DMA), and reset-reject (RR) control bits in the Subsystem Control port are set to 0 by the subsystem. EI and DMA remain 0 unless specifically set by the program while the device-reset operation is in progress. All activities not related to subsystem reset are ended.

When the subsystem senses the transition of the reset-control bit from 1 to 0, it completes reset actions for the subsystem and all attached devices.

At the completion of this command, the Interrupt Status port and all bits in the Command and Attention ports are set to 0, and all device interrupts are cleared.

7	6	5	4	3	2	1	0
RST	R	RR	R	SD	SD	DMA	EI

Figure 1-56. Subsystem Control Port (Hardware-Controlled Subsystem Reset)

Programming Notes:

1. Using the hardware-controlled Subsystem Reset command ends all other commands in progress on the subsystem and causes all interrupts from attached devices to be lost.
2. This command should be used in error-recovery procedures by programs in the system master when all other recommended retry actions have failed.
3. If the subsystem fails to complete a Reset Subsystem command within the maximum time defined for it by each implementation, the software unit may have reset the reset bit in the Subsystem Control port to 0 too rapidly. In this case, setting the reset bit to 1 again will restart the Reset Subsystem command sequence.

4. Notice that the normal completion of this command sets the Subsystem Control port DMA and EI control bits to 0. These control bits should be set to 1 by the program in the system master after it has switched the reset-control bit to 0 or after the busy bit in the Command Busy Status port is set to 0. If these control bits are not reset, DMA and interrupts for the subsystem remain disabled after the Reset Subsystem command has been completed.

5. A Subsystem Reset command causes the busy bit in the Command Busy/Status port to be set to 1 and the reject bit in the Command Busy/Status port to be set to 0. Commands submitted normally through the Attention port have the same effect on the busy and reject bits. By examining these bits, programs cannot distinguish between commands submitted through the Attention port and commands that cause a subsystem reset (which take longer). Therefore, to avoid resource conflicts, programs should maintain an external indication of any subsystem-reset activity.

Chapter 2. Physical Level

The physical level provides support for gaining access to data and control areas in shared memory and I/O address space and for interrupting the system master.

At the physical level, the architecture defines the control areas in I/O address space that are used to deliver immediate commands and command control blocks to the various devices in a subsystem. It also defines how devices gain access to data and command control blocks in shared memory, as well as how a device or subsystem interrupts a system master.

Structure

The following figure shows the structure of the physical level in terms of the components used to provide the command and command-control-block delivery.

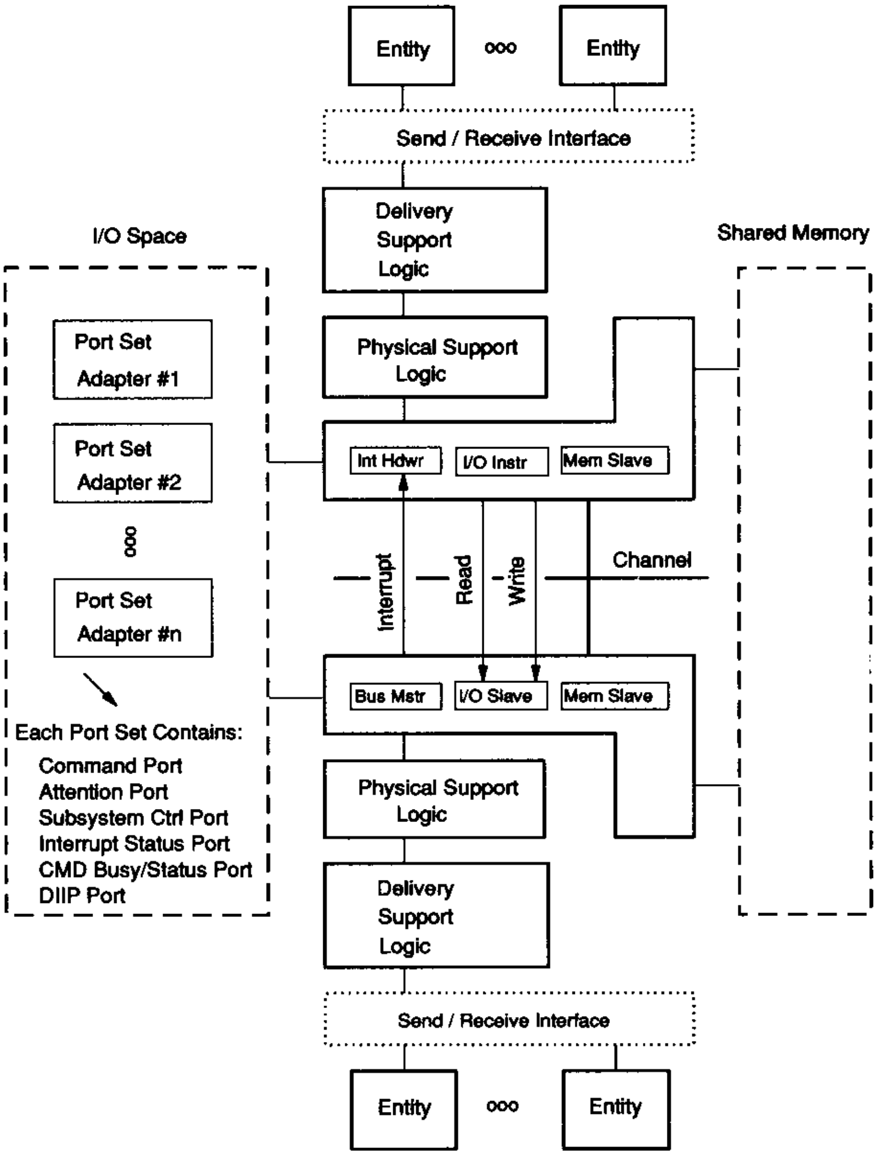


Figure 2-1. Locate Mode Physical Level

Figure 2-1 shows the physical-level structure for a configuration with a single system master and a single subsystem. The physical-level services are provided by a combination of hardware support, physical-support logic, and control areas in I/O address space.

System-Master Hardware Support

The hardware support in a system master includes an interrupt controller, the instructions used to gain access to information in I/O address space, and shared-memory-slave support.

The interrupt controller supports the use of interrupts as a means of signalling the system master from one or more subsystems.

The I/O instructions provide the system master with access to information in control areas in I/O address space that are shared with the subsystem.

The memory-slave support provides access to data and control information either by bus masters or by memory instructions.

Subsystem Hardware Support

The hardware support in a subsystem includes I/O-slave support and bus-master support.

The I/O-slave support provides a means for the logic in a system master to initialize the subsystem, enable access to the subsystem, signal the subsystem, and identify the subsystem as the source of the system interrupts.

The bus-master support provides access to data and control information in shared memory. The subsystem interface to the system is through the following ports:

- Command port
- Attention port
- Subsystem Control port
- Interrupt Status port
- Command Busy/Status port
- Device Interrupt Identifier port (optional).

Support Logic

The physical-support logic implements the services, protocols, and functions that support the sharing of control areas by different command and command-control-block delivery flows. It also provides the support necessary for access to the shared memory used by a subsystem for command control blocks and data in data buffers.

Memory Address Space

One of the most important aspects of the physical level is that it provides access to data and control information in shared memory. The delivery-level protocols need access to command control blocks, which are physically located in system-master memory that is shared with a subsystem.

The locations of the data buffers, used during data transfer, are also assigned locations in shared memory. The locations of the data buffers are determined by the delivery-level support and can change from command control block to command control block.

The command control blocks are assigned locations within shared memory. The starting location of the command control block is provided by the command-delivery support for each delivery request.

The shared-memory addresses that are supported are defined during system configuration.

I/O Address Space

The physical-level services provide access to a set of control areas (ports) in I/O address space that are shared by the command and command-control-block delivery for all the entities in a system or subsystem.

Note: This minimizes the amount of I/O address space required, but it requires the management of the sharing of this address space.

The range of I/O addresses assigned to each subsystem is determined during system configuration. A description of the control areas is defined by the architecture for the Locate mode and can be found in "I/O Address Space" on page 1-15.

Physical-Level Services

The services provided by the physical level are used to gain access to data and control information and to support interrupting a system master from any subsystem on the Channel. The services provided are:

- Push
- Pull
- Interrupt.

These services are used by the delivery-level logic in both the system master and subsystems to gain access to control areas in I/O address space, to gain access to command control blocks and data buffers in shared memory, and to cause an interrupt of the system master.

Note: A push from a system master to a control area in the I/O address space of the Attention port causes an interrupt of the associated subsystem upon completion.

Push and Pull

The delivery-support logic uses the I/O-address-space form to move control parameters between the system master and the subsystem. It uses the shared-memory form to gain access to command control blocks and data buffers.

A system master uses I/O instructions (IN or OUT) to gain access to control areas in I/O address space.

A subsystem uses bus-master, DMA-type operations to gain access to the shared memory for the specific subsystem or entity that performs the service.

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

The following conceptual service primitive illustrates the parameters required when the push service is invoked:

```
PUSH(unit id, addr space, from addr, to addr, length, return code)
```

The ***unit ID*** parameter identifies the system master or subsystem that applies.

The ***address space*** parameter indicates whether the ***to address*** parameter refers to a location in shared memory or in I/O address space.

The ***from address*** parameter identifies the location of the information to be pushed into the location that is indicated by the ***to address*** parameter.

The ***to address*** parameter identifies the destination in I/O address space or shared memory of the information that is indicated by the ***from address*** parameter.

The ***length*** parameter indicates the number of bytes to be pushed into the location in I/O or shared memory, identified by the ***to address*** parameter.

The ***return code*** parameter indicates the success or failure of the push operation.

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

The following conceptual service primitive illustrates the parameters required when the pull service is invoked:

```
PULL(unit, addr space, from addr, to addr, length, return code)
```

The ***unit ID*** parameter identifies the system master or subsystem that applies.

The ***address space*** parameter indicates whether the ***to address*** parameter refers to a location in shared memory or in I/O address space.

The ***from address*** parameter identifies the location of the information to be pulled into the location that is indicated by the ***to address*** parameter.

The ***to address*** parameter identifies the destination in the caller's shared memory of the information that is indicated by the ***from address*** parameter.

The ***length*** parameter indicates the number of bytes to be pulled into the location that is identified by the ***to address*** parameter.

The ***return code*** parameter indicates the success or failure of the pull operation.

Interrupt

In addition to the push and pull services, the physical level also provides a service to interrupt the system master.

Note: As stated previously, there are command-delivery protocols that can be used to cause an interrupt to a subsystem. This is accomplished by using the push service primitive to put the appropriate control information into the control area in I/O address space that is assigned to the subsystem Attention port, not by using the interrupt service.

The following conceptual service primitive illustrates the parameters required when the service is invoked:

```
INTERRUPT (return code)
```

There are no parameters required by the interrupt service primitive; however, the interrupt service does have a return code.

The ***return code*** parameter indicates the success or failure of the interrupt operation.

Data Delivery

In subsystems, the bus-master support can also be used to transfer data to and from entities in the subsystem. The command control blocks contain the location of the data in shared memory to which the subsystem entities have access. The push and pull services can be used by these entities as well as by the delivery-support logic to gain access to the data in shared memory.

Note: Different forms of address transformation might be required to establish addressability between a system master or subsystem's local shared memory and the shared memory in the command control blocks. The specific address transformation is determined by the particular system master or subsystem implementation.

Physical-Level Protocols

The protocols defined at the physical level for access to control areas in I/O address space are the same as those required for Channel operations to an I/O slave.

The protocols defined for access to command control blocks and data buffers in shared memory are the same as those required for Channel operations to a shared-memory slave.

The protocol defined for interrupting a system master from a subsystem is the same as the Channel protocol for interrupting a system master.

The protocols at the delivery level for command and command-control-block delivery define the uses of the various control areas. See Chapter 3, "Delivery Level" on page 3-1 for additional information.

Notes:

Chapter 3. Delivery Level

The delivery level provides the set of services and protocols that support the delivery of requests from programs in a system master to devices in a subsystem. The primary form for these requests is one or more control blocks that have been grouped together by addresses. This grouping of control blocks includes the command control block, a status block, and an optional indirect list. When the request consists of more than one control block, the next one is pointed to by a chain address maintained for that purpose in each command control block.

The delivery level also supports the delivery of immediate and hardware-controlled commands to a subsystem. The immediate commands are limited to control-type functions that occupy one doubleword. The hardware-controlled commands are used to directly control the basic operation of the subsystem.

The Locate mode architecture defines the delivery-level protocols that are used with bus-master type subsystems designed to operate as shared-logic control units. It also defines the functions and identifies the support logic required in both the system master and the subsystem to implement the services and protocols of the delivery level.

Structure

The following describes the delivery-level structure and the services that support command and control-block delivery between entities.

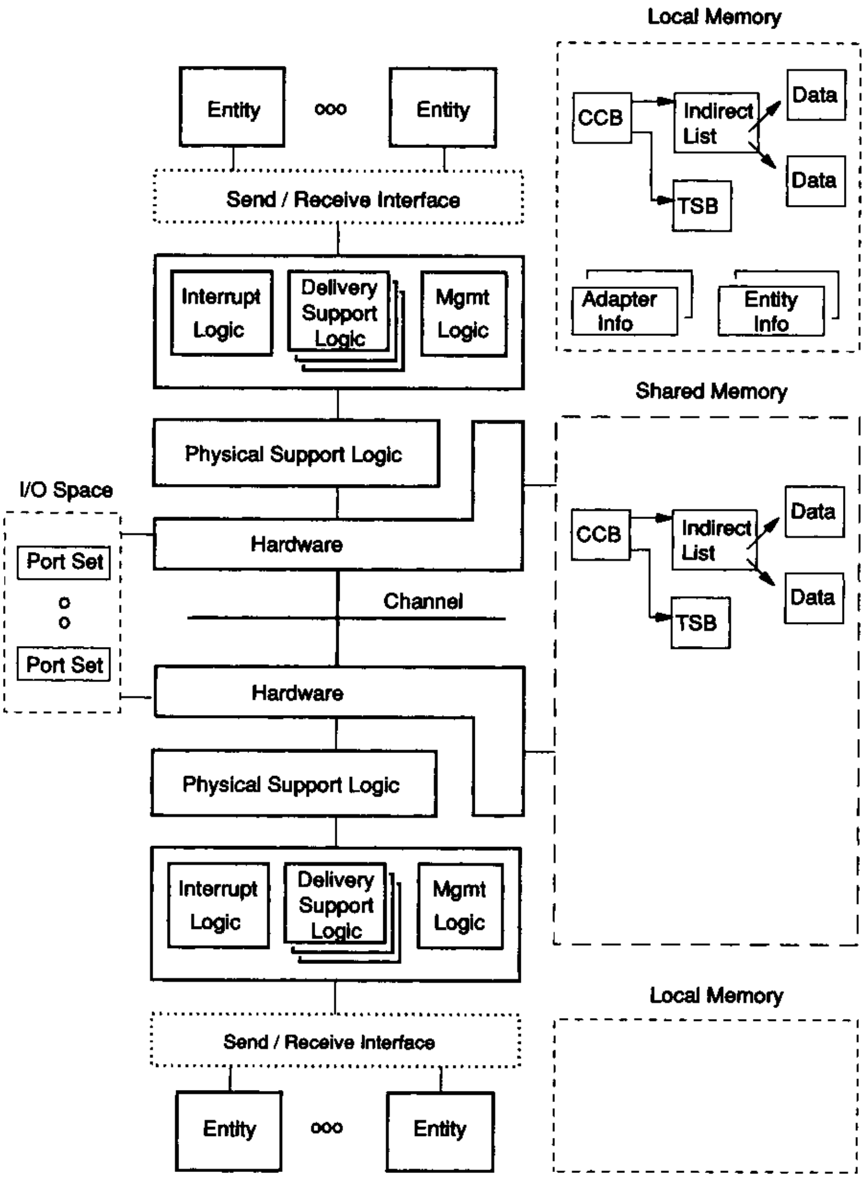


Figure 3-1. Overall Delivery-Level Structure

Figure 3-1 shows that the delivery support in the system master has multiple instances of the request-delivery logic and the local control areas needed to maintain the states of both multiple Locate mode subsystems as well as multiple devices associated with each of them.

The figure shows that in a system master, there is typically a single instance of the interrupt handler that supports the completion of the request-delivery protocol. It also shows that in a subsystem, there is a single instance of both the request-delivery support and the system-interrupt support. These must be managed to allow them to support the multiple request-delivery protocols.

Figure 3-1 on page 3-2 also shows that delivery support is provided to programs by the send and receive interfaces and that the delivery logic uses the service of the underlying physical level.

The send and receive interfaces provide the local operating-environment policy support for interfaces to the delivery-level support and for scheduling this support on the processing-level threads.

Processing-level threads of execution are used in a multitasking programming environment.

Figure 3-1 on page 3-2 shows two views of a sample delivery request (set of coupled command control blocks). The request control-block structure needs to be understood in the local address space seen by the system-master entity and the request delivery-level support, as well as in the shared memory understood by the subsystem entities and delivery-level support.

This happens when a program in a system master is running in virtual or protected mode. The program in the system master would use logical addresses when building control blocks or referring to fields within the command control blocks. On the other hand, the device in the subsystem would be using physical addresses when fetching, storing, or referring to fields within the command control blocks.

From the delivery-support view, all addresses must be physical addresses in shared memory. The programs in the system master can provide a local view by prefixing each control block with the necessary logical addresses.

Because the request delivery-level protocols use the control areas in the I/O space to deliver requests and to process any related interrupts, these are also shown as part of the overall structure.

Locate mode commands can be delivered by one of two methods:

- The hardware-controlled method involves writing to the Subsystem Control port, which causes the subsystem to react immediately to the functions encoded in the port bits. For the definition of these functions, see "Subsystem Control Port" on page 1-20.
- The software-controlled method involves writing to the Command and Attention ports and waiting for the command handler in the subsystem to accept or reject the command. For the definition of these functions see "Command Port" on page 1-18 and "Attention Port" on page 1-18.

Resetting the subsystem always forces the subsystem to a known state, regardless of the state of the subsystem and its attached devices. The specific actions for a reset of the subsystem are described in "Reset Subsystem (Hardware Controlled)" on page 1-90.

Immediate-Command Delivery

Immediate commands are mainly device directed and are control oriented. Similar to control-block commands, immediate commands use the Command and Attention ports for delivery.

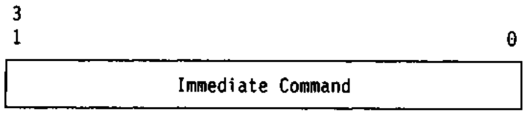


Figure 3-2. Command Port—Immediate Commands

The 32-bit immediate command is put into the Command port.

In addition to the Command port, the immediate-command delivery requires the use of the Attention port.

An immediate command that is written to the Command and Attention ports cannot be executed until the device has verified that the whole command has been accepted by the subsystem. After acceptance has been verified, the busy bit in the Command Busy/Status port is set to 0. The command type is determined by examining the attention code in the Attention port. When the command is determined to be an immediate command, the operation code in the Command port is decoded to determine the device actions to be performed. The following figure shows the format of the Attention port when it is used for immediate-command delivery.

```
Reset Device/Reset Subsystem

 7          4  3          0
┌────────────┬────────────┐
│  0 0 0 0   │Device Number│
└────────────┴────────────┘


Reset Control Block Interrupt/Reset Interrupt Status Port

 7          4  3          0
┌────────────┬────────────┐
│  1 1 1 0   │Device Number│
└────────────┴────────────┘


All Other Immediate Commands

 7          4  3          0
┌────────────┬────────────┐
│  0 0 0 1   │Device Number│
└────────────┴────────────┘
```

Figure 3-3. 8-Bit Attention Port – Immediate Commands

Bits 3 – 0	For a description of the Device Number subfield (bits 3-0), refer to the definition of the Attention port in "Attention Port" on page 1-18.

Control-Block Command Delivery

Control-block commands are mainly data-transfer oriented and use the Command and Attention ports for delivery.

Control-block commands are identified by examining the attention code in the Attention port. For a control-block command, the control-block starting address is contained in the Command port.

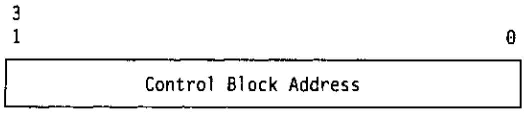


Figure 3-4. Command Port – Control-Block Commands

The address of a control block placed in the Command port must be the 32-bit, doubleword-aligned physical address of the location in shared memory where the first or only command control block is located.

In addition to the Command port, the command-control-block delivery requires the use of the Attention port. The following figure shows the format of the Attention port when it is used for command-control-block delivery.

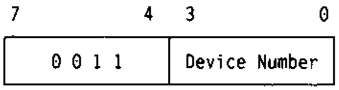


Figure 3-5. 8-Bit Attention Port – Control-Block Commands

Bits 3 – 0 For a description of the Device Number subfield (bits 3-0), refer to the definition of the Attention port in "Attention Port" on page 1-18.

Delivery-Level Services

The services provided by the delivery level are used to deliver requests from an entity (software) in a system master to an entity (device) in a subsystem. Three types of service are provided:

- Immediate-command delivery
- Hardware-controlled command delivery
- Command-control-block delivery.

A description of each of these services is provided in the following sections.

Immediate-Command Delivery Service

The immediate-command delivery service is used to pass an immediate command to a device or subsystem, using the services provided by the underlying physical level for access to the appropriate control areas in I/O address space.
The following conceptual service primitive illustrates the parameters required when invoking the immediate-command delivery service:

```
DELIVER_IMMEDIATE (destination, attention code, command, return code)
```

The ***destination*** parameter identifies the adapter (subsystem) and entity (device) to which the immediate command is to be delivered.

The ***attention code*** parameter contains the value to be placed in the Attention port on command delivery.

The ***command*** parameter contains the immediate command to be delivered.

The ***return code*** parameter is returned to the caller and indicates the successful or unsuccessful acceptance of the command.

Hardware-Controlled Command Delivery Service

The hardware-controlled command delivery service is used to set the appropriate hardware-control bit in the Subsystem Control port associated with a feature adapter and to use the services provided by the underlying physical level to set up and access the appropriate control areas in I/O space. The following conceptual service primitive illustrates the parameters required when invoking the command delivery service:

```
DELIVER_CONTROL (destination, value, return code)
```

The ***destination*** parameter identifies which adapter (subsystem) the hardware-controlled command is directed to.

The ***value*** parameter contains the hardware-control value to be delivered.

The first byte of the ***value*** parameter is used to identify the specific hardware-controlled command. It must be one of the following:

- Reset Subsystem
- Reset Reject Condition
- Set Subsystem Interrupt Enablement
- Set DMA Enablement
- Set Other Bits in the Subsystem Control Port.

When Set Other Bits is indicated, byte 2 of the ***value*** parameter contains the value to be set in the Subsystem Control port.

The ***return code*** parameter is returned to the caller and indicates the successful or unsuccessful acceptance of the command.

Control-Block Delivery Service

The control-block delivery service is used to pass the address of a chained or nonchained control block to an entity (device) in an adapter (subsystem), using the services provided by the underlying physical level to set up and access the appropriate control areas (ports) in I/O address space. The following conceptual service primitive illustrates the parameters required when invoking the control-block delivery service:

```
DELIVER_BLOCK (destination, location, return code)
```

The ***destination*** parameter identifies the adapter and entity to which the address of the chained or nonchained control block is to be delivered.

The ***location*** parameter contains the address in shared memory of the chained or nonchained control block to be delivered.

The ***return code*** parameter is returned to the caller and indicates the successful or unsuccessful acceptance of the command.

Delivery-Level Protocols

There are two types of delivery-level protocols: command-delivery protocols and signalling protocols.

Command-Delivery Protocol

The following figure shows the flow of commands through the Command and Attention ports. It is followed by a description of the protocols used to deliver these commands.

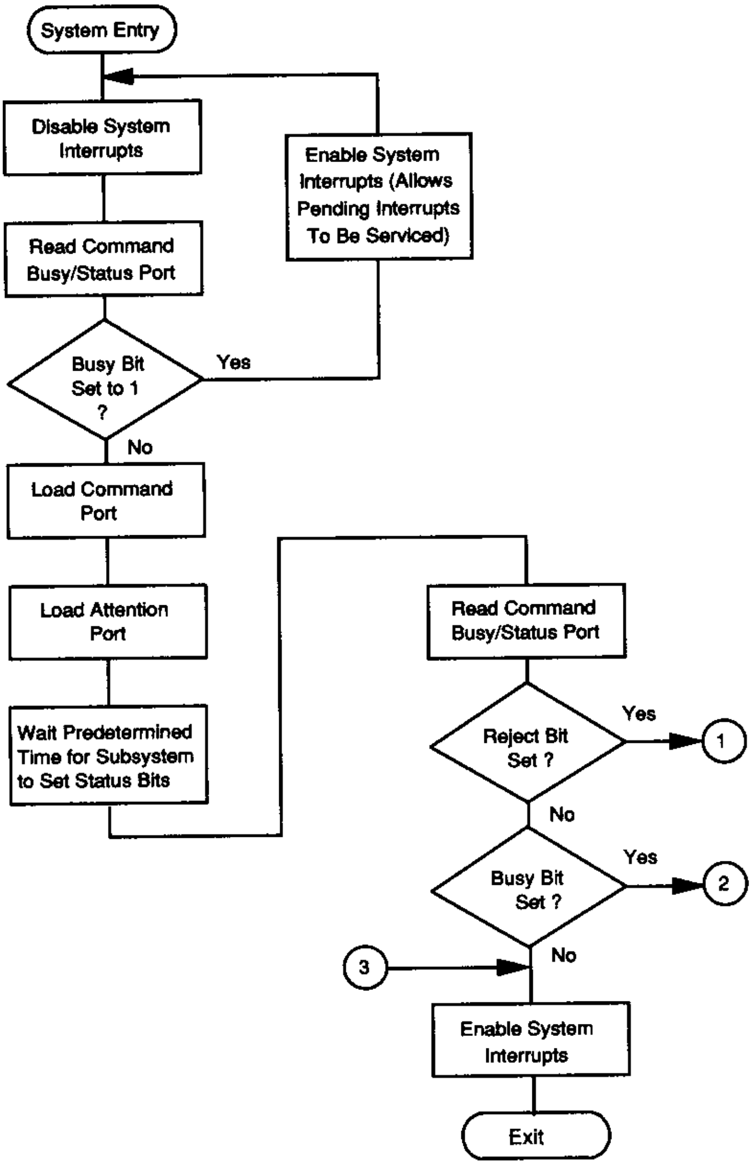

System Entry
Disable System Interrupts
Read Command Busy/Status Port
Busy Bit Set to 1 ?
Yes
No
Enable System Interrupts (Allows Pending Interrupts To Be Serviced)
Load Command Port
Load Attention Port
Wait Predetermined Time for Subsystem to Set Status Bits
Read Command Busy/Status Port
Reject Bit Set ?
Yes
1
No
Busy Bit Set ?
Yes
2
3
No
Enable System Interrupts
Exit

Figure 3-6 (Part 1 of 2). Command-Delivery Flow

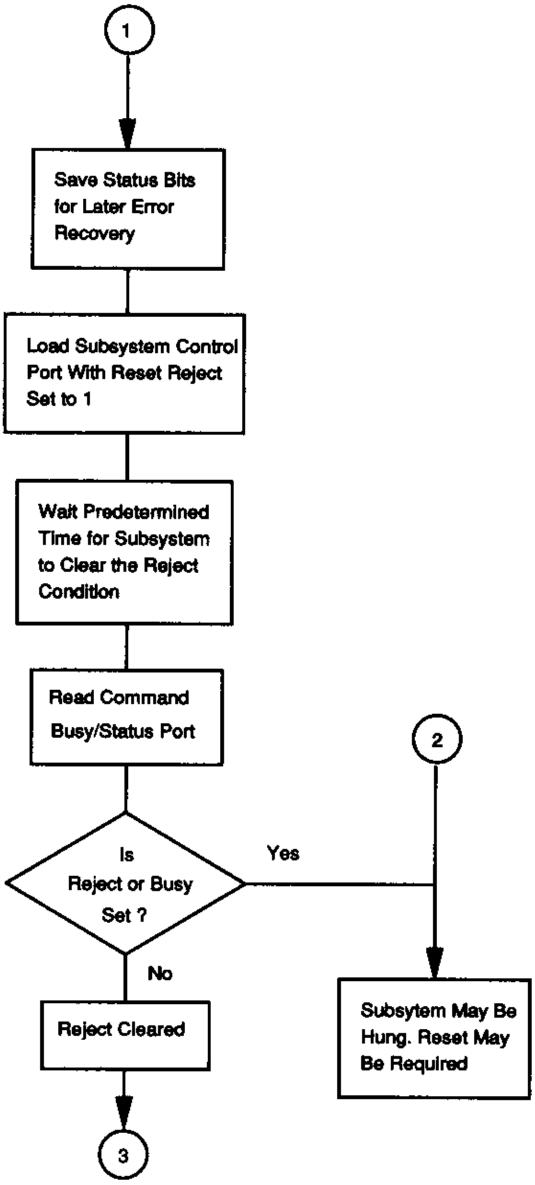

1
Save Status Bits for Later Error Recovery
Load Subsystem Control Port With Reset Reject Set to 1
Wait Predetermined Time for Subsystem to Clear the Reject Condition
Read Command Busy/Status Port
2
Is Reject or Busy Set ?
Yes
No
Reject Cleared
Subsytem May Be Hung. Reset May Be Required
3

Figure 3-6 (Part 2 of 2). Command-Delivery Flow

System interrupts should be disabled before attempting to write a command to the subsystem. The Command Busy/Status port should be read to ensure that the busy bit is off, indicating that the Command

and Attention ports are available. The desired command (or address, in the case of a control block command) must be written to the Command port before the Attention port is written to with the desired attention code and device number. After the two ports are written, the Command Busy/Status port must be read to ensure that the command was not rejected by the subsystem.

Note: The amount of time that a subsystem requires to accept or reject a command is implementation dependent and can be determined in advance by examining the minimum command-status time in the configuration data, which is returned in response to a Read Configuration command.

If the busy and reject bits are both set to 0, the program can assume that the command has been accepted by the subsystem. System interrupts can then be enabled, and the system is free to perform other tasks.

If the reject bit is set to 1 in the Command Busy/Status port, it is an indication that the subsystem has rejected the command. The program should examine the status bits in the Command Busy/Status port to determine the reason for rejection and take appropriate action to resolve the condition.

If the reject bit is 1, it must be explicitly reset by the system by setting the reset-reject bit to 1 in the Subsystem Control port. To ensure that the reject condition has been cleared, the program can read the Command Busy/Status port to ensure that the reject and busy bits have been set to 0 by the subsystem. If the condition has not been cleared, the system might need to use more severe error recovery actions; that is, a Reset Device or Reset Subsystem command could be required.

Signalling Protocol

This section describes the protocols governing interrupts in the Locate mode, how interrupts are presented to the system master, and how they can be cleared.

Interrupts are used as the signalling mechanism between the subsystem and the system master in the Locate mode. They indicate that the subsystem, or a device attached to the subsystem, has completed a command and requires the system master to act upon the completion by servicing the interrupt request.

For simplicity in the discussion of interrupts, the term "device interrupt" is used to refer to an interrupt that either originates from a device attached to the subsystem, or the subsystem itself when a command is directed to it as device 0, or when the reset-subsystem (RST) bit in the Subsystem Control port is toggled.

Interrupt Identification

In the Locate mode, a subsystem has a control area (the Interrupt Status port) in I/O address space that is set by the subsystem with the identification of the device that caused the interrupt, as well as the general reason for the interrupt. Programs in the system master must read this information to determine the source and cause of the interrupt.

Each control-block command also has a unique area (the termination status block) in shared memory where interrupt information and completion status can be stored by the subsystem. Programs in the system master can read this area to determine how a particular command control block has been completed.

Physical and Logical Interrupts

In the Locate mode, the architecture defines and uses two types of interrupts:

- Physical
- Logical.

A physical interrupt occurs when a subsystem activates the interrupt line to the system master, alerting it that an I/O event has occurred during the execution of a command issued to a device on the subsystem. This I/O event is associated with the successful or unsuccessful completion of a command control block, an immediate command, or a hardware-controlled command. When the system master recognizes the activated line, it typically preempts the program currently being executed and directs an interrupt-handling program to the I/O event. Physical interrupts are directly associated with immediate commands and direct hardware commands, and they can be signalled as a result of command control blocks.

Logical interrupts are associated only with command control blocks. A logical interrupt occurs when a completed command control block causes End Status Word 1 of the termination status block to be written into the shared memory with the control-block-interrupt-requested bit set to 1 and the device's bit set to 1 in the Device Interrupt Identifier port. This logically informs the system master of the interrupt, potentially before the physical interrupt can be delivered to the system master. An option of the Locate mode architecture allows a program in the system master to process logical interrupts for the command control block before the physical interrupt is presented. Logical interrupts are reset by the Reset Control Block Interrupt command. This is described in detail in "Resetting Interrupts" on page 3-38.

Physical-Interrupt Enablement

Physical-interrupt enablement determines whether an interrupt that has been detected by a device will be presented as a physical interrupt to the system master by the subsystem. In the Locate mode, interrupts can be disabled at the subsystem or device level.

When interrupts are disabled at the subsystem level, the subsystem and its attached devices cannot present a physical interrupt to the system master until the subsystem is enabled for interrupts.

The subsystem can be disabled for interrupts by setting the enable-interrupt (EI) bit in the Subsystem Control port to 0.

An individual device can be disabled for physical interrupts. A device is disabled for interrupts by setting the disable-device-interrupt (DDI) bit to 1 in any immediate command directed to a device. (It is enabled by setting the disable-device-interrupt bit in an immediate command to 0.)

When disabled, a device cannot present any further interrupt requests as physical interrupts to the system master. An interrupt request that is generated by a device after the device is disabled must be saved by the device for later presentation to the system master.

Once an interrupt value is written into the Interrupt Status port, it will cause an interrupt to the system master if the subsystem is enabled, even if the device that was the source of the interrupt has been subsequently disabled.

Interrupt Queuing within the Subsystem

A subsystem has a single Interrupt Status port in I/O address space for communicating interrupt data to the system master. Once an interrupt value is written into the Interrupt Status port, the port cannot be used to denote any other interrupt until it is explicitly cleared by a Reset Interrupt Status Port command issued from the system master. Because there can be multiple devices contending for this resource (and the Interrupt Status port can be used by only one device at a time), the subsystem is responsible for ensuring that each attached device is given an opportunity to send a physical interrupt request to the system master.

A device can detect an interrupt condition and be unable to present it to the system master at the time of detection. This occurs when the Interrupt Status port is currently in use, or when the device has been disabled for interrupts. To deal with this situation, each device must retain sufficient information to allow later presentation of the interrupt. An interrupt should never be lost in this case.

Interrupt Types

The Locate mode architecture defines two forms of interrupt handling:

- Non-DIIP or simple form, which serializes all interrupts through a single Interrupt Status port
- DIIP or multiple form, which uses the Interrupt Status port for non-control-block and hardware-failure interrupts but uses the Device Interrupt Identifier port (DIIP) for control-block interrupts.

Non-DIIP Interrupts (Simple Form)

In this form of interrupt architecture, all physical interrupts are presented to the system master by the subsystem recording device-level information in a single Interrupt Status port in I/O address space. Once the Interrupt Status port is written by the subsystem with a valid interrupt value, the Interrupt Status port is unavailable for a new interrupt cause until software in the system master explicitly clears it. Clearing the Interrupt Status port is usually accomplished by the Reset Interrupt Status Port command.

An interrupt value in the Interrupt Status port contains two fields:

- The device ID that caused the interrupt
- The reason the interrupt was sent (the Interrupt Identifier Code field).

  The Interrupt Identifier Code field is set by the subsystem to indicate whether a command control block, immediate command, or hardware-controlled command was successfully completed.

In the Locate mode, all interrupts are presented by a device in the same order in which the system master submitted the corresponding command. This restriction does not apply to the Move mode. If commands must be executed in another order, the Move mode must be used.

Note: This can be done in the implementation if the subsystem queues all interrupt requests in the order in which the device raised them.

The Locate mode requires that commands to a device be executed in the order in which they were submitted from the system master. This restriction does not apply to the Move mode. If commands must be executed in another order, the Move mode must be used.

Device – Control-Block Interrupts: This section describes the actions taken by a device when a command control block is completed and needs to raise an interrupt to the system master.

The codes stored in the Interrupt Status port for command control-block interrupts are:

- Command control block completed, no error
- Command control block completed, error
- Command control block completed, hardware failure.

The interrupt-valid (IV) bit in the Command Busy/Status port is set to 1 when the Interrupt Status port is written with the interrupt code.

Normally, when a command control block is completed and requires an interrupt, the termination status block associated with the command control block has at least End Status Word 1 stored with the interrupt-requested (INT) bit set to 1 before the physical interrupt is requested.

A program can elect to inhibit the storing of all termination-status-block status in the non-DIIP-interrupt case when it is completed without error. When this option is selected, a physical interrupt request is enqueued for the device without any corresponding logical interrupt. This option is offered to improve system performance. It should be used with care, especially with control-block chains, because it is more difficult for a program in the system master to determine the completion status of a particular control block. (If a program were written so that only the last control block in a chain requested an interrupt on normal completion, no problems would occur.)

The inhibiting of the storing of termination-status-block status is ignored and a logical interrupt is raised if the command control block is completed with any of the following conditions:

- The command control block was completed with an error that was not suppressed. An example of an error that cannot be suppressed is an irrecoverable hardware failure during a read.
- The command control block is suspended as the result of a Suspend command.

These conditions cause a logical interrupt to be raised and the storing of at least End Status Word 1 with the interrupt-request (INT) bit set to 1 in the termination status block.

Note: If termination-status-block status is stored, a program running in the system master can observe that a device has completed a command control block with the interrupt-request (INT) bit set before the actual interrupt is requested. This is possible if the program tests the termination status block associated with the completed command control block before the interrupt is received by the system master.

This technique is not recommended, however, because an interrupt cannot be cleared in the non-DIIP-interrupt case unless the subsystem has placed the interrupt value in the Interrupt Status port.

The correct method of detecting a pending interrupt is to check the interrupt-valid (IV) bit in the Command Busy/Status port. When this bit is equal to 1, the Interrupt Status port can be read to determine the interrupt type and device origin. Any interrupt present in the Interrupt Status port is cleared by the Reset Interrupt Status Port command.

A command control block can also fail as a result of a hardware failure when the subsystem or device is attempting either to read a control block or to store status information in a termination status block. In either of these cases, termination-status-block status is not stored, and an interrupt code of hex 8 (Hardware failure – control block command) is presented in the Interrupt Status port. The execution of any control-block chain for the device is terminated when this error is encountered.

Device – Non-Control-Block Interrupts: The codes for non-control-block interrupts are:

- Immediate command completed, no error
- Immediate command completed, error
- Immediate command completed, hardware failure
- Reset subsystem or device completed, no error.

The interrupt-valid (IV) bit in the Command Busy/Status port is set to 1 when the Interrupt Status port is written.

Subsystem – Physical Interrupts: When a subsystem determines that a physical interrupt must be raised to the system master, the subsystem must ensure that the following actions are taken:

1. The Interrupt Status port must be written with a valid interrupt identifier and device number. The device must be enabled to interrupt the system master.
2. The interrupt-valid bit in the Command Busy/Status port must be set to 1 to indicate that the Interrupt Status port contains a valid interrupt. The subsystem must be enabled to interrupt the system master before the interrupt-valid bit can be set to 1.
3. The subsystem must activate the physical-interrupt-request line for the level it was assigned at initialization time.

After the system master has serviced the interrupt, it must reset the interrupt by performing the following actions:

1. The Interrupt Status port must be reset by issuing a Reset Interrupt Status Port command. This causes the subsystem to clear the Interrupt Status port and reset the interrupt-valid bit in the Command Busy/Status port. This, in turn, causes the physical-interrupt-request line to become inactive.

2. The system master must reset the interrupt controller (if it has one) by issuing an End of Interrupt command.

   The subsystem can now raise a new physical interrupt to the system if any interrupts are pending within the subsystem.

System Master – Physical Interrupts: The following discussion deals with programming considerations in the system master with the non-DIIP-interrupt-handling model of Locate mode architecture.

An interrupt is sent from the subsystem to the system master if the subsystem is enabled to send interrupts to the system master and the Interrupt Status port has an interrupt value in it.

When the system master receives a physical interrupt on an interrupt level assigned to the subsystem, it should invoke a program to read the interrupt status of the subsystem to determine whether there is an interrupt to service. This program is called a *first-level interrupt-handling routine* for the subsystem. It is invoked in a state in which the system master is disabled for interrupts.

The first-level interrupt-handling routine determines the interrupt state of a subsystem by reading its Command Busy/Status port in I/O address space to determine whether the interrupt-valid (IV) bit is equal to 1. If this is the case, the subsystem or one of its attached devices has requested a physical interrupt. If the IV bit is 0, the first-level interrupt-handling routine can return with the report that it did not have an interrupt to service.

If the IV bit is set to 1, the first-level interrupt-handling routine must read the Interrupt Status port for the subsystem to obtain the identity of the device presenting the interrupt and the reason. The Device ID field identifies the device, and the Interrupt ID field indicates the type and reason for the interrupt.

The interrupt should be cleared by the first-level interrupt-handling routine as soon as possible to avoid blocking the interrupt level and to free the subsystem to present interrupts from other devices that can be attached. The interrupt is cleared by the Reset Interrupt Status Port command. The Attention port value specified in the Reset Interrupt Status Port command must have a device ID equal to the device ID read from the Interrupt Status port at the time of the interrupt. In some cases, the clearing action must be done before analysis of the cause of the interrupt is completed. This can be true when dealing with chained command control blocks in a multitasking environment. In this case, the clearing operation is done in the first-level interrupt-handling routine, and a *second-level interrupt-handling routine* is scheduled. The second-level interrupt-handling routine is like the program that issued the command chain. To be most effective, the second-level interrupt-handling routine should be waiting for a signal to continue processing. The first-level interrupt-handling routine provides the needed signal, and it unblocks execution of the second-level interrupt-handling routine. Having freed the subsystem and the interrupt level, the first-level interrupt-handling routine returns control to its caller.

When the second-level interrupt-handling routine is unblocked and signalled, it needs input arguments. Most likely, the argument supplied to a second-level interrupt-handling routine is the value read by the first-level interrupt-handling routine from the Interrupt Status port.

When a second-level interrupt-handling routine approach is used, the first-level interrupt-handling routine can prevent reentry into the second-level interrupt-handling routine for interrupts by disabling the device that caused the interrupt. This is done by setting the disable-device-interrupt (DDI) bit to 1 before issuing the Reset Interrupt Status Port command. The second-level interrupt-handling routine issues an immediate command to the device to reenable interrupts when it has finished processing a chain or single control block and wants to receive more signals from the first-level interrupt-handling routine.

If an interrupt is not cleared from the Interrupt Status port, it will cause a later physical interrupt when the subsystem and the interrupt level are reenabled.

Non-Control Block Completed without Error, Reset Subsystem

The clearing and processing of an interrupt of this type is straightforward. It consists of the following steps:

1. Clear the Interrupt Status port with a Reset Interrupt Status Port command with the Attention port device ID set to the device ID read from the Interrupt Status port.

2. Find the program that issued the command to the device and cause it to gain control (unblock it if in a multitasking environment).
3. The program that issued the command should easily determine the specific command completed, because only one non-control-block command can be active for a device at a time.
4. Issue an End of Interrupt command to reset the system-master interrupt controller to free the interrupt level.

The End of Interrupt command referred to is the command described in the documentation for the system-master interrupt controller. It is not a Locate mode architected command, but it is included here for implementation information.

When an interrupt controller exists between the subsystem and the system master, the resetting of the interrupt controller must be explicitly performed in addition to the reset, which is required by the subsystem to clear an interrupt. Interrupt Reset commands in this case should be issued as follows:

1. Reset Interrupt Status Port or Reset Control Block Interrupt
2. End of Interrupt to the interrupt controller.

This ensures that the subsystem interrupt is cleared so that the subsystem can present a new interrupt when the interrupt controller is cleared. If the interrupt controller is cleared without the subsystem interrupt being cleared first, a physical interrupt from the interrupt controller will be received as a result of the prior subsystem interrupt.

Non-Control Block Completed with Error or Hardware Failure

Handling an interrupt of this type is more complex than handling a non-control-block command that was completed without error. The processing is identical for both cases to the point of recovery for the non-control-block command. The additional complexity occurs in handling the recovery for interrupts of this type. Recovery is determined, for the most part, by each implementation. For more detail, see “Exception and Error Handling” on page 5-4.

System Master – Control Block Interrupts: Interrupt handling for control blocks can involve the use of first- and second-level interrupt-handling routines, discussed in "System Master – Physical Interrupts" on page 3-31. This is the case because the second-level interrupt-handling routine is typically given the task of determining which control block in the command chain was the source of the interrupt, and whether the chain had gone to completion. The task of locating a particular control block in a chain can involve a search through termination status blocks and interrogation of control-block Enable Word 1. In a multitasking environment, it is best to have this search done in a task that is separate from a first-level interrupt-handling routine. This leaves the first-level interrupt-handling routine free to service other devices on the subsystem.

A second-level interrupt-handling routine would probably be signalled while the device it is servicing has been disabled for interrupts. This action would be taken by the first-level interrupt-handling routine as part of clearing the Interrupt Status port with a Reset Interrupt Status Port command. Disabling the device performs two functions. First, it assures the second-level interrupt-handling routine that it will not be reentered from the first-level interrupt-handling routine until it specifically reenables the device. Second, it assures the first-level interrupt-handling routine that it will not be called upon to service additional interrupts from a device until the second-level interrupt-handling routine has processed its work request from the first-level interrupt-handling routine. This allows devices on the adapter to continue to obtain interrupt service from the first-level interrupt-handling routine while others are being serviced by the second-level interrupt-handling routine. A second-level interrupt-handling routine, when signalled, is supplied with the data read from the Interrupt Status port when the interrupt was received by the first-level interrupt-handling routine.

Searching Control-Block Chains: The second-level interrupt-handling routine can also be designed to determine which control block in a command chain has interrupted and whether the chain is complete. This task can be greatly simplified if certain programming conventions concerning control-block chains are followed. Some examples of these conventions are:

- Always request a successful completion interrupt in the last (or only) control block in a command chain.

- Ensure that End Status Word 1 in each termination status block in the chain is set to 0 before the chain is issued to the device.
- Once a chain is initiated, do not modify control blocks that have not been reported as complete.

By searching the termination status blocks of a chain of commands, a second-level interrupt-handling routine can find the specific command control block that interrupted and can determine the state of a chain of control blocks. After analyzing the results of the search, the second-level interrupt-handling routine can take one of the following paths:

- If the chain has ended and has not been suspended, the second-level interrupt-handling routine should reenable interrupts for the device. Status should be returned to the program that created the chain. The interrupt code passed to the second-level interrupt-handling routine by the first-level interrupt-handling routine determines the kind of ending status given.
- If the chain has not ended and has not been suspended, the second-level interrupt-handling routine should log any error found, enable the device for interrupts, and block itself waiting for the next signal from the first-level interrupt-handling routine for the command-control-block chain.
- If the chain has been suspended, the second-level interrupt-handling routine needs to log any errors found. If the chain has ended, the second-level interrupt-handling routine does not need to save state to restart the chain at a later point. The second-level interrupt-handling routine can return a final result of the program that created the chain or control block. Status should be returned to the program that created the chain. The interrupt code passed to the second-level interrupt-handling routine by the first-level interrupt-handling routine determines the kind of ending status given. If the chain has not ended, the address of the next control block in the chain can be used to restart the chain at a later point if necessary. Any error found is noted.

  Another control-block command stream can be started at the device after the device is enabled for interrupts. This can be done by issuing a write to the Command and Attention ports. The second-level interrupt-handling routine will block itself waiting for new interrupts if a new command stream is started.

In the cases cited previously where the device is still active on a control-block chain, the second-level interrupt-handling routine reenables interrupts by using the Command Reset Interrupt Status port with its disable-device-interrupt bit set to 0. This command will be accepted by a device that is busy executing a control block.

Control Block Completed without Error

When the interrupt ID that is returned to the interrupt-handling program in the system unit indicates that the control-block command was completed without error, there are two cases to consider:

- If the interrupt-handling program knows that the control block is not part of a chain, it can quickly determine which control-block command caused the interrupt, without examining the termination status blocks. This follows from the fact that only one control block was active.
- If the interrupt-handling program does not know the chain status of the control block, it should use a chain-searching technique to locate the interrupting control block.

Although the architecture permits a program to suppress storing termination status blocks for control-block commands that are completed without error, the use of this option must be weighed against the extra analysis that could be needed in interrupt-handling programs to handle chains of control blocks that use interrupts on normal completion to track progress.

A Reset Interrupt Status Port command will clear this interrupt condition from the subsystem and make the Interrupt Status port available.

Control Block Completed with Error

When the interrupt ID indicates that an error occurred that was not caused by the hardware, the interrupt-handling program needs to examine the termination status block associated with the control block to determine the cause of the error.

If a control block is not part of a chain, the interrupting control block can easily be identified because a device can have only one control block active at a time.

When the command control block is part of a chain operation, the interrupt-handling program might be required to search the control-block chain to determine which control block requested an interrupt. This is done by examining, in chain order, each control block that is thought to be incomplete, and testing for completion with termination-status-block status indicating that it requested an interrupt. The first termination status block found indicating a request for an interrupt locates the failing command control block. For more details on exception handling see "Exception and Error Handling" on page 5-4.

Control-Block-Command Hardware Failure

When the interrupt ID indicates that an error caused by the hardware occurred while a control block was being processed, the interrupt-handling program cannot accurately identify the control block when it is part of a chain. This is because a termination status block is not stored for hardware-related errors, even though the control-block chain is terminated.

If a control block is not part of a chain, the interrupting control block can easily be identified because a device can have only one control block active at a time.

DIIP Interrupts (Multiple Form)

In this form of interrupt handling, command-control-block interrupts are not restricted to presentation in the serially-reusable Interrupt Status port. Instead, they are presented through the Device Interrupt Identifier port (DIIP), which provides a means of signalling interrupts for multiple devices in a single port. Not only can multiple devices present logical interrupts in a single DIIP, but each device that signals the presence of a logical interrupt in the DIIP by setting its designated device bit can indicate that several command control blocks have been completed with their interrupt-request bits set to 1 in End Status Word 1 of the termination status blocks.

In this interrupt mode, every command control-block interrupt is a logical interrupt and results in at least End Status Word 1 being written to memory before a physical interrupt request is made. This differs from the non-DIIP-interrupt case, in which termination-status-block status can be disabled for completion of error-free command control blocks when an interrupt is requested.

As defined in the description of the DIIPs, each device is assigned a unique bit in the DIIP that, when set to 1, indicates that at least one command-control-block logical interrupt has been raised for the assigned device. Each time a device detects a command-control-block logical interrupt, an internal count of logical interrupts presented by the device is incremented by 1. If this internal count of logical interrupts presented by the device is greater than 0, the assigned DIIP bit for the device is set to 1, signalling the presence of a logical interrupt in the DIIP. If the device that requested the command-control-block interrupt is enabled for interrupts and the subsystem is also enabled for interrupts, a physical interrupt will be requested.

When the DIIP is supported, at least one DIIP must exist. The implementation defines as many DIIPs as are needed to address the number of devices attached to it. When multiple DIIPs exist, they must occupy consecutive addresses in shared I/O space. The first DIIP defines its least-significant bit 0 to indicate that command-control-block logical interrupts have been raised for the subsystem. Bits 1-15 are then assigned to the first 15 devices attached to the subsystem. Any remaining devices are assigned to other DIIPs, with each port supporting 16 devices. Device assignment continues in each port, starting at the least-significant bit and extending to the most-significant bit. Unassigned bits of a DIIP must always be set to 0.

Device Control-Block Interrupts: When a command control block is completed with an interrupt request, at least End Status Word 1 is stored in the termination status block associated with the command control block, indicating that a logical interrupt request is pending for this command. The device issuing the request increments an internal logical interrupt count by 1 for every logical interrupt request. If the logical interrupt count is greater than 0, the device detects the presence of a logical interrupt by setting its associated bit to 1 in the Device Interrupt Identifier port (DIIP). A physical interrupt for a device is presented to the system master if the following conditions are met:

- The DIIP bit for the device is set to 1.
- The device is enabled for interrupts.
- The subsystem is enabled to send interrupts.

The DIIP interrupt architecture allows a program to reset logical interrupts as soon as they are detected. Typically, a program detects the need to process logical interrupts for a device by reading a subsystem DIIP. The number and specifics of the command control blocks that have been completed with logical interrupts are then determined by examining the End Status Word 1 of control-block commands associated with the device whose DIIP bit is 1.

Logical interrupts that have been processed by programs in the system master are cleared for a specific device by issuing a Reset Control Block Interrupt command. This command contains a count value that allows a program in the system master to reset as many as 15 logical interrupts for a device at one time. The count value specified in the command is subtracted from the internal logical interrupt count for the device. If the resulting new logical interrupt count for the device is less than or equal to 0 as a result of the Reset Control Block Interrupt command, the DIIP bit associated with the device is set to 0 in the appropriate DIIP. This signals the fact that the device currently has no outstanding logical interrupts left to process, and it prevents any further interruption of the system master on behalf of the device for logical interrupts.

When the DIIP architecture is implemented, most control-block interrupts are presented to the system master through the DIIPs. The only exception to this rule occurs when a command control block cannot be read by a device or the subsystem or when completion status could not be stored in the control block's termination status block by the subsystem or device.

If the command control block could not be read or the termination-status-block status could not be written, the interrupt request is not presented in the DIIP, but is signalled as interrupt code hex 8 (Hardware failure – control block command) in the Interrupt Status port. This type of interrupt is handled in an identical fashion to the non-DIIP interrupt, and the resulting interrupt must be signalled to the system master in the Interrupt Status port of the subsystem.

Note: If the failing command control block is an element in a command chain, it might not be possible to locate the failing control block, because no termination-status-block status has been stored.

When the Interrupt Status port is written to and the DIIP is supported, the same rules apply for setting the interrupt-valid bit in the Command Busy/Status port to 1. Interrupt-handling software in the system master should test the interrupt-valid bit so that it can determine whether the Interrupt Status port needs to be read to handle the interrupt request recorded there. A practice of testing and reading the Interrupt Status port before reading any DIIPs ensures that control-block hardware failures are handled on a higher priority than are logical interrupts caused by recoverable command-control-block completions.

As in the non-DIIP-interrupt case, a control-block hardware-failure interrupt is cleared from the Interrupt Status port by the Reset Interrupt Status Port command. Interrupt data remains in the Interrupt Status port until it is explicitly reset by the system master.

Software in the system master can detect that a control-block logical interrupt has been raised for a device before it receives a physical interrupt.

The recommended method for detecting control-block interrupts is to read the DIIPs for any nonzero bits. Nonzero bits represent devices that have at least one control-block logical interrupt. A nonzero value for a DIIP bit indicates that at least one control-block logical interrupt exists for a program to handle.

Note: This method of detecting control-block interrupts before receiving the physical interrupt is simpler and uses fewer system-master processor cycles in the DIIP-interrupt case than in the non-DIIP-interrupt case. In the non-DIIP-interrupt case, it is possible to detect a logical interrupt before the physical interrupt is requested. The program in the system master can do this by polling the termination status blocks for one with the interrupt-request bit set to 1. This is not recommended because it can waste processor cycles that could be used by other tasks.

Once a logical interrupt is found, the DIIP-interrupt case is superior to the non-DIIP-interrupt case because it allows a logical interrupt to be reset without having to wait for the interrupt value to be presented in the subsystem's Interrupt Status port.

Device Non-Control-Block Interrupt: If a non-control-block logical interrupt is found by a device, it is signalled to the system master in exactly the same manner as in the non-DIIP-interrupt case. The interrupt code is signalled in the Interrupt Status port. The specific codes are:

- Immediate/hardware-controlled command completed, no error
- Immediate/hardware-controlled command completed, with error
- Immediate/hardware-controlled command completed, hardware failure
- Reset subsystem or device completed, no error.

Subsystem – Physical Interrupts (DIIP Form): A subsystem can raise a physical interrupt to the system master, following the same steps as in the non-DIIP case (see pages 3-16 through /SFEND/). In addition, a physical interrupt can be raised if a device is enabled for interrupts and its associated DIIP bit is set to 1. The interrupt-valid bit is set to 1 only if the Interrupt Status port contains a valid interrupt value. The DIIP bits are reset by the Reset Control Block Interrupt command (see "Reset Control Block Interrupt" on page 1-80 for details).

System Master – Physical Interrupts: The following discussion deals with programming considerations in the system master with the Device Interrupt Identifier port (DIIP) interrupt-handling model of the Locate mode architecture.

A subsystem will attempt to interrupt the system master if the following conditions are met:

- The subsystem is enabled to send interrupts to the system master.
- Either of the following conditions is true:
  - The Interrupt Status port has an interrupt value in it.
  - Any device is enabled for interrupts, and its associated DIIP bit is set to 1.

If these conditions are met, the system master will receive an interrupt from the subsystem.

Unlike the non-DIIP interrupt, a physical interrupt in the DIIP-interrupt case can indicate that at least one device has presented a control-block logical interrupt and a device has interrupt data present in the Interrupt Status port.

In order to determine the source of a physical interrupt, a program in the system master must read both the Command Busy/Status port and the DIIPs. As in the non-DIIP-interrupt case described in "System Master – Physical Interrupts" on page 3-20, interrupt handling can be viewed as consisting of two levels: a first-level interrupt-handling routine and a second-level interrupt-handling routine.

When a system master receives a physical interrupt, it invokes the first-level interrupt-handling routine associated with the subsystem. This is based upon the interrupt level assigned to the subsystem at configuration and initialization time.

The first-level interrupt-handling routine determines the interrupt state of a subsystem by reading its Command Busy/Status port and DIIPs. If the interrupt-valid bit in the Command Busy/Status port is 0 and all the bits in the DIIPs are also 0, the interrupt is not for this subsystem (interrupt sharing). The interrupt handler simply returns control to its caller with an indication that it did not have an interrupt to service.

If an interrupt exists for the subsystem or any of its devices, a procedure is followed that allows the first-level interrupt-handling routine to schedule the second-level interrupt-handling routine to process device requests for interrupt servicing. A first-level interrupt-handling routine typically tries to schedule all the requests it received on a single invocation to the appropriate second-level interrupt-handling routines. Because second-level interrupt-handling routines are scheduled, the first-level interrupt-handling routine must not accept further interrupts from the device. This is done by disabling the devices. While the devices are disabled, the first-level interrupt-handling routine ignores the DIIP bits for these devices when it is reentered on an interrupt request. (The second-level interrupt-handling routines may not have finished resetting all interrupts at that time.)

Devices are disabled by the first-level interrupt-handling routine in two ways:

- When an interrupt is found in a DIIP.

  A Reset Control Block Interrupt command is used with the disable-device-interrupt (DDI) bit set to 1. The count value in the command is set to 0. This allows the second-level interrupt-handling routine to reset as many logical interrupts as it encounters in its processing, using a true interrupt count.

- When an interrupt is found in the Interrupt Status port.

  A Reset Interrupt Status Port command is used with the disable-device-interrupt (DDI) bit set to 1.

Second-level interrupt-handling routines need to use the services provided by the first-level interrupt-handling routine to reenable a device for interrupts. This allows the second-level interrupt-handling routine to be capable of being called again by the first-level interrupt-handling routine when the first-level interrupt-handling routine detects that the DIIP bit for its device is set to 1.

When all DIIPs have been processed, the first-level interrupt-handling routine returns control to its caller. It indicates that it has cleared an interrupt when it schedules any second-level interrupt-handling routine. If no second-level interrupt-handling routine was scheduled on entry to the first-level interrupt-handling routine, no interrupts were found or cleared by it on this invocation.

As each second-level interrupt-handling routine is scheduled by the first-level interrupt-handling routine, it passes on information explaining why it is being invoked. In the case of an interrupt found in the Interrupt Status port, it would most likely be the contents of the Interrupt Status port Interrupt ID field. For routines that are scheduled to handle logical interrupts, a code is passed that indicates that a search of the command control blocks is needed.

The procedure for dealing with an interrupt found in the Interrupt Status port in the DIIP-interrupt case is similar to that in the non-DIIP-interrupt case. If the interrupt-valid (IV) bit in the Command Busy/Status port is set to 1, the first-level interrupt-handling routine must read the Interrupt Status port for the subsystem to determine the device and the cause of the interrupt. The Interrupt ID field indicates the type and cause of the interrupt, and the Device ID field indicates the identity of the device.

The interrupt should be cleared as soon as possible to avoid blocking the interrupt level and to free the subsystem to present interrupts from other attached devices. The interrupt is cleared by the Reset Interrupt Status Port command. The Attention port must be set to the Device ID read from the Interrupt Status port. In this case, the clearing action is done before analysis of the cause of the interrupt is completed. The clearing operation is done in the first-level interrupt-handling routine, and a second-level interrupt-handling routine is scheduled.

The second-level interrupt-handling routine can be the program that originally issued the command. To be most effective, the second-level interrupt-handling routine should be waiting for a signal to continue processing. The first-level interrupt-handling routine provides the needed signal, and it unblocks the execution of the second-level interrupt-handling routine. In the DIIP case, the first-level interrupt-handling routine continues processing to determine whether it needs to schedule second-level interrupt-handling routines for DIIP interrupts.

Non-Control Block Completed without Error, Reset Subsystem or Device

When the interrupt ID from the Interrupt Status port indicates that a non-control-block command was completed without error, processing of the interrupt consists of a Reset Interrupt Status Port command, followed by an End of Interrupt command to reset the interrupt controller on the system master to free the interrupt level.

Note: The End of Interrupt command referred to is the command described in the documentation for the system-master interrupt controller. It is not a Locate mode architected command, but it is included here for implementation information.

Non-Control Block Completed with Error, Non-Control-Block Hardware Failure

When the interrupt ID indicates that a non-control-block command was completed with an error, recovery depends upon the interrupt code returned and the command that was submitted. The program must keep track of all submitted non-control-block commands that have not been completed. There can be only one non-control-block command active for each device. The interrupt is cleared in the same way as a non-control-block command that was completed without error. For more details on error handling, see "Exception and Error Handling" on page 5-4.

Control-Block-Command Hardware Failure

When the interrupt ID indicates that a hardware failure occurred while a control-block command was being processed, the program might not be able to locate the exact control block that failed, if the control block is an element of a control-block chain. This is because, in these cases, termination-status-block status is not stored.

In the case of a single nonchained control block, a device can have only one control-block address active at a time, so the interrupting control block is immediately known. In the case in which a control-block chain might be present, a second-level interrupt-handling routine and a control-block-searching procedure can be used. In this case, the first control block that has an End

Status Word 1 of 0 is located. The execution of the control-block chain is terminated by the device when such a condition is signalled.

For more details on error handling, see "Exception and Error Handling" on page 5-4.

DIIP Interrupt Present

To process a logical interrupt in the DIIP interrupt architecture, a program in the system master must read the termination status blocks associated with the control-block commands. This happens because only one DIIP bit is returned to the system master to indicate that the device has posted logical interrupts in shared storage.

Because this is the case, a programming convention needs to be employed to simplify the task of locating the termination status block of a control block that is complete. The same convention and searching approach that is described in the non-DIIP-interrupt case can be used; that is:

- Always request a successful completion interrupt in the last (or only) control block in a command chain.
- Ensure that the End Status Word 1 in each termination status block in the chain is set to 0 before the chain is issued to the device.
- Do not modify control blocks that have not been reported as complete once a chain is initiated.
- Enable termination-status-block status storage on the completion of every control block.

Using this convention, a second-level interrupt-handling routine can be written that uses a search program like the one described to find control-block interrupts in the non-DIIP-interrupt case. The difference in the DIIP case is that a second-level interrupt-handling routine uses a loop to call the search routine. The objective of the loop is to call the search routine to find logical interrupts and to count them. As each interrupting control-block command is found, any error or exception condition that is found is processed. For a discussion of error handling, see "Exception and Error Handling" on page 5-4. Calls to the search routine continue if all the following conditions are met:

- The search routine indicates that an interrupt has been found.
- The end of the chain has not been reached.
- The chain has not been suspended.

If any of these conditions are not met, searching stops.

After calls to the search program are stopped in the second-level interrupt-handling routine, several paths are possible. On any path, the logical interrupts found are reset. Resetting is done by the Reset Control Block Interrupt command. As many as 15 interrupts can be reset with one use of the command. If more than 15 interrupts are found, multiple Reset Control Block Interrupt commands are used.

The actions for stopping the search are as follows:

- If the chain has ended and has not been suspended, the second-level interrupt-handling routine should reset the interrupts found and reenable interrupts for the device. The second-level interrupt-handling routine should return status to the program that created the chain. The ending status provided by the second-level interrupt-handling routine is determined from the End Status Word 1 of the last interrupting control block.
- If the chain has not ended and has not been suspended, the second-level interrupt-handling routine should log any errors found. The second-level interrupt-handling routine resets the logical interrupts found and enables the device for interrupts. The second-level interrupt-handling routine blocks itself waiting for the next signal from the first-level interrupt-handling routine for the control-block chain.
- If the chain has been suspended, the second-level interrupt-handling routine needs to log any errors found. The second-level interrupt-handling routine resets the number of logical interrupts found. If the chain has ended, status does not need to be saved by the second-level interrupt-handling routine to restart the chain at a later point. Final status can be returned to the program that created the chain or control block. The ending status provided by the second-level interrupt-handling routine is determined from the End Status Word 1 of the last interrupting control block found by the search program. If the chain has not ended, the address of the next control block in the

chain, returned by the search routine, can be used to restart the chain at a later point. Any error found is noted.

Another command-control-block stream can be started at the device after the number of logical interrupts found is reset and the device is enabled for interrupts. This is done by issuing a write to the Command and Attention ports.

The second-level interrupt-handling routine will block itself waiting for new interrupts if a new command stream is started.

In the cases cited previously where the device is still active on a control-block chain, the second-level interrupt-handling routine reenables interrupts for a device by calling the first-level interrupt-handling routine, which issues a Reset Control Block command with a count of 0 and with its disable-device-interrupt bit set to 0. This command will be accepted by a device that is busy executing a control block.

The order in which devices are processed for control-block interrupts and the number of control-block interrupts that are processed for each device are determined solely by the interrupt-handling software.

Note: System software is free to read DIIPs at any time, but in a typical case, these ports would be read after the system master received the physical interrupt.

DIIP versus Non-DIIP: The description of DIIP (multiple form) interrupt handling can be concluded by summarizing its advantages over the non-DIIP (simple form) interrupt case. The advantages are:

- Multiple interrupt sources are presented in the DIIP case.

  In the DIIP case, interrupt-handling software in the system master can determine the interrupt status of all its devices on a single physical interrupt of the system master and decide the order of service. In the non-DIIP case, only one device source is presented to the system master.

- Multiple logical interrupts can be reset.

  Multiple control-block logical interrupts from one device can be reset with a single execution of the Reset Control Block Interrupt command in the DIIP case. The non-DIIP case allows only one to be reset.

- Logical interrupts can be reset quickly.

  In the DIIP case, logical interrupts can be reset as soon as they are detected in the system master. Software in the system master does not have to wait for a value to be presented in the Interrupt Status port, as in the non-DIIP case.

- Device logical-interrupt status can be found quickly.

  The use of single bits for a device in a DIIP allows programs in the system master to determine whether any logical interrupts are pending for a device without having to poll each termination status block, as in the non-DIIP case.

- Logical interrupts are presented quickly.

  In the DIIP case, logical interrupts are queued quickly at the device by incrementing a simple count. They are presented to the system master immediately by setting the assigned DIIP bit each time a device requests a logical interrupt and its internal count is nonzero. This means that control-block logical interrupts can be responded to in the order the program in the system master selects, rather than waiting for the subsystem to serialize presentation through a single Interrupt Status port.

Resetting Interrupts

This section describes the actions and effects of Reset type commands on the Interrupt Status port and the Device Interrupt Identifier ports (DIIPs).

Interrupt Status Port: The value in the Interrupt Status port remains constant once it is set by the subsystem or a device, until software in the system master clears the port with a Reset Interrupt Status Port command or a Reset Subsystem command.

The normal method of clearing the Interrupt Status port is the Reset Interrupt Status Port command. This command clears the Interrupt Status port when the port contains a valid interrupt value and the device that caused the interrupt matches the value specified in the Attention port. After the Interrupt Status port is cleared, the interrupt-valid (IV) bit in the Command Busy/Status port is set to 0.

Device Interrupt Identifier Ports: The Device Interrupt Identifier ports (DIIPs) are reset by the Reset Control Block Interrupt, Reset Device, or Reset Subsystem command.

The normal method of clearing logical interrupts and DIIPs is the Reset Control Block Interrupt command.

When the Reset Control Block Interrupt command is used, the reset count specified in the command should match the number of logical interrupts to be cleared. If the reset count used is greater than or equal to the number of logical interrupts counted internally by the device, the subsystem sets the DIIP bit for the device to 0 and will not set the DIIP bit for the device until enough control-block interrupts occur for the device to set its internal logical-interrupt count to a positive value. When the DIIP bit for a device is 0, it cannot raise a physical interrupt.

A Reset Device command sets the DIIP bit for the device to 0 and sets the device's internal logical-interrupt count to 0. An interrupt is requested when execution of the Reset Device command is complete, because a Reset Device command always requests an interrupt upon successful completion. (For details, see "Reset Device" on page 1-77.)

Notes:

Chapter 4. Processing Level

In general, the services and protocols at the processing level are not defined by the architecture. However, the architecture does define configuration and initialization, exception handling, and the control-block and data-chaining protocols. The control-block and data-chaining protocols are described in this section. A discussion of configuration, initialization, and exception handling can be found in "Exception and Error Handling" on page 5-4.

There are two types of control-block chaining:

- Command chaining.

  Command chaining allows a series of control blocks to be presented to a device for execution with a single request. The request points to the first control block in a chain of control blocks. This frees the system master to perform other work.

- Data chaining.

  Data chaining uses an indirect list pointed to from the control block to perform data transfers into a set of separate shared-memory areas (pages). This capability is useful when dealing with data buffers in a virtual shared-memory system that are logically contiguous but are mapped by the operating system into separate pages.

Command Chaining

A command-control-block address is delivered from the system master to the subsystem through I/O instructions to write to the Command and Attention ports. Once this command address is delivered and accepted, the device performs the operation specified by the command control block. If command control blocks are issued to the device one at a time, a program in the system master will wait for the command control block to be completed before it issues the next control block to the device. Such a practice involves at least three I/O operations from the system master to initiate the control block and verify its acceptance. At least three I/O operations are needed to verify and reset an interrupt from the device on completion.

If a program needs to issue several commands to a device, instructions are saved in the system master by using command chaining. This is illustrated in Figure 4-1 on page 4-3 (Control Blocks without Command Chaining) and Figure 4-2 on page 4-4 (Control Blocks with Command Chaining).

Figure 4-1 on page 4-3 shows an example of three command control blocks initiated as individual commands from the system master. As indicated, each I/O command requires at least two I/O operations to be delivered to the subsystem through the Attention and Command ports. An I/O operation to the Command Busy/Status port is required to verify command acceptance.

Before control block 2 is sent to the device, control block 1 must be completed and must interrupt the system master. This requires at least three I/O operations from the system master:

- One I/O operation to read the Interrupt Status port
- Two I/O operations to write to the Command and Attention ports to clear the Interrupt Status port.

A program in the system master can then issue the next command control block (control block 2). The procedure is the same as the procedure for control block 1. When control block 2 is completed, control block 3 is issued.

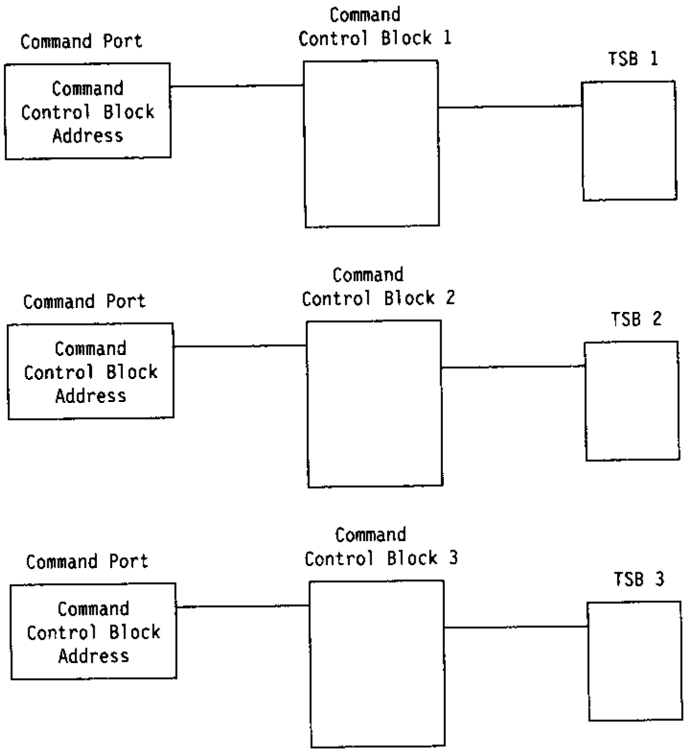


Figure 4-1. Control Blocks without Command Chaining

When command-control-block chaining is used, the system master issues one set of I/O commands to the subsystem to initiate the chain, and the device performs the three I/O operations independently of the system.

In the following figure, the operation defined in control block 1 is executed, and the completion status is written to the termination status block. After this is done, the device reads the next control block (control block 2), that is, independent of the system master. Control block 2 is executed by the device, and the completion status is written to its termination status block. The device then reads and executes control block 3. After control block 3 has been executed, its completion status is written to its termination status block, and a system-master interrupt is requested. This notifies the program that all three commands have been successfully executed.

The system master is free to perform other work while the device is executing control blocks 1, 2, and 3 and does not have to execute extra instructions to submit each control-block command. This is useful in a multitasking environment. See "Signalling Protocol" on page 3-13 for details of programming conventions, the architecture for interrupts, and command-control-block chains.

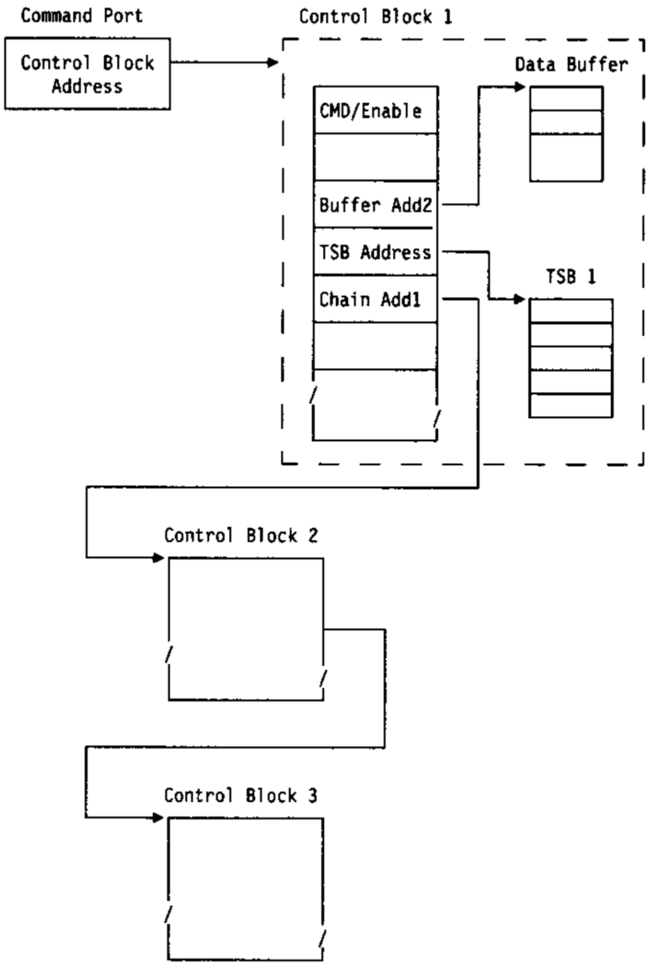


Figure 4-2. Control Blocks with Command Chaining

Conditional Command Chaining

The Locate mode architecture supports the ability to conditionally continue processing command control blocks in a command chain if the current control block is completed with a specified condition. This is called conditional chaining, and it uses a separate control-block-chain address field (Chain Address 2) to specify the next control block to be executed. The form of command chaining illustrated in Figure 4-2 on page 4-4 is called Chain No Error. It uses the other chain-address field in the control block (Chain Address 1) to specify the next control block to be executed when the current control block is completed without error. An individual control block can specify one or both forms of command chaining. When only one form is enabled, the chain continues down the specified path when the control block is completed in a manner that satisfies the chain specification. When both forms of chaining are specified in a control block, two branches in the chain are enabled. The control-block execution path is Chain Address 2 if that condition is met. If the condition is not met and the control block is completed without error, the Chain Address 1 execution path is followed. If neither condition is met, the chain is halted.

An example of the use of both Conditional Chaining and Chain No Error is shown in the following figure. In the example, control block 1 uses the chain condition specification, Chain on Error, and also enables Chain No Error.

If control block 1 is completed with an error and the conditional chain condition is met, the chain continues using Chain Address 2. In the following figure, the path of chain execution is Chain Address 2.

If the condition is not met and control block 1 is completed without error, the chain continues using Chain Address 1. In this case, the path of execution follows Chain Address 1.

Using chaining in this manner allows a program to provide a predetermined error-recovery path. The conditional-chain path is used for commands that are performed dependent on the specified condition.

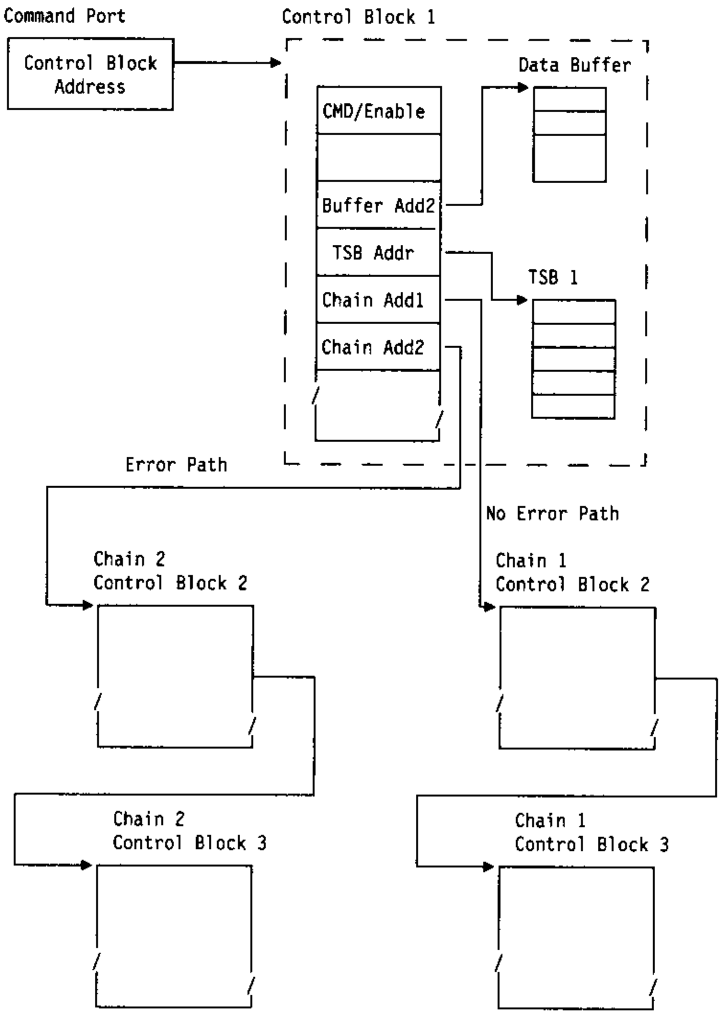


Figure 4-3. Command Chaining with Conditional and Chain No Error Enabled

Command-Chaining Options

Command chaining is specified by setting the chain-no-error bit or the conditional-chaining bit in Enable Word 1 of a command control block. This results in four types of chaining:

- No chaining
- Chain no error (Enable Chain Address 1)
- Conditional chain (Enable Chain Address 2)
- Two-way chain (Enable Chain Address 1 and 2).

The types of chaining are described in detail below.

- No chaining.

  The operation in the control block stands alone. After the device has completed processing of the control block, the device is ready to execute another command from the program. This is specified when the chain-no-error and conditional-chaining bits in Enable Word 1 of the current command control block are both 0.

- Chain no error (unconditional chain).

  This type of command chaining is enabled if the chain-no-error bit in Enable Word 1 of the command control block is set to 1 and the conditional-chaining bit is set to 0. If the current control block is completed without error or is completed with an error that has been suppressed, chaining continues, using control-block Chain Address 1 to read the next control block. An example of this type of chain is a write to a disk, followed by a read to verify the data.

  If a control block terminates with an unsuppressed error, the chain is halted.

- Conditional chaining.

  This type of command chaining specifies that chaining is to use the Chain Address 2 field of the current control block when the control block is completed in a manner that matches the chain-condition-specification bits in Enable Word 1 of the current control block.

  If the current control block is not completed in a manner that matches the chain-condition-specification bits in Enable Word 1 of the current control block, chaining is halted.

  The conditions that can be specified by the settings of the chain-condition-specification bits are:

000 ***Chain on Error***

An unsuppressed error/exception occurred. The errors excluded from this condition are: specification error, hardware failure reading a control block, and hardware failure storing a termination status block.

001 ***Chain on More Status***

The device or subsystem completed the current command control block in a state that indicates that more status is available at the device than could be stored in a termination status block.

010 – 111 These codes are reserved.

Note: A program in the system master can determine whether a command control block was completed by selecting the conditional chain address (Chain Address 2). This is indicated by the setting of the chain-direction bit in the termination-status-block End Status Word 1. To ensure that the termination status block is stored when conditional chaining is specified, the program should set the termination-status-block bit in End Enable Word 1 of the control block to 0 when the chain condition is Chain on More Status. The chain condition Chain on Error causes termination-status-block End Status Word 1 to be stored, ignoring the setting of the termination-status-block bit in Enable Word 1.

- Two-way chaining.

  This type of chaining describes the actions to be taken when both Conditional Chaining and Chain No Error are enabled in the same control block (both the chain-no-error and conditional-chaining bits are set to 1).

  Note: When both forms of chaining are specified in a single control block, testing is performed first to determine whether the Conditional Chaining specification is met. If it is not met, testing is performed to determine whether the Chain No Error condition is met.

  Two-way command chaining specifies that chaining is to continue using the Chain Address 2 field of the current control block, if the current control block is completed in a manner that matches the conditions specified in the chain-condition-specification bits in Enable Word 1.

If the current control block is not completed in a manner that matches the chain-condition-specification bits in Enable Word 1 of the current control block, chaining continues, using the Chain Address 1 field of the current control block, provided the control block was completed without error or with an error that was suppressed.

Chaining is halted if the chain condition does not match the ending condition of the current control block or if the control block is completed with an unsuppressed error.

Control-Block Execution

Control blocks in a chain must be executed in the sequence in which they are presented. This allows programs to depend on a predetermined order of execution. The device cannot perform the next control block until it has completed all operations on the current control block, including the storing of the termination status block.

In the Locate mode, the ordering restriction on command execution also applies to all commands outside of a chain. The restriction is applied to simplify and speed up interrupt-handling software in the system master.

Data Chaining

Data chaining, often referred to as indirect lists, allows a single control block to gather data from several areas of shared memory and transfer it to or from a destination or source.

Data chaining is enabled with the indirect-list-1 (IL1) and indirect-list-2 (IL2) bits in the control-block Enable Word 1 and control-block Enable Word 2, respectively. In the base control block, only IL1 is used, because only one byte count is present to specify the length of the indirect list in system shared memory. This data area, whether used as a source or destination, is accessed by incrementing addresses. With the extended command control block, both Byte Count 1 and Byte Count 2 can be used to specify two data areas addressed by Shared Memory Address 1 and Shared Memory Address 2. Each address can be specified as incrementing or decrementing during the data transfer.

Note: IL1 is associated with the shared-memory address chosen by the setting of the memory-address-1-selected (MA1) and memory-address-2-selected (MA2) bits in Enable Word 1. From this point on, the term "Memory Address x" will be used to refer to either shared-memory address, as selected by the MA1 and MA2 bits.

If IL1 is enabled, Memory Address 1 or Memory Address 2 points to the doubleword-aligned beginning of an indirect list of buffer descriptors. The indirect list is in error if Memory Address x points to an invalid address, the indirect list does not begin on a doubleword boundary, or the length of the indirect list is not a multiple of eight bytes. A specification exception might be noted in these cases.

For reference, an indirect list is illustrated in Figure 4-4 on page 4-11.

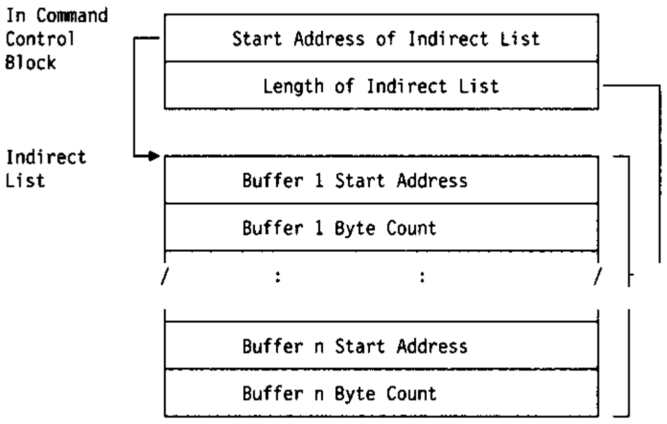


Figure 4-4. Indirect List

Each element (buffer descriptor) in the indirect list consists of eight bytes. The first four bytes define the buffer start address—the location in shared memory to be used as the source or destination of a data transfer. The second four bytes define the buffer length—the size, in bytes, of the data transfer.

The total data-transfer length defined for the control block is the sum of the buffer lengths of all the elements in the indirect list.

The Byte Count 1 field in the control block, associated with Memory Address x, contains the length, in bytes, of the indirect list. The indirect list is in error if the byte count is not a multiple of eight bytes. Because each element consists of eight bytes, the number of elements in the list is Byte Count 1 divided by 8. A specification error exists if any element points to an invalid address. Testing of element validity must be performed before data transfer begins. If no specification testing is done on each address, the results of the operation are not defined by the architecture. An example of a command control block with an indirect list is shown in Figure 4-5 on page 4-12.

Note: The indirect list is useful for software systems that are running user programs in a virtual shared-memory mode. In this case, a user request to read or write from a contiguous area of virtual space might require I/O to a set of noncontiguous real shared-memory pages. The indirect buffer list would then describe the physical pages that map the contiguous virtual area. The start of such an area might fall on an arbitrary byte boundary.

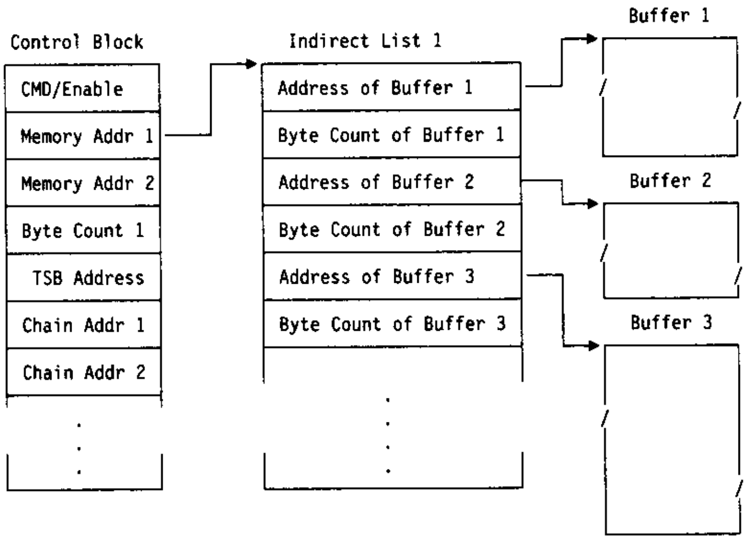


Figure 4-5. Control-Block Command with Indirect List

Reading with a Base Control Block and Indirect Lists

In a read operation, the control block is read from the system master by the device. If IL1 is equal to 1 and EXT is equal to 0, Memory Address x is read to determine the starting address of an indirect list in shared memory. An indirect list must start on a doubleword boundary. The length of the indirect list, in bytes, is found in Byte Count 1. The length of an indirect list must be a multiple of eight bytes.

An indirect list consists of individual buffer descriptors that are eight bytes long and contain a 4-byte buffer address, followed by a 4-byte buffer length. The buffer descriptors give the starting address and length of each element in the indirect list. The total number of data bytes that can be transferred using an indirect list in a read command is equal to the sum of the buffer lengths of all the buffer descriptors in the indirect list. An implementation might choose to compute this total value and reject a command with a specification error if the total data-transfer size exceeds its capability. In this case, no data transfer occurs.

As data is transferred during the read operation, data is not allowed to exceed the length specified for an individual buffer descriptor. A buffer length of 0 transfers no bytes for that buffer descriptor. Additional source data from the device is transferred into the next buffer descriptor in the list if the end of the indirect list has not been reached and more source data exists.

A residual byte count can be developed by the implementation, initializing a value to the size of the buffer and subtracting 1 for each byte that is moved to the buffer from the device. Data movement to the buffer stops when the residual-byte-count value is 0. The buffer is said to be filled.

Data transfers continue in the read operation if any unfilled buffer descriptors remain in the indirect list and more data remains at the source. The indirect list is accessed by incrementing a value set to the contents of Memory Address x by 8 as each buffer descriptor is filled. Processing of the indirect list is completed when the number of list elements that have been filled equals the value in Byte Count 1 divided by 8. A residual buffer address can be developed by the implementation by initializing a value to the contents of Memory Address x and incrementing by 8 as each buffer descriptor is processed. See “Locate Mode Residual Data Values” on page 5-19 for details on developing the residual byte count and the residual buffer address.

If the operation terminates before successful completion, a program in the system master can read the residual byte count and residual buffer address in the corresponding termination status block to determine the progress of the operation and the recovery procedures. Storing of these values in the termination status block is optional (except for short- or long-length exception conditions) and is indicated by the conditional termination-status-block bit in Enable Word 1 of the command control block. If storing of these values is not supported, the program should assume that a complete retry of the operation is needed. For details on this topic, see "Locate Mode Residual Data Values" on page 5-19.

When the device has more data to send to the program than there is space for in a buffer, a long-length exception condition exists. This error condition is always indicated in End Status Word 1. The error might be suppressed if the option is supported by the implementation. For a definition of this condition, see "Long-Length Exception" on page 5-11.

When the program defines buffer descriptors that require more data than the device has to send, a short-length exception condition exists. This error condition is always indicated in End Status Word 1. The error might be suppressed if the implementation supports this option. For a definition of this condition, see "Short-Length Exception" on page 5-9.

Apart from long-length and short-length exception conditions, transfer on a read might be terminated because of an irrecoverable error at the device. In this case, a major error is indicated in End Status Word 1, and an I/O interrupt is requested. Device-dependent error data can also be stored in the termination status block. For more detail on this topic, see "Exception Conditions" on page 5-9.

Reading with an Extended Control Block and Indirect Lists

This case is the same as the base control-block case, except that either Memory Address 1, Memory Address 2, or both can be specified as a combination of indirect lists or direct addressing, and incrementing or decrementing of shared-memory addresses. For further details, see the definition of the indirect-list-1 bit on 1-33 and the indirect-list-2 bit on 1-42.

Writing with a Base Control Block and Indirect Lists

In a write operation, the control block is sent from the system master to the device. The control block is read, and if the indirect-list-1 (IL1) bit is equal to 1 and Ext is equal to 0, Memory Address 2 is read to determine the starting address of an indirect list in shared memory. An indirect list must start on a doubleword boundary. The length of an indirect list must be a multiple of eight bytes. The length of the indirect list, in bytes, is found in Byte Count 1.

An indirect list consists of individual buffer descriptors that are eight bytes long and contain a 4-byte buffer address, followed by a 4-byte buffer length. The buffer descriptors give the starting address and length of each element in the indirect list. The total number of data bytes that can be transferred using an indirect list in a write command is equal to the sum of the buffer lengths of all the buffer descriptors in the indirect list. An implementation might choose to compute this total value and reject a command with a specification error if the total data-transfer size exceeds its capability. In this case, no data transfer occurs.

As data is transferred during the write operation, data is not accessed beyond the extent of the individual buffer descriptor. A buffer length of 0 transfers no bytes for that buffer descriptor.

As bytes are moved from an individual buffer element, a residual byte count can be developed by the implementation, initializing a value to the size of the buffer and subtracting 1 for each byte that is moved to the device from the buffer. Data movement from the buffer stops when the residual-byte-count value is 0. The buffer is said to be empty.

Data transfers continue in the write operation if any full buffer descriptors remain in the indirect list. The indirect list is accessed by incrementing a value set to the contents of Memory Address x by 8 as each buffer descriptor is emptied. Processing of the indirect list is completed when the number of list elements that have been emptied equals the value in Byte Count 1 divided by 8. A residual buffer address can be developed by the implementation by initializing a value to Memory Address x, and incrementing by 8 as each buffer descriptor is processed.

If the operation terminates before successful completion, a program in the system master can read the residual byte count and buffer address in the corresponding termination status block to determine the progress of the operation and to determine recovery procedures. Storing of these values in the termination status block is optional and is indicated by the termination-status-block bits in End Status Word 1. If storing of these values is not supported, the program should assume that a complete retransfer is needed. For more information on this topic, see "Locate Mode Residual Data Values" on page 5-19.

Transfer on a write might be terminated because of an irrecoverable error at the device. In this case, a major error is indicated in End Status Word 1, and an I/O interrupt is queued. Device-dependent error data can also be stored in the termination status block. For more information on this topic, see "Exception Conditions" on page 5-9.

Writing with an Extended Control Block and Indirect Lists

This case is the same as the base control-block case, except that either Memory Address 1, Memory Address 2, or both can be specified as a combination of indirect lists or direct addressing, and incrementing or decrementing of shared-memory addresses. For further details, see the definition of the indirect-list-1 bit on 1-33 and the indirect-list-2 bit on 1-42.

Writing Physical Data Blocks and Indirect Lists

The architecture does not require that a device write the elements of an indirect list as a single physical data block. However, if the device is capable of being written and then later read, the architecture does require that the same indirect list used on the write be capable of retrieving the same physical data blocks on the read.

Chapter 5. Design Considerations

This section contains a discussion of topics covering broad areas in Locate mode architecture. It addresses the following subjects:

- Locate mode setup and initialization
- Locate mode configuration
- Locate mode exception handling.

Locate Mode Setup and Initialization

This section contains some general architectural rules on the topic of setup and initialization. Many of these areas are implementation defined and are not part of the SCB architecture.

The following are used by Locate mode delivery and must be taken into consideration when the system and subsystem are configured:

- I/O address space

  During setup, the I/O base address space and its length are defined.

- System interrupt level
- System arbitration level
- Maximum number of devices that can be attached to the subsystem.

The specific control area definitions in I/O address space for Locate mode delivery are defined in "I/O Address Space" on page 1-15.

Locate Mode Configuration

This section contains information on the configuration of subsystems that support the Locate mode. Much of this area is implementation defined. The following are some of the architectural rules.

Locate Mode Ports

The following ports are required implementations of the Locate mode architecture:

- One Attention port
- One Command Interface port
- One Subsystem Control port
- One Command Busy/Status port
- One Interrupt Status port.

The size of the Attention and Interrupt Status ports depends on whether a Type 1 or Type 2 register interface is used. It is determined by the maximum number of devices supported by the subsystem. A given subsystem can use only the Type 1 or Type 2 register interface and only a single size for its Attention and Interrupt Status ports. This value must be determined before any commands are issued by programs.

Device Interrupt Identifier Port

The Device Interrupt Identifier ports are optional. If these ports are supported, the number of Device Interrupt Identifier ports required is determined by the following equation:

(1 + number of devices attached to the subsystem)/16

The definition of these ports is found in "Device Interrupt Identifier Port" on page 1-28.

Obtaining Configuration Information

The configuration information for a particular subsystem varies with the installation. The subsystem also has values that are important to the operation of programs. The Read Configuration command is used to obtain both configuration and device-dependent information and can be directed either to an individual device or to the subsystem. See page 1-63 for a complete description of the Read Configuration command.

The following is a list of examples of information returned by the Read Configuration command:

- The attachment ID of the subsystem
- The POS register values (Micro Channel systems)
- The revision level of the subsystem
- The interrupt and arbitration levels assigned to the subsystem
- The maximum number of devices that can be attached to the subsystem
- An indication of whether the subsystem supports the Device Interrupt Identifier port
- An indication of whether the device-dependent information is returned
- An indication of whether the subsystem needs to be initialized
- The I/O address of the Device Interrupt Identifier port
- The time it takes for a device to perform a reset
- The data block returned by the Run Diagnostics command
- The size and definition of the programs
- The time a subsystem takes to perform a reset
- The time a device takes to reset a control-block interrupt
- The time a device takes to reset the Interrupt Status port
- The minimum time that must elapse between the submission of a command and the reading of the Command Busy/Status port for acceptance data.

Exception and Error Handling

The Locate mode defines exception and error conditions and provides a standardized method of reporting these conditions. The following topics are discussed:

- Exception and error handling
- Exceptions and error conditions
- The actions to take when a condition is detected
- The data reported in a status block for a condition
- A summary of exception and error conditions that can occur.

  Errors and exceptions exist at two levels in the Locate mode: the device level and the subsystem or command-delivery level. Many of the details of error handling and recovery are device dependent; however, some general guidelines are described in the following.

Device Level

At the device level, most exceptions or errors are indicated by an interrupt to the system master. The treatment and handling of Locate mode signalling is described in detail in "Signalling Protocol" on page 3-13.

To process an exception, an error handling program determines which command caused the exception and reads the associated termination status block for status. This topic is discussed in "System Master – Physical Interrupts" on page 3-31. A discussion of how to process residual status is found in "Locate Mode Residual Data Values" on page 5-19.

At the device level, errors and exceptions are generally handled in the following manner:

1. Determine which device caused the exception.

   This information is found either in the Interrupt Status port or in the Device Interrupt Identifier port (DIIP) for the subsystem.

2. Determine the type of exception.

   The type of the exception is found in the Interrupt Status port interrupt code or in a nonzero DIIP bit. If it was a control-block command, the termination status block might need to be examined to determine the exact cause of the exception.

3. Determine which command caused the exception.

4. Read the termination status block for ending status.

   If the interrupt code was a control-block command completed with error, read the termination status block to get the details of the exception.

When the above steps are complete, proceed as follows:

- Short-length exception (suppressed).

  Determine the number of data bytes successfully read, using residual status. Do not treat it as an error.

  Note: To determine the number of bytes successfully transferred in a control-block command, refer to "Number of Bytes Transferred" on page 5-21.

- Short-length exception (not suppressed).

  Determine the number of data bytes successfully read, using residual status. Report any error.

- Specification exception.

  Use any implementation-defined data, found in the termination status block of the command, to find the reason for the error. Report any error.

- Long-length exception (suppressed).

  Determine the number of data bytes successfully read, using residual status. Do not treat it as an error.

- Long-length exception (not suppressed).

  Determine the number of data bytes successfully read, using residual status. Report any error.

- Device overrun on write.

  Determine the number of data bytes successfully written if residual status is present. Report any error.

- Major error on data transfer.

  Determine the number of data bytes successfully transferred if residual status is present. Report any error.

- Hardware failure – control-block command.

  It might not be possible to find the control-block command that failed. Report any error.

- Hardware failure – non-control-block command.

  Retry the failed command, or report an error if the device is not functional.

- Non-control-block command completed with error.

  Determine which command failed. Report any error.

- Device does not respond after maximum time.

  Retry the failed command, or report an error if the device is not functional.

Subsystem Level

At the subsystem level, exceptions or errors are noted by signalling an interrupt to the system master or by setting the reject and status bits in the Command Busy/Status port.

The treatment and handling of Locate mode signalling is described in "Signalling Protocol" on page 3-13.

At the subsystem level, errors and exceptions are generally handled as follows:

- Control-block command errors.

  These are handled the same way as in the device or entity case. See "Device Level" on page 5-4.

- The subsystem does not respond after the maximum time.

  Determine which command is in progress:

  - Hardware-controlled Reset Subsystem command.

    When the command is a hardware-controlled Reset Subsystem command, report any error, retry the command, and take the subsystem out of service if the retry is unsuccessful.

  - Software-controlled Reset Subsystem command.

    When the command is a software-controlled Reset Subsystem command, retry the command and issue a hardware-controlled Reset Subsystem command if the retry is unsuccessful.

  - The subsystem is taking too much time to clear the busy bit on command submission.

    Read the Command and Attention ports. Issue a software-controlled Reset Subsystem command, and attempt to resubmit the command.

- Command rejected in Command Busy/Status port.

  The status bits in the Command Busy/Status port are read, and processing proceeds as follows:

  - Device unavailable.

    Determine which command failed. Clear the rejection condition from the subsystem. Retry the command, or report any error.

  - Invalid command.

    Determine which command failed. Clear the rejection condition from the subsystem. Report any error.

  - Device busy.

    Determine which command failed. Clear the rejection condition from the subsystem. Report any error, or wait a specified amount of time and retry the command.

- Device control-block execution suspended.

  Determine which command failed. Clear the rejection condition from the subsystem. Report any error. The caller can reenable control-block execution with a Resume command if the suspension is the result of a Suspend command.

- Invalid device number.

  Determine which command failed. Clear the rejection condition from the subsystem. Report any error.

- Device limits reached.

  Determine which command failed. Clear the rejection condition from the subsystem. Report any error.

  Note: The caller needs to process and clear interrupts for the device with the queue full by ensuring that the device is enabled for interrupts, using the Reset Interrupt Status Port command. This is necessary for the interrupt-handling software to read the Interrupt Status port and issue a Reset Interrupt Status Port command for the device indicated, if the interrupt-valid (IV) bit of the Command Busy/Status port is set to 1.

Exception Conditions

There are five types of exception conditions in the Locate mode:

- Short-length exception
- Long-length exception
- Specification exception
- Device-overrun exception
- Major error on data transfer.

Each is described in this section.

Short-Length Exception

- Definition of a short-length exception.

  A short-length exception can occur when data is being read during the execution of a control-block command. The condition exists when the shared data area specified in the control block (the target buffer area) requests more data than is available from the source device.

  The condition can arise in any of the control-block commands that use the semantics of the control-block read commands to transfer data from a device to an area in shared memory. The condition is not restricted to the control-block Read command; it can be detected in the following control-block commands:

  - Read
  - Read Configuration
  - Read Completion Status
  - Run Diagnostic Test.

- Standard actions on detection.

  When the condition is detected, the following actions are always taken:

  1. The residual byte count is stored.

     The residual byte count is stored in the termination status block. The stored value is the number of bytes that remain to be transferred for the current buffer. See "Residual Byte Count" on page 5-19.

2. The residual buffer address is stored.

   The residual buffer address is stored in the termination status block. The stored value is the address of the buffer element being used in the current transfer. See "Residual Buffer Address" on page 5-20.

3. The short-length-exception bit in End Status Word 1 is set to 1.

4. The TSB-available bits in End Status Word 1 are set to show that residual status is stored in the termination status block.

- Optional actions on detection.

  When the condition is detected, the following optional actions can be taken:

  1. Device-dependent status is stored.

     An implementation can choose to store device-dependent data in the termination status block when the condition is detected. If this is done, the TSB-available bits in End Status Word 1 are set to reflect this.

  2. Exception suppressed.

     If the exception has been suppressed in the current control block by setting the suppress-exception-short bit in Enable Word 1 to 1, the condition is not treated as a major exception. An interrupt is not requested for this control block unless the disable-interrupt bit in Enable Word 1 is set to 0. If the disable-interrupt bit is set to 0, the interrupt-requested bit in End Status Word 1 is set to 1, the done bit in the termination status block is set to 0, and an interrupt is requested for the condition.

     A command chain that is defined to continue on No Error (by setting the chain-no-error bit in Enable Word 1 to 1) is not terminated if the error is suppressed. If conditional chaining was specified, the conditional chain can continue if the ending condition matches the conditional specification.

3. Exception not suppressed.

   If the exception has not been suppressed in the current control block by setting the suppress-exception-short bit in Enable Word 1 to 0, the condition is treated as a major exception. The major-error and interrupt-requested bits in End Status Word 1 are set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

   A command chain that was defined to continue on No Error (by setting the chain-no-error bit in Enable Word 1 to 1) cannot continue. If conditional chaining is enabled in the current control block, with the conditional-chaining bit set to 1, the conditional chain can continue if the condition matches the ending status.

Long-Length Exception

- Definition of long-length exception.

  A long-length exception can occur when data is being read during the execution of a control-block command. The condition exists when the shared-data area specified in the control block (the target buffer area) requests less data than is available from the source device.

  The condition can arise in any of the control-block commands that use the semantics of the Read command to transfer data from a device to an area in shared memory. It is not restricted to the control-block Read command, and it can be detected in the following control-block commands:

  - Read
  - Read Configuration
  - Read Completion Status
  - Run Diagnostic Test.

- Standard actions on detection.

  When the condition is detected, the following actions are always taken:

  1. The residual byte count is stored.

     The residual byte count is stored in the termination status block. The stored value is 0, because all bytes specified in the current buffer have been transferred (there is no residual byte count). See "Residual Byte Count" on page 5-19.

  2. The residual buffer address is stored.

     The residual buffer address is stored in the termination status block. The stored value is the address of the buffer element being used in the current transfer. In this case, it is the last buffer in an indirect list or the only buffer, when an indirect list is not used. See "Residual Buffer Address" on page 5-20.

  3. The long-length-exception bit in End Status Word 1 is set to 1.
  4. The TSB-available bits in End Status Word 1 are set to show that residual status is stored in the termination status block.

- Optional actions on detection.

  When the condition is detected, the following optional actions can be taken:

  1. Device-dependent status is stored.

     An implementation can choose to store device-dependent data in the termination status block when the condition is detected. If this is done, the TSB-available bits in End Status Word 1 are set to reflect this.

  2. The exception is suppressed.

     If the exception has been suppressed in the current control block and the suppress-exception-long bit in Enable Word 1 is set to 1, the condition is not treated as a major exception. An interrupt is not requested for this control block unless the disable-interrupt bit in Enable Word 1 is set to 0. If the disable-interrupt bit is set to 0, the interrupt-requested bit in End Status Word 1 is set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

A command chain that is defined to continue on No Error by setting the chain-no-error bit in Enable Word 1 to 1 is not terminated when this exception is suppressed. If conditional chaining is specified, the command chain can continue if the ending condition matches the conditional specification.

3. The exception is not suppressed.

   If the exception has not been suppressed in the current control block and SEL in Enable Word 1 is set to 0, the condition is treated as a major exception. The major-error and interrupt-requested bits in End Status Word 1 are set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

   A command chain that is defined to continue on No Error by setting the chain-no-error bit in Enable Word 1 to 1 cannot continue. If conditional chaining is enabled in the current control block by setting the conditional-chaining bit to 1, the conditional chain can continue if the condition matches the ending status.

Specification Exception

- Definition of specification exception.

  A specification exception can occur during the reading of any control-block command in the Locate mode. The condition occurs when an illegal or unsupported value or option is detected in a control-block field. Specification testing is optional in an implementation. The condition can be detected in any command control blocks.

- Standard actions on detection.

  When the condition is detected, the following actions are always taken:

  1. The command is terminated, and no data transfer occurs.
  2. If additional status is to be provided, it is stored in the termination status block. This is described in the list of optional actions for this condition.

3. This condition is treated as a major error. The specification-check bit in End Status Word 1 is set to 1, indicating that the condition has been detected. The major-error/exception bit in the termination status block in End Status Word 1 is also set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

   Any control-block chaining that is in effect when the condition is detected is halted. Conditional chaining is not supported when a specification error is detected.

- Optional actions on detection.

  When the condition is detected, the following optional actions can be taken:

  - Device-dependent data is optionally stored.

    An implementation-defined value can be written to the termination status block in the device-dependent area to identify the field or option that has been found to be illegal.

  - The TSB-available bits are set to show device-dependent data.

    If device-dependent data is stored in the termination status block, the TSB-available bits in End Status Word 1 are set to reflect this.

Device-Overrun Exception

- Definition of device-overrun exception.

  A device-overrun exception can occur when data is being read or written during the execution of a control-block command. The condition exists when data is being transferred at a rate that the device cannot accept. When the condition is detected, data transfer to the device is halted.

  The condition can arise in any of the control-block commands that transfer data to or from an area in shared memory.

- Standard actions on detection.

  When the condition is detected, the following actions are always taken:

  1. Data transfer for the control-block command requesting the data transfer is terminated.
  2. If additional status is to be provided, it is stored in the termination status block. This is described in the list of optional actions for this condition.
  3. The condition is treated as a major error. The device-overrun bit in End Status Word 1 is set to 1, indicating that the condition has been detected. The major-error/exception bit in the termination status block in End Status Word 1 is also set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

     A control-block command chain that is defined to continue on No Error by setting the chain-no-error bit in Enable Word 1 to 1 cannot continue. If conditional chaining is enabled in the current command control block by setting the conditional-chaining bit to 1, the conditional chain can continue if the condition matches the ending status.

- Optional actions on detection.

  When the condition is detected, the following optional actions can be taken:

  1. The residual byte count is optionally stored.

     The residual byte count is optionally stored in the termination status block. The stored value is the number of bytes remaining to be transferred for the current buffer. See "Residual Byte Count" on page 5-19.

  2. The residual buffer address is optionally stored.

     The residual buffer address is stored in the termination status block. The stored value is the address of the buffer element being used in the current transfer. See "Residual Buffer Address" on page 5-20.

3. The TSB-available bits are set to show that residual status was stored.

   The TSB-available bits in End Status Word 1 are set to show that residual status is stored in the termination status block, if needed.

4. Device-dependent status is optionally stored.

   An implementation can optionally store device-dependent data in the termination status block when the condition is detected.

5. The TSB-available bits are set to show device-dependent data.

   If device-dependent data is stored in the termination status block, the TSB-available bits in End Status Word 1 are set to reflect this.

Major Error on Data Transfer

- Definition of major error on data transfer.

  A major error on data transfer can occur when data is being read or written during the execution of a control-block command. The condition exists when the data requested cannot be provided because an error has arisen that cannot be corrected. Note that this condition differs from errors for device-overrun, long-length, and short-length exceptions. This condition also sets the major-error/exception bit in End Status Word 1. The condition arises in the following ways:

  – Data is being read.

    In this case, the data at the device cannot be delivered to the shared-memory buffer area defined in the control block because an error has arisen that cannot be corrected. This error cannot be a long-length or short-length exception.

  – Data is being written.

    In this case, the data from the shared-memory buffer area defined in the control block cannot be delivered to the device because an error has arisen that cannot be corrected. This error cannot be a device-overrun exception.

The condition can arise in any control-block command that transfers data to or from shared memory.

- Standard actions on detection.

  When the condition is detected, the following actions are always taken:

  1. Data transfer for the control-block command requesting the read or write is terminated.

  2. If additional status is to be provided, it is stored in the termination status block. This is described in the list of optional actions for this condition.

  3. An error condition that cannot be corrected is treated as a major error. The major-error/exception bit in the termination status block in End Status Word 1 is set to 1, indicating that the condition has been detected. The interrupt-requested bit in End Status Word 1 is set to 1, the done bit is set to 0, and an interrupt is requested for the condition.

     A control-block command chain that is defined to continue on No Error by setting the chain-no-error bit in Enable Word 1 to 1 cannot continue. If conditional chaining is enabled in the current command control block by setting the conditional-chaining bit to 1, the conditional chain can continue if the condition matches the ending status.

- Optional actions on detection.

  When the condition is detected, the following optional actions can be taken:

  1. The residual byte count is optionally stored.

     The residual byte count is optionally stored in the termination status block. The stored value is the number of bytes that remain to be transferred for the current buffer. See "Residual Byte Count" on page 5-19.

  2. The residual buffer address is optionally stored.

     The residual buffer address is stored in the termination status block. The stored value is the address of the buffer element being used in the current transfer. See "Residual Buffer Address" on page 5-20.

3. The TSB-available bits are set to show that residual status is stored.

   The TSB-available bits in End Status Word 1 are set to show that residual status is stored in the termination status block.

4. Device-dependent status is optionally stored.

   An implementation can optionally store device-dependent data in the termination status block when the condition is detected.

5. The TSB-available bits are set to show device-dependent data.

   If device-dependent data is stored in the termination status block, the TSB-available bits in End Status Word 1 are set to reflect this.

Locate Mode Residual Data Values

The termination status block for a control-block command contains fields that are designed to show the progress a control-block command has made in data transfer when it has been completed. These fields are the Residual Byte Count and Residual Buffer Address fields. Further definition and programmed use of these fields are discussed in the following section.

A summary of the residual data provided, organized by control-block-command ending status, is in "Summary – Residual Buffer Address and Residual Byte Count" on page 5-22.

Residual Byte Count

The termination status block contains the Residual Byte Count field that is used to indicate the number of bytes that remain to be transferred when the command is completed.

The use of the Residual Byte Count field is completely defined by the architecture when a single storage operand is specified in the control block. The use of this field is implementation defined when two shared-memory operands are specified in the same control block.

When the Residual Byte Count field is 0 and a long-length exception has not been indicated for the command, all data for the command has been successfully read or written. When a long-length exception exists, this field is 0, but additional data remains at the device. An implementation can provide the extent of this unread data in a device-dependent field in the termination status block.

When the Residual Byte Count field is not 0, it indicates that the operation has terminated; therefore, the data that was defined by the command could not be transferred. In this case, a residual byte count remains. To determine the number of bytes of data that have been successfully transferred by the command, a program in the system master needs to read the associated control block and determine the expected byte count of the transfer. To do this, a program might need to use the second data value provided in the termination status block, the residual buffer address.

Residual Buffer Address

The termination status block contains the Residual Buffer Address field, which is used to indicate the buffer that was in use when the command was completed.

The use of the Residual Buffer Address field is completely defined by the architecture when a single storage operand is specified in the control block. The use of this field is implementation defined when two shared-memory operands are specified in the same control block.

When indirect lists are not being used, the address returned for this value corresponds to the buffer field specified in the control block. The address that is presented corresponds to either the address of Memory Address 1 or the address of Memory Address 2. The value is determined by the setting of the memory-address-1-selected (MA1) bit or the memory-address-2-selected (MA2) bit in Enable Word 1.

For example, if the control block is located at Memory Address hex 40000 and MA2 is selected, the residual buffer address is hex 40008 (the base control-block address + hex 8).

If indirect lists are being used, the address returned is the physical address of the list element in use when the command was completed.

For example, assume that an indirect list is being used, it is located at Memory Address hex 60000, and it contains 4 buffer elements (is 32 bytes long). The list starts at shared-memory location hex 60000 and ends at location hex 6001F. In this case, the residual buffer address returned for a control block using this indirect list must be one of the following:

- Hex 60000 – the first buffer descriptor in the indirect list
- Hex 60008 – the second buffer descriptor in the indirect list
- Hex 60010 – the third buffer descriptor in the indirect list
- Hex 60018 – the last buffer descriptor in the indirect list.

Number of Bytes Transferred

To determine the number of bytes that have been successfully read or written by a control-block command, the residual byte count can be subtracted from the number of bytes requested in the command. This computation is of most interest when a short-length exception is detected. It defines the number of bytes that were successfully read. The computation involves the following cases:

- Control block in base format — no indirect list specified.

  The total data-transfer size is given by the contents of Byte Count 1. The number of bytes successfully moved is computed as:

  Bytes moved = Byte Count 1 - residual byte count

- Control block in extended format — no indirect list specified (only one memory address specified).

  The total data-transfer size is given by the contents of Byte Count 1 if Memory Address 1 is 1, or by the contents of Byte Count 2 if Memory Address 2 is 1. The Residual Byte Count field selected in the termination status block is Residual Byte Count 1 if Memory Address 1 is 1, or Residual Byte Count 2 if Memory Address 2 is 1. The number of bytes successfully moved is computed as:

  Bytes moved = total data-transfer size - residual byte count

- Control block in extended format — no indirect list specified (two memory addresses specified).

  The total data-transfer size is not defined by the architecture in this case. It is determined by the control-block operation code. An example of this is a case in which data is defined to be transferred from Memory Address 1 to Memory Address 2. If this convention were used, Memory Address 2 would be used to determine the number of data bytes read.

- Indirect list specified — only one memory address specified.

  The total data-transfer size is given by the byte counts of the indirect-list elements up to and including the element where the transfer stopped. From this total the residual byte count given in the termination status block needs to be subtracted.

- Indirect list specified – two memory addresses specified.

  The total data-transfer size is not defined by the architecture in this case. It is determined by the control-block operation code. An example of this is a case in which data is defined to be transferred from Memory Address 1 to Memory Address 2. If this convention were used, Memory Address 2 would be used to determine the number of data bytes read.

Summary – Residual Buffer Address and Residual Byte Count

This section provides a summary of the contents of the Residual Buffer Address and Residual Byte Count fields for the various ending conditions of control-block commands. The Residual Buffer Address and Residual Byte Count fields must be stored in the termination status block when a short-length or long-length exception condition is encountered. For all other errors and exceptions, storing of these fields in the termination status block is optional.

In the summary data in Figure 5-1, the terms "read" and "write" refer to any control-block operations using the semantics of the Read and Write control-block commands to transfer data to or from shared memory. For example, the term "read" refers not only to the Read command, but to other commands such as Read Configuration; the term "write" refers not only to the Write command, but to other commands such as Initialize Device.

Control Block Completion	Residual Buffer Address	Residual Byte Count
Normal end on read	Memory Address 2 in control block if no indirect list. If indirect list, address of last list element.	Zero.
Short-length exception on read	Memory Address 2 in control block if no indirect list. If indirect list, address of last list element.	Number of bytes remaining to transmit for current buffer element.
Long-length exception on read	Memory Address 2 in control block if no indirect list. If indirect list, locates last list element.	Zero.
Device error on read	Memory Address 2 in control block if no indirect list. If indirect list, address of current list element.	The number of bytes that remain to be read in the current buffer element.
Specification error in control block	Not set.	Not set.
Normal end on write	Memory Address 2 in control block if no indirect list. If indirect list, address of last list element.	Zero.
Device error on write, or device overrun	Memory Address 2 in control block if no indirect list. If indirect list, address of current list element.	The number of bytes that remain to be written in the current buffer element.
Normal end on no data transfer (Noop)	Not set.	Not set.

Figure 5-1. Residual Status Data after Control-Block Execution

Notes:

Index