

**To:** John B. Lohmeyer  
Chairman X3T9.2  
NCR  
1635 Aeroplaza Dr  
Colorado Springs, CO 80916

**Date:** Jan 28, 1993  
**From:** Richard Whalen  
Digital Equipment Corporation  
**Phone:** (508)-841-2684  
**Loc:** SHR3-2/W28, Shrewsbury MA 01545  
**E-Mail:** whalen@starch.dec.com

**Subject:** Copy of SIMport specifcaiton for January SCSI working group minutes

---

John,

Enclosed is a copy of the specification that was presented to the CAM working group at the January SCSI working group meeting. This copy of the specification has a statement in it giving explicit permission to ANSI X3T9.2 to publish it and distribute it. Please let me know if anything additional is needed in order for this to be included as part of the minutes.

Sincerely,

Richard Whalen

# **SIMport**

## **HBA Interface Specification, V A.9, 1/7/93**

---

**This document specifies a SIM interface for intelligent host bus adapters. SIMport provides a SCSI Interface Module (SIM) like interface as defined in the specification entitled "SCSI-2 Common Access Method, Transport and SCSI Interface Module."**



---

## **COPYRIGHT NOTICE**

Copyright © January 1992 by Digital Equipment Corporation.

POSSESSION, USE, DUPLICATION OR DISSEMINATION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENTATION IS AUTHORIZED ONLY PURSUANT TO A VALID WRITTEN LICENSE FROM DIGITAL OR THIRD PARTY OWNER OF THE COPYRIGHT.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERROR WHICH MAY APPEAR IN THIS DOCUMENT.

Digital Equipment Corporation has separately applied for patents on some of the material disclosed in this specification. A license to use any patents issuing as a result of these applications in SCSI CAM SIMport implementations will be granted with reasonable terms if this specification is accepted as part of the SCSI CAM specification.

Digital Equipment Corporation grants permission to ANSI X3T9.2 to publish this document and circulate it to the public at large.

The following are trademarks of Digital Equipment Corporation.

DEC, VMS

All Rights Reserved.  
Printed in U.S.A.



---

# Contents

<b>Preface</b> .....	<b>xi</b>
0.1      Revision History .....	<b>xi</b>
0.2      Purpose of this Specification .....	<b>xii</b>
0.3      Conventions .....	<b>xii</b>

## Glossary

## 1 Introduction

1.1      Goals .....	1-1
1.1.1      Interface modeled after CAM XPT to SIM interface .....	1-1
1.1.2      Provide a host independent interface .....	1-1
1.2      Reference Documents .....	1-1
1.2.1      CAM XPT SIM specification .....	1-1

## 2 Overview

2.1      SCSI Interface Module .....	2-1
2.2      Initialization .....	2-2
2.2.1      The Adapter Block .....	2-2
2.2.2      SIMport Queues .....	2-2
2.2.3      Adapter Initialization and State Control .....	2-2
2.2.4      Channel Initialization and State Control .....	2-3
2.3      Normal SCSI Channel Operation .....	2-3
2.3.1      CCB Processing .....	2-3
2.3.2      Response Queue Processing .....	2-4
2.4      Channel SIM Queues .....	2-4
2.5      Verify Adapter Sanity Timer .....	2-4
2.6      Channel Errors .....	2-5
2.7      Adapter Errors .....	2-5
2.8      Interrupts .....	2-5
2.8.1      Adapter Interrupts .....	2-5
2.8.2      Host Interrupts .....	2-5
2.9      Free Queue Element Allocation .....	2-6

## 3 Interface Components

3.1	Adapter Block .....	3-1
3.1.1	Queue Header .....	3-2
3.2	Queues .....	3-2
3.2.1	Driver-Adapter Command Queue .....	3-2
3.2.2	Adapter-Driver Response Queue .....	3-2
3.2.3	Driver-Adapter Free Queue .....	3-2
3.2.4	Adapter-Driver Free Queue .....	3-3
3.3	Queue Carrier .....	3-3
3.3.1	Queue Carrier Extension .....	3-4
3.3.1.1	SIM Queue Priority Flag .....	3-4
3.3.1.2	Return Status Format.....	3-4
3.4	Queue Buffer .....	3-4
3.5	Adapter Registers.....	3-4
3.5.1	Adapter Maintenance Control and Status Register (AMCSR).....	3-5
3.5.2	Adapter Block Base Register (ABBR).....	3-5
3.5.3	Driver-Adapter Command Queue Insertion Register (DACQIR) .....	3-5
3.5.4	Driver-Adapter Free Queue Insertion Register (DAFQIR) .....	3-5
3.5.5	Adapter Status Register (ASR) .....	3-6
3.5.6	Adapter Failing Address Register (AFAR).....	3-7
3.5.7	Adapter Failing Parameter Register (AFPR).....	3-7
3.6	Interrupt Holdoff Timer.....	3-8

## 4 Initialization and State Control

4.1	Initialization Steps.....	4-1
4.2	SIM Driver Initialization Steps.....	4-1
4.3	Adapter Initialization Steps .....	4-2
4.4	State Transitions.....	4-4
4.5	Adapter States.....	4-4
4.5.1	Uninitialized State .....	4-4
4.5.2	Disabled State .....	4-4
4.5.3	Enabled State .....	4-4
4.6	Channel States .....	4-4
4.7	Device States .....	4-5

## 5 Command and Response Processing

5.1	General Operating Characteristics .....	5-1
5.1.1	Command Response Protocol .....	5-1
5.1.2	Command Priority .....	5-1
5.1.3	Unrecognized Commands.....	5-1
5.2	Adapter Specific Commands and Responses .....	5-2
5.2.1	Reset Adapter Command .....	5-3
5.2.2	Initialize Adapter Command .....	5-3
5.2.3	Set Adapter State Command .....	5-3
5.2.4	Adapter State Set Response Message .....	5-4
5.2.5	Set Parameters Command .....	5-6
5.2.6	Parameters Set Response Message .....	5-7
5.2.7	Set Channel State Command.....	5-7
5.2.8	Channel State Set Response Message .....	5-8
5.2.9	Set Device State Command.....	5-9
5.2.10	Device State Set Response Message .....	5-9
5.2.11	Verify Adapter Sanity Command .....	5-10
5.2.12	Adapter Sanity Verified Response Message.....	5-10
5.2.13	Read Counters Command .....	5-10

5.2.14	Counters Read Response Message .....	5-11
5.2.15	BSD Request Message .....	5-13
5.2.16	BSD Response Command .....	5-14
5.2.17	Bus Reset Request Message .....	5-15
5.2.18	Bus Reset Response Command .....	5-16
5.3	Maintenance Commands and Responses .....	5-17
5.4	CAM defined Commands and Responses .....	5-17
5.4.1	Execute SCSI I/O Command .....	5-18
5.4.1.1	Command Timeout and Command Abort .....	5-20
5.4.1.2	Queued Operation Tag Assignment .....	5-20
5.4.1.3	Autosense .....	5-20
5.4.2	SCSI I/O Executed Response Message .....	5-20
5.4.3	Execute Target I/O Command .....	5-20
5.4.4	Target I/O Executed Response Message .....	5-21
5.4.5	Release SIM Queue Command .....	5-21
5.4.6	Abort SCSI I/O Command .....	5-21
5.4.7	Reset SCSI Bus Command .....	5-21
5.4.8	Reset SCSI Device Command .....	5-22
5.4.9	Terminate SCSI I/O Process Command .....	5-22
5.4.10	Set ASYNC Callback Command .....	5-22
5.4.11	AEN Response Message .....	5-23
5.4.12	Path Inquiry Command .....	5-23
5.4.13	Path Inquiry Response Message .....	5-24
5.4.14	Enable LUN Command .....	5-25

## 6 Unsolicited Message Processing

6.1	Free Queue Element Management .....	6-1
6.2	Unsolicited Reselection Message .....	6-1
6.3	Channel Disabled Message .....	6-2
6.4	Device Disabled Message .....	6-3

## 7 Target Mode Operation

7.1	Processor Mode .....	7-1
7.1.1	Inquiry command .....	7-1
7.1.2	Receive command .....	7-1
7.1.3	Request Sense command .....	7-1
7.1.4	Send command .....	7-1
7.2	Phase Cognizant Mode .....	7-1

## 8 HBA Engines

## 9 Interrupts

9.1	Host Interrupts .....	9-1
9.1.1	Command Complete Interrupt .....	9-1
9.1.2	Adapter Miscellaneous Interrupt .....	9-1
9.1.3	Systems with single interrupt only support .....	9-1
9.2	Adapter Interrupts .....	9-1
9.2.1	Initialization Interrupt .....	9-1
9.2.2	Reset Interrupt .....	9-1

9.2.3	Command Queue Insertion Interrupt .....	9-2
9.2.4	Free Queue Insertion Interrupt .....	9-2

## 10 Buffer Memory Mapping

10.1	Buffer Segment Map .....	10-1
10.2	Buffer Segment Map Options .....	10-3

## 11 Issues

### A CAM and SIMport function codes

A.1	Supported CAM functions codes .....	A-1
A.2	SIMport Adapter Specific function codes .....	A-2
A.3	SIMport Status codes .....	A-3

### B Unaligned Host Memory Buffers

B.1	Unaligned Buffers .....	B-1
B.1.1	Writing unaligned host buffers .....	B-2
B.1.2	Reading unaligned host buffers .....	B-2

### C Bus Reset and Bus Device Reset

C.1	Adapter detected bus reset .....	C-1
C.1.1	Adapter reset processing steps .....	C-1
C.1.2	SIM Driver reset processing steps .....	C-2
C.2	Adapter initiated bus reset .....	C-2
C.3	Adapter detected Bus Device Reset .....	C-2
C.4	Bus Device Reset .....	C-2
C.4.1	Adapter initiated Bus Device Reset .....	C-3
C.4.2	Host initiated Bus Device Reset .....	C-3

### D SIMport on 64-bit architectures ~~CAM Command Format~~

D.1	CCB format on systems with 64 bit addresses .....	D-1
-----	---	-----

### E CAM command formats

E.1	SIMport supported CAM CCB formats .....	E-1
E.1.1	Execute SCSI I/O Command .....	E-1
E.1.2	Release SIM Queue Command .....	E-2
E.1.3	Abort SCSI I/O Command .....	E-2
E.1.4	Reset SCSI Bus Command .....	E-2
E.1.5	Reset SCSI Device Command .....	E-2
E.1.6	Terminate SCSI I/O Process Command .....	E-2
E.1.7	Set ASYNC Callback Command .....	E-2
E.1.8	Path Inquiry Command .....	E-3
E.1.9	Enable LUN Command .....	E-3

## F Host-Adapter polling mode

## G SIMport Queues

G.1	Queue Structure Overview .....	G-1
G.1.1	Queue Mechanics .....	G-1
G.1.2	Addressing .....	G-3
G.2	Carrier Structure .....	G-4
G.3	Queue Insertion Procedures .....	G-5
G.3.1	Driver-Adapter Queues .....	G-5
G.3.2	Adapter-Driver Queues .....	G-7
G.4	Free Queues .....	G-8
G.5	Recovering Queue Buffers and Carriers on Adapter Crash .....	G-8

## H SIMport Data Structures

### Figures

Figure 2-1	CAM Model .....	2-1
Figure 3-1	Adapter Block .....	3-1
Figure 3-2	SIMport Queue Carrier Format .....	3-3
Figure 3-3	Adapter Block Base Register (ABBR) .....	3-5
Figure 3-4	Driver-Adapter Command Queue Insertion Register (DACQIR) .....	3-5
Figure 3-5	Driver-Adapter Free Queue Insertion Register (DAFQIR) .....	3-5
Figure 3-6	Adapter Status Register (ASR) .....	3-6
Figure 3-7	Adapter Failing Address Register (AFAR) .....	3-7
Figure 3-8	Adapter Failing Parameter Register .....	3-7
Figure 3-9	Command, Response, Message flow .....	3-9
Figure 4-1	Adapter/Driver initialization information exchange .....	4-3
Figure 4-2	Adapter and Channel State Transitions .....	4-4
Figure 5-1	Adapter Specific Command/Response Format .....	5-2
Figure 5-2	Set Adapter State Command .....	5-3
Figure 5-3	Adapter State Set Response Message .....	5-4
Figure 5-4	Set Parameters Command .....	5-6
Figure 5-5	Set Channel State Command .....	5-7
Figure 5-6	Channel State Set Response Message .....	5-8
Figure 5-7	Set Device State Command .....	5-9
Figure 5-8	Device State Set Response Message .....	5-10
Figure 5-9	Read Counters Command .....	5-11
Figure 5-10	Counters Read Response Message .....	5-12
Figure 5-11	BSD Request Message .....	5-13
Figure 5-12	BSD Response Command .....	5-14
Figure 5-13	Bus Reset Request Message .....	5-16
Figure 5-14	Execute SCSI I/O CCB Private Data .....	5-18
Figure 5-15	Adapter Specific Command/Response Format .....	5-23
Figure 5-16	Format of Enable LUN Command .....	5-25
Figure 6-1	Unsolicited Message Format .....	6-1
Figure 6-2	Unsolicited Reselection Message .....	6-2



Figure 6-3 Channel Disabled Message .....	6-2
Figure 6-4 Channel Disabled Message Parameter Block .....	6-3
Figure 10-1 Buffer Segment Map (BSM) .....	10-2
Figure 10-2 A Buffer Segment Descriptor (BSD) .....	10-2
Figure B-1 DMA of unaligned host buffers .....	B-2
Figure C-1 Adapter/Driver reset information exchanged .....	C-1
Figure C-2 Adapter/Driver bus device reset information exchanged .....	C-3
Figure D-1 64 Bit SIMport CCB Header format .....	D-1
Figure G-1 Queue Carriers and Queue Buffers .....	G-1
Figure G-2 Queue Structure .....	G-2
Figure G-3 Carrier Format .....	G-4
Figure G-4 Carrier Physical Address Pointer Fields .....	G-4
Figure G-5 Driver-Adapter Queue Structure .....	G-6
Figure G-6 Adapter-Driver Queue Structure .....	G-7

## Tables

Table 0-1 Revision History .....	xi
Table 3-1 Adapter Block Fields .....	3-2
Table 3-2 SIMport Queue Carrier field definitions .....	3-3
Table 3-3 SIMport Queue Carrier Flags definitions .....	3-4
Table 3-4 ASR bit field definitions .....	3-6
Table 3-5 AFPR error codes .....	3-7
Table 5-1 Adapter Specific Commands and Response Messages .....	5-2
Table 5-2 Set Adapter State Command Fields .....	5-3
Table 5-3 Adapter State Set Response Message Fields .....	5-4
Table 5-4 Adapter State Set Response Message Status .....	5-5
Table 5-5 Adapter State Set Response Message Flags .....	5-5
Table 5-6 Adapter State Set Response Message Xfer_Align Field .....	5-5
Table 5-7 Set Parameters Command Fields .....	5-6
Table 5-8 Set Parameters Command Flags Fields .....	5-7
Table 5-9 Parameters Set Response Message Status .....	5-7
Table 5-10 Set Channel State Command Fields .....	5-8
Table 5-11 Channel State Set Response Message Fields .....	5-8
Table 5-12 Channel State Set Response Message Status .....	5-9
Table 5-13 Set Device State Command Fields .....	5-9
Table 5-14 Device State Set Response Message Fields .....	5-10
Table 5-15 Device State Set Response Message Status .....	5-10
Table 5-16 Read Counters Fields .....	5-11
Table 5-17 Read Counters Flags .....	5-11
Table 5-18 Counters Read Response Message Fields .....	5-12
Table 5-19 Counters Read Response Message Status .....	5-12
Table 5-20 Required Counters Names and Offsets .....	5-13
Table 5-21 BSD Request Message Fields .....	5-14
Table 5-22 BSD Response Command Fields .....	5-14
Table 5-23 BSD Response Command Status .....	5-15
Table 5-24 BSD Response Command Buf_id .....	5-15
Table 5-25 Bus Reset Request Message Fields .....	5-16
Table 5-26 Bus Reset Request Reason Field .....	5-16



Table 5-27 Bus Reset Response Command Status .....	5-17
Table 5-28 CAM defined Commands and Response Messages .....	5-17
Table 5-29 Execute SCSI I/O CCB Private Data Fields .....	5-19
Table 5-30 Path Inquiry Field Responsibilities .....	5-24
Table 5-31 Enable LUN Command Fields .....	5-25
Table 6-1 Unsolicited Reselection Message Fields .....	6-2
Table 6-2 Channel Disabled Message Fields .....	6-3
Table 6-3 Channel Disabled Message Status .....	6-3
Table 6-4 Device Disabled Message Status .....	6-3
Table 10-1 Buffer Segment Map Fields .....	10-2
Table 10-2 Buffer Segment Descriptor Fields .....	10-3
Table A-1 CAM Function Codes .....	A-1
Table A-2 SIMport Function Codes .....	A-2
Table A-3 SIMport status codes .....	A-3
Table E-1 Execute SCSI I/O Command Field Usage .....	E-1
Table E-2 Execute SCSI I/O VU Flags .....	E-2
Table E-3 Set ASYNC Callback Command Fields .....	E-2
Table E-4 Enable LUN Command Fields .....	E-3
Table G-1 Carrier Structure .....	G-4
Table G-2 Carrier Physical Address Pointer Fields .....	G-5



---

# Preface

Please direct all comments and suggestions to:

Richard Whalen  
Storage Systems Architecture  
Digital Equipment Corporation  
334 South Street SHR3-2/W28  
Shrewsbury, Massachusetts 01545-4112  
E-Mail: whalen@starch.enet.dec.com  
Voice: 508-841-2684

## 0.1

## Revision History

---

Table 0-1 Revision History

Revision	Date	Authors	Description
A.1	6-Jan-1992	M. Mackay	First draft release
A.2	26-May-1992	M. Mackay R. Whalen	Second draft release. This update reflects architectural changes only. For example, text which clarifies or duplicates the CAM specification are not included in this revision.
A.3	24-Jul-1992	R. Whalen	Third draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT.
A.4	11-Sep-1992	R. Whalen	Fourth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Added a fourth header level which allowed some reorganization of chapter 5. Unspecified information in Appendix E.1 added.
A.5	16-Oct-1992	R. Whalen	Fifth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Specification of AFPR added.
A.6	2-Nov-1992	R. Whalen	Sixth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Replaced Appendix G with new text suggesting alternatives for recovering queue resources on adapter crash.
A.7	20-Nov-1992	R. Whalen	Seventh draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. This update contains a re-working of the status codes to eliminate gaps and duplications. Also the format of the adapter specific commands and responses has been reorganized so that fields always end up in the same place and are of the same size.
A.8	18-Dec-1992	R. Whalen	Eighth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. The majority of the changes remove DEC specific considerations in preparation for presenting the specification at the upcoming CAM-2 meeting for possible inclusion in the CAM-2 spec. This includes a complete definition of the operation of SIMport

## 0.2 Purpose of this Specification

The purpose of this specification is to define the interface to a family of intelligent Host Bus Adapters. The intent is to provide a document such that both SIM driver development and adapter development can proceed independently.

## 0.3 Conventions

[ ]	The square brackets are used to enclose implementation notes, issues, and missing text. When issues and missing text are resolved they will be removed.
n	The lower case letter 'n' is used in parameter tables to indicate an integer value.
pa	The lower case letters 'pa' are used in parameter tables to indicate a host physical address value.
va	The lower case letters 'va' are used in parameter tables to indicate a host virtual address value.
SBZ	SBZ in structure formats and parameter tables implies the field Should Be Zero for future architecture compatibility.
N.A.	Not applicable.
T.B.D.	To be determined or supplied.
Numbers	Both decimal and hexadecimal numbers are used in this document. Hexadecimal numbers are followed by a subscripted 16.

---

## Glossary

### **AB**

The Adapter Block, AB, is a component of SIMport. It is a host resident data structure used to pass initialization information from the host to the HBA.

### **ABBR**

Adapter Block Base Register. The ABBR contains the physical address of the AB.

### **Adapter**

The interface between the host bus and the SCSI bus.

### **ADFQ**

The Adapter-Driver Free Queue, ADFQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from a SIMport adapter to the host. The command buffers in this case are completed immediate commands that do not require a host response.

### **ADRQ**

The Adapter-Driver Response Queue, ADRQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from a SIMport adapter to the host. The command buffers in this case are either response messages from previously issued commands or unsolicited messages generated by the adapter in response to unexpected events.

### **AEN**

A SCSI Asynchronous Event Notification.

### **AFAR**

The Adapter Failing Address Register contains the physical address of host memory associated with ASR errors.

### **AFPR**

The Adapter Failing Parameter Register passes implementation specific information on an ASR error.

### **AMCSR**

The Adapter Maintenance Control and Status Register controls the state and operating parameters of the adapter.

### **ASR**

The Adapter Status Register records adapter status that requires host intervention.

### **BDR**

Bus Device Reset of a target on the SCSI bus.

## **BSD**

The Buffer Segment Descriptor, BSD, is a component of SIMport. A BSD defines the address and size of a host memory segment or of a Buffer Segment Map (BSM).

## **BSM**

The Buffer Segment Map, BSM, is a component of SIMport. BSMs are used to define a physically discontinuous host memory buffer. A BSM defines a list of BSDs, where each BSD defines a host memory segment.

## **CAM**

The Common Access Method, CAM, is a ANSI standard which defines the software interface between device drivers and the Host Bus Adapters or other means by which SCSI peripherals are attached to a host processor.

## **CCB**

The CAM Control Block, CCB, is the data structure used to pass CAM command information across the various CAM architectural layers.

## **CDB**

The SCSI Command Descriptor Block, or a pointer to the CDB.

## **Channel**

A SCSI bus on the adapter.

## **DACQ**

The Driver-Adapter Command Queue, DACQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from the host to a SIMport adapter.

## **DACQIR**

The Driver Adapter Command Queue Insertion Register is used to notify the adapter of new entries on the DACQ.

## **DAFQ**

The Driver-Adapter Free Queue, DAFQ, is a component of SIMport. It is a host resident queue type data structure also used to pass command buffers (CCBs) from the host to a SIMport adapter. The command buffers in this case are to be used by the adapter in the generation of unsolicited message.

## **DAFQIR**

The Driver Adapter Free Queue Insertion Register is used to notify the adapter of new entries on the DAFQ.

## **doublet**

Two bytes (16 bits) of data.

## **HBA**

The Host Bus Adapter, HBA, is the physical layer of CAM that interfaces a SIM with a SCSI bus.

## **Host Memory Segment**

A host memory segment is a physically contiguous section of host memory. It is defined (in a BSD) by the physical address of the start of the section and the number of bytes in the section.

## **Hexaword**

A unit of memory equal to 16 2-byte words. A data structure that is hexaword aligned will have the low 5 bits <4:0> of its address equal to 0 (zero).



**LSB**

Least Significant Bit.

**LUN**

A Logical Unit, LUN, is an addressable entity of a SCSI device. A single SCSI device may have up to eight LUNs.

**octalet**

Eight bytes (64 bits) of data.

**MSB**

Most Significant Bit.

**quadlet**

Four bytes (32 bits) of data.

**SIM**

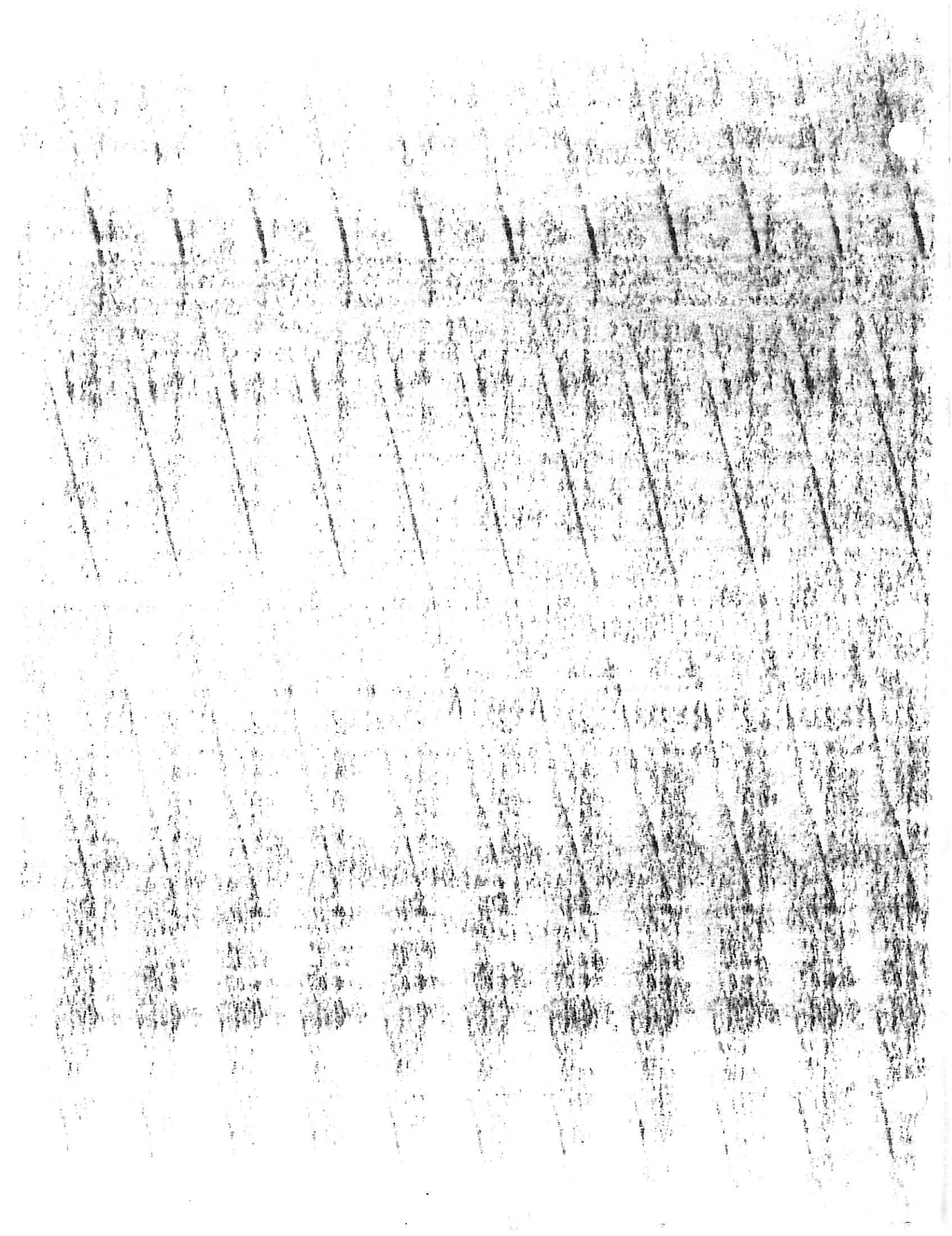
The SCSI Interface Module, SIM, is the layer of CAM that interfaces the CAM XPT layer with the CAM HBA layer.

**SIM Queue**

A CAM defined queue used to hold SCSI I/O commands that are waiting for SCSI bus arbitration. A separate queue exists for each LUN.

**XPT**

The XPT is the transport layer of CAM. It directs CAM commands to appropriate SIMs.





- b) To provide a means of supporting the ATA-2 protocol on other physical interfaces, such as that of the newly-emerging local buses.
- c) To improve transfer protocols.
- d) To improve reliability of data transfers.
- e) To provide a means to increase the number of devices on the interface.
- f) To eliminate obsolete functionality.
- g) To reduce the command overhead by introducing the concept of queuing.
- h) Other capabilities which fit within the general scope of implementing the ATA-2 on a broader range of applications.
- i) An effort will be made to maintain compatibility.

### 2.3. Existing Practice in Area of Proposed Standard or Technical Report

The proposed project involves evolutionary expansion of the draft AT Attachment standard to provide additional capabilities.

### 2.4. Expected Stability of Proposed Standard or Technical Report with Respect to Current and Potential Technological Advance

The nature of the proposed project is to insure that the AT Attachment has an upward, highly compatible growth path.

## 3. Description of Proposed Project

### 3.1. Type of Document (Standard or Technical Report)

Standard.

### 3.2. Definition of Concepts and Special Terms (if any)

None.

### 3.3. Expected Relationship with Approved X3 Reference Models (e.g., DBMS, OSI)

The ATA-2 standard is for use in closed systems.

### 3.4. Recommended Program of Work

The following program of work is planned for the ATA-2 standard:

- \* Solicit continuing participation by the present AT Attachment participants through X3T9.2 procedures and new participants through press releases. Invite comments by end-user organizations and invite proposals from organizations that may have a contribution to a viable ATA-2 standard.
- \* Establish functional requirements for ATA-2 functional additions along with downward compatibility requirements.
- \* Prepare a draft standard based on proposals submitted and other information gathered during the initial investigation.
- \* Consider the results of ATA-2 testing as may be available to the committee through the voluntary efforts of the various participants in X3T9 and its assigned task group.

- \* Submit the draft proposed standard to X3 for further processing.

### 3.5. Resources - Individuals and Organizations Competent in Subject Matter

The current membership of X3T9.2 includes representatives from all parts of the computer industry from semiconductor chip manufacturers to large mainframe system manufacturers as well as Government agencies. The members of X3T9.2 have expressed their desire to participate and cooperate in the development of this proposed standard.

There are sufficient resources to complete the development of this standard without delaying work on other standards.

### 3.6. Recommended X3 Development Technical Committees (Existing or New)

It is recommended that the development work be done in task group X3T9.2 which was responsible for developing the draft AT Attachment standard.

### 3.7. Anticipated Frequency and Duration of Meetings

Task group X3T9.2 meets for two days bi-monthly. Specific task ad hoc groups are called as may be required for one to three days between the regular meetings but their results are not binding.

### 3.8. Target Date for dpANS to X3 (Milestone 10)

December 1994.

### 3.9. Estimated Useful Life of Standard or Technical Report

It is anticipated that this standard will have a life of over 10 years.

## 4. Implementation Impacts

### 4.1. Impact on Existing User Practices and Investments

The proposed ATA-2 standard will provide an upward growth path complementary to the existing practices and investments. It is likely that any isolated negative impacts would occur in any case through non-standard evolution or revolution.

### 4.2. Impact on Supplier Products and Support

The proposed ATA-2 standard will provide an upward growth path complementary to the existing practices and investments. It is likely that any isolated negative impacts would occur in any case through non-standard evolution or revolution.

### 4.3. Techniques and Costs for Compliance Verification

The committee will consider the results of ATA-2 testing as may be available to the committee through the voluntary efforts of the various participants in X3T9 and its assigned task group. With this method all costs are borne by the organizations of the various participants and have for the most part been mainly an adjunct of their normal development costs.

### 4.4. Legal Considerations

No new legal considerations are expected that are not already attendant with AT Attachment and in accordance with accepted X3 patent policies.

5. Closely Related Standards Activities

5.1. Existing Standards

None.

5.2. X3 Standards Development Projects

X3T9.2/90-143 AT Attachment Interface.

5.3. X3/SPARC Study Groups

None.

5.4. Other Related Domestic Standards Efforts

None.

5.5. ISO Standards Development Projects

It is anticipated that this standard will be proposed to JTC1/SC25/WG4.

5.6. Other Related International Standards Development Projects

None.

5.7. Recommendations for Coordinating Liaison

None.

5.8. Recommendations for Close Liaison

None.

TANDBERG DATA

FAX FORM

TO: NCR Corporation, Colo Spgs

FAX NO: +1 719 597 8225

ATT: John Lohmeyer

URGENT: Yes

DATE: January 6, 1993

FROM: Irene Larsen

DATA STORAGE DIVISION  
FAX NO: + 47 2 18 95 50

SUBJECT: SCSI standard

COPY:

PAGE 1 of 1

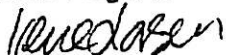
Dear Mr. Lohmeyer

I have a question regarding the SCSI-2 standard which I hope you can answer.

Section 5.1.5.1 of the standard describes asynchronous information transfer. It says that when transferring data to the target, the target shall first assert REQ. The initiator shall drive the DB signals, delay at least one deskew delay plus a cable skew delay and assert the ACK signal.

Does this mean that it has to be at least a deskew delay plus a cable skew delay between REQ asserted to ACK asserted? Or could it be less as long as the data bus is driven at least a deskew plus a cable skew delay before ACK is asserted?

Kind Regards



Irene Larsen

TANDBERG DATA A/S, OSLO, NORWAY  
Kjelsåsveien 161  
P.O. Box 9 Korsvoll  
N-0808 OSLO 8, NORWAY

PHONE: + 47 2 18 90 90

J:\BASIC\DIG\ANSI.FAX

January 6, 1993

Dear Ms. Larsen,

My *personal opinion* on your question is that the deskew delay plus cable skew cable exist to insure at least 0 ns. of setup time in the target from the DATA BUS being valid to the ACK signal being asserted. I think the statements you reference from SCSI-2 could be viewed as implying that REQ must be true prior to beginning this delay, however SCSI-2 also says that the initiator may change or release the DATA BUS upon it detecting the REQ signal going false between bytes. This latter statement was included in SCSI-2 to permit the initiator to change the DATA BUS value earlier so that the time delay could also start earlier.

Both implementations exist in the industry. I have not heard of interoperability problems with either approach.

Please be aware that the above is my personal opinion and it does not represent an official interpretation of SCSI-2. You may submit a "Request for Interpretation" on approved ANSI standards which will eventually result in an official response from the X3 Committee. Unfortunately, this process does take some time (3 to 6 months) and it cannot begin until SCSI-2 is formally approved by ANSI (probably another 3 to 6 months).

If you wish to submit a formal Request for Interpretation, please send it to:

Ms. Lynn Barra  
CBEMA (X3 Secretariat)  
1250 Eye St. NW  
Suite 200  
Washington, DC 20005-3922

Best Regards,



John Lohmeyer

X3T9.2/93-052

1/14/93, 4:48 PM

To: SPI Working Group, ANSI X3T9.3

Copy: Bill Spence, Chairman  
Larry Lamers, Secretary

Sbjct: Test of differential drivers for SCSI

From: Kevin Gingerich, Texas Instruments

Ref: Suggested changes to Figure 8 (Differential driver test circuit) for SCSI-3 SPI document, John Goldie, National Semiconductor

Are all of the grounds of each SCSI device on a bus at the same potential energy at all times? This question needs answering before any specification of the test requirements for the differential interface circuits. If the answer is yes, then there is zero common-mode voltage on the signal wires and the differential buffers may be integrated with the CMOS controllers. If the answer is no, then the proposed test requirements in reference 2 does not accurately represent the electrical environment that can exist on the SCSI bus.

Point 4 of the referenced document states that the test circuit of figure 8 of X3T9.2/91-010R7 models a DC shift in the grounds and that assigning six ground wires in the SCSI cable will prevent it. Indeed, the six grounds will prevent low frequency ground shifts. However, they most likely will also create as many ground-loops with earth ground magnifying higher frequency inductive noise coupling with the outside world and inducing ac common-mode noise. The bus driver must absorb this noise energy in addition to driving the DC load presented by the differential SCSI bus. My experience has been that there will be installations with electromagnetic environments that will induce ground potential shifts. The probability as well as the magnitude of these shifts increases with cable length.

EIA RS-485(1983) defines the common-mode requirements for the bus as, "Generators and receivers conforming to this standard can operate with a common-mode voltage between -7 V and 7 V (instantaneous). The common-mode voltage is defined to be any uncompensated combination of generator-receiver ground potential difference and longitudinally coupled peak noise voltage measured between the receiver circuit ground and cable with the generator ends of the cable short circuited to ground, plus the generator offset voltage (V<sub>OS</sub>)."

This standard is complied with, in part, by drivers with a minimum of 60 mA source and sink current capability. This number comes from driving a differential load of 60  $\Omega$  and 32 receiver inputs at 12 k $\Omega$  each. The differential signal output current would be

$$I_o = 1.5V \times \left( \frac{1}{0.06k\Omega} + \frac{1}{12k\Omega} \times 32 \right) = 29mA.$$

1/14/93, 4:48 PM

With ground potential shifts of -7 V to 7 V and 5 V of driver common-mode output, the driver must drive the load over a voltage range of -7 V to 12 V. The maximum common-mode output current from or to 32 receivers is then

$$I_{oc} = 12V \times \left( \frac{1}{12k\Omega} \times 32 \right) = 32mA.$$

The SCSI load changes the termination and bias network of the baseline RS-485 load while relaxing the minimum differential output voltage to 1 V. The unbalancing of the load affects the + and - signal unequally resulting in two equations for the output current. Using the same analysis on the proposed differential test circuit of figure 4 from the referenced document, the signal output current requirements are

$$I_{o-} = 1V \times \frac{1}{0.075k\Omega} + 4.3V \times \frac{1}{0.165k\Omega} = 39.3mA \quad \text{for assertion and}$$

$$I_{o-} = -1V \times \frac{1}{0.075k\Omega} + 4.3V \times \frac{1}{0.165k\Omega} - 1V \times \left( \frac{1}{12k\Omega} \times n \right) = 12.7 - \frac{n}{12}mA$$

for negation and where n = the number of receivers. For the + line

$$I_{o+} = -1V \times \frac{1}{0.075k\Omega} - 1V \times \frac{1}{0.165k\Omega} - 1V \times \left( \frac{1}{12k\Omega} \times n \right) = -19.4 - \frac{n}{12}mA$$

for assertion and

$$I_{o+} = 1V \times \frac{1}{0.075k\Omega} = 13.3mA \quad \text{for negation.}$$

As can be seen above, the highest driver output current demands are for  $I_{O-}$  and  $I_{O+}$  when asserted.

To model ground potential shift in the proposed test circuit add a voltage source between the generator ground and the test load. The contribution from the common-mode voltage for either output is then

$$I_{oc} = -V_{oc} \times \left( \frac{1}{0.165k\Omega} + n \times \frac{1}{12k\Omega} \right).$$

Summing the worst load and common-mode currents,

$$I_{OUT-} = I_{o-} + I_{oc} = 39.3 - V_{CM} \times \left( \frac{1}{.165} + \frac{n}{12} \right) \text{ and}$$

$$I_{OUT+} = I_{o+} + I_{oc} = -19.4 - V_{CM} \times \left( \frac{1}{.165} + \frac{n}{12} \right).$$

1/14/93, 4:48 PM

From the equations above, a table can be constructed showing the relationships of driver output currents, the maximum number of nodes on the bus, the allowable common-mode voltage range and ground potential shifts across the bus.  $V_{GND}$  is calculated assuming the RS-485 maximum common-mode output voltage of -1 V to 3 V.

Max Driver Current	Number of Nodes	$V_{CM\ min}$	$V_{CM\ max}$	$ V_{GND} $
60 mA	8	-3.1 V	6.0 V	2.1 V
60	16	-2.8	5.6	1.8
60	32	-2.4	4.7	1.4
70	8	-4.6	7.5	3.6
70	16	-4.2	6.8	3.2
70	32	-3.5	5.8	2.5
80	8	-6.1	9.0	5.1
80	16	-5.5	8.2	4.5
80	32	-4.7	6.9	3.7
90	8	-7.5	10.5	6.5
90	16	-6.9	9.5	5.9
90	32	-5.8	8.1	4.8
100	8	-9.0	12.0	8.0
100	16	-8.2	10.9	7.2
100	32	-7.0	9.2	6.0

Table 1. Allowable common-mode voltage range of a SCSI bus for various combinations of driver output currents and nodes on the bus.

Since Differential SCSI does not appear to be broken, does it need fixing? There are several possible reasons that problems have not surfaced with Differential SCSI. Most RS-485 drivers are over-designed and can typically deliver more than 60 mA, SCSI systems are not experiencing these worst-case loading or common-mode conditions, and the system impact may not be readily visible to the users.

The committee needs to define the system requirement for usable common-mode voltage range and the driver requirements should follow or accept the liability of a specification compliant bus that doesn't work. Texas Instruments agrees to provide RS-485 devices with the additional requirements of the Differential Test Circuit (figure 8) from SPI R7 or 86 mA drivers.

Kevin Gingerich  
Texas Instruments  
(214)997-3378  
Fax: (214)997-3165  
internet: 4307725@mcimail.com



## QUOTATION

J. M. Cohen  
Technical Editing  
9975 Mansfield Road - #28  
Keithville, LA 71047  
(318) 687-7971

Quote #93:4501-X3T9.2  
January 31, 1993

TO: John Lohmeyer  
Chairman X3T9.2  
1645 AeroPlaza Drive  
Colo Sprgs, CO 80916

Item	Description	Price
1	SCSI-2 Document Final Edit	\$3000.00
1.1	incorporate changes as marked by the ISO Editor*	
1.2	print review copy of document to validate changes	
1.3	print camera copy of document for publication	
	Total	\$3000.00

\* Item 1.1 does not included changes to the figures.

QUOTATION IS VALID FOR 90 DAYS FROM DATE OF ISSUE.

## QUOTATION

J. M. Cohen  
Technical Editing  
9975 Mansfield Road - #28  
Keithville, LA 71047  
(318) 687-7971

Quote #93:4502-X3T9.2  
January 31, 1993

TO: John Lohmeyer  
Chairman X3T9.2  
1645 AeroPlaza Drive  
Colo Sprgs, CO 80916

Item	Description	Price
1	AT ATTACHMENT INTERFACE FOR DISK DRIVES	\$650.00
1.1	convert document from text to WordPerfect ISO style	
1.2	print review copy of document to validate changes	
1.3	incorporate ISO editor's suggested changes	
1.4	print review copy of document to validate changes	
1.5	print camera copy of document for publication	
	Total	\$650.00

\* Item 1.1 does not included conversion of the figures.

QUOTATION IS VALID FOR 90 DAYS FROM DATE OF ISSUE.

# QUOTATION

J. M. Cohen  
 Technical Editing  
 9975 Mansfield Road - #28  
 Keithville, LA 71047  
 (318) 687-7971

Quote #93:4503-X3T9.2  
 January 31, 1993

TO: John Lohmeyer  
 Chairman X3T9.2  
 1645 AeroPlaza Drive  
 Colo Sprgs, CO 80916

Item	Description	Price
1	SCSI-2 COMMON ACCESS METHOD TRANSPORT AND SCSI INTERFACE MODULE	\$650.00
1.1	convert document from text to WordPerfect ISO style	
1.2	incorporate ISO editor's suggested changes	
1.3	print review copy of document to validate changes	
1.4	print camera copy of document for publication	
	Total	\$650.00

\* Item 1.1 does not included conversion of the figures.

QUOTATION IS VALID FOR 90 DAYS FROM DATE OF ISSUE.

March 12, 1993

John Lohmeyer  
Chairman, X3T9.2  
3718 N. Rock Road  
Wichita, KS 67226

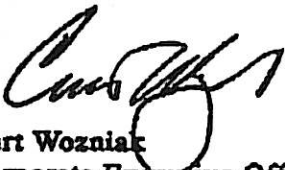
I. Dal Allan  
Small Form Factor Committee  
14426 Black Walnut Court  
Saratoga, CA 95070

Subject: Copyright for Single Connector Attachment document

Dear Mr. Lohmeyer and Mr. Allan

The *Single Connector Attachment for Small SCSI Disk Drives* document is copyrighted by Sun Microsystems Inc. ("Sun"). No part of the document may be reproduced, modified, or distributed in any form by any means without prior written authorization from Sun. The Small Form Factor committee and members of ANSI Technical Committee X3T9.2 are authorized by this letter to copy in any form any portion of this document and subsequent revisions of this document as may be provided by Sun for the purposes of considering and reviewing the document in connection with the standardization activities of those two committees; provided that the complete text of the copyright notice on the document must be included in all complete copies. Copies of portions of the document must also include recognition of Sun's copyright by including the short form notice shown in the attached. Upon completion of the standardization activities, Sun will continue to have all rights to publish and revise this document. The Small Form Factor Committee will have rights to publish the final specification resulting from its standardization activities. All other copyright rights not explicitly granted to The Small Form Factor committee and members of ANSI Technical Committee X3T9.2 are reserved by Sun.

Sincerely,



Curt Wozniak  
Corporate Executive Officer  
SMCC Engineering

enc.

## Copyright Statement for Single Connector Attachment document

Use the following short form copyright notice on portions of the document which may be copied separately:

Copyright 1992 Sun Microsystems, Inc. All rights reserved.

This material is protected by copyright law (Title 17 U.S. Code), and is used by the Small Form Factor Committee and members of ANSI Technical Committee X3T9.2 with permission.

The following statement shall be included in any complete copy of the *Single Connector Attachment for Small SCSI Disk Drives* document prepared by the Small Form Factor Committee or by any member of the ANSI X3T9.2 organization. The present revision of this document contains this copyright statement.

© 1992 Sun Microsystems, Inc.—Printed in the United States of America.  
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This document is protected by copyright and distributed under licenses restricting its use, copying, modification or distribution. No part of document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

**RESTRICTED RIGHTS LEGEND:** Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

### TRADEMARKS

Sun, Sun Microsystems, the Sun logo, are trademarks or registered trademarks of Sun Microsystems, Inc. AMP and Champ are trademarks or registered trademarks of AMP Incorporated. All other product names mentioned herein are the trademarks of their respective owners.

**THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.**

**THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.**

**X3T9.2/93-057**

# **Single Connector Attachment for Small SCSI Disk Drives**

**Revision 1.0**

**Sun Microsystems Computer Corporation  
2550 Garcia Avenue  
Mountain View, CA 94043-1100**

**Seagate Technology  
PO Box 12313  
10321 West Reno  
Oklahoma City, OK 73127-2313**

**Conner Peripherals  
3081 Zanker Road  
San Jose, CA 95134-2128**

© 1992 Sun Microsystems, Inc.—Printed in the United States of America.  
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

**RESTRICTED RIGHTS LEGEND:** Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

#### **TRADEMARKS**

Sun, Sun Microsystems, the Sun logo, are trademarks or registered trademarks of Sun Microsystems, Inc. AMP and Champ are trademarks or registered trademarks of AMP Incorporated. All other product names mentioned herein are the trademarks of their respective owners.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

## Publication History

Revision Number	Description	Date
1.0	Initial Publication of Document	11/9/92





## Table of Contents

1	Scope	9
2	Applicable Documents	9
3	Introduction	9
4	Connector Definition	9
5	Location Within Drive	10
6	Pinout	12
6.1	Single Connector Attachment design considerations	14
6.2	Single Connector Attachment signal definitions:	14
6.2.1	Power	14
6.2.2	Spindle Sync	14
6.2.3	LED OUT	14
6.2.4	Motor Start Controls	15
6.2.5	SCSI ID selection	15
6.2.6	SCSI Signals	16
6.2.7	Reserved Signals	16
6.3	SCSI Options	16
7	Connector and Drive Environment	17
8	Connector Properties	17

## Tables

Table 1	Single Connector Pinout	13
Table 2	Output Characteristics of LED Driver Signal	15
Table 3	Definition of Motor Start Controls	15
Table 4	Electronic Requirements for Motor Start Controls	15
Table 5	Definition of SCSI Device ID Selection Signals	16
Table 6	Electronic Requirements for SCSI ID drive	16

## Figures

Figure 1	Definition of Dimensions	10
Figure 2	Positioning of Connector	11
Figure 3	Maximum Protrusion Zone	12



## Foreword

Sun Microsystems, Seagate Technology, and Conner Peripherals have created a design for a single connector SCSI drive suitable for direct attachment to backplanes and motherboards. All three companies feel that the technology, while not suitable for all SCSI applications, will be useful in a wide variety of SCSI system designs. The first design is intended for 1" high 3.5" disk drives, but the connector has been selected so that the same relative dimensions can be applied to 1.6" high 3.5" disk drives and to 2.5" disk drives. The single connector carries all the required SCSI signals as well as the necessary power and subsystem control signals.

Sun Microsystems, Seagate Technology, and Conner Peripherals expect that this design will prove useful to the disk drive and computer system industries and will make the design available to interested companies.



# Single Connector Attachment for SCSI Disk Drives

## 1 Scope

This document defines a Single Connector Attachment (SCA) system designed for 8-bit and 16-bit SCSI devices. The single connector carries all standard SCSI signals as defined by the SCSI-3 Parallel Interface proposed standard. In addition, all required power and auxiliary signals are carried by the same single connector. The SCA mechanical definition allows the device to be plugged into a board socket. The dimensions are provided for 1" high 3.5" disk devices, but the same connector structure is appropriate for 1.6" high 3.5" devices as well as 2.5" devices.

## 2 Applicable Documents

SCSI-3 Parallel Interface (SPI), August 26, 1992, document X3T9.2/91-010 Revision 7

Small Computer System Interface 2 (SCSI-2), October 17, 1991, document X3T9/89-042 or X3T9.2/86-109, Revision 10H

## 3 Introduction

The Single Connector Attachment (SCA) system includes signals for a complete 8-bit or 16-bit single-ended SCSI. In addition, +5 V and +12 V power, spindle synch, I/D select, motor start control, and an LED drive signal are included in the connector definition. The SCA connector is designed and placed to allow plugging a drive directly into a backplane. The SCA connector provides the necessary electrical connection, but mechanical stability and device retention must be provided by other mechanisms, including mounting brackets, guide rails, clips, or screw attachments.

The connector selected for the SCA is the 80 position ribbon (ribbon, leaf, or single beam) connector made by AMP®, and sold as the Champ 0.050" Series 1. Various connector options are available to meet the different mounting requirements of the connector to the SCSI device and the different drive plugging requirements. Other connector manufacturers have compatible connector designs.

The SCA connector will allow drive-to-board mating.

Since power and address information are provided to the drives through the connector, special cables must be provided if daisy-chaining of drives is required.

The SCA is designed principally for the direct plugging of drives into a backplane. Even though a hot plugging capability is being investigated, it is still recommended that power be removed before removing or inserting a drive.

## 4 Connector Definition

The drive connector is a right angle or straddle mount plug, part number AMP Champ 557613-1 or AMP Champ 557114-5. The connector to which the drive plug mates is the AMP Champ 2-557103-1 vertical receptacle or the AMP Champ 2-557101-1 right angle receptacle.

The connector technology meets the following requirements:

- 80 signal contacts to provide power and all interface and control signals
- small connector to fit in the form factor of:
  - a) 1" high 3.5" disk drives

b) 1.6" high 3.5" disk drives

c) 2.5" disk drives

- Right angle or straddle mount device connector to allow flexibility in drive design
- Vertical or right angle board mount receptacle to allow flexibility in mating drive to backplane
- Shrouded contacts for mechanical protection of mating surfaces
- Polarized housing to prevent incorrect insertion of drive into backplane sockets
- Tolerant alignment guide-in to allow blind mating
- Greater than 500 mating cycles
- Acceptable insertion and withdrawal forces (90 and 20 grams/contact)
- Electrical properties suitable for the application

## 5 Location Within Drive

The SCA connector is fixed in two dimensions with respect to the drive form factor. The connector is flush with the end of the drive in the Y dimension (see Figure 1) and centered side to side in the end of the drive in the X dimension. In the Z dimension, a small amount of variation among drive vendors or models is allowed. Among drives of the same model, a very tight tolerance in the third dimension is still required to allow the design of drive mounting guides and brackets that will properly guide the connector to the board socket. See Note 2 of Figure 2. The slight variation in connector location requires the space reserved for the drive on systems that accept multiple models of drive to be large enough to accommodate this variation. See Figure 3 for the allowable protrusion zone.

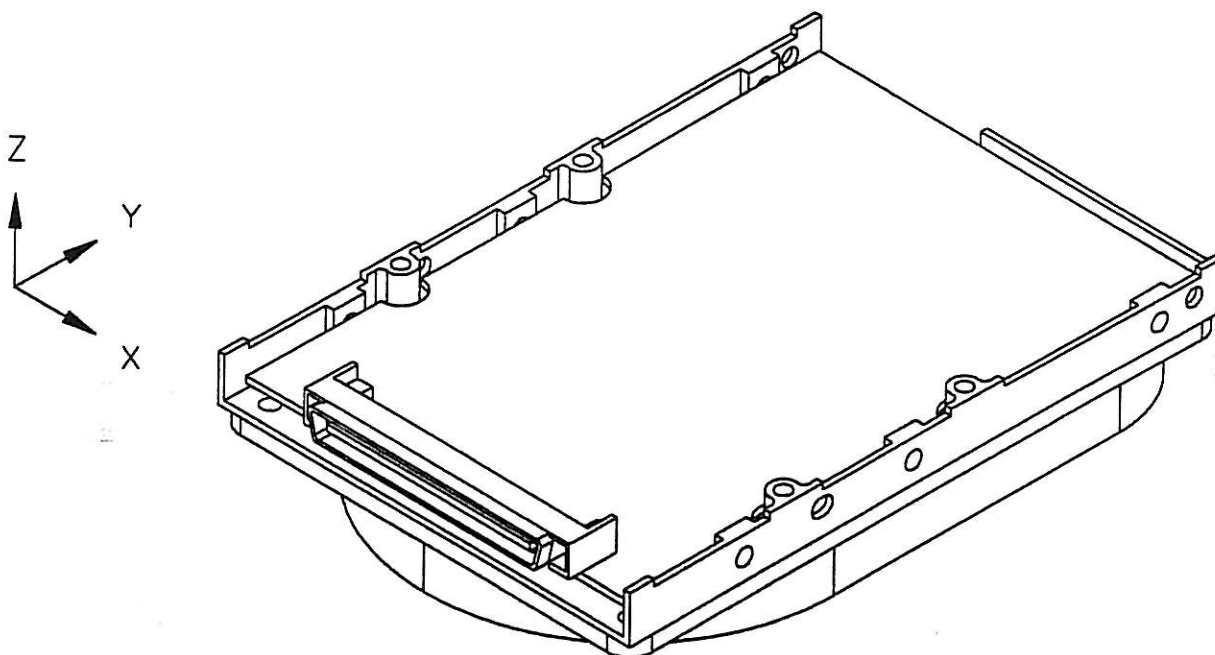
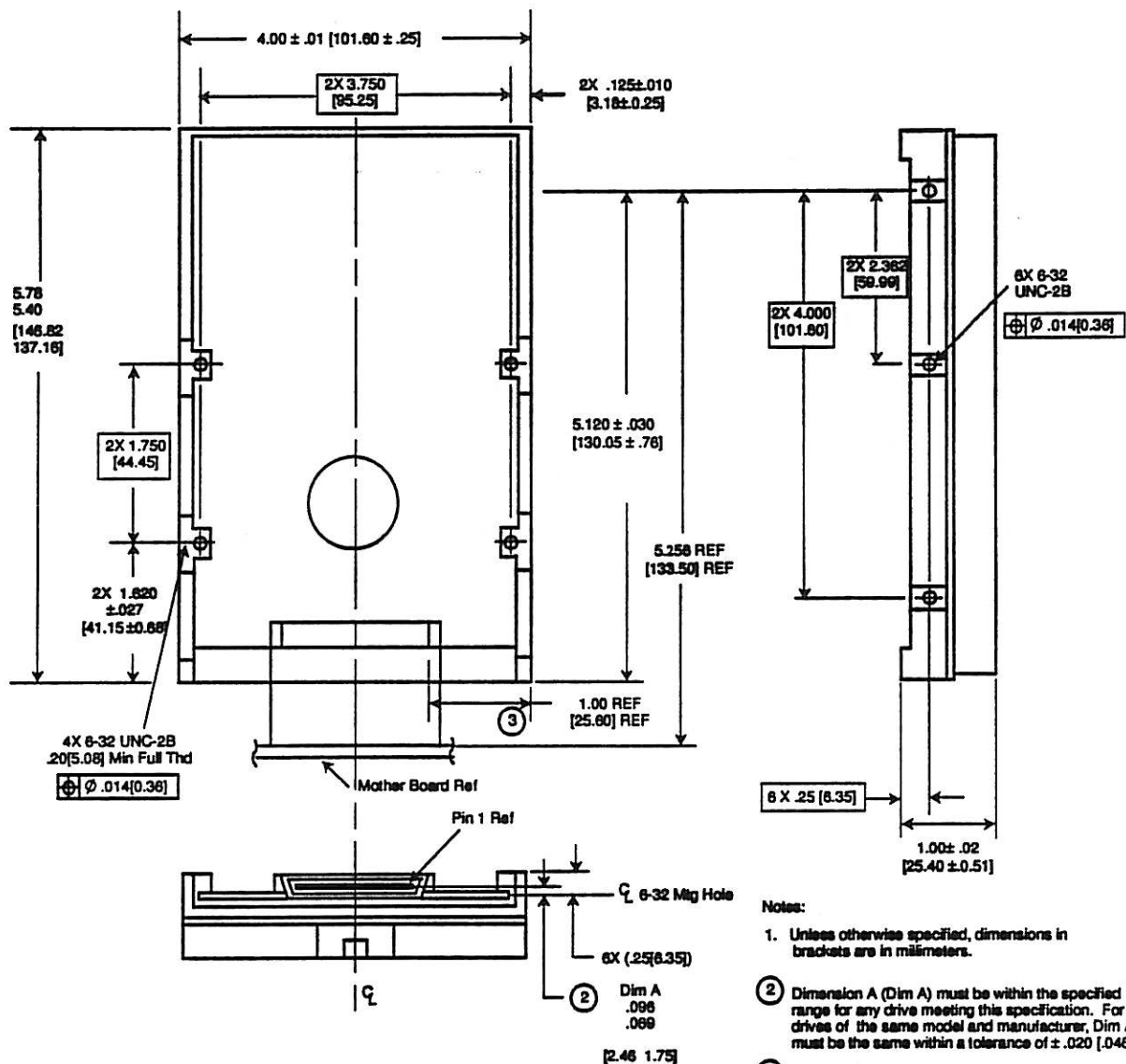


Figure 1: Definition of Dimensions

When the SCA disk drive is mated to a vertical board mount socket, there is  $3.45 \pm 0.7$  mm clearance between the disk drive and the mated backplane. See Note 4, Figure 2.

Figure 2 provides the dimensions and tolerances associated with an SCA drive using a 1" high 3.5" disk drive.

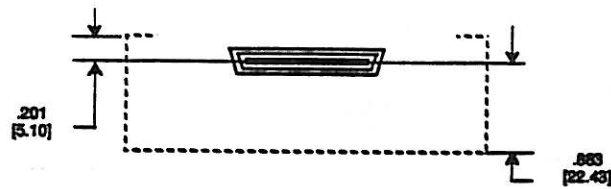


## Notes:

1. Unless otherwise specified, dimensions in brackets are in millimeters.
2. Dimension A (Dim A) must be within the specified range for any drive meeting this specification. For drives of the same model and manufacturer, Dim A must be the same within a tolerance of  $\pm .020$  [0.46].
3. Connector mating portion is centered and flush with drive edge.
4. Drawing is not to scale.

Figure 2, Positioning of Connector





**Figure 3, Maximum Possible Protrusion Zone of any Drive with respect to Connector**

## 6 Pinout

The SCA connector pinout accommodates the signals for 8-bit and 16-bit single-ended SCSI, +5 V and +12 V power, spindle synch, SCSI Device I/D Selection, spin-up control, and an LED driver. The pin out is shown in Table 1. Pin 1 is required to be located as shown in Figure 2.

**Table 1: Single Connector Pinout**

Connector Contact	Signal Name	Signal Name	Connector Contact
1	12 VOLT	12 V GROUND	41
2	12 VOLT	12 V GROUND	42
3	12 VOLT	12 V GROUND	43
4	12 VOLT	12 V GROUND	44
5	RESERVED/NC	RESERVED/NC	45
6	RESERVED/NC	RESERVED/NC	46
7	DB(11)	GROUND	47
8	DB(10)	GROUND	48
9	DB(9)	GROUND	49
10	DB(8)	GROUND	50
11	I/O	GROUND	51
12	REQ	GROUND	52
13	C/D	GROUND	53
14	SEL	GROUND	54
15	MSG	GROUND	55
16	RST	GROUND	56
17	ACK	GROUND	57
18	BSY	GROUND	58
19	ATN	GROUND	59
20	DB(P0)	GROUND	60
21	DB(7)	GROUND	61
22	DB(6)	GROUND	62
23	DB(5)	GROUND	63
24	DB(4)	GROUND	64
25	DB(3)	GROUND	65
26	DB(2)	GROUND	66
27	DB(1)	GROUND	67
28	DB(0)	GROUND	68
29	DB(P1)	GROUND	69
30	DB(15)	GROUND	70
31	DB(14)	GROUND	71
32	DB(13)	GROUND	72
33	DB(12)	GROUND	73
34	5 VOLT	5 V GROUND	74
35	5 VOLT	5 V GROUND	75
36	5 VOLT	5 V GROUND	76
37	SYNC	ACTIVE LED OUT	77
38	RMT_START	DLYD_START	78
39	SCSI ID(0)	SCSI ID (1)	79
40	SCSI ID (2)	SCSI ID (3)	80

## 6.1 Single Connector Attachment design considerations

No termination power supply lines are included. The SCSI termination circuits are on the platform backplane.

Special cable structures are required if the drives are being connected using cables instead of direct backplane insertion. The cable structures must provide the SCSI Device ID and control information, the power, and the required characteristic impedance and loading for the SCSI signals. Cable structures must be designed with special care to prevent the coupling of power supply noise into the SCSI signals.

The backplane applications will use power planes for power distribution.

## 6.2 Single Connector Attachment signal definitions:

### 6.2.1 Power

Four 12 VOLT signals provide +12 volt power to the device. The current return for the +12 volt power supply is through the 12 V GROUND signals. The maximum total current that can be provided to a drive through the 12 VOLT signal pins is 3 amps. The supply current and return current must be distributed as evenly as possible among the pins. The maximum current typically occurs while the drive motor is starting.

Three 5 VOLT signals provide +5 volt power to the device. The current return for the +5 volt power supply is through the 5 V GROUND signals. It is expected that the 5 V GROUND will also establish the digital logic ground for the device. The supply current and return current must be distributed as evenly as possible among the pins. The maximum total current that can be provided to a drive through the 5 VOLT signals is 2 amps.

The maximum current specified is related to the connector's characteristics. Additional limitations may be associated with the power supply's current budgets and with the system's power dissipation budget. Those limitations are not controlled by this specification.

### 6.2.2 Spindle Sync

The spindle sync is assigned a single pin, SYNC. The synchronization protocol and the electronic requirements for the SYNC signal are defined in the drive specification. Industry standards presently require that the drives interconnected for synchronization be the same or equivalent models. Spindle synchronization is managed by the SCSI command set. The signal current requirements shall not exceed 100 milliamperes and the signal voltage shall not be higher than 5.25 or lower than -0.25 volts. The minimum driver capability required by the SYNC signal shall be sufficient to drive the receivers on 30 identical disk drives.

The SYNC signal when driving should be capable of driving a minimum of 30 identical disk drives.

The SYNC signal is a source for noise and may be affected by noise. The design of the SYNC signal interconnections should take this into account by properly laying out the SYNC signals on the backplane or motherboard. Proper layout must consider routing relative to other signals, the proper line impedance, and terminations if necessary. The selection of the electronic transceiver must also take into account the possibility of noise. The signal levels, signal risetime, receiver thresholds, and receiver hysteresis must be considered as part of that selection.

### 6.2.3 LED OUT

The ACTIVE LED OUT signal is driven by the drive when the drive is performing a SCSI operation. The ACTIVE LED OUT signal is required to be implemented and is used to indicate that the disk drive is operating. Other optional indications can be provided by properly flashing the LED. The host system is not required to generate any visual output when the ACTIVE LED OUT signal is raised, but if such a visual output is provided, it must be white or green to indicate that normal activity is being performed.

The ACTIVE LED OUT signal is designed to pull down the cathode of an LED. The anode is attached to the proper +5 voltage supply through an appropriate current limiting resistor. The LED and the current limiting resistor are external to the drive.

**Table 2: Output Characteristics of LED Driver Signal**

STATE	CURRENT DRIVE AVAILABLE	OUTPUT VOLTAGE
DRIVE LED OFF	$0 < I_{OH} < 100 \mu A$	
DRIVE LED ON	$I_{OL} < -30 mA$	$0 < V_{OL} < 0.8 V$

#### 6.2.4 Motor Start Controls

The method of starting the drive's motor is established by the signals RMT\_START and DLYD\_START, as described in Table 3. The state of these signals can either be wired into the backplane socket or driven by logic on the backplane. The OPEN and GND states are established as described in Table 4.

**Table 3:**  
**Definition of Motor Start Controls**

Case	DLYD_START	RMT_START	Motor Spin Function
1	OPEN	OPEN	Motor spins up at DC power on.
2	OPEN	GND	Motor spins up only when SCSI "start" command is received.
3	GND	OPEN	Motor spins up after a delay of 12* seconds times the numeric SCSI target ID of the drive from DC power on.
4	GND	GND	Reserved. Drives not implementing this option shall execute power control according to the rules of Case 2.

\* This value may be reduced by drive suppliers to reflect the worst case time duration of peak current drains at the 12 volt or 5 volt source (or both) during motor spin up. In no case should the delay exceed 12 seconds.

**Table 4: Electronic Requirements for Motor Start Controls**

STATE	VOLTAGE	CURRENT
OPEN	$2.4 < V_{IH} < V_{CC} + 0.5$	$0 < I_{IH} < \pm 100 \mu A$
GND	$-0.5 V < V_{IL} < 0.4 V$	$0 < I_{OH} < -3 mA$

#### 6.2.5 SCSI ID selection

The SCSI device address of the attached drive is determined by the state of the signals SCSI ID(0-3). Table 5 indicates the relationship between the level of the SCSI ID signals and the selected SCSI device address. The OPEN and GND states are established as specified in Table 6.

**Table 5:**  
**Definition of SCSI Device ID Selection Signals**

Address	SCSI ID(0)	SCSI ID(1)	SCSI ID(2)	SCSI ID(3)
0	OPEN	OPEN	OPEN	OPEN
1	GND	OPEN	OPEN	OPEN
2	OPEN	GND	OPEN	OPEN
3	GND	GND	OPEN	OPEN
4	OPEN	OPEN	GND	OPEN
5	GND	OPEN	GND	OPEN
6	OPEN	GND	GND	OPEN
7	GND	GND	GND	OPEN
8*	OPEN	OPEN	OPEN	GND
9*	GND	OPEN	OPEN	GND
10*	OPEN	GND	OPEN	GND
11*	GND	GND	OPEN	GND
12*	OPEN	OPEN	GND	GND
13*	GND	OPEN	GND	GND
14*	OPEN	GND	GND	GND
15*	GND	GND	GND	GND

\* Addresses in the range from 8 to 15 are only supported by drives implementing the 16-bit SCSI option.

**Table 6: Electronic Requirements for SCSI ID Selection**

STATE	VOLTAGE	CURRENT
OPEN	$2.4 < V_{IH} < V_{CC} + 0.5$	$0 < I_{IH} < \pm 100 \mu A$
GND	$-0.5 V < V_{IL} < 0.4 V$	$0 < I_{OH} < -3 mA$

### 6.2.6 SCSI Signals

The SCSI signals implement a standard SCSI-2 interface. Standard SCSI voltage and current levels are supplied to the drives and expected from the drives. The 8-bit SCSI interface is defined by the SCSI-2 standard. The 16-bit SCSI interface is defined by the SCSI-3 Parallel Interface proposed standard. The SCSI signals defined are for standard single-ended drivers.

The ground leads opposing each SCSI signal are reserved signals. If differential SCSI implementations are ever required, the opposing signal will be used as the differential signal.

### 6.2.7 Reserved Signals

Reserved signals shall have no electronic connection to the disk drive circuitry or to motherboard circuitry until the use for those signals is defined by this specification.

## 6.3 SCSI Options

The device shall use the appropriate mandatory commands of the SCSI-2 command set. The support of optional SCSI-2 commands is negotiated between the drive vendor and customer.

The device shall have SCSI parity always enabled.

Support of a SCSI terminator is negotiated between the drive vendor and customer. If implemented on the drive, the terminator shall be a SCSI regulated terminator. The terminator on the drive must be removable or must have a mechanism allowing it to be disabled.

If the drive does not support 16-bit SCSI data transfers, then the signals DB(8) through DB(15) and DB(P1) shall not be electronically connected to the drive circuitry. The motherboard may optionally connect or not connect the high order SCSI signals to its internal SCSI host adapter. If the signals are supported by the motherboard, the signals must follow the standard SCSI rules for routing, characteristic impedance, and termination whether or not the attached drives connect to the signals.

## 7 Connector and Drive Environment

The drive connector will mate with the AMP Champ 2-557103-1 vertical receptacle or the AMP Champ 2-557101-1 right angle receptacle. The connector has been modified to allow for 0.040" of alignment tolerance along the X and Z axes. The connector will not be required to support the weight of the drive, although some vibration modes may place forces on the connector. Drives will be installed with appropriate brackets that will locate the drive and lock it in place.

## 8 Connector Properties

The "ribbon" or "leaf" contact connector, AMP Champ 0.050" Series 1, meets the following requirements:

Pin count	80
Contact resistance	35 m $\Omega$ maximum
Contact current rating	1 A per isolated contact
Durability	500 mating cycles minimum at maximum allowed misalignment
Insertion force	90 g/contact
Withdraw force	20 g/contact
Insulation resistance	1000 M $\Omega$ minimum at 250VDC; 500VDC desired
Dielectric withstanding voltage	500 V AC (rms), one minute
Hot plugging	Under Study
Contact plating	Appropriate to application requirements
Housing material	Appropriate to application requirements
Board retention features	Provided by backplane and associated hardware

**Intentionally Left Blank**

<b>To:</b>	John B. Lohmeyer	<b>Date:</b>	Jan 28, 1993
	Chairman X3T9.2	<b>From:</b>	Richard Whalen
	NCR		Digital Equipment Corporation
	1635 Aeroplaza Dr	<b>Phone:</b>	(508)-841-2684
	Colorado Springs, CO 80916	<b>Loc:</b>	SHR3-2/W28, Shrewsbury MA 01545
		<b>E-Mail:</b>	whalen@starch.dec.com

**Subject: Copy of SIMport specificaiton for January SCSI working group minutes**

---

John,

Enclosed is a copy of the specification that was presented to the CAM working group at the January SCSI working group meeting. This copy of the specification has a statement in it giving explicit permission to ANSI X3T9.2 to publish it and distribute it. Please let me know if anything additional is needed in order for this to be included as part of the minutes.

Sincerely,

Richard Whalen



# **SIMport**

## **HBA Interface Specification, V A.9, 1/7/93**

---

**This document specifies a SIM interface for intelligent host bus adapters. SIMport provides a SCSI Interface Module (SIM) like interface as defined in the specification entitled "SCSI-2 Common Access Method, Transport and SCSI Interface Module."**

---

## **COPYRIGHT NOTICE**

Copyright © January 1992 by Digital Equipment Corporation.

POSSESSION, USE, DUPLICATION OR DISSEMINATION OF THE SOFTWARE DESCRIBED IN THIS DOCUMENTATION IS AUTHORIZED ONLY PURSUANT TO A VALID WRITTEN LICENSE FROM DIGITAL OR THIRD PARTY OWNER OF THE COPYRIGHT.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERROR WHICH MAY APPEAR IN THIS DOCUMENT.

Digital Equipment Corporation has separately applied for patents on some of the material disclosed in this specification. A license to use any patents issuing as a result of these applications in SCSI CAM SIMport implementations will be granted with reasonable terms if this specification is accepted as part of the SCSI CAM specification.

Digital Equipment Corporation grants permission to ANSI X3T9.2 to publish this document and circulate it to the public at large.

The following are trademarks of Digital Equipment Corporation.

DEC, VMS

All Rights Reserved.  
Printed in U.S.A.



---

# Contents

<b>Preface</b> .....	xi
0.1      Revision History .....	xi
0.2      Purpose of this Specification .....	xii
0.3      Conventions .....	xii

## Glossary

## 1 Introduction

1.1      Goals .....	1-1
1.1.1      Interface modeled after CAM XPT to SIM interface .....	1-1
1.1.2      Provide a host independent interface .....	1-1
1.2      Reference Documents .....	1-1
1.2.1      CAM XPT SIM specification .....	1-1

## 2 Overview

2.1      SCSI Interface Module .....	2-1
2.2      Initialization .....	2-2
2.2.1      The Adapter Block .....	2-2
2.2.2      SIMport Queues .....	2-2
2.2.3      Adapter Initialization and State Control .....	2-2
2.2.4      Channel Initialization and State Control .....	2-3
2.3      Normal SCSI Channel Operation .....	2-3
2.3.1      CCB Processing .....	2-3
2.3.2      Response Queue Processing .....	2-4
2.4      Channel SIM Queues .....	2-4
2.5      Verify Adapter Sanity Timer .....	2-4
2.6      Channel Errors .....	2-5
2.7      Adapter Errors .....	2-5
2.8      Interrupts .....	2-5
2.8.1      Adapter Interrupts .....	2-5
2.8.2      Host Interrupts .....	2-5
2.9      Free Queue Element Allocation .....	2-6

## 3 Interface Components

3.1	Adapter Block .....	3-1
3.1.1	Queue Header .....	3-2
3.2	Queues .....	3-2
3.2.1	Driver-Adapter Command Queue .....	3-2
3.2.2	Adapter-Driver Response Queue .....	3-2
3.2.3	Driver-Adapter Free Queue .....	3-2
3.2.4	Adapter-Driver Free Queue .....	3-3
3.3	Queue Carrier .....	3-3
3.3.1	Queue Carrier Extension .....	3-4
3.3.1.1	SIM Queue Priority Flag .....	3-4
3.3.1.2	Return Status Format .....	3-4
3.4	Queue Buffer .....	3-4
3.5	Adapter Registers .....	3-4
3.5.1	Adapter Maintenance Control and Status Register (AMCSR) .....	3-5
3.5.2	Adapter Block Base Register (ABBR) .....	3-5
3.5.3	Driver-Adapter Command Queue Insertion Register (DACQIR) .....	3-5
3.5.4	Driver-Adapter Free Queue Insertion Register (DAFQIR) .....	3-5
3.5.5	Adapter Status Register (ASR) .....	3-6
3.5.6	Adapter Failing Address Register (AFAR) .....	3-7
3.5.7	Adapter Failing Parameter Register (AFPR) .....	3-7
3.6	Interrupt Holdoff Timer .....	3-8

## 4 Initialization and State Control

4.1	Initialization Steps .....	4-1
4.2	SIM Driver Initialization Steps .....	4-1
4.3	Adapter Initialization Steps .....	4-2
4.4	State Transitions .....	4-4
4.5	Adapter States .....	4-4
4.5.1	Uninitialized State .....	4-4
4.5.2	Disabled State .....	4-4
4.5.3	Enabled State .....	4-4
4.6	Channel States .....	4-4
4.7	Device States .....	4-5

## 5 Command and Response Processing

5.1	General Operating Characteristics .....	5-1
5.1.1	Command Response Protocol .....	5-1
5.1.2	Command Priority .....	5-1
5.1.3	Unrecognized Commands .....	5-1
5.2	Adapter Specific Commands and Responses .....	5-2
5.2.1	Reset Adapter Command .....	5-3
5.2.2	Initialize Adapter Command .....	5-3
5.2.3	Set Adapter State Command .....	5-3
5.2.4	Adapter State Set Response Message .....	5-4
5.2.5	Set Parameters Command .....	5-6
5.2.6	Parameters Set Response Message .....	5-7
5.2.7	Set Channel State Command .....	5-7
5.2.8	Channel State Set Response Message .....	5-8
5.2.9	Set Device State Command .....	5-9
5.2.10	Device State Set Response Message .....	5-9
5.2.11	Verify Adapter Sanity Command .....	5-10
5.2.12	Adapter Sanity Verified Response Message .....	5-10
5.2.13	Read Counters Command .....	5-10

5.2.14	Counters Read Response Message .....	5-11
5.2.15	BSD Request Message .....	5-13
5.2.16	BSD Response Command .....	5-14
5.2.17	Bus Reset Request Message .....	5-15
5.2.18	Bus Reset Response Command .....	5-16
5.3	Maintenance Commands and Responses .....	5-17
5.4	CAM defined Commands and Responses .....	5-17
5.4.1	Execute SCSI I/O Command .....	5-18
5.4.1.1	Command Timeout and Command Abort .....	5-20
5.4.1.2	Queued Operation Tag Assignment .....	5-20
5.4.1.3	Autosense .....	5-20
5.4.2	SCSI I/O Executed Response Message .....	5-20
5.4.3	Execute Target I/O Command .....	5-20
5.4.4	Target I/O Executed Response Message .....	5-21
5.4.5	Release SIM Queue Command .....	5-21
5.4.6	Abort SCSI I/O Command .....	5-21
5.4.7	Reset SCSI Bus Command .....	5-21
5.4.8	Reset SCSI Device Command .....	5-22
5.4.9	Terminate SCSI I/O Process Command .....	5-22
5.4.10	Set ASYNC Callback Command .....	5-22
5.4.11	AEN Response Message .....	5-23
5.4.12	Path Inquiry Command .....	5-23
5.4.13	Path Inquiry Response Message .....	5-24
5.4.14	Enable LUN Command .....	5-25

## 6 Unsolicited Message Processing

6.1	Free Queue Element Management .....	6-1
6.2	Unsolicited Reselection Message .....	6-1
6.3	Channel Disabled Message .....	6-2
6.4	Device Disabled Message .....	6-3

## 7 Target Mode Operation

7.1	Processor Mode .....	7-1
7.1.1	Inquiry command .....	7-1
7.1.2	Receive command .....	7-1
7.1.3	Request Sense command .....	7-1
7.1.4	Send command .....	7-1
7.2	Phase Cognizant Mode .....	7-1

## 8 HBA Engines

## 9 Interrupts

9.1	Host Interrupts .....	9-1
9.1.1	Command Complete Interrupt .....	9-1
9.1.2	Adapter Miscellaneous Interrupt .....	9-1
9.1.3	Systems with single interrupt only support .....	9-1
9.2	Adapter Interrupts .....	9-1
9.2.1	Initialization Interrupt .....	9-1
9.2.2	Reset Interrupt .....	9-1

9.2.3	Command Queue Insertion Interrupt .....	9-2
9.2.4	Free Queue Insertion Interrupt .....	9-2

## 10 Buffer Memory Mapping

10.1	Buffer Segment Map .....	10-1
10.2	Buffer Segment Map Options .....	10-3

## 11 Issues

### A CAM and SIMport function codes

A.1	Supported CAM functions codes .....	A-1
A.2	SIMport Adapter Specific function codes .....	A-2
A.3	SIMport Status codes .....	A-3

### B Unaligned Host Memory Buffers

B.1	Unaligned Buffers .....	B-1
B.1.1	Writing unaligned host buffers .....	B-2
B.1.2	Reading unaligned host buffers .....	B-2

### C Bus Reset and Bus Device Reset

C.1	Adapter detected bus reset .....	C-1
C.1.1	Adapter reset processing steps .....	C-1
C.1.2	SIM Driver reset processing steps .....	C-2
C.2	Adapter initiated bus reset .....	C-2
C.3	Adapter detected Bus Device Reset .....	C-2
C.4	Bus Device Reset .....	C-2
C.4.1	Adapter initiated Bus Device Reset .....	C-3
C.4.2	Host initiated Bus Device Reset .....	C-3

### D SIMport on 64-bit architectures ~~CAM Command Format~~

D.1	CCB format on systems with 64 bit addresses .....	D-1
-----	---	-----

### E CAM command formats

E.1	SIMport supported CAM CCB formats .....	E-1
E.1.1	Execute SCSI I/O Command .....	E-1
E.1.2	Release SIM Queue Command .....	E-2
E.1.3	Abort SCSI I/O Command .....	E-2
E.1.4	Reset SCSI Bus Command .....	E-2
E.1.5	Reset SCSI Device Command .....	E-2
E.1.6	Terminate SCSI I/O Process Command .....	E-2
E.1.7	Set ASYNC Callback Command .....	E-2
E.1.8	Path Inquiry Command .....	E-3
E.1.9	Enable LUN Command .....	E-3

## F Host-Adapter polling mode

## G SIMport Queues

G.1	Queue Structure Overview .....	G-1
G.1.1	Queue Mechanics .....	G-1
G.1.2	Addressing .....	G-3
G.2	Carrier Structure .....	G-4
G.3	Queue Insertion Procedures .....	G-5
G.3.1	Driver-Adapter Queues .....	G-5
G.3.2	Adapter-Driver Queues .....	G-7
G.4	Free Queues .....	G-8
G.5	Recovering Queue Buffers and Carriers on Adapter Crash .....	G-8

## H SIMport Data Structures

### Figures

Figure 2-1	CAM Model .....	2-1
Figure 3-1	Adapter Block .....	3-1
Figure 3-2	SIMport Queue Carrier Format .....	3-3
Figure 3-3	Adapter Block Base Register (ABBR) .....	3-5
Figure 3-4	Driver-Adapter Command Queue Insertion Register (DACQIR) .....	3-5
Figure 3-5	Driver-Adapter Free Queue Insertion Register (DAFQIR) .....	3-5
Figure 3-6	Adapter Status Register (ASR) .....	3-6
Figure 3-7	Adapter Failing Address Register (AFAR) .....	3-7
Figure 3-8	Adapter Failing Parameter Register .....	3-7
Figure 3-9	Command, Response, Message flow .....	3-9
Figure 4-1	Adapter/Driver initialization information exchange .....	4-3
Figure 4-2	Adapter and Channel State Transitions .....	4-4
Figure 5-1	Adapter Specific Command/Response Format .....	5-2
Figure 5-2	Set Adapter State Command .....	5-3
Figure 5-3	Adapter State Set Response Message .....	5-4
Figure 5-4	Set Parameters Command .....	5-6
Figure 5-5	Set Channel State Command .....	5-7
Figure 5-6	Channel State Set Response Message .....	5-8
Figure 5-7	Set Device State Command .....	5-9
Figure 5-8	Device State Set Response Message .....	5-10
Figure 5-9	Read Counters Command .....	5-11
Figure 5-10	Counters Read Response Message .....	5-12
Figure 5-11	BSD Request Message .....	5-13
Figure 5-12	BSD Response Command .....	5-14
Figure 5-13	Bus Reset Request Message .....	5-16
Figure 5-14	Execute SCSI I/O CCB Private Data .....	5-18
Figure 5-15	Adapter Specific Command/Response Format .....	5-23
Figure 5-16	Format of Enable LUN Command .....	5-25
Figure 6-1	Unsolicited Message Format .....	6-1
Figure 6-2	Unsolicited Reselection Message .....	6-2



Figure 6-3 Channel Disabled Message .....	6-2
Figure 6-4 Channel Disabled Message Parameter Block .....	6-3
Figure 10-1 Buffer Segment Map (BSM) .....	10-2
Figure 10-2 A Buffer Segment Descriptor (BSD) .....	10-2
Figure B-1 DMA of unaligned host buffers .....	B-2
Figure C-1 Adapter/Driver reset information exchanged .....	C-1
Figure C-2 Adapter/Driver bus device reset information exchanged .....	C-3
Figure D-1 64 Bit SIMport CCB Header format .....	D-1
Figure G-1 Queue Carriers and Queue Buffers .....	G-1
Figure G-2 Queue Structure .....	G-2
Figure G-3 Carrier Format .....	G-4
Figure G-4 Carrier Physical Address Pointer Fields .....	G-4
Figure G-5 Driver-Adapter Queue Structure .....	G-6
Figure G-6 Adapter-Driver Queue Structure .....	G-7

## Tables

Table 0-1 Revision History .....	xi
Table 3-1 Adapter Block Fields .....	3-2
Table 3-2 SIMport Queue Carrier field definitions .....	3-3
Table 3-3 SIMport Queue Carrier Flags definitions .....	3-4
Table 3-4 ASR bit field definitions .....	3-6
Table 3-5 AFPR error codes .....	3-7
Table 5-1 Adapter Specific Commands and Response Messages .....	5-2
Table 5-2 Set Adapter State Command Fields .....	5-3
Table 5-3 Adapter State Set Response Message Fields .....	5-4
Table 5-4 Adapter State Set Response Message Status .....	5-5
Table 5-5 Adapter State Set Response Message Flags .....	5-5
Table 5-6 Adapter State Set Response Message Xfer_Align Field .....	5-5
Table 5-7 Set Parameters Command Fields .....	5-6
Table 5-8 Set Parameters Command Flags Fields .....	5-7
Table 5-9 Parameters Set Response Message Status .....	5-7
Table 5-10 Set Channel State Command Fields .....	5-8
Table 5-11 Channel State Set Response Message Fields .....	5-8
Table 5-12 Channel State Set Response Message Status .....	5-9
Table 5-13 Set Device State Command Fields .....	5-9
Table 5-14 Device State Set Response Message Fields .....	5-10
Table 5-15 Device State Set Response Message Status .....	5-10
Table 5-16 Read Counters Fields .....	5-11
Table 5-17 Read Counters Flags .....	5-11
Table 5-18 Counters Read Response Message Fields .....	5-12
Table 5-19 Counters Read Response Message Status .....	5-12
Table 5-20 Required Counters Names and Offsets .....	5-13
Table 5-21 BSD Request Message Fields .....	5-14
Table 5-22 BSD Response Command Fields .....	5-14
Table 5-23 BSD Response Command Status .....	5-15
Table 5-24 BSD Response Command Buf_id .....	5-15
Table 5-25 Bus Reset Request Message Fields .....	5-16
Table 5-26 Bus Reset Request Reason Field .....	5-16

Table 5-27 Bus Reset Response Command Status .....	5-17
Table 5-28 CAM defined Commands and Response Messages .....	5-17
Table 5-29 Execute SCSI I/O CCB Private Data Fields .....	5-19
Table 5-30 Path Inquiry Field Responsibilities .....	5-24
Table 5-31 Enable LUN Command Fields .....	5-25
Table 6-1 Unsolicited Reselection Message Fields .....	6-2
Table 6-2 Channel Disabled Message Fields .....	6-3
Table 6-3 Channel Disabled Message Status .....	6-3
Table 6-4 Device Disabled Message Status .....	6-3
Table 10-1 Buffer Segment Map Fields .....	10-2
Table 10-2 Buffer Segment Descriptor Fields .....	10-3
Table A-1 CAM Function Codes .....	A-1
Table A-2 SIMport Function Codes .....	A-2
Table A-3 SIMport status codes .....	A-3
Table E-1 Execute SCSI I/O Command Field Usage .....	E-1
Table E-2 Execute SCSI I/O VU Flags .....	E-2
Table E-3 Set ASYNC Callback Command Fields .....	E-2
Table E-4 Enable LUN Command Fields .....	E-3
Table G-1 Carrier Structure .....	G-4
Table G-2 Carrier Physical Address Pointer Fields.....	G-5



---

# Preface

Please direct all comments and suggestions to:

Richard Whalen  
Storage Systems Architecture  
Digital Equipment Corporation  
334 South Street SHR3-2/W28  
Shrewsbury, Massachusetts 01545-4112  
E-Mail: whalen@starch.enet.dec.com  
Voice: 508-841-2684

## 0.1

## Revision History

---

**Table 0-1 Revision History**

---

Revision	Date	Authors	Description
A.1	6-Jan-1992	M. Mackay	First draft release
A.2	26-May-1992	M. Mackay R. Whalen	Second draft release. This update reflects architectural changes only. For example, text which clarifies or duplicates the CAM specification are not included in this revision.
A.3	24-Jul-1992	R. Whalen	Third draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT.
A.4	11-Sep-1992	R. Whalen	Fourth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Added a fourth header level which allowed some reorganization of chapter 5. Unspecified information in Appendix E.1 added.
A.5	16-Oct-1992	R. Whalen	Fifth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Specification of AFPR added.
A.6	2-Nov-1992	R. Whalen	Sixth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. Replaced Appendix G with new text suggesting alternatives for recovering queue resources on adapter crash.
A.7	20-Nov-1992	R. Whalen	Seventh draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. This update contains a re-working of the status codes to eliminate gaps and duplications. Also the format of the adapter specific commands and responses has been reorganized so that fields always end up in the same place and are of the same size.
A.8	18-Dec-1992	R. Whalen	Eighth draft release. This update reflects changes discussed in the notes file in SSAG::ARCH\$REVIEW:SIMPORT. The majority of the changes remove DEC specific considerations in preparation for presenting the specification at the upcoming CAM-2 meeting for possible inclusion in the CAM-2 spec. This includes a complete definition of the operation of SIMport

## u.2

## Purpose of this Specification

The purpose of this specification is to define the interface to a family of intelligent Host Bus Adapters. The intent is to provide a document such that both SIM driver development and adapter development can proceed independently.

## 0.3

## Conventions

[ ]	The square brackets are used to enclose implementation notes, issues, and missing text. When issues and missing text are resolved they will be removed.
n	The lower case letter 'n' is used in parameter tables to indicate an integer value.
pa	The lower case letters 'pa' are used in parameter tables to indicate a host physical address value.
va	The lower case letters 'va' are used in parameter tables to indicate a host virtual address value.
SBZ	SBZ in structure formats and parameter tables implies the field Should Be Zero for future architecture compatibility.
N.A.	Not applicable.
.D.	To be determined or supplied.
Numbers	Both decimal and hexadecimal numbers are used in this document. Hexadecimal numbers are followed by a subscripted 16.

### **AB**

The Adapter Block, AB, is a component of SIMport. It is a host resident data structure used to pass initialization information from the host to the HBA.

### **ABBR**

Adapter Block Base Register. The ABBR contains the physical address of the AB.

### **Adapter**

The interface between the host bus and the SCSI bus.

### **ADFQ**

The Adapter-Driver Free Queue, ADFQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from a SIMport adapter to the host. The command buffers in this case are completed immediate commands that do not require a host response.

### **ADRQ**

The Adapter-Driver Response Queue, ADRQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from a SIMport adapter to the host. The command buffers in this case are either response messages from previously issued commands or unsolicited messages generated by the adapter in response to unexpected events.

### **AEN**

A SCSI Asynchronous Event Notification.

### **AFAR**

The Adapter Failing Address Register contains the physical address of host memory associated with ASR errors.

### **AFPR**

The Adapter Failing Parameter Register passes implementation specific information on an ASR error.

### **AMCSR**

The Adapter Maintenance Control and Status Register controls the state and operating parameters of the adapter.

### **ASR**

The Adapter Status Register records adapter status that requires host intervention.

### **BDR**

Bus Device Reset of a target on the SCSI bus.

## **BSD**

The Buffer Segment Descriptor, BSD, is a component of SIMport. A BSD defines the address and size of a host memory segment or of a Buffer Segment Map (BSM).

## **BSM**

The Buffer Segment Map, BSM, is a component of SIMport. BSMs are used to define a physically discontinuous host memory buffer. A BSM defines a list of BSDs, where each BSD defines a host memory segment.

## **CAM**

The Common Access Method, CAM, is a ANSI standard which defines the software interface between device drivers and the Host Bus Adapters or other means by which SCSI peripherals are attached to a host processor.

## **CCB**

The CAM Control Block, CCB, is the data structure used to pass CAM command information across the various CAM architectural layers.

## **CDB**

The SCSI Command Descriptor Block, or a pointer to the CDB.

## **Channel**

A SCSI bus on the adapter.

## **DACQ**

The Driver-Adapter Command Queue, DACQ, is a component of SIMport. It is a host resident queue type data structure used to pass command buffers (CCBs) from the host to a SIMport adapter.

## **DACQIR**

The Driver Adapter Command Queue Insertion Register is used to notify the adapter of new entries on the DACQ.

## **DAFQ**

The Driver-Adapter Free Queue, DAFQ, is a component of SIMport. It is a host resident queue type data structure also used to pass command buffers (CCBs) from the host to a SIMport adapter. The command buffers in this case are to be used by the adapter in the generation of unsolicited message.

## **DAFQIR**

The Driver Adapter Free Queue Insertion Register is used to notify the adapter of new entries on the DAFQ.

## **doublet**

Two bytes (16 bits) of data.

## **HBA**

The Host Bus Adapter, HBA, is the physical layer of CAM that interfaces a SIM with a SCSI bus.

## **Host Memory Segment**

A host memory segment is a physically contiguous section of host memory. It is defined (in a BSD) by the physical address of the start of the section and the number of bytes in the section.

## **Hexaword**

A unit of memory equal to 16 2-byte words. A data structure that is hexaword aligned will have the low 5 bits <4:0> of its address equal to 0 (zero).

**LSB**

Least Significant Bit.

**LUN**

A Logical Unit, LUN, is an addressable entity of a SCSI device. A single SCSI device may have up to eight LUNs.

**octalet**

Eight bytes (64 bits) of data.

**MSB**

Most Significant Bit.

**quadlet**

Four bytes (32 bits) of data.

**SIM**

The SCSI Interface Module, SIM, is the layer of CAM that interfaces the CAM XPT layer with the CAM HBA layer.

**SIM Queue**

A CAM defined queue used to hold SCSI I/O commands that are waiting for SCSI bus arbitration. A separate queue exists for each LUN.

**XPT**

The XPT is the transport layer of CAM. It directs CAM commands to appropriate SIMs.







# Introduction

## 1.1

### Goals

#### 1.1.1 Interface modeled after CAM XPT to SIM interface

SIMport should model the interface between the CAM XPT layer and the CAM SIM layer.

#### 1.1.2 Provide a host independent interface

SIMport should be host operating system independent.

## 1.2

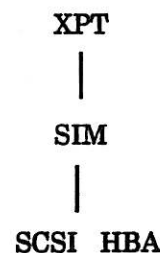
### Reference Documents

Concepts and components of this specification are based on the following specifications.

#### 1.2.1 CAM XPT SIM specification

Refer to the specification entitled "SCSI-2 Common Access Method Transport and SCSI Interface Module Rev 3.0 Apr 27, 1992" for information about the CAM architecture.

SIMport is a host bus adapter (HBA) interface definition. SIMport is designed to provide access to HBAs that support the functions of the SCSI Interface Module (SIM) as defined by the SCSI-2 Common Access Method (CAM). The SIM is a functional layer of the CAM architecture. The SIM is layered under the CAM transport (XPT) layer and above the SCSI HBA layer (see figure 2-1).



**Figure 2-1 CAM Model**

SIMport defines a communication mechanism between the XPT layer (on the host) and a SIM layer (on the adapter) such that most if not all the CAM defined functions of the SIM layer can reside in the HBA.

## 2.1

### SCSI Interface Module

SIM, as defined by the CAM XPT/SIM specification provides the following functions:

- Perform all SCSI interface functions, including protocol steps and error detection and recovery, as required.
- Manage data transfer path hardware, including address mapping and DMA resources.
- Perform queuing of multiple operations for different LUNs as well as the same LUN and assign tags for Tag Queuing (if supported).
- Freeze and un-freeze the SIM Queues as necessary to accomplish queue recovery.
- Assure that completed operations are posted back to the initiating peripheral driver.
- Assure that supported asynchronous events are detected and posted, as required.
- Implement a timer mechanism, using values provided by the peripheral driver.

The SIM layer of CAM has a well-defined interface. A function call (*sim\_init*) is defined to initialize the SIM. A function call (*sim\_action*) is defined to pass CAM Control Blocks (CCBs) to the SIM. CCBs contain commands to both initiate and control SCSI I/O. In addition, the SIM interface defines a function callback mechanism. The callback mechanism is used to notify CAM upper protocol layers of SCSI I/O CCB command completion as well as Asynchronous Event notification (e.g., bus reset).

## Initialization

SIMport allows for multiple SCSI Channels on a single adapter. To separate the control of the adapter from the control of the SCSI channels, the adapter and channels are defined as separate entities.

The adapter part is responsible for establishing and maintaining communication with the SIM driver. Queues are used as the communication mechanism. The adapter part has complete control over the channels.

A channel part will be responsible for a single SCSI bus. It will accept commands from the adapter part to initialize as well as perform SCSI bus I/O.

### 2.2.1 The Adapter Block

The Adapter Block (AB) is a block of host memory shared with the adapter that contains queue headers and configuration parameters. The queues are the mechanism for passing command and response information between the adapter and the host.

### 2.2.2 SIMport Queues

SIMport Queues are used to pass CAM Control Blocks (CCBs) between the host and the adapter. The queues consists of queue headers, queue carriers, and queue buffers. A queue header consists of the queue head and queue tail pointers. A queue carrier forms the queue linkage and is used to pass pointers to queue buffers. The queue buffers contain CCBs.

The queue structure was chosen after consideration of interlocked queues and rings. Interlocked queues offer dynamic expandability, but are expensive due to the number of serialized memory references required and contention for memory interlocks. Rings offer relatively inexpensive memory access, but are not simply expandable, may result in cache contention, and the normal execution code must include some overhead to support ring overrun. These queues offer dynamic expandability and the simple synchronized access of rings. See appendix G for greater detail on the queues and operations on them.

SIMport defines four queues. The Driver-Adapter Command Queue (DACQ) is used to pass CCB commands to the adapter. The Adapter-Driver Response Queue (ADRQ) is used to pass CCB command response messages and unsolicited event messages to the host. The Driver-Adapter Free Queue (DAFQ) is used to provide free queue elements to the adapter. The Adapter-Driver Free Queue (ADFQ) is used to return free queue elements to the host.

Note that CCB formatted queue buffers that are placed on the command queue (DACQ) are called commands and that CCB formatted queue buffers that are placed on the response queue (ADRQ) are called messages.

### 2.2.3 Adapter Initialization and State Control

After the first call to the *sim\_init* routine within the SIM driver, the SIM driver writes the MIN bit in the Adapter Maintenance Control and Status Register (AMCSR) to reset the adapter and waits for the reset to complete. Next the SIM driver constructs the Adapter Block, including the initialization of the queues. The initialization of the queues includes placing a *Set Adapter State* command on the command queue (DACQ). The SIM driver then writes the Adapter Block Base Register (ABBR) in an attempt to initialize the adapter. Writes to adapter registers interrupt the adapter.

The value written to the ABBR is the physical address of the Adapter Block (AB). The adapter reads and verifies the information in the AB and performs some internal initialization. At this point, the adapter knows the physical address of the queue headers and thus all subsequent communication between the adapter and the SIM driver is performed using the SIMport queuing mechanism.

After some internal initialization, the adapter detects and reads the *Set Adapter State* command from the DACQ. The command instructs the adapter to transition to the disabled state. The adapter responds to this request with an *Adapter State Set* response message. The adapter writes the *Adapter State Set* response message to the response queue (ADRQ). The SIM receives the response message indicating that the adapter is now out of the *uninitialized* state and now in the *disabled* state. See appendix F for a mechanism that allows the SIM to poll adapter registers in lieu of interrupts.

If the adapter cannot transition to the *disabled* state, the adapter updates adapter status registers and issues an Adapter Miscellaneous Interrupt to the host. The host is expected to read the appropriate adapter status registers to determine the reason for failure.

The *Adapter State Set* response message notifies the SIM that the adapter is in the disabled state. As part of this message, the adapter informs the SIM of adapter specific configuration parameters and options. In the *disabled* state, the SIM driver and the adapter may exchange information via the command and response queues to establish implementation and configuration parameters. In addition, this state allows diagnostic and debugging information exchange.

When the information exchange completes, the SIM driver requests the adapter to transition to the *enabled* state by sending another *Set Adapter State* command. The adapter responds with an *Adapter State Set* response message indicating success or failure.

## 2.2.4 Channel Initialization and State Control

The SIM driver initializes each channel with the *Set Channel State* command. The adapter responds with a *Channel State Set* response message indicating success or failure.

The *Set Channel State* command allows the initialization to model the CAM SIM initialization, where a separate call to *sim\_init* is made for each channel. The *sim\_init* call passes a *path\_id* to assign to the channel and the channel must be ready for CCB commands on return from *sim\_init*. The adapter maintains the mapping of *path\_id* to adapter *channel\_id* such that the *path\_id* in subsequent CCB commands is used to address the channel.

The first call to *sim\_init* should cause the SIM driver to do a complete adapter initialization, starting from resetting the adapter. Subsequent calls to *sim\_init* should detect that the adapter is enabled and simply send the *Set Channel State* command.

## 2.3

## Normal SCSI Channel Operation

Normal SCSI Channel operation consists of two processing threads. One thread is activated by the XPT CAM layer calling the *sim\_action* routine, passing a CCB to process. The other thread is activated via an interrupt from the adapter. The interrupt notifies the SIM driver that new messages are on the response queue (ADRQ).

### 2.3.1 CCB Processing

The SIM driver places each CCB command, received from the XPT (via *sim\_action*), on the command queue (DACQ) to the adapter. There are two types of CCB commands: immediate CCB commands and queued CCB commands.

Immediate CCB commands are expected to be completed immediately. The implication is that the CCBs for immediate CCB commands are implicitly returned to the XPT layer on return from the *sim\_action* call. Thus, the SIM driver has two choices. One is to queue the original CCB command to the command queue, wait for the adapter to return a response on the response queue, and then return from the *sim\_action* call. The other is to copy the original CCB to a private CCB, queue the private CCB to the adapter, and then return the original CCB to the caller without waiting for a response.

Most immediate CCB commands always return a success status; thus for these commands, the SIM driver should copy the CCB to a private CCB and return the original CCB indicating the success status.

An optimization is used such that the SIM driver is required to copy the immediate CCBs that always return success. The SIM driver does not wait for status and the adapter does not have to DMA to the status field. Also, this reduces host interrupts since the adapter returns the queue element to the ADFQ instead of the ADRQ.

Some immediate CCB commands do return a meaningful status. Specifically, the Path Inquiry CCB command. The assumption here is that the SIM driver should issue these types of immediate CCB commands during initialization and that during this time it is acceptable for the SIM driver to wait for the response on the response queue.

Queued CCB commands are Execute SCSI I/O requests and remain queued within the adapter on return from the *sim\_action* call. The CCB for queued CCB commands is returned when the SCSI I/O completes. Thus for queued CCB commands, the SIM driver queues the original CCB to the adapter and returns 'request in progress' to the XPT layer.

When the adapter receives a queued CCB, it places the CCB on the appropriate internal SCSI channel SIM Queue. When the SCSI channel completes the queued CCB command, it places a command response message on the response queue (ADRQ).

### 2.3.2 Response Queue Processing

The SIM driver is interrupted by the adapter to inform the SIM driver of new messages on the response queue (ADRQ). There are two main types of messages that can be on the response queue. One type is a command response message, indicating that a previously queued command has completed. The other type is an unsolicited message, typically used to inform the SIM of an unexpected channel event.

Command completion is initiated by a SCSI channel when it completes a SCSI I/O. The channel constructs a command response message and places it on the ADRQ. The response message is the completed SCSI I/O CCB. CAM status is not written to the CAM status field of the CCB, but to the Private Data area of the CCB. The SIM driver obtains the CCB from the ADRQ and copies the CAM status from the Private Data area to the CAM status field in the CCB before it performs the command complete callback.

Unsolicited messages are typically initiated by the channel when it detects an unexpected event (e.g., bus reset). The adapter obtains a queue element from the free queue (DAFQ), writes the SIMport function CCB format to the queue buffer, and places it on the response queue (ADRQ). The message typically invokes an asynchronous callback to the XPT or peripheral driver layers of CAM (although the callback does not occur until all outstanding adapter I/O is returned).

## 2.4

### Channel SIM Queues

Adapter resources must be such that it can always execute immediate commands. For queued commands (i.e., Execute SCSI I/O CCBs), the adapter must maintain SIM Queues as defined by the CAM XPT/SIM specification. A SIM Queue should exist for each LUN for each device. If adapter resources are exhausted, the adapter must be able to enqueue queued commands to internal SIM Queues, but leave the queue carriers and queue buffers in host memory.

Queued commands on SIM Queues are dispatched when the SCSI bus is free using a round-robin algorithm.

## 2.5

### Verify Adapter Sanity Timer

A command is supported, *Verify Adapter Sanity*, so that the SIM can verify the operation of the adapter DACQ and ADRQ. Essentially, the SIM driver is expected to run an inter-



val timer. Whenever the interval timer expires the SIM driver will ensure that the adapter has returned a previously issued *Verify Adapter Sanity* command. If it has, the SIM driver will issue another *Verify Adapter Sanity* command, re-start the timer, and return. If the *Verify Adapter Sanity* command has not been returned, the SIM driver will assume the adapter is broken and proceed to reset it. The minimum timer value is specified by the *Adapter State Set* message during initialization, see section 5.2.4.

## 2.6

### Channel Errors

Channel errors do NOT stop adapter operation. That is, the adapter command/response mechanism remains operational. A channel error is reported as an unsolicited message. Any pertinent register information is placed in the message (queue buffer) which is queued to the response queue. The message essentially informs the host that a channel error has occurred and may require SIM driver intervention to re-enable the channel.

## 2.7

### Adapter Errors

Adapter errors DO stop the adapter and all channel operations. The adapter will write all pertinent information about the error to appropriate registers and interrupt the host with an Adapter Miscellaneous Interrupt. The interrupt informs the host that the adapter is broken. The SIM driver should read all pertinent adapter registers and log the error to the host based error logger. The SIM driver must do a complete re-initialization as indicated above to resume normal adapter/channel operations.

## 2.8

### Interrupts

Interrupts in SIMport are used sparingly. The main reason for this is to reduce complexity. Multiple interrupt threads typically require synchronizing using locking primitives to prevent race conditions. An example would be the adapter interrupting the host to request more free queue entries when the host is executing code that accesses the free queue. That is, either the host CCB command processing thread or the host response queue processing thread is interrupted by the adapter to request more free queue entries. The host would need a locking mechanism to synchronize access to the free queue.

SIMport reduces the use of interrupts by requiring most adapter and host communication to use the command and response queue mechanism (which is interrupt driven). If, for example, it were determined that a mechanism were required for the adapter to request more free queue entries, the mechanism would be an unsolicited message placed on the response queue. This would synchronize the request with other work in progress.

#### 2.8.1 Adapter Interrupts

There are four interrupts from the SIM driver to the adapter, each corresponding to an adapter based register write. For initialization, the SIM driver writes the Adapter Block Base Register. To notify the Adapter of new commands on the DACQ, the SIM driver writes the Command Queue Insertion Register. To notify the Adapter of new entries on the Driver-Adapter Free Queue the SIM driver writes the Adapter Free Queue Insertion Register. To reset the adapter, the SIM driver writes the MIN bit in the Adapter Maintenance Control and Status Register.

#### 2.8.2 Host Interrupts

There are two interrupts from the adapter to the SIM driver. To inform the SIM driver of new messages on the ADRQ, the adapter issues a Command Complete interrupt to the host. To inform the SIM driver that the adapter requires servicing, the adapter issues an Adapter Miscellaneous interrupt. The adapter does not interrupt the host to request more free queue entries (see section 2.9).

A Command Complete interrupt is subject to the *interrupt holdoff* timer. The interrupt holdoff timer mechanism is an optional adapter feature that can be enabled during



adapter initialization. When enabled, the adapter limits Command Complete interrupts such that only one interrupt is generated per holdoff timer value, as opposed to an interrupt per response queue message. The holdoff timer value is specified by the host during adapter initialization with the *Set Parameters* command, see section 5.2.5.

## 2.9

### Free Queue Element Allocation

During initialization, the SIM driver allocates to the adapter (on the DAFQ) a specific number of free queue entries for use by the adapter for unsolicited messages. The SIM driver is required to maintain this number. When the host receives an unsolicited message from the adapter, it is required to return the queue element to the adapter via the free queue (DAFQ) or provide another queue element.

Since the host is required to maintain the free queue level (on the DAFQ) as indicated, the adapter does not interrupt the host to request more free queue entries. The reason for this is that the host must have already been interrupted to process the unsolicited messages and thus an additional interrupt would be redundant and may increase the time it takes for the host to provide free queue entries.

The adapter will wait indefinitely for a free queue element to be placed on the DAFQ. This is not a problem for adapter target mode operation, since free queue elements are not used for target mode operations. Under CAM, the host is required to pre-allocate CCBs to the adapter for target mode operation.

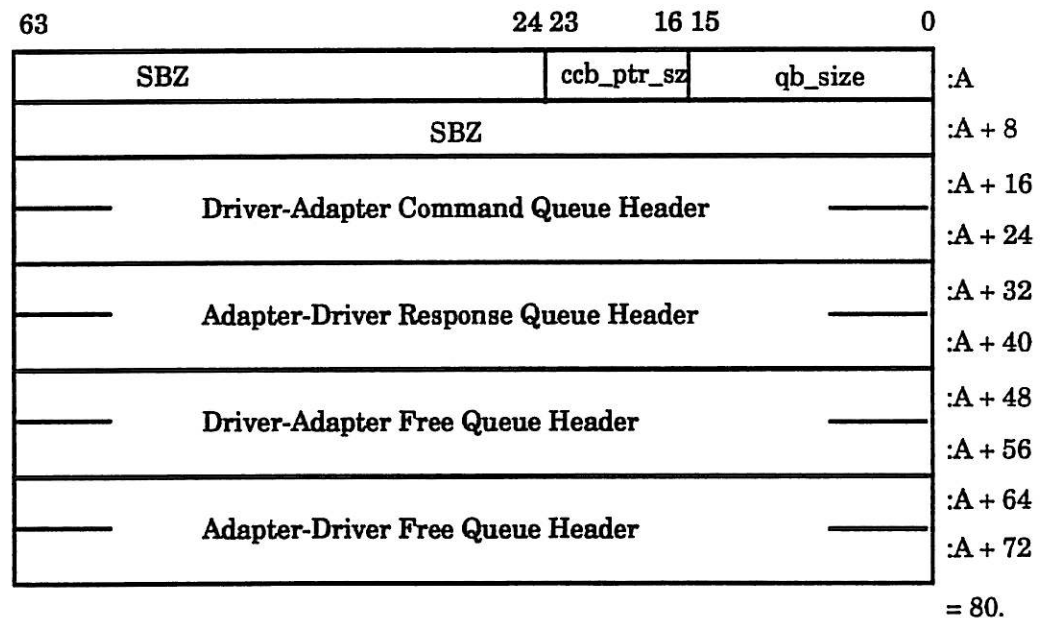
## Interface Components

This chapter defines the interface components between the SIM driver and the adapter. The interface components consist of the Adapter Block, SIMport style queues, Queue Carriers, Queue Buffers, and Adapter Registers. Figure 3-9 illustrates the interface components as well as indicates the flow of queued commands, immediate commands, unsolicited messages, and callback completion control.

### 3.1

#### Adapter Block

The Adapter Block (AB) is a hexaword aligned, physically contiguous, host resident data structure. The AB is shared by the SIM driver and the adapter. It contains the SIMport style queue headers and configuration parameters. The adapter is passed the physical address of the AB during adapter initialization through the Adapter Block Base Register. The AB is read-only to the adapter. The adapter maintains private copies of all driver-adapter queue heads (DACQ, DAFQ) and adapter-driver queue tails (ADRQ, ADFQ).



**Figure 3-1 Adapter Block**

Table 3-1 lists, for each Adapter Block field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 3-1 Adapter Block Fields**

Field	Size	Value	Description
qb_size	<15:0>	n	Number of bytes in the fixed length Queue Buffer.
ccb_ptr_size	<23:16>	4,8	Number of bytes allocated for a CCB pointer field.

### 3.1.1 Queue Header

Queue headers reside in the Adapter Block. The Queue Header is used to maintain a pointer to the head of the queue and a pointer to the tail of the queue. Queue Header head and tail pointers are physical addresses for adapter use and virtual addresses for SIM driver use. Most adapters will choose to cache the queue headers in the Adapter Block to improve performance. The following is a list of Queue Header pointers that are physical addresses for exclusive use by the adapter. The remaining pointers are virtual addresses for SIM driver use.

- Driver-Adapter Command Queue head pointer
- Driver-Adapter Free Queue head pointer
- Adapter-Driver Response Queue tail pointer
- Adapter-Driver Free Queue tail pointer

The SIM driver, during adapter initialization, initializes the queues by placing 'stopper' queue carriers on all queues. In addition, during adapter initialization, the SIM driver places a *Set Adapter State* command on the command queue (DACQ).

## 3.2

### Queues

The queues are one-way (linked list) queues, either host to adapter or adapter to host. They are 'producer/consumer' type queues, where one side (the host or the adapter) places elements on the tail of the queue and the other side removes elements from the head of the queue. The queues allow simultaneous access by both the SIM driver and the adapter without synchronization primitives. The queues allow the SIM driver to access the queue using host virtual addresses, while the adapter uses physical addresses. In addition, the queues allow the adapter to post multiple responses with a single interrupt. See Appendix G for details on operating on the queues.

#### 3.2.1 Driver-Adapter Command Queue

All commands for the adapter, except initialize and reset, are passed to the adapter on the Driver-Adapter Command Queue (DACQ). The DACQ is to be processed such that it simulates the CAM XPT to SIM function *sim\_action*. That is, it is designed to emulate a called interface. The implication is that the adapter will always accept commands over the DACQ and never block.

#### 3.2.2 Adapter-Driver Response Queue

All Adapter command response messages and unsolicited event messages are passed to the host on the Adapter-Driver Response Queue (ADRQ). The ADRQ is to be processed such that it simulates the CAM SIM to XPT command and asynchronous event callback.

#### 3.2.3 Driver-Adapter Free Queue

All free queue elements are passed to the adapter on the Driver-Adapter Free Queue (DAFQ). The adapter will obtain free queue entries from the DAFQ to create unsolicited

messages for the SIM driver. During initialization, the SIM driver will place a fixed number of free queue entries on the DAFQ. The SIM driver is responsible for maintaining the number of free queue entries on the DAFQ to the level established during adapter initialization. The response to the initial *Set Adapter State* command contains the requested number of free queue entries the adapter needs for efficient unsolicited message generation. The host is required to place at least one queue element on the DAFQ before the adapter is enabled via the *Set Adapter State* command.

### 3.2.4 Adapter-Driver Free Queue

All free queue elements are returned to the host on the Adapter-Driver Free Queue (ADFQ). The SIM driver passes immediate commands to the adapter on the DACQ. For immediate commands that do not require a command response message, the queue elements should be returned to the host via the ADFQ.

## 3.3

### Queue Carrier

The Queue Carrier is a quadword (octalet) aligned, physically contiguous, host resident data structure. The format of a Queue Carrier is a modified version of the format defined in Appendix G. The next pointer field in the Queue Carrier is used to form the one-way linked list that represents the queue. The Queue Carrier contains a pointer to a Queue Buffer. As well as the next pointer field and the queue buffer field, the SIMport Queue Carrier contains an additional 64 bits. This allows a common SIMport header to be used for all commands (CAM defined and SIMport defined) and contains a function code and a status field such that all commands are executed based on the SIMport header function code and the completion status is always posted to the SIMport header status field. As well as providing a canonical location for the status, the extended queue carrier allows the adapter to reduce the number of times that it has to read the queue buffer when the adapter's resources are saturated by using the function code to determine which commands are adapter specific. Figure 3-2 illustrates the format of the SIMport Queue Carrier.

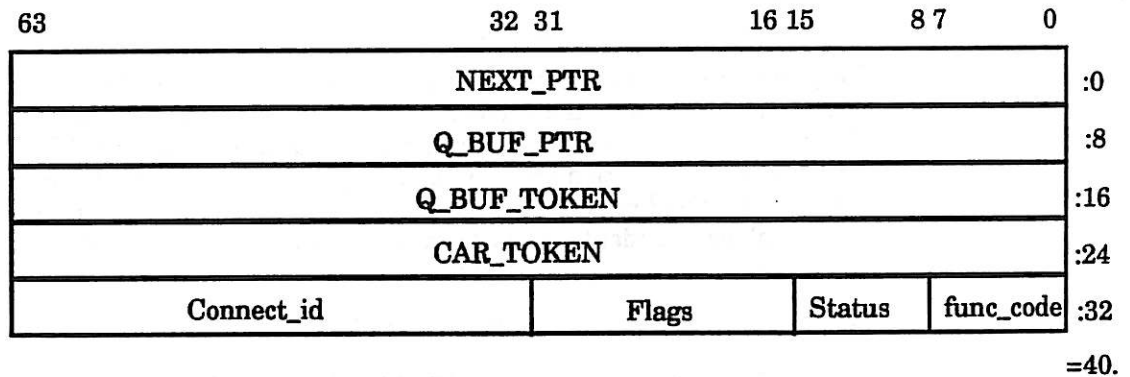


Figure 3-2 SIMport Queue Carrier Format

Table 3-2 lists the contents and description of the fields in the Queue Carrier.

Table 3-2 SIMport Queue Carrier field definitions

Field	Size	Driver to Adapter Value	Adapter to Driver Value	Description
NEXT_PTR	<63:0>	pa	va	Pointer to next carrier or stopper indicator.

Q_BUF_PTR	<63:0>	pa	va	Pointer to Q_BUFFER. Q_BUF_PTR is undefined when the carrier is a stopper.
Q_BUF_TOKEN	<63:0>	va	undefined	Queue Buffer port driver address
CAR_TOKEN	<63:0>	va	va	Carrier port driver address
func_code	<7:0>	n		CAM or SIMport function code
Status	<15:8>	n		Status of CAM or SIMport command
Flags	<31:16>	see table 3-3		Flags for queue insert priority and status return
Connect_id	<63:32>	see spec		CAM defined connect id

Pointers are physical addresses when used by the adapter and virtual addresses when used by the driver.

Table 3-3 lists the definition of the flags field.

**Table 3-3 SIMport Queue Carrier Flags definitions**

Field	Bit	Description	Set by
SIMQueuePriority	16	1: Head Insertion, 0: Tail Insertion	Host
ReturnStatusFormat	17	1: SIMport specific, 0: CAM normal	HBA

### 3.3.1 Queue Carrier Extension

The Queue Carrier Extension consists of the Func\_code, Status, Flags and Connect\_id fields of the Queue Carrier. The host shall write the proper values to the Func\_code, Status, Flags and Connect\_id fields as part of the process of placing a new command on the DACQ. The adapter shall write the proper values to the Func\_code, Status, Flags, and Connect\_id fields as part of the process of placing a response on the ADRQ.

#### 3.3.1.1 SIM Queue Priority Flag

The QueuePriority flag controls SIM queue insertion order. This should normally be set to 0 (zero) so that commands are executed in the order that they are issued. When set to 1 the command is inserted at the head of the SIM queue for the LUN; this is generally only used when recovering from error conditions.

#### 3.3.1.2 Return Status Format

This flag is used to distinguish between SIMport specific error codes and CAM error codes on CAM defined function codes.

## 3.4

### Queue Buffer

The Queue Buffer is a hexaword aligned, physically contiguous, host resident data structure. The format of a Queue Buffer is as defined by the CAM XPT/SIM specification, with SIMport formats defined by this specification. The Queue Buffer is used to pass command and response information between the host and the adapter. A pointer to a queue buffer is passed in a Queue Carrier.

## 3.5

### Adapter Registers

Adapter registers are used to reset the adapter, initialize the adapter, and to inform the adapter of new entries on the Driver-Adapter queues. Adapter registers are all 64 bit registers (although for some adapter implementations only the lower 32 bits are valid).

### 3.5.1 Adapter Maintenance Control and Status Register (AMCSR)

The Adapter Maintenance Control and Status Register is used to control the operation of the adapter.

Bit zero is the Maintenance Initialize (MIN) bit. When set by the host, the MIN bit clears adapter hardware state (including status registers), invokes a firmware reset, and transitions the adapter to the uninitialized state.

**Note that the host must wait an adapter specific time interval (for reset to complete) before it accesses any additional adapter registers.**

Bit three is the Interrupt Enable (IE) bit. When set, the adapter interrupts the host for Command Complete interrupts and Adapter Miscellaneous interrupts. Command Complete interrupts may be subject to control of the Interrupt Holdoff Timer (section 3.6 ).

### 3.5.2 Adapter Block Base Register (ABBR)

The Adapter Block Base Register is used to initialize the adapter. The ABBR contains the upper most bits of the physical address of the Adapter Block (see section 5.2.2). The Adapter Block is a hexaword aligned structure, bits <4:0> of the physical address SBZ. Writing the ABBR is valid only when the adapter is in the uninitialized state. Figure 3-3 illustrates the format of the ABBR.

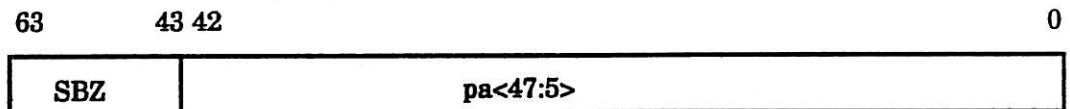


Figure 3-3 Adapter Block Base Register (ABBR)

### 3.5.3 Driver-Adapter Command Queue Insertion Register (DACQIR)

The Driver-Adapter Command Queue Insertion Register is used to notify the adapter of new entries on the DACQ. The value written to this register is the physical address of the new stopper carrier of the DACQ. Figure 3-4 illustrates the format of the DACQIR.

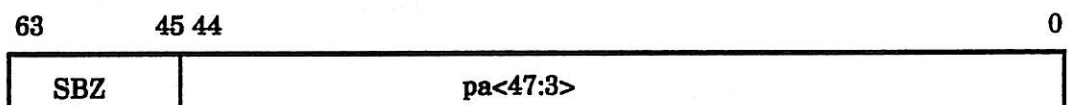


Figure 3-4 Driver-Adapter Command Queue Insertion Register (DACQIR)

### 3.5.4 Driver-Adapter Free Queue Insertion Register (DAFQIR)

The Driver-Adapter Free Queue Insertion Register is used to notify the adapter of new entries on the DAFQ. The value written to this register is the physical address of the new stopper carrier of the DAFQ. Figure 3-5 illustrates the format of the DAFQIR.

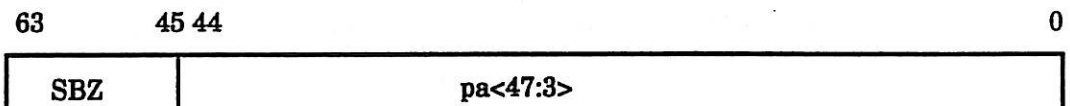


Figure 3-5 Driver-Adapter Free Queue Insertion Register (DAFQIR)



### 3.5.5 Adapter Status Register (ASR)

The Adapter Status Register is used to record adapter status information that requires host intervention. Figure 3-6 illustrates the format of the ASR.

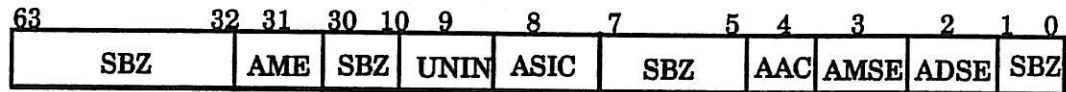


Figure 3-6 Adapter Status Register (ASR)

Table 3-4 lists the bit fields in the ASR and their meaning.

Table 3-4 ASR bit field definitions

Field	Bit	Description
SBZ	0,1	Should be zero.
ADSE	2	If bit set, a host data structure error has occurred.
AMSE	3	If bit set, a host memory system error has occurred while accessing a data structure.
AAC	4	If bit set, an abnormal condition exists in the adapter firmware.
SBZ	5-7	Should be zero.
ASIC	8	If bit set, the adapter is requesting a completion interrupt, as opposed to a miscellaneous interrupt.
UNIN	9	If bit set, the adapter is in the uninitialized state.
SBZ	10-30	Should be zero.
AME	31	If bit set, an adapter maintenance (hardware) error has occurred.
SBZ	32-63	Should be zero.

Except for the UNIN bit and the ASIC bit, all bits in the ASR indicate adapter fatal error conditions. When any error bits are set, the adapter issues an Adapter Miscellaneous interrupt and then enters the uninitialized state (although the UNIN bit is not set). The host must read the ASR to determine the cause of the error. To resume normal adapter functions, a complete adapter initialization must be performed (starting with an adapter reset command). For error log purposes, the host should also read and record the AFAR and the AFPR registers. These registers contain adapter specific error information. Set error bits in the ASR remain until an adapter reset command is issued by the host. No more than one of the bits that indicate an adapter fatal error shall be set at one time.

The UNIN bit indicates that the adapter is in the uninitialized state and that no error conditions exist. Essentially, a set UNIN bit indicates that the adapter is waiting for the Adapter Block address (via a write to the ABBR register). The UNIN bit is set only after power-up or after a reset command is issued to the adapter (see section 3.5.1). An interrupt to the host is not issued when this bit is set. When set, the UNIN bit can be the only bit set in the ASR. A set UNIN bit in the ASR remains until either a fatal adapter error occurs or a message is placed on the ADRQ.

The ASIC bit is intended to be used by hosts that do not support multiple interrupt vectors or operate with interrupts disabled. A completion interrupt to the host is issued

The ASIC bit in the ASR remains set until either a fatal adapter error or the adapter is reset. For these cases, error bits are written or the UNIN bit is written and the ASIC bit is cleared. The host should examine the response queue in these cases to ensure all completed commands are processed before the re-initialization of the adapter proceeds.

**The Adapter Failing Address Register is used to indicate the physical address of host memory associated with ASR errors. Some implementations may not be able to provide an exact failing address, only an address within  $N$  number of bytes of the failing address. Such implementations may choose to zero out the least significant bits to the extent that accuracy is insured and granularity is not compromised. This register is only cleared during an adapter reset.**

63

---

pa

### 3.5.7 Adapter Failing Parameter Register (AFPR)

63	8 7 0
implementation_specific	error_code

The `error_code` field of the AFPR gives additional details about the type of error reported in the ASR. Table 3-5 lists the codes for each of the ASR fatal adapter errors; these are mapped to SIMport error codes when the same problem is being reported.

ASR Bit	Value	Description
ADSE	4	Illegal Adapter Block format. A field offset, pointer value, or length value within the Adapter Block was found to be incorrect.
	8	Illegal Queue Buffer alignment. A Queue Buffer must reside entirely within one 64KB block and must be hexaword aligned.
	16	A Queue Carrier field contains invalid data.
	17	A BSM or BSD contains an invalid field or a field that is not valid in the current context.



	18	Buffer Segment Map link - the offset for current BSM does not equal offset + length of previous BSM.
AMSE		
	1	Error occurred while accessing an Adapter Block.
	2	Error occurred while accessing a Queue Carrier.
	3	Error occurred while accessing a Queue Buffer.
	4	Error occurred while accessing a BSM or BSD.
AAC		See adapter specification for explanation
AME		See adapter specification for explanation

---

## 3.6

### Interrupt Holdoff Timer

The interrupt holdoff timer is an optional part of the architecture that can be used to reduced the number of command complete interrupts when then interarrival time is small. The interrupt holdoff timer specifies the minimum time (in microseconds) between command complete interrupts. When this timer is set to a non-zero value the adapter starts a timer on the first completion interrupt. While the timer is active the adapter will not request any additional completion interrupts. When the timer expires the adapter requests a second completion interrupt only if one or more requests have become pending. The timer is restarted whenever the adapter requests a completion interrupt; the next channel interrupt request will again start the timer.



# Initialization and State Control

## 4.1

### Initialization Steps

The following sections detail the steps performed by the host SIM driver and the SIMport adapter during adapter initialization. These steps are required after system power-up, after an adapter hardware board level reset, after an adapter software SIMport command reset, and after any fatal error conditions reported by the adapter via the ASR register. Figure 4-1 illustrates the steps involved in a SIMport adapter initialization.

## 4.2

### SIM Driver Initialization Steps

A complete adapter initialization starts with a SIMport *Reset* command. The *Reset* command should ensure the adapter is in the uninitialized state. Some adapters are not capable of supporting a complete hardware reset via the defined SIMport *Reset* command and thus it may be necessary to perform an adapter specific reset. Refer to the appropriate adapter functional specification for details about adapter reset functions.

The following is a list of steps the host performs to initialize a SIMport adapter.

- 1 Typically, after power is applied to an adapter or a hardware board level reset is performed, the adapter performs internal diagnostics. During this time, the adapter registers may be in an indeterminate state and a SIMport *Reset* command may not be acknowledged and could even cause the adapter diagnostics to fail. You should review the adapter functional specification for when internal diagnostics are run and how long such diagnostics take. If the initialization is performed after such diagnostics are run, the SIM driver should wait an adapter specific time for the diagnostics to complete.
- 2 After any wait loop for adapter diagnostics, the host issues a SIMport *Reset* command by setting the MIN bit of the adapter AMCSR register.
- 3 The host waits a T.B.D. amount of time for the *Reset* command to complete and then reads the ASR register. The *UNIN* bit in the ASR register should be set, indicating that the adapter is in the uninitialized state and no errors have been detected. If the *UNIN* bit is not set, the host should examine the other bits in the ASR register for an error bit. If no other bits are set, a complete adapter specific hardware board level reset should be issued and the host should repeat step 2.
- 4 The host constructs the Adapter Block (AB) and performs any other SIM driver internal initialization. As part of the AB construction the host places stopper queue carriers on all queues defined in the AB and writes the address of the stopper queue carriers to the DACQIR and the DAFQIR. Next, the host writes the ABBR register with the physical address of the AB. At this point the IE bit in the AMCSR may be set if it is desired to use the adapter in interrupt mode. The first command to be placed on the Driver-Adapter Command Queue (DACQ) must be a SIMport *Set Adapter State* command requesting the adapter to transition to the disabled state. This command can be placed on the queue either before or after the physical address of the AB is written to the ABBR.

- 5 The host waits for a response to the *Set Adapter State* command, which is recognized by either polling the Adapter-Driver Response Queue (ADRQ) or receiving an interrupt. While waiting for the response the host should maintain a timeout timer with a T.B.D. time period. If the timer expires before a response message is received on the ADRQ, the host should read the ASR to determine the adapter error condition. When a message is on the ADRQ, the host should read the message, verify that it is an *Adapter State Set* response message, and interpret the configuration parameters within the message.
- 6 The host must place queue elements on the Driver-Adapter Free Queue (DAFQ). The queue elements are to be used by the adapter to generate unsolicited messages. The number of queue elements required by the adapter is indicated in the *Adapter State Set* response message. Minimally, one queue element must be placed on the DAFQ before the adapter transitions to the enabled state.
- 7 The host, optionally, issues a *Set Parameters* command based on configuration information contained in the *Adapter State Set* response message. If this step is performed, the host should again wait for a response by either polling the ADRQ or wait for an interrupt. After receiving the response to this command and the host verifies the command was executed by the adapter successfully.
- 8 Next, the host issues a second *Set Adapter State* command, requesting the adapter to transition to the enabled state. The host should wait for an interrupt or poll for a response message indicating that the transition to enabled state succeeded.
- 9 At this point the adapter is enabled and ready to accept commands to the channels. The first channel command the host issues is the *Set Channel State* command, instructing the channel to transition to the enabled state. The host should wait for an interrupt or poll for a response message indicating that the channel successfully transitioned to the enabled state.
- 10 The host should repeat step 9 for each channel.

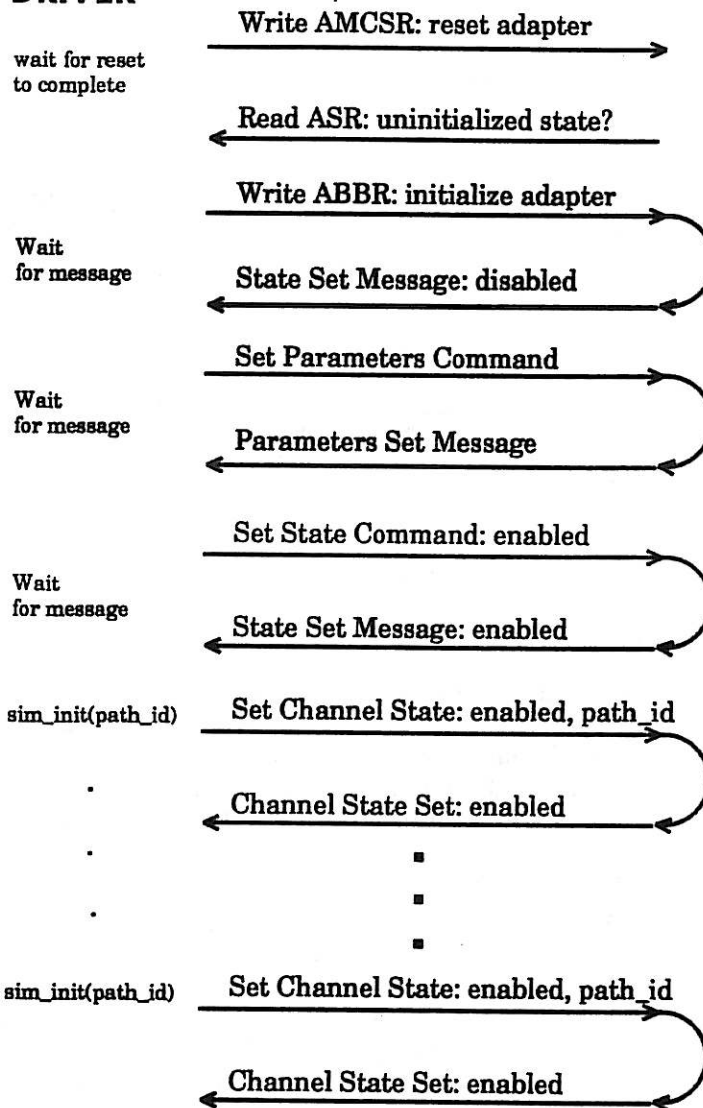
### 4.3

#### Adapter Initialization Steps

- 1 After internal diagnostics succeeds, the adapter waits for a host write to the AMCSR register. The adapter verifies that the *MIN* bit has been set. If this is the case, the adapter performs internal initialization, clears the ASR register, and sets the *UNIN* bit in the ASR register.
- 2 The adapter now waits for a write to the ABBR register. When this occurs, the adapter reads the AB and verifies its contents. If this succeeds, the adapter begins DACQ processing.
- 3 If any errors are detected by the adapter, the adapter clears the ASR, sets the appropriate error bit in the ASR, writes appropriate error information to the AFAR and AFPR, and waits for a reset.
- 4 The first command on the DACQ is the *Set Adapter State* command. The adapter reads this command, verifies its contents, and posts the appropriate response to the ADRQ. Any parameter errors in the command are communicated in the response message to the host, as opposed to the ASR register. A successful response to the this command indicates to the host that the adapter is now in the disabled state. Before placing the response on the ADRQ, the adapter clears the *UNIN* bit and sets the *ASIC* bit in the ASR.
- 5 The remainder of the adapter initialization steps are essentially a repeat of steps 3 and 4. That is, as initialization commands are presented to the adapter, the adapter interprets them and posts a response to the host. Command parameter errors are reported to the host in the response message to the host. Fatal adapter errors are reported via the ASR register and no response message posted.

## HOST DRIVER

## Adapter



**Figure 4-1 Adapter/Driver initialization information exchange**

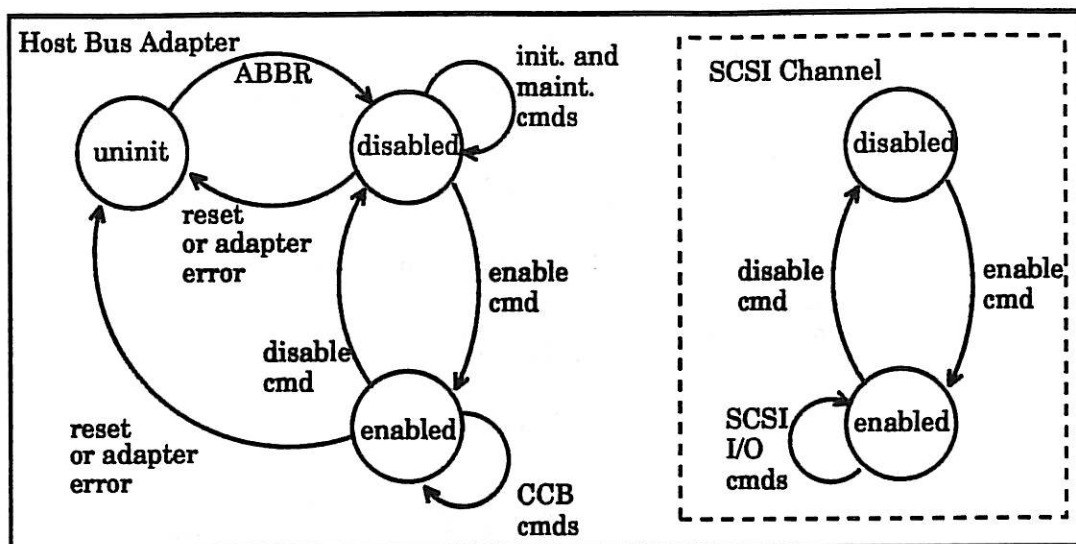


Figure 4-2 Adapter and Channel State Transitions

## 4.5

## Adapter States

The Adapter is either uninitialized, disabled, or enabled. On power-up and after a reset the adapter is in the uninitialized state. A register write of the Adapter Block Base Register (with the physical address of a valid Adapter Block and a *Set Adapter State: Disabled* command on the DACQ) will transition the adapter to the disabled state. A *Set Adapter State: Enabled* command will transition the adapter to the enabled state.

## 4.5.1 Uninitialized State

In the uninitialized state, the adapter does not access either the host bus or the SCSI bus. This state is entered on power-up or can be forced by a write to the MIN bit in the Adapter Maintenance Control and Status Register (i.e., a reset command). Adapter errors also cause the adapter to transition to the uninitialized state, in which case the adapter writes the Adapter Status Register and generates an Adapter Miscellaneous interrupt to the host. The UNIN bit in the ASR is set after a reset is complete to indicate that the adapter is in the uninitialized state.

## 4.5.2 Disabled State

In the disabled state, the adapter accesses the host bus via the queuing mechanism. In this state the adapter will monitor the command queue for initialization and maintenance commands and respond to these commands with response messages on the response queue. Commands which imply access to the SCSI channel will be rejected.

## 4.5.3 Enabled State

In the enabled state, the adapter allows access to both the host bus and the SCSI channels. In this state, the adapter will accept commands to both initialize the SCSI channels as well as accept commands to submit and control SCSI I/O to the channels.

## 4.6

## Channel States

A channel is either enabled or disabled. When disabled, a channel does not initiate or respond to selections. When enabled, a channel initiates and responds to selections as required by the host. All channels are implicitly disabled on power-up and when the

adapter is reset. All channels are implicitly disabled when the adapter is disabled. A channel is enabled via the *Set Channel State* command. A channel can be disabled also with the *Set Channel State* command. A bus reset will also transition a channel to the disabled state.

If a channel transitions to the disabled state due to an error condition, a *Channel Disabled* unsolicited message is sent to the host. The *Channel Disabled* message indicates the error condition. Such errors cause the channel to return all outstanding I/O with an error status (e.g., bus reset). While in the disabled state, a channel will continue to return new I/O with an error status. The host must re-enable the channel with a *Set Channel State* command. A *Channel State Set* response message is returned indicating that all I/O has been returned and any new I/O will be processed as normal (see appendix C). Note that under these conditions, the adapter is required to return the I/O in the order issued on a per LUN basis.

A channel can be explicitly disabled via the *Set Channel State* command. A request to disable a channel causes the channel to complete and return all outstanding I/O before the channel enters the disabled state. The *Channel State Set* response message is returned when the disabled state is entered. This response message follows all I/O response messages.

## 4.7

### Device States

A channel device is either enabled or disabled. All channel devices are implicitly enabled when the channel is enabled. All channel devices are implicitly disabled when a channel is disabled. A bus device reset causes a channel device to enter the disabled state.

A channel device enters the disabled state when a bus device reset is issued for the device. Similar to the channel error I/O cleanup (see section 4.6), the channel sends a *Device Disabled* unsolicited message to the host. Next, the channel returns all outstanding I/O with the bus device reset status. New I/O directed to the device is also returned with the error status. The SIM driver is required to issue a *Set Device State* command to re-enable the device. A *Device State Set* response message is returned, following all returned I/O (see appendix C).

A channel device can not be disabled with a *Set Device State* command.



# Command and Response Processing

This chapter defines the format of commands and command response messages. It is divided into Adapter Specific commands and responses, maintenance commands and responses, and CAM defined commands and responses.

**Note that all CAM Control Block (CCB) related offsets in this specification are for CCBs with 32-bit pointers. For CCBs with 64-bit pointers, the offsets must be adjusted accordingly. See appendix D.1 for details.**

## 5.1

### General Operating Characteristics

#### 5.1.1 Command Response Protocol

Except for the reset and initialize commands, all commands to the adapter are enqueued to the tail of the Driver-Adapter Command Queue (DACQ). The adapter dequeues commands from the head of the DACQ. All command response messages to the host are enqueued to the tail of the Adapter-Driver Response Queue (ADRQ). The host dequeues messages from the head of the ADRQ. See the appendix G for details about the queuing protocol used for these queues.

Not all commands to the adapter have command responses. For commands that do have command responses, the same CCB that is used for the command is used for the command response message. Typically, the status field is the only modification to the CCB for the command response messages. For commands that do NOT have a response, the queue element is returned to the host over the Adapter-Driver Free Queue (ADFQ).

Most commands are initiated by the host and executed by the adapter, resulting in the adapter posting a response message to the host. The *BSD Response/BSD Request* and the *Bus Reset Response/Bus Reset Request* commands/messages are exceptions. The *BSD Request* and *Bus Reset Request* messages are initiated by the adapter (over the ADRQ). The host is required to act on these request messages and issue the corresponding *BSD Response* or *Bus Reset Response* commands (over the DACQ). Note for the *BSD Response* and *Bus Reset Response* commands, the adapter does not return the queue element to the host. Since the queue element originated from the Driver-Adapter Free Queue (DAFQ), the adapter retains the queue element.

#### 5.1.2 Command Priority

With the exception of the *Execute SCSI I/O* commands, all commands to the adapter are executed in the order submitted. The *Execute SCSI I/O* commands are first placed on the appropriate SIM Queues, at either the tail or head, depending upon the value of the insertion flag (see section 3.3.1.1). When internal adapter resources are available and the SCSI bus is free, a round-robin scheduling algorithm is used to dispatch commands from the SIM Queues onto the SCSI bus.

#### 5.1.3 Unrecognized Commands

Unrecognized commands are CCB commands that are either not supported or have invalid fields such that the command is unrecognizable. The adapter returns all such com-



mands on the response queue (ADRQ) with a status indicating the error (e.g., Invalid Request, Invalid Path ID, etc.). Note that this is true for commands which normally do not have a response and which normally the queue element is returned on the free queue (ADFQ).

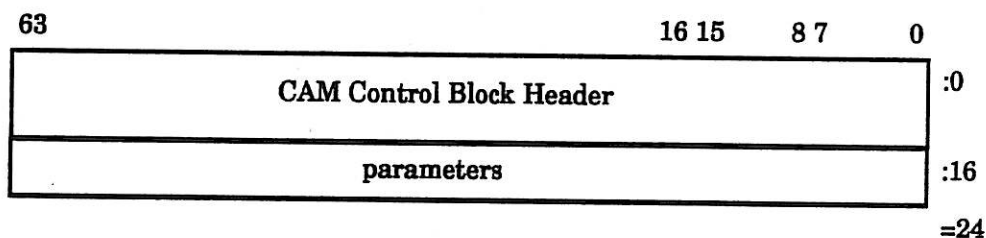
## Adapter Specific Commands and Responses

This section defines the format used to pass Adapter Specific commands to the adapter and the corresponding format the adapter uses to report command response messages. The CAM function codes used for Adapter Specific commands and response messages start at 80 hex. Table 5-1 lists the Adapter Specific commands and the required adapter response messages. Table A-2 lists the values of the SIMport function codes.

**Table 5-1 Adapter Specific Commands and Response Messages**

Command	Response Message
Reset Adapter	N.A.
Initialize Adapter	Adapter State Set
Set Adapter State	Adapter State Set
Set Parameters	Parameters Set
Set Channel State	Channel State Set
Set Device State	Device State Set
Verify Adapter Sanity	Adapter Sanity Verified
Read Counters	Counters Read
BSD Response	BSD Request
Bus Reset Response	Bus Reset Request

Figure 5-1 illustrates the general SIMport CCB format used for Adapter Specific commands and response messages.



**Figure 5-1 Adapter Specific Command/Response Format**

The CAM Control Block Header is as specified in the CAM XPT/SIM specification. SIMport function codes are a subset of CAM function codes (see Appendix A). The parameter fields are specific to each command and response.

The following sections detail the format for each command, as well as the format of any required response message from the adapter. All numeric values for command and response fields are in hex.

Note that all commands and response messages have the standard CCB header, although the header is not shown.

### 5.2.1 Reset Adapter Command

The SIM driver sets the MIN bit of the AMCSR to reset the adapter. This command resets the adapter and disables all SCSI channels. There is no response to this command. The adapter enters the uninitialized state and all the channels enter the disabled state.

**Note that the host must wait an adapter specific time interval after a reset command before any additional adapter registers (including the ASR) are accessed.**

After waiting the adapter specific time interval, the host should read the ASR register to ensure that the adapter is in the uninitialized state. The UNIN bit should be the only bit set in the ASR after a reset command is complete.

### 5.2.2 Initialize Adapter Command

To initialize the adapter, the host should construct the Adapter Block, construct a *Set Adapter State* command and place it on the Driver-Adapter Command Queue, and then write the physical address of the Adapter Block to the Adapter Block Base Register (ABBR). The *Set Adapter State* command should request the adapter to transition to the disabled state. The SIM driver must first ensure that reset is complete before it writes the ABBR (see the Reset Adapter command).

The required response to the Initialize command is the response to the *Set Adapter State* command, an *Adapter State Set* response message. The response message should indicate that the adapter is in the disabled state. If the adapter cannot transition to the disabled state, it will interrupt the host with an Adapter Miscellaneous Interrupt, after the ASR is updated. The SIM driver should read the ASR to determine the cause of the initialization failure. See appendix F for Driver-Adapter polling mode alternatives to interrupts.

### 5.2.3 Set Adapter State Command

The *Set Adapter State* command is used to set the state of the adapter. Figure 5-2 illustrates the SIMport function CCB format.

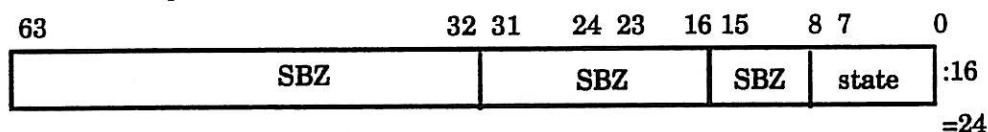


Figure 5-2 Set Adapter State Command

Table 5-2 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

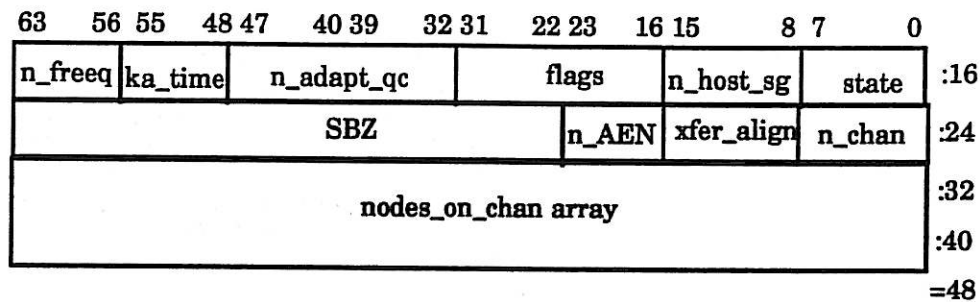
Table 5-2 Set Adapter State Command Fields

Field	Size	Value	Description
state	<7:0>	1, 2	The requested state- 1: enabled, 2: disabled.

A *Set Adapter State* command indicating 'disabled,' implicitly disables all channels. The required response to this command is an *Adapter State Set* response message.

## 5.2.4 Adapter State Set Response Message

The *Adapter State Set* response message is the required response to a *Set Adapter State* command. Figure 5-3 illustrates the SIMport function CCB format.



**Figure 5-3 Adapter State Set Response Message**

Table 5-3 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 5-3 Adapter State Set Response Message Fields**

Field	Size	Value	Description
state	<7:0>	1, 2	The current state- 1: enabled, 2: disabled.
The following fields are only valid when the adapter state transitions from UNINitialized to DISABLED			
n_host_sg	<15:8>	n	The number of host 4K segments of host memory to allocate for exclusive adapter use. If non-zero, requires host to issue <i>Set Parameters</i> command.
flags	<31:16>	see table 5-5	Supported features flags.
n_adapt_qc	<47:32>	n	The maximum number of queued commands the adapter can maintain without overflow to host memory.
ka_time	<55:48>	n	The minimum time in seconds the host must allow the adapter for <i>Verify Adapter Sanity</i> command execution.
n_freeq	<63:56>	n	The minimum number of free queue entries the host should allocate to the adapter for use by the adapter for the generation of unsolicited messages. The host shall place at least one queue element on the DAFQ to enable the adapter.
n_chan	<7:0>	1-16	The number of SCSI channels in the adapter.
xfer_align	<15:8>	see table 5-6	This byte informs the host of the transfer alignment capabilities of the adapter.
n_AEN	<23:16>	n	Recommended number of AEN buffers that the host should queue to the adapter. See section 5.4.10.
nodes_on_chan	<8 bits>	1-16	An array of bytes containing the maximum number of nodes that may be on each channel.

Table 5-4 specifies the values for the *status* field in the Queue Carrier.

**Table 5-4 Adapter State Set Response Message Status**

Status	Value	Description
success	1	State set as requested.
inv_command	0	Either the command is not supported or the command format is incorrect.
host_mem	-3	Before the adapter will transition to the enable stated, the SIM driver must allocate, for exclusive adapter use, <i>n_host_sg</i> number of 4K host memory segments and issue a <i>Set Parameter</i> command to notify the adapter of the physical addresses of the allocated host memory.
host_fqe	-4	Before the adapter will transition to the enabled state, the SIM driver must allocate, for exclusive adapter use, at least one queue element and place it on the DAFQ.

Table 5-5 specifies the values for the *flags* field.

**Table 5-5 Adapter State Set Response Message Flags**

Field	Bit	Description
Int. Holdoff	0	If bit set, the interrupt holdoff timer is supported.
Channel Node	1	If bit set, the adapter supports soft setting of the SCSI bus node identifier via the <i>Set Channel State</i> command.
Linked BSMS	2	If bit set, the adapter supports linked BSMS.
BSD Request	3	If bit set, the adapter issues <i>BSD Request</i> unsolicited messages.
Target Mode	<5:4>	One bit for each of the two possible target modes supported by CAM. If bit 4 is set the adapter supports Processor mode. If bit 5 is set the adapter supports Phase Cognizant mode.

The *Int. Holdoff* message flag, when set, indicates that the adapter supports a interrupt holdoff timer. The host can enable and set this timer via the *Set Parameter* command. See section 3.6 for additional details on the interrupt holdoff timer.

Table 5-6 specifies the values for the *xfer\_align* field.

**Table 5-6 Adapter State Set Response Message Xfer\_Align Field**

Value	Description
0	The adapter is capable of byte aligned data transfers or any multiple thereof.
1	The adapter is capable of doublet (word) aligned data transfers or any multiple thereof.
2	The adapter is capable of quadlet (longword) aligned data transfers or any multiple thereof.
3	The adapter is capable of octalet (quadword) aligned data transfers or any multiple thereof.
4	The adapter is capable of octaword aligned data transfers or any multiple thereof.

Note that the *xfer\_align* field indicates the number of low-ordered bits which are masked or ignored by the adapter in host physical addressing information. For example, a value of 2 in the *xfer\_align* field indicates that bits 0 and 1 of host physical addressing information should be zero. See appendix B for information about unaligned host memory buffers.

### 5.2.5 Set Parameters Command

The *Set Parameters* command is used to set various parameters of the adapter. The adapter indicates the supported parameters in the *Adapter State Set* response message (issued after a write to the ABBR). All adapter parameter settings which imply adapter functions default to disabled, implying the parameter must be set to enable the function.

Figure 5-4 illustrates the SIMport function CCB format.

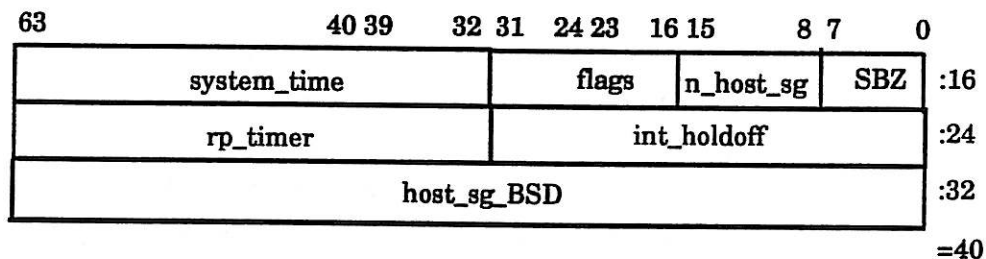


Figure 5-4 Set Parameters Command

Table 5-7 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

Table 5-7 Set Parameters Command Fields

Field	Size	Value	Description
n_host_sg	<15:8>	n	The actual number of 4K segments of host memory allocated for exclusive adapter use.
flags	<31:16>	see table 5-8	Flags controlling adapter operation.
system_time	<63:32>	n	System time in seconds since 0 hours, 0 minutes, 0 seconds, January 1, 1970 Coordinated Universal Time. This is the format defined by the IEEE Standard Portable Operating System Interface for Computer Environments.
int_holdoff	<31:0>	n	Completion interrupt holdoff timer control. The Interrupt Holdoff Time is the minimum time the adapter must wait between issuing Command Complete interrupts. The value is expressed in microseconds.
rp_timer	<63:32>	n	The Bus Reset Request timer is the minimum time the adapter must wait for a <i>Bus Reset Response</i> command from the host. If this timer expires, the adapter proceeds with the bus reset. If not set by the host, the adapter will wait indefinitely for the <i>Bus Reset Response</i> command. The timer period is expressed in milliseconds.
host_sg_BSD	<63:0>	see table 10-2	A Buffer Segment Descriptor (BSD) defining the size and location of host memory allocated for exclusive adapter use. Required only if the n_host_sg field in the <i>Adapter State Set</i> response message is non-zero.

The *host\_sg\_BSD* field is a BSD that may point directly to the requested host memory or to a BSM defining the location(s) of the requested host memory (see chapter 10).

**Table 5-8 Set Parameters Command Flags Fields**

Field	Bit	Description
Enable_Counters	0	When set the counters that can be read with the Read Counters command are maintained.

The required response to this command is a *Parameters Set* response message.

### 5.2.6 Parameters Set Response Message

The *Parameters Set* response message is the required response to a *Set Parameter* command. There are no fields defined in the CCB by SIMport.

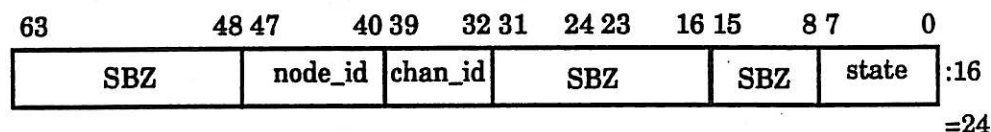
Table 5-9 specifies the values for the status field in the Queue Carrier.

**Table 5-9 Parameters Set Response Message Status**

Status	Value	Description
success	1	Parameter set as requested.
inv_command	0	Either the command is not supported or the command format is incorrect.
not_disabled	-2	Adapter must be in the disable state.
inv_int_holdoff	-5	Interrupt Holdoff timer invalid or not supported.
inv_host_sg	-6	Insufficient host memory allocated.

### 5.2.7 Set Channel State Command

The *Set Channel State* command is used to set the state of the channel. This command is used to enable a channel during initialization or after a channel error. Figure 5-5 illustrates the SIMport function CCB format.



**Figure 5-5 Set Channel State Command**

Table 5-10 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

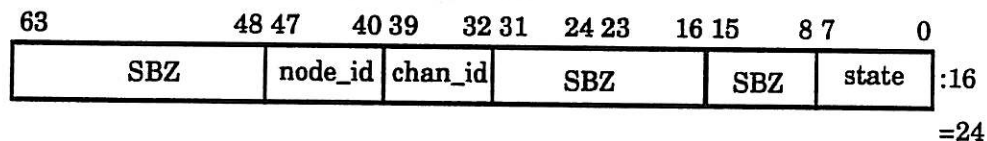
**Table 5-10 Set Channel State Command Fields**

Field	Size	Value	Description
state	<7:0>	1, 2	The requested state- 1: enabled, 2: disabled.
chan_id	<39:32>	0-(n_chan-1)	The internal adapter identifier of the channel to set.
node_id	<47:40>	0-(n_node-1)	The SCSI bus node identifier for the channel. Only meaningful if soft setting of the node id supported by the adapter. A value of FF <sub>16</sub> does NOT set the id, but also does not return error, allowing the current id to be read.

The required response to this command is a *Channel State Set* response message.

### 5.2.8 Channel State Set Response Message

The *Channel State Set* response message is the required response to a *Set Channel State* command. Figure 5-6 illustrates the SIMport function CCB format.



**Figure 5-6 Channel State Set Response Message**

Table 5-11 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 5-11 Channel State Set Response Message Fields**

Field	Size	Value	Description
state	<7:0>	1, 2	The current state- 1: enabled, 2: disabled.
chan_id	<39:32>	0-(n_chan-1)	The internal adapter identifier for the channel.
node_id	<47:40>	0-(n_node-1)	The assigned SCSI bus node identifier for the channel. Always filled in when the status is success, whether or not the node identifier was set.

Table 5-12 specifies the values for the status field in the Queue Carrier.

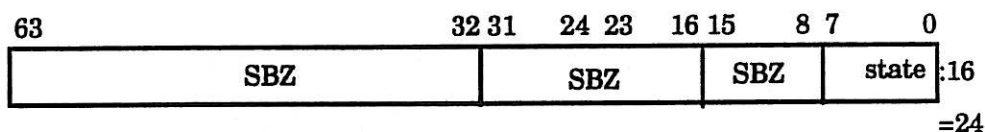


**Table 5-12 Channel State Set Response Message Status**

Status	Value	Description
success	1	State set as requested.
inv_command	0	Either the command is not supported or the command format is incorrect.
not_enabled	-1	The adapter must be enabled before a channel is enabled.
inv_chan_id	-7	Invalid channel identifier or channel does not exist.
inv_node_id	-8	Invalid SCSI bus node identifier or soft setting not supported.

### 5.2.9 Set Device State Command

The *Set Device State* command is used to set the state of a SCSI Channel device. This command is used to re-enable a device after a bus device reset. Figure 5-7 illustrates the SIMport function CCB format.



**Figure 5-7 Set Device State Command**

Table 5-13 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

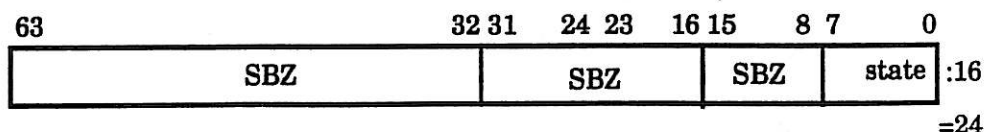
**Table 5-13 Set Device State Command Fields**

Field	Size	Value	Description
state	<7:0>	1	The requested state- 1: enabled

The required response to this command is a *Device State Set* response message.

### 5.2.10 Device State Set Response Message

The *Device State Set* response message is the required response to a *Set Device State* command. Figure 5-8 illustrates the SIMport function CCB format.





## Figure 5-8 Device State Set Response Message

Table 5-14 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

Table 5-14 Device State Set Response Message Fields

Field	Size	Value	Description
state	<7:0>	1, 2	The current state- 1: enabled, 2: disabled.

Table 5-15 specifies the values for the status field in the Queue Carrier.

Table 5-15 Device State Set Response Message Status

Status	Value	Description
success	1	State set as requested.
inv_command	0	Either the command is not supported or the command format is incorrect.
not_enabled	-1	The adapter and/or channel must be enabled before a device is enabled.
cant_disable	-9	A device channel cannot be disabled by host command.

### 5.2.11 Verify Adapter Sanity Command

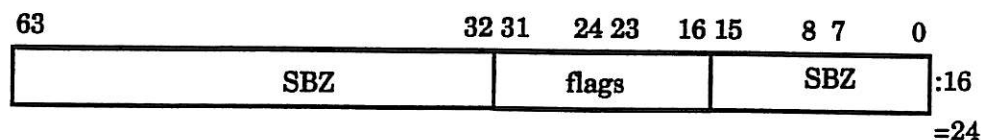
The *Verify Adapter Sanity* command allows the SIM driver to verify the operation of the adapter. The SIMport function code for *Verify Adapter Sanity* is 84 hex. The required response to this command is a *Adapter Sanity Verified* response message. The minimum timer value is specified by the *Adapter State Set* message during initialization, see section 5.2.4.

### 5.2.12 Adapter Sanity Verified Response Message

The *Adapter Sanity Verified* response message is the required response to an *Verify Adapter Sanity* command. It has the same format as the command. Since this command can never fail, there are no status values defined for it. If it is on the response queue, then the command succeeded.

### 5.2.13 Read Counters Command

The *Read Counters* command is used to read maintenance and performance counters from the adapter. The counter values are returned in the *Counters Read* response message. Figure 5-9 illustrates the SIMport function CCB format.



## Figure 5-9 Read Counters Command

Table 5-16 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

Table 5-16 Read Counters Fields

Field	Size	Value	Description
flags	<31:16>	see table 5-17	Flags that control subfunctions of the Read Counters command

The required response to this command is a *Counters Read* response message.

Table 5-17 Read Counters Flags

Field	Bit	Value	Description
Zero_Counters	0	1	The counters are zeroed after being read into the response message.

### 5.2.14 Counters Read Response Message

The *Counters Read* response message is the required response to the *Read Counters* command. The *Counters Read* response message returns all required counters and any implementation specific counters defined by the adapter. Figure 5-10 illustrates the SIMport function CCB format.



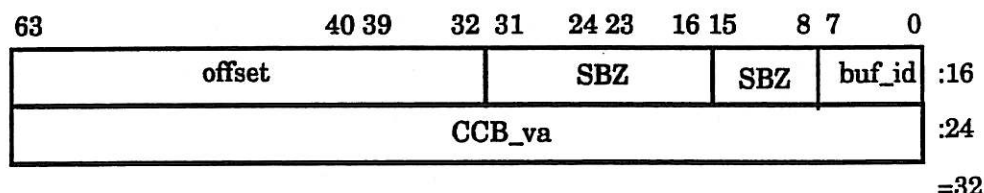
These counters are 32 bits in length and operated upon as unsigned integers. The value FFFFFFFF<sub>16</sub> is not valid and indicates an error or overflow condition. Unused space in the required counters portion will be filled with FFFFFFFF<sub>16</sub> so that the values can be recognized as invalid if additional counters are added for later implementations. The list of required counters for the *Counters Read* response message are listed in table 5-20.

**Table 5-20 Required Counters Names and Offsets**

Counter Name	Offset	Units
Time (since the counters were last zeroed)	0 (20)	Milliseconds
Number of host bus faults	4 (24)	
Number of SCSI Commands sent	8 (28)	Commands
Number of SCSI Commands received (target mode)	12 (32)	Commands
Total DATA PHASE bytes sent	16 (36)	Bytes
Total DATA PHASE bytes received	20 (40)	Bytes
Number of SCSI bus resets	24 (44)	
Number of BUS DEVICE RESETS sent	28 (48)	
Number of selection/reselection timeouts	32 (52)	
Number of parity errors detected	36 (56)	
Number of unsolicited reselections	40 (60)	
Number of unrecognized, unexpected, or unsupported messages received	44 (64)	Messages
Number of MESSAGE REJECT messages received	48 (68)	Messages
Number of unexpected disconnects	52 (72)	
Number of unexpected phase changes (phase mismatches)	56 (76)	
Number of synchronous data transfer period violations (Handshake-to-handshake timeout)	60 (80)	
unused (available) space	64-83 (84-103)	

### 5.2.15 BSD Request Message

The *BSD Request* message is a request from the adapter to the host to provide additional Buffer Segment Descriptors (BSDs) for a previously issued *Execute SCSI I/O*. The request specifies a buffer offset from the start of the buffer. The BSDs are returned to the adapter in a *BSD Response* command (see section 5.2.16) and should define the buffer segments starting from the specified buffer offset field. Figure 5-11 illustrates the SIMport function CCB format.



**Figure 5-11 BSD Request Message**

Table 5-21 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

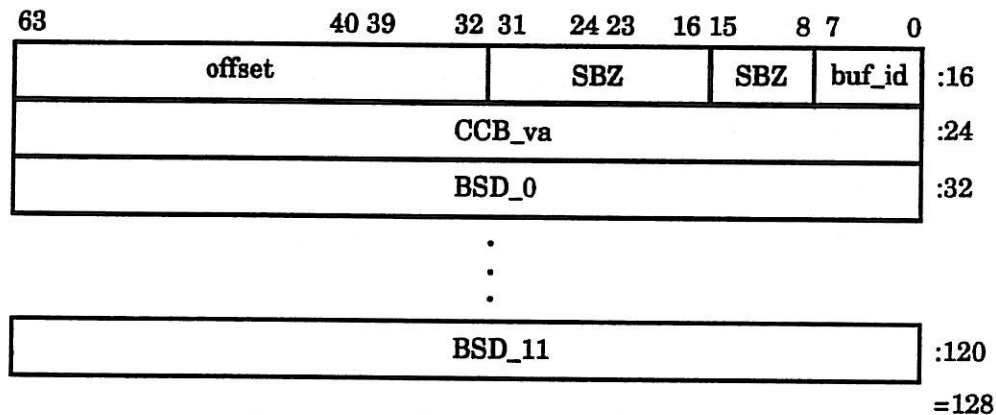
**Table 5-21 BSD Request Message Fields**

Field	Size	Value	Description
buf_id	<7:0>	see table 5-24	The buffer field within the private data area of the CCB to which this command is associated.
offset	<63:32>	n	The byte offset from the start of the data buffer. The data buffer is defined by the CCB with address <i>CCB_va</i> .
CCB_va	<63:0>	va	Host virtual address of the Execute SCSI I/O CCB.

The host is required to response to this message with a *BSD Response* command.

### 5.2.16 BSD Response Command

The *BSD Response* command is really the host response to a *BSD Request* message (see section 5.2.15). This command informs the adapter of new Buffer Segment Descriptors (BSDs) for a previously issued *Execute SCSI I/O CCB* command. Figure 5-12 illustrates the SIMport function CCB format.



**Figure 5-12 BSD Response Command**

Table 5-22 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 5-22 BSD Response Command Fields**

Field	Size	Value	Description
buf_id	<7:0>	see table 5-24	The buffer field within the private data area of the CCB to which this command is associated.
offset	<63:32>	n	The byte offset from the start of the data buffer. The data buffer is defined by the CCB with address <i>CCB_va</i> .
CCB_va	<63:0>	va	Host virtual address of the Execute SCSI I/O CCB.
BSD_n	<63:0>	see table 10-2	Buffer Segment Descriptor (see section 10.1)

Table 5-23 specifies the values for the *status* field in the Queue Carrier.

**Table 5-23 BSD Response Command Status**

Status	Value	Description
success	1	BSDs set as requested.
inv_command	0	Either the command is not supported or the command format is incorrect.
inv_offset	-10	Invalid offset value or offset not within buffer.
inv_CCB	-11	Invalid CCB address.
inv_buf_id	-12	Invalid buf_id field value.

Table 5-24 specifies the values for the *buf\_id* field.

**Table 5-24 BSD Response Command Buf\_id**

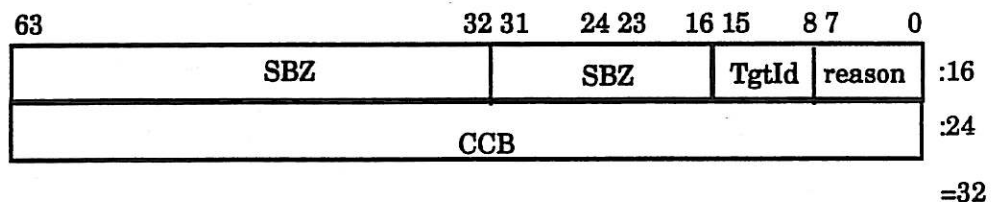
Field	Value	Description
data	1	BSDs are for the data buffer
CDB	2	BSDs are for the CDB buffer
sense	3	BSDs are for the Sense buffer
message	4	BSDs are for the Message buffer

See section 10.1 for the format of a BSD.

This command is a response to the *BSD Request* message from the adapter and thus does not have a response back to the host. The queue element is retained by the adapter to replace the free queue element used to issue the *BSD Request* message.

### 5.2.17 Bus Reset Request Message

The *Bus Reset Request* message is a request from the adapter to the host to reset the SCSI bus. The message indicates the reason for the requested bus reset. Figure 5-13 illustrates the SIMport function CCB format.



## Figure 5-13 Bus Reset Request Message

Table 5-25 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

Table 5-25 Bus Reset Request Message Fields

Field	Size	Value	Description
reason	<7:0>	see table 5-26	Reason code for reset request.
TgtId	<15:8>	0-15	SCSI id of target, or FF <sub>16</sub> if the target is not known.
CCB	<63:0>	CCB va	The virtual address of the CCB active on the bus. If zero then no CCB is associated with the bus problem

Table 5-26 specifies the values for the *reason* field.

Table 5-26 Bus Reset Request Reason Field

Name	Value	Description
Reject_BDR	1	Target rejected a Bus Device Reset attempt.
Phase_Error	2	Target does not respond to attention after n unexpected phase changes.
Data_Out	3	Unexpected phase change to DATA-OUT phase.
Rsvd_Phase	4	target entered a reserved phase during abort attempt
No_Mag_Out	5	target did not enter MESSAGE OUT upon selection during an attempt to reconnect for sending either an ABORT or BDR

The required host response to this message is a *Bus Reset Response* command. Note that the adapter enters a 'reset pending' state until the host issues the *Bus Reset Response* command. In this state, additional *Execute SCSI I/O* commands are placed on the appropriate SIM queues, but no additional SCSI I/O is started.

Note that an optional reset pending timer can be supported by the adapter for high availability systems. See the *Set Parameters* command for details.

### 5.2.18 Bus Reset Response Command

The *Bus Reset Response* command is really the host response to a *Bus Reset Request* message (see section 5.2.17). SIMport does not define any fields in the CCB. Note that the action authorized by the host is presented in the status field of the Queue Carrier and not in the response message.

Table 5-27 specifies the values for the *status* field in the Queue Carrier.

**Table 5-27 Bus Reset Response Command Status**

Status	Value	Description
success	1	Adapter should reset the SCSI bus, assuming the reported <i>reason</i> is still valid.
inv_command	0	Either the command is not supported or the command format is incorrect.
no_reset	-13	Adapter should not reset the SCSI bus. Note that the adapter, assuming the reported <i>reason</i> is still valid, will transition to the <i>Channel Disabled</i> state. In this case, normal channel disable procedures (including the return of all outstanding I/O) is initiated.

This command is a response to the *Bus Reset Request* message from the adapter and thus does not have a response back to the host. The queue element is retained by the adapter to replace the free queue element used to issue the *Bus Reset Request* message.

When the adapter receives the *Bus Reset Response* command and the command indicates to proceed with the reset, the adapter should first verify that the *reason* is still valid and then proceed with the bus reset. If the *reason* is no longer valid, the adapter should not perform the bus reset and continue normal command processing. If the command indicates to NOT proceed with the reset, the adapter should transition the affected channel to the disabled state (without a bus reset) and proceed to return all outstanding I/O (for that channel) to the host.

## 5.3 Maintenance Commands and Responses

Maintenance commands are allowed while the adapter is in the disabled state. The same command and response mechanism used for initialization is used for maintenance commands. The definition of such commands is beyond the scope of this document.

## 5.4 CAM defined Commands and Responses

CAM defined commands are as defined by the CAM XPT/SIM specification. Table 5-28 lists the commands supported by SIMport. See appendix A for the opcode value for each CAM supported command. See Appendix D for the CAM defined command formats and SIMport required fields.

**Table 5-28 CAM defined Commands and Response Messages**

Command	Response Message
Execute SCSI I/O	SCSI I/O Executed
Execute Target I/O	Target I/O Executed
Release SIM Queue	N.A.
Abort SCSI I/O Command	N.A.
Reset SCSI Bus	N.A.
Reset SCSI Device	N.A.
Terminate SCSI I/O Process	N.A.
Set ASYNC Callback	AEN
Path Inquiry	Path Inquiry
Enable LUN	T.B.D.



The following sections detail each CAM defined command, as well as any require response from the adapter.

### 5.4.1 Execute SCSI I/O Command

The *Execute SCSI I/O* command is used to initiate SCSI I/O on an adapter SCSI channel. Refer to the CAM XPT/SIM specifications for a complete definition of this command. Refer to table E-1 for information about CCB field values for this command.

SIMport is designed to allow the adapter to be host independent and thus provides the adapter with the physical addresses of all buffers. The private data area (see figure 5-14) of a SCSI I/O CCB is used to pass the required physical address information to the adapter. In addition, the private data area is used to pass status and residual count information back to the host (see section 5.4.2).

The following is assumed about the host resident buffers:

- If the CDB is a pointer, it can cross a page boundary. Furthermore, the CDB is limited in length by the size of the 'CDB Length' field. Since this field is one byte, then the CDB is at most 256 bytes and thus can cross at most one page boundary.
- The sense buffer can cross a page boundary. Furthermore, the sense buffer is limited in length by the size of the 'Sense Buffer Length' field. Since this field is one byte, then the sense buffer is at most 256 bytes and thus can cross at most one page boundary.
- The message buffer can cross a page boundary. Furthermore, the message buffer is limited in length by the size of the 'Message Buffer Length' field. Since this field is two bytes, then the message buffer is at most 64K bytes and thus can cross many page boundaries.

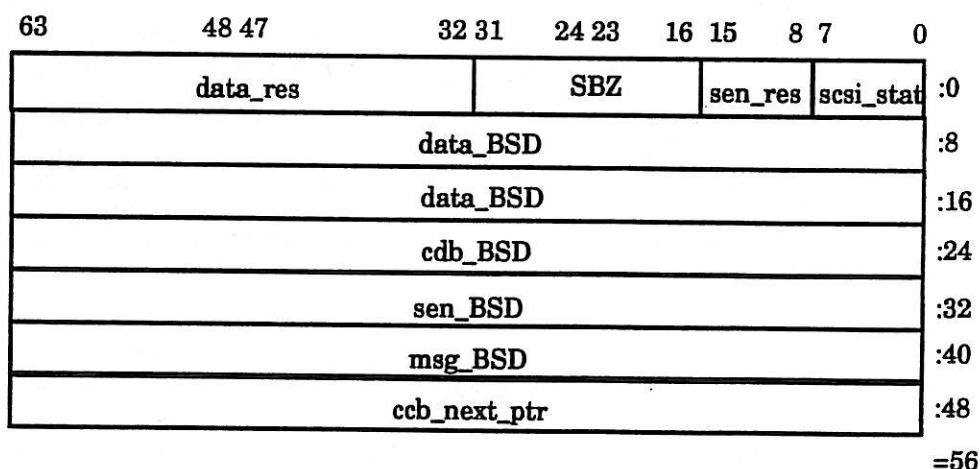


Figure 5-14 Execute SCSI I/O CCB Private Data

Note that the fields *sen\_res*, *scsi\_stat*, and *data\_res* are actually output values for the *SCSI I/O Executed* response message. These values must be initialized by the host to success values (as defined by CAM). This is a performance optimization allowing the adapter to not DMA to these fields for successfully completed *Execute SCSI I/O* commands.

Table 5-29 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 5-29 Execute SCSI I/O CCB Private Data Fields**

Field	Size	Value	Description
<code>scsi_stat</code>	<7:0>	n	SCSI Status code as defined by the SCSI-2 spec.
<code>sen_res</code>	<15:8>	n	AutoSense buffer residual length as defined by the CAM XPT/SIM spec.
<code>data_res</code>	<63:32>	n	Data buffer residual length as defined by the CAM XPT/SIM spec.
<code>data_BSD</code>	<63:0>	see table 10-2	First Buffer Segment Descriptor for the SCSI data buffer.
<code>data_BSD</code>	<63:0>	see table 10-2	Second Buffer Segment Descriptor for the SCSI data buffer.
<code>cdb_BSD</code>	<63:0>	see table 10-2	Buffer Segment Descriptor for the SCSI CDB.
<code>sen_BSD</code>	<63:0>	see table 10-2	Buffer Segment Descriptor for the SCSI sense buffer.
<code>msg_BSD</code>	<63:0>	see table 10-2	Buffer Segment Descriptor for the SCSI message buffer. This field is only used for Target Mode operations.
<code>ccb_next_ptr</code>	<63:0>	pa	Physical address of next CCB, if linked command.

See table E-2 for a description of the SIMport flags used with this command.

The Buffer Segment Descriptor (see section 10.1) fields define the host physical addresses of the indicated buffers. Note that two descriptors are allotted for the SCSI data buffer. This is a performance optimization. It is expected that the average SCSI data buffer will consist of at most two physically contiguous buffer segments. See chapter 10 for information about buffer memory mapping, including a definition of a Buffer Segment Descriptor.

If a Buffer Segment Map (BSM) is needed to define the host data buffer, then only the first `data_BSD` may contain valid data. The second `data_BSD` field **shall be zero** in this case. The `type` field of the first `data_BSD` field must be set to 1, indicating that the BSD defines a BSM. If multiple BSMs are required, the host should use the `next_bsm_BSD` field in the first BSM to form a linked list of BSMs. See section 10.2 for possible alternatives.

If a BSM is not required to define the host data buffer, then both `data_BSDs` may be used to define the host data buffer. The BSD `type` fields, in this case, must be set to zero to indicate that the BSDs define host buffer segments. The first `data_BSD` is used to define the start of the host data buffer. The second `data_BSD` is used to define the remainder of the host data buffer. If only one `data_BSD` is used, the second `data_BSD` **shall be zero**.

Note that due to the segment `byte_count` field size (within a BSD), host memory segments have a maximum size of 64K bytes; thus, for large physically contiguous host buffers, a Buffer Segment Map may be required even though the host buffer is physically contiguous.

The remaining BSD fields (`cdb_BSD`, `sen_BSD`, and `msg_BSD`) **should be zero**, if they are not used. The `ccb_next_ptr` field also **should be zero**, if it is not used.

The required response to this command is an *SCSI I/O Executed* response message.

### 5.4.1.1 Command Timeout and Command Abort

The timeout period is specified in the CCB by the peripheral driver and is used to detect an errant target device. Thus, the adapter is expected to start the timer when the command is sent to the target and not when it is received by the adapter. Command timeout does not apply when operating in target mode.

When a command timer expires, the adapter should attempt to abort the command. If the abort attempt is rejected by the target device, the adapter should 'upgrade' the abort to a bus device reset (BDR) message. If the BDR attempt fails, the adapter should send a *Bus Reset Request* message to the host over the ADRQ and wait in a 'reset pending' state until the host responds with a *Bus Reset Response* command. The *Bus Reset Response* command should indicate whether or not the adapter should proceed with a bus reset.

### 5.4.1.2 Queued Operation Tag Assignment

Queued operation tag assignment only applies to *Execute SCSI I/O* commands which have the Queue Action Enable bit set in the CAM flags field of the CCB. The SIM layer of the adapter is responsible for queue tag allocation and reconnection of the I\_T\_L\_Q nexus (see SCSI-2 X3.131-1991). For an intelligent adapter tag assignment and management is done internal to the adapter, for a simple adapter the host port driver may be required to manage the tags.

### 5.4.1.3 Autosense

If autosense is not disabled (CAM flags, byte 1, bit 5), then the SIM driver must ensure that there is a valid autosense buffer described in the *sen\_BSD* field of the private data area of *Execute SCSI I/O*. Also, the *sen\_res* field must be initialized to zero by the host. The adapter takes care of issuing the Request Sense command to the target/LUN when a Check Condition status is returned.

If autosense is disabled and Check Condition status is returned, then the adapter checks the SIM Queue Freeze Disable bit to see if the SIM Queue should be frozen, which would allow the host to do Request Sense, or any other error recovery.

### 5.4.2 SCSI I/O Executed Response Message

The format of the *SCSI I/O Executed* response message is identical to the *Execute SCSI I/O* command. The only differences are the status and residual count fields in the private data area. It is the responsibility of the SIM driver to copy the *data\_res*, *sen\_res*, and *scsi\_stat* fields from the private data area of the CCB and the (CAM) *status* field from the queue carrier to the correct fields in the CCB before the command complete callback is performed. This copying is required for the CAM status field to avoid problems when the peripheral driver is polling the CAM status field. The copying of the other fields is an adapter performance optimization.

Note that the fields *sen\_res*, *scsi\_stat*, and *data\_res* must be initialized by the host to success values (as defined by CAM) before the command is placed on the command queue to the adapter. This is a performance optimization allowing the adapter to not DMA to these fields for successfully completed *Execute SCSI I/O* commands.

### 5.4.3 Execute Target I/O Command

It is not valid for the host to place an *Execute Target I/O* command on the command queue (DACQ) to the adapter. To provide this functionality a list of queue elements containing *Execute Target I/O* formatted commands is provided to the adapter via the *Enable LUN* command. The format of an *Execute Target I/O* command is identical to the format of an *Execute SCSI I/O* command. The difference is in the use of the CAM defined target mode fields and the CAM defined function code of 31<sub>16</sub>.

The adapter, when selected for a target mode operation, must first locate the queue element containing the appropriate *Execute Target I/O* command from the list provided in

the *Enable LUN* command. A CDB match must be made and the CAM defined *Target CCB Available* flag must be set in the CAM flags field. The target mode operation is completed from information provided in the *Execute Target I/O* command.

The required response to this command is a *Target I/O Executed* response message.

#### 5.4.4 Target I/O Executed Response Message

The format of the *Target I/O Executed* response message is identical to the *SCSI I/O Executed* response message. The host can distinguish a completed target mode operation by the CAM defined *Execute Target I/O* opcode of 31<sub>16</sub>. The adapter is required to clear the CAM defined *Target CCB Available* flag before it places the queue element on the Adapter-Driver Response Queue (ADRQ). The adapter cannot use the same queue element for a subsequent target mode operation until the host peripheral driver sets the *Target CCB Available* flag, after processing the completed I/O.

#### 5.4.5 Release SIM Queue Command

The *Release SIM Queue* command is used to unfreeze a SIM Queue. Refer to the CAM XPT/SIM specifications for a complete definition of this command.

This command always succeeds and thus does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

The SIMport adapter maintains a freeze counter for each LUN to avoid a race condition where two commands with status other than GOOD complete on the same target/LUN before the HBA executes a Release SIM Queue command from the host. These non-negative counters are zeroed on device enable (implicit on channel enable), incremented each time that a command completes with status other than GOOD, and decremented each time a Release SIM Queue command is executed. Commands are only executed from the SIM Queue when this counter has the value of zero.

#### 5.4.6 Abort SCSI I/O Command

The *Abort SCSI I/O* command is used to abort a previously issued *Execute SCSI I/O* command. Refer to the CAM XPT/SIM specifications for a complete definition of this command. The CCB to be Aborted Pointer field contains the physical address of the CCB to be aborted.

This command does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

If the channel still has the command, it attempts to abort it. If the channel has already completed the command this command is ignored. The command is aborted, removed from the SIM queue, and placed on the response queue immediately, whether or not it is at the head of the queue. Note that the actual status of the command to abort is returned in the status field of the aborted command. The status returned in the aborted command will be a failure status and will cause the SIM queue to freeze unless the SIM Queue Freeze Disable bit is set in the *Execute SCSI I/O* command that was aborted.

#### 5.4.7 Reset SCSI Bus Command

The *Reset SCSI Bus* command is used to reset the bus of a SCSI channel. Refer to the CAM XPT/SIM specifications for a complete definition of this command.

This command causes the indicated SCSI channel to return all outstanding I/O with bus reset status and then to enter the disabled state. While in the disabled state, all new I/O



is also returned with bus reset status. The host is required to issue a *Set Channel State* command to re-enable the channel.

This command always succeeds and thus does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

#### 5.4.8 Reset SCSI Device Command

The *Reset SCSI Device* command is used to reset a device on a SCSI channel. Refer to the CAM XPT/SIM specifications for a complete definition of this command.

This command causes the indicated SCSI device to return all outstanding I/O with bus device reset status and then to enter the disabled state. While in the disabled state, all new I/O (directed to the reset device) is also returned with bus device reset status. The host is required to issue a *Set Device State* command to re-enable the device. The first *Execute SCSI I/O* command for any LUN on a target that has been reset must include new negotiation instructions for width, ~~and~~ speed and whether or not synchronous transfers are done. This is necessary because the adapter and target can get out of synchronization on previously negotiated parameters if the target detects an error on the *Reset SCSI Device* command and transitions to BUS FREE without notifying the adapter.

This command always succeeds and thus does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

#### 5.4.9 Terminate SCSI I/O Process Command

The *Terminate SCSI I/O Process* command is used to terminate a previously issued *Execute SCSI I/O* command. Refer to the CAM XPT/SIM specifications for a complete definition of this command. The CCB to be Aborted Pointer field contains the physical address of the CCB to be terminated.

This command does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

If the channel still has the command, it attempts to terminate it. If the channel has already completed the command this command is ignored. Note that the actual status of the command to terminate is returned in the status field of the terminated command. The status returned in the terminated command will be a failure status and will cause the SIM queue to freeze unless the SIM Queue Freeze Disable bit is set in the *Execute SCSI I/O* command that was terminated.

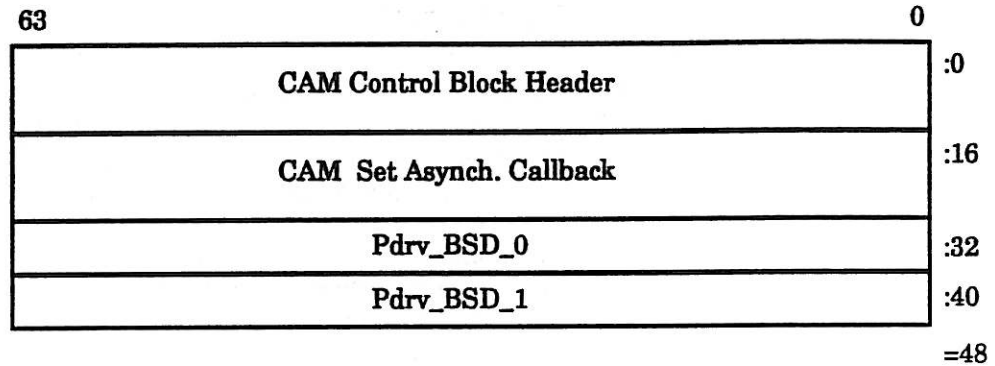
#### 5.4.10 Set ASYNC Callback Command

The *Set Async Callback* command is used to pass the AEN buffer address and size to the adapter. *Set Async Callback* callback commands are treated as an adapter wide resource instead of being assigned to a particular Bus/Target/LUN nexus as specified in the CAM XPT/SIM spec. The adapter will fill in the Connect ID field of the command buffer with the identification of the nexus that triggered the event. To reduce the possibility of losing events when the interarrival time is small, the adapter will specify a recommended number of buffers that the host should queue to it in the *Adapter State Set* response message (see section 5.2.4). These buffers shall be replaced by the host as they are used by the adapter.

Note that the only information that the adapter uses from this command is the address and size of the AEN buffer. That is, the adapter only examines bit 3 of the *Asynchronous Event Enable* field (SCSI AEN). If the SCSI AEN bit is set, the adapter saves the *Periph-*

eral Driver Buffer Pointer field buffer defined by Pdrv\_BSD\_0, Pdrv\_BSD\_1, and the Size of Allocated Peripheral Buffer fields.

Figure 5-15 illustrates the CCB format used for this command. Note that the standard CAM defined CCB is used, but it is extended to include two Buffer Segment Descriptors (see section 10.1). The BSDs define the location of the host resident AEN buffer in physical address format. Since an AEN buffer can cross at most one page boundary (a one byte length field), at most, two segment descriptors are required. Table E-3 describes the CCB fields that must be provided for this command.



**Figure 5-15 Adapter Specific Command/Response Format**

The queue element used to issue this command when used to enable the AEN function, is retained by the adapter for use in reporting the AEN event. An AEN event is reported as a 'delayed' response to this command. If the SCSI AEN bit is clear, the adapter interprets this command as a request to disable the AEN function for the indicated <either buffer (specified by the pa of the BSD) or for all buffers, T.B.D.> . The queue element used to issue this command is immediately returned to the host over the ADRQ. The queue elements that were retained when the AEN function was enabled, are returned to the host via the ADFQ.

The required (but possibly delayed) response to this command is an AEN response message.

#### 5.4.11 AEN Response Message

The AEN response message is used to inform the host that the AEN host resident buffer for the indicated nexus has been written with AEN data. The address and size of the AEN host resident buffer is established via the *Set Asynch Callback* CCB command. The queue element used to issue the *Set Asynch Callback* command is retained and used to issue the AEN response message. Thus, the format of the AEN response message is identical to the *Set Asynch Callback* command. **Note that the host is required to re-issue the *Set Asynch Callback* command in order for the adapter to generate subsequent AEN response message for the same Bus/Target/LUN nexus.**

The AEN response message is also used to acknowledge a *Set Asynch Callback* command issued to disable the AEN function. Again, the same queue element used to issue the *Set Asynch Callback* command is used as the AEN response message. The host can distinguished between the two types of AEN response messages (i.e., one to indicate the AEN data buffer written and the other acknowledging the AEN disable function) by the SCSI AEN bit. When the SCSI AEN bit is set, the AEN response message indicates that the AEN buffer contains valid data. When the SCSI AEN bit is clear, the AEN response message is an acknowledgment that the AEN function has been disabled.

#### 5.4.12 Path Inquiry Command

The *Path Inquiry* command is used to obtain CAM and SCSI parameter information. Refer to the CAM XPT/SIM specifications for a complete definition of this command.

The required response to this command is a *Path Inquiry* response message.

### 5.4.13 Path Inquiry Response Message

The *Path Inquiry* response message is the required response to a *Path Inquiry* command. It has the same format as the *Path Inquiry* command. Note that the CCB CAM status field is written directly by the adapter (since this command does not have a private data area).

Note that bits set in the Asynchronous Event Capabilities field imply that the adapter will issue *unsolicited messages* as defined in chapter 6. SIMport does not use *Set Asynch Callback* to report Unsolicited SCSI Bus Reset, Unsolicited Reselection, or Sent BDR to Target.

In a CAM environment, some fields in the Path Inquiry response are filled in by the HBA, while other fields are filled in by the host SIM layer. In a non-CAM environment the fields not filled in by the HBA are not necessarily valid. The HBA must OR in its values for the bits defined in the *Reasons* sub-field of the Asynchronous Event Capabilities field so as to not destroy the values that may have been previously set by the host. Table 5-30 lists which component is responsible for filling in which fields.

Table 5-30 Path Inquiry Field Responsibilities

Field	Size	Responsible party
CAM_STATUS	1	HBA
Features Supported		
Version Number	1	HBA
SCSI Capabilities	1	HBA
Target Mode Support	1	HBA
Miscellaneous	1	Host
HBA Capabilities	16	HBA
Size of private data area	4	Host
Asynchronous Event Capabilities		
Vendor Unique	1	
Reserved	2	
Reasons	1	
New Devices Found		Host
SIM De-Registered		Host
SIM Registered		Host
Sent BDR to Target		HBA
SCSI AEN		HBA
Unsolicited Reselection		HBA
Unsolicited SCSI Bus Reset		HBA
Highest Path ID Assigned	1	Host
SCSI Device ID	1	HBA
Vendor ID of SIM	1	Host
Vendor ID of HBA	1	HBA

## 5.4.14 Enable LUN Command

SIMport only supports CAM Processor Mode and thus only CDBs for Inquiry, Receive, Request Sense, and Send are supported. A modified CCB, illustrated in figure 5-16, is used for this command. The difference is that the Enable LUN CCB contains a pointer to the queue carrier of the first target CCB, and that the list of target CCBs is defined by a linked list of queue carriers which specify the CCBs in their queue buffer field.

This command always succeeds and thus does not have a response message. The host SIM copies the original CCB into a free queue element and gives the copy to the HBA. The original CCB is returned with a CAM status of "Request completed without error." The HBA places the copied CCB on the ADFQ upon completion of the command.

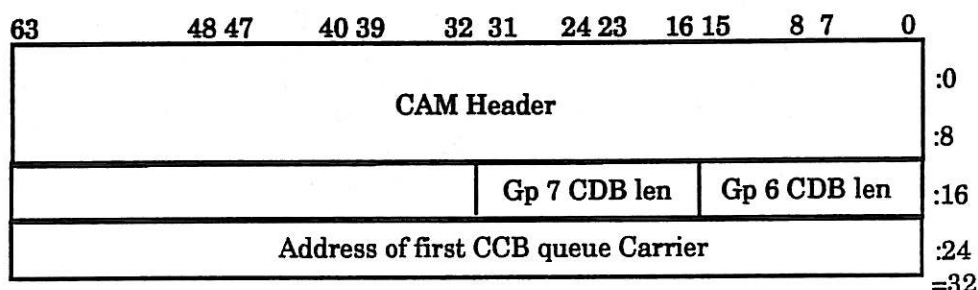


Figure 5-16 Format of Enable LUN Command

Table 5-31 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

Table 5-31 Enable LUN Command Fields

Field	Size	Value	Description
Gp 6 CDB Len	<15:0>	n	see CAM spec
Gp 7 CDB Len	<31:16>	n	see CAM spec
Address of first CCB	<63:0>	pa	Physical address of first CCB queue carrier in linked list of CCB queue carriers.



# Unsolicited Message Processing

This chapter defines the format of unsolicited messages. Unsolicited messages are generated by the adapter and typically are used to notify the host of unexpected events. Figure 6-1 illustrates the general format of an unsolicited message.

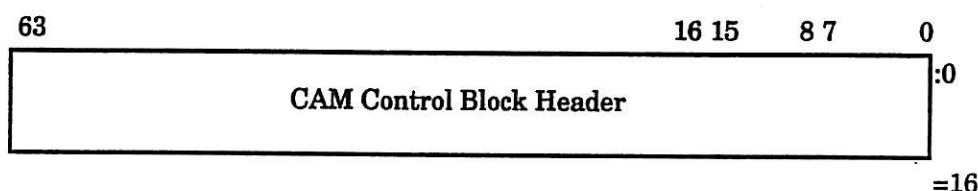


Figure 6-1 Unsolicited Message Format

## 6.1

### Free Queue Element Management

Unsolicited messages are placed on the response queue (ADRQ). The adapter is required to obtain queue entries for unsolicited messages from the free queue (DAFQ). The host, during adapter initialization, is required to place a fixed number of free queue entries on the DAFQ. The number required by the adapter is specified in the adapter *State Set* response message. After processing an unsolicited message from the response queue (ADRQ), the host is required to return either the same queue entry, or another queue entry, to the DAFQ so to maintain the free queue level.

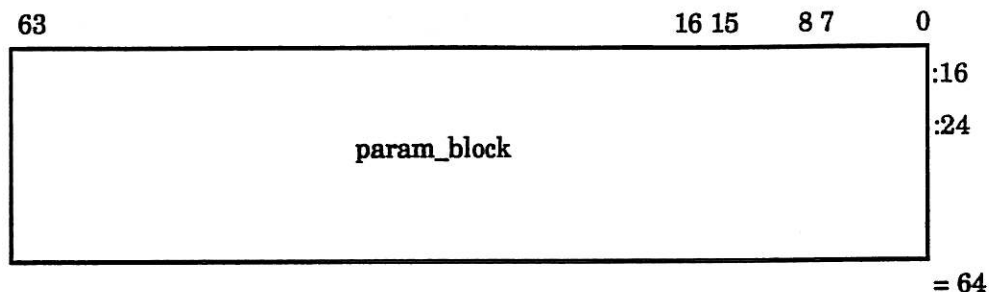
## 6.2

### Unsolicited Reselection Message

A SIMport adapter will indicate its support for generating this message in the *Path Inquiry* response (bit 1 of the 'Asynchronous Event Capabilities' field).

If the support bit is set, then this unsolicited message shall be placed on the ADRQ whenever an unsolicited reselection occurs. The adapter specific function code for *Unsolicited Reselection* is 88<sub>16</sub>. If the host is running CAM, then the SIMport driver will initiate an asynchronous callback to the upper layer, assuming that this event was pre-registered via the 'Set Asynchronous Callback' CCB. Figure 6-2 illustrates the the *Unsolicited Reselection* message. If the host is not running CAM then the message should be error-logged.

If the support bit is not set, then this unsolicited message will never be generated. A counter is kept in the adapter for unsolicited reselections, the value of this counter is available via the *Read Counters* command.



**Figure 6-2 Unsolicited Reselection Message**

Table 6-1 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

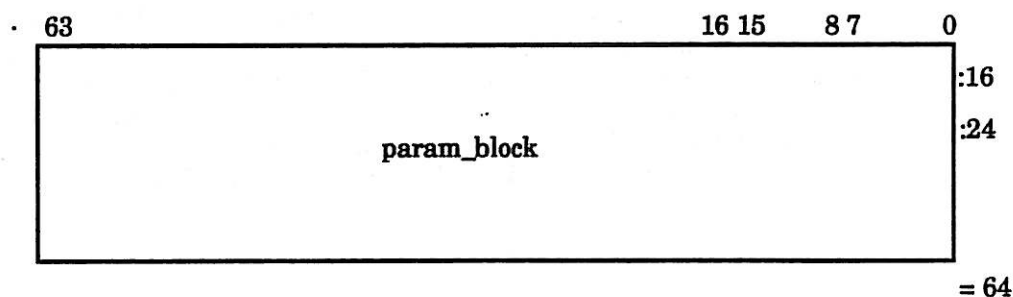
**Table 6-1 Unsolicited Reselection Message Fields**

Field	Size	Value	Description
param_block	see below	see figure	The parameter block is used to report addition information related to the status.

## 6.3

### Channel Disabled Message

The *Channel Disabled* message is used to inform the host of an error condition which has caused a channel to transition to the disabled state. The adapter specific function code for *Channel Disabled* is 89<sub>16</sub>. Figure 6-3 illustrates the SIMport function CCB format. If the status code (disabled reason code) is not 'Bus Reset', it is recommended that after the channel has been re-enabled the host either issue the 'Reset SCSI Bus' command or a series of 'Reset SCSI Device' commands in order to resynchronize device context with the adapter



**Figure 6-3 Channel Disabled Message**

Table 6-2 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 6-2 Channel Disabled Message Fields**

Field	Size	Value	Description
param_block	see below	see figure 6-4	The parameter block is used to report additional information related to the status.

Table 6-3 specifies the values for the *status* field in the Queue Carrier.

**Table 6-3 Channel Disabled Message Status**

Status	Value	Description
bus reset	2	Bus Reset detected on the SCSI bus
Reset Request Denied	3	The Host denied the adapter's request to reset the SCSI bus.

Figure 6-4 illustrates the format of the parameter block of the *Channel Disabled* message.  
[Figure T.B.D.]

**Figure 6-4 Channel Disabled Message Parameter Block**

## 6.4

### Device Disabled Message

The *Device Disabled* message is used to inform the host of a bus device reset which has caused the channel device to transition to the disabled state. The adapter specific function code for *Device Disabled* is  $8A_{16}$ . SIMport does not define any fields in the CCB.

Table 6-4 specifies the values for the *status* field in the Queue Carrier.

**Table 6-4 Device Disabled Message Status**

Status	Value	Description
device reset	4	Bus Device Reset sent to device on the SCSI bus

# Target Mode Operation

[This chapter should describe target mode operation, in general. Note that the commands used for target mode operation (i.e., Execute Target I/O and Enable Lun) are described under the appropriate command sections of this document. This chapter is to further clarify target mode operations, including target mode queue element management]

## **7.1 Processor Mode**

### **7.1.1 Inquiry command**

### **7.1.2 Receive command**

### **7.1.3 Request Sense command**

### **7.1.4 Send command**

## **7.2 Phase Cognizant Mode**

## HBA Engines

**SIMport does not currently specify any HBA engines. Engine Inquiry is not listed as a supported function code and hence will return a CAM status of Invalid Request, indicating that the specified engine is not installed. This action complies with the CAM spec.**

This chapter defines the host and adapter interrupts.

## 9.1 Host Interrupts

The host is interrupted for two reasons. One is to notify the host that new entries are on the response queue (ADRQ). The other is to notify the host that the Adapter Status Register (ASR) has changed and thus requires servicing.

### 9.1.1 Command Complete Interrupt

The command complete interrupt notifies the host that the adapter has placed new entries on the Adapter-Driver Response Queue (ADRQ). This interrupt is subject to the interrupt control holdoff timer, see section 3.6.

### 9.1.2 Adapter Miscellaneous Interrupt

The Adapter Miscellaneous interrupt notifies the host that the adapter requires service. The required service is indicated in the Adapter Status Register (see section 3.5.5). The Adapter Miscellaneous interrupt is typically used to inform the host of an adapter error condition that has caused the adapter to transition to the uninitialized state. If possible, this interrupt should be higher priority than the Command Complete Interrupt.

### 9.1.3 Systems with single interrupt only support

For systems which do not support multiple interrupt vectors, the ASR must be read to distinguish between the Command Complete and Adapter Miscellaneous interrupts (see section 3.5.5).

## 9.2 Adapter Interrupts

The adapter is interrupted for the following reasons.

- To reset the adapter.
- To initialize the adapter.
- To indicate new commands on the Driver-Adapter Command Queue.
- To indicate new entries on the Driver-Adapter Free Queue.

Each adapter interrupt corresponds to a write to an adapter register.

### 9.2.1 Initialization Interrupt

To initialize the adapter the host should write the Adapter Block Base Register (ABBR). See sections 5.2.2 and 3.5.2 for additional details.

### 9.2.2 Reset Interrupt

To reset the adapter the host should set the MIN bit of the Adapter Maintenance Control and Status Register (AMCSR). See sections 5.2.1 and 3.5.1 for additional details.

### **9.2.3 Command Queue Insertion Interrupt**

When the host places a command on the Driver-Adapter Command Queue (DACQ) it is required to write the DACQ Insertion Register (DACQIR). See section 3.5.3 for additional details.

### **9.2.4 Free Queue Insertion Interrupt**

When the host places a queue entry on the Driver-Adapter Free Queue (DAFQ) it is required to write the DAFQ Insertion Register (DAFQIR). See section 3.5.4 for additional details.

## Buffer Memory Mapping

SIMport requires that the SIM driver provide to the adapter all host memory addresses in physical address format. For host memory buffers that are not physically contiguous, the host is required to provide to the adapter a Buffer Segment Map (BSM). A BSM is a list of Buffer Segment Descriptors (BSDs), where each BSD in the list defines a physically contiguous segment of the host memory buffer. A BSD contains the physical address of the start of a buffer segment and the number of bytes in the buffer segment. For scatter-gather type buffers, a Buffer Segment Map is required that defines all physically contiguous segments for the entire scatter-gather list.

CAM allows data buffers for SCSI I/O to be large. A single data buffer can be up to 4GB in length. CAM also has support for scatter-gather type data buffers, where each entry in the scatter-gather list can be a 4GB buffer and the list can have up to 64K entries. To avoid a multi-level buffer segment mapping scheme as well as to reduce the requirements for physically contiguous memory, a fixed length BSM is used. Note that for large discontinuous host buffers, the BSM will not have enough segment descriptors to define the entire host buffer.

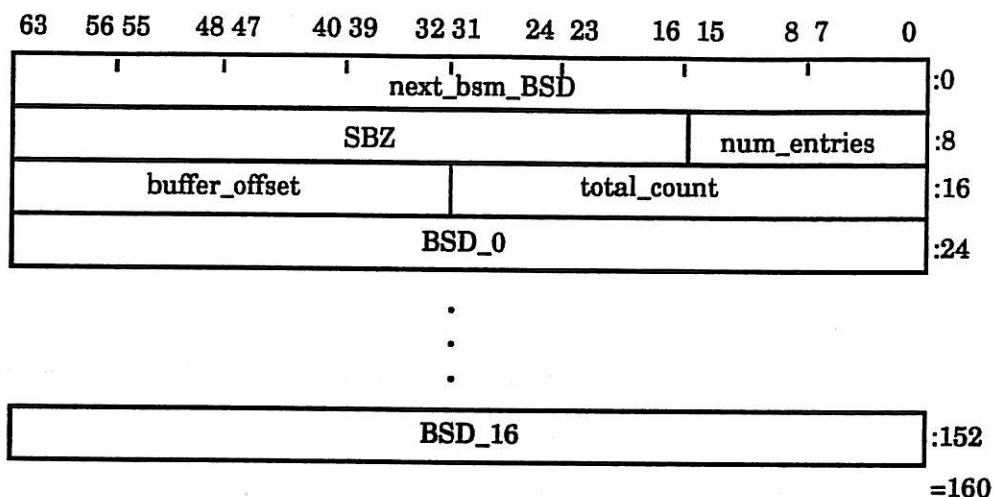
To support large discontinuous host buffers, the host has the option of either defining a linked list of BSMs which define all the segments in the buffer or to define one BSM which defines only the segments at the start of the buffer. For the latter, the adapter is required to issue request messages to the host to obtain additional BSDs dynamically.

### 10.1

#### Buffer Segment Map

The Buffer Segment Map (BSM) is hexaword aligned, physically contiguous, host resident data structure that defines a host memory buffer (or a virtually contiguous portion of a host memory buffer) in terms of a sequential list of Buffer Segment Descriptors (BSDs). Each BSD defines a physically contiguous segment of the host memory buffer and contains the physical address of the start of the segment and a count of the number of bytes in the segment. A BSM is usually associated with a SCSI I/O CCB request, where a pointer to the BSM is supplied in the private data area. The BSD is also used to point to a BSM. A type field in the BSD is used to indicate whether the BSD points to a host buffer segment (type 0) or to a BSM (type 1). Figure 10-1 illustrates the format of a BSM.





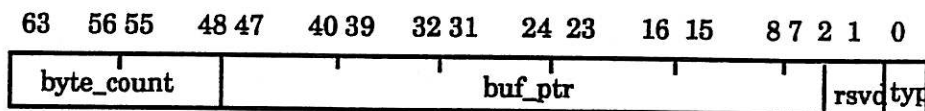
**Figure 10-1 Buffer Segment Map (BSM)**

Note that the *BSD\_n* fields within the BSM can only be used to point to host buffer segments and never to additional BSMs. Also note that the *next\_bsm\_BSD* field can only be used to point to a BSM and never to a host buffer segment. Table 10-1 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 10-1 Buffer Segment Map Fields**

Field	Size	Value	Description
next_bsm_BSD <63:0>		see table 10-2	A <i>BSD</i> which defines the location of the next BSM, if the link option is used. A zero Type Field in the <i>BSD</i> indicates the end of the linked list.
num_entries <15:0>		1-17	Number of valid <i>BSDs</i> in this BSM.
total_count <31:0>		n	Total number of bytes mapped by this BSM. i.e., sum total of <i>byte_count</i> fields in each <i>BSD</i> .
buffer_offset <63:32>		n	Byte offset from the start of the host buffer for this BSM.
BSD_n <63:0>		see table 10-2	A Buffer Segment Descriptor defines one physically contiguous segment of the host buffer.

Figure 10-2 illustrates the format of a *BSD*.



**Figure 10-2 A Buffer Segment Descriptor (BSD)**

Table 10-2 lists, for each field, the field name, the field size, the allowed field values, and a brief description of the field.

**Table 10-2 Buffer Segment Descriptor Fields**

Field	Size	Value	Description
typ	<0:0>	0,1	Type of buffer pointer. 0: physical address of the host buffer, 1: physical address of a BSM for the buffer.
rsvd	<1:1>	0,1	Reserved.
buf_ptr	<47:2>	pa	The physical address of the host buffer (or of the BSM for the host buffer) shifted such that no bits that are known to be zero are presented. If the value of align_xfer (see section 5.2.4) is 2, then this field is shifted to the right 2 bits such that the LSB of the field represents bit 2 of the actual address.
byte_count	<63:48>	n	Size in bytes of the host buffer segment or of the BSM.

The *typ* field within the BSD determines the meaning of the *byte\_count* and *buf\_ptr* fields. A *typ* 0 BSD defines the starting physical address of a physically contiguous segment of a host buffer and the number of bytes in the segment. A *typ* 1 BSD defines the starting physical address of a physically contiguous BSM and the number of bytes in the BSM. Note that although the BSM has a maximum size, the *byte\_count* fields reflects the amount of significant data (since not all of the *BSD\_n* fields are valid).

## 10.2

### Buffer Segment Map Options

To support large discontinuous host buffers, the host may use either a linked list of BSMs which define all of the segments in the buffer or one BSM which defines only the segments at the start of the buffer, depending upon which methods are supported by the adapter as specified in the *Adapter State Set* command (see section 5.2.4). For the latter, the adapter is required to issue to the host *BSD Request* messages to obtain additional BSDs dynamically.

If a single BSM is used, the adapter can determine this by a zero type field in the *next\_bsm\_BSD* field and by comparing the *total\_count* field in the BSM with the data length field in the CCB. The adapter can request the host to provide additional BSDs for the host buffer by placing a *BSD Request* message on the response queue to the host. The *BSD Request* message instructs the host to provide additional BSDs that map a specific offset from the start of the buffer. The host is required to return the additional BSDs in a *BSD Response* command.

The *next\_bsm\_BSD* field in the BSM allows the host to provide a linked list of BSMs that map the entire host buffer. This is optional for the SIM driver. To avoid the processing of the *BSD Request* message, the SIM driver can supply all required BSDs in a linked list of BSMs. A value of zero in the type field of the *next\_bsm\_BSD* field indicates that this is the only or the last BSM in the list.

For additional details about the *BSD Response* command see section 5.2.16. For additional details about the *BSD Request* message see section 5.2.15.

Note that SIMport requires the use of the linked list or *BSD Response* message options only for large scatter-gather lists or for large virtually mapped host buffers (greater than 128K when the host page size is 4K).

For host data buffers that are less than or equal to the host memory page size but allowed to cross one page boundary, a BSM is not required. For these cases, the two BSDs con-

tained in the private data area of the CCB are used to point to the physical addresses of the two segments (See Appendix B for exceptions).

[Note that the linked list approach is not a good solution if the adapter supports 'Modify Data Pointer.' For example, if a target device issues a 'Modify Data Pointer' to the end of a very large buffer, the adapter would have to traverse the linked list to locate the correct BSM. For the single BSM approach, the adapter would request the host to provide the BSDs for the specific offset as provided by the 'Modify Data Pointer' operation. Thus for efficient support of 'Modify Data Pointer' for large buffers, it seems appropriate for the host to support the *BSD Request* message. The alternative would be a multi-layered scheme.

The host could provide a multi-layered mapping structure, in an indexed sequential fashion, where a list of addresses of each BSM is supplied. The list would essentially be the set of all the *next\_bsm\_BSD* link fields. This appears to be efficient only if the upper layers of the mapping structures can be cached in the adapter such that the adapter can locate the correct BSM quickly.]

- 1 Target mode operation and Enable LUN required parameters will be specified at some later time.
- 2 Adapter time-out timer. (section 4.2 SIM Driver Initialization Steps)  
10 seconds has been proposed.
- 3 Reset command completion time. (section 4.2 SIM Driver Initialization Steps)  
2 seconds has been proposed.
- 4 Format of channel disabled parameter block. (figure 6-4, page 6-3)  
It has been proposed that the block be removed if there are no suggestions for contents.
- 5 Adapter detected Bus Device Reset status code. (section C.3, page C-2)  
The reserved code 21<sub>16</sub> has been proposed.
- 6 Does a Set Asynch Callback with AEN enable clear disable a specific AEN buffer or all AEN buffers. (section 5.4.10 Set ASYNC Callback Command, page 5-22)

# CAM and SIMport function codes

## A.1

### Supported CAM functions codes

Table A-1 defines the supported CAM function codes. Values are in hex.

**Table A-1 CAM Function Codes**

CAM function		value	meaning
Nop	0		Execute nothing
Execute SCSI I/O	1		Execute the requested SCSI I/O
Path Inquiry	3		Get SCSI channel information
Release SIM Queue	4		Unfreeze indicated SIM Queue
Set Asynch. Callback	5		Set asynchronous callback buffer
Abort SCSI I/O	10		Abort the specified SCSI I/O CCB
Reset SCSI Bus	11		Reset indicated SCSI Channel
Reset SCSI Device	12		Reset indicated SCSI Device
Terminate SCSI I/O	13		Terminate the specified SCSI I/O CCB
Enable LUN	30		Enable LUN for target mode operation
SIMport Adapter Specific	80-FF		Vendor Unique Adapter Specific code

## A.2

### SIMport Adapter Specific function codes

Table A-2 defines the SIMport function codes. Values are in hex.

**Table A-2 SIMport Function Codes**

<b>SIMport function</b>	<b>value</b>	<b>meaning</b>
Set Adapter State	80	Enable/disable adapter
Set Parameters	81	Set adapter parameters
Set Channel State	82	Enable/disable channel
Set Device State	83	Enable channel device
Verify Adapter Sanity	84	Verify DACQ and ADRQ operation
Read Counters	85	Read internal adapter counters
BSD Request	86	Request for BSD data
Bus Reset Request	87	Request to reset SCSI bus
Unsolicited Reselection	88	Unsolicited Reselection unsolicited message
Channel Disable	89	Channel disabled unsolicited message
Device Disable	8A	Device disabled unsolicited message
Reserved	8B-FF	Reserved for future use

## SIMport Status codes

Table A-3 defines the SIMport status codes.

Table A-3 SIMport status codes

Status Name	value	meaning
success	1	Successful completion of command
inv_command	0	Unsupported or improperly formatted command
not_enabled	-1	The adapter, channel, device or function is not enabled.
not_disabled	-2	The Adapter is not in the disabled state
host_mem	-3	Before the adapter will transition to the enable stated, the host driver must allocate, for exclusive adapter use, <i>n_host_sg</i> number of 4K host memory segments and issue a <i>Set Parameter</i> command to notify the adapter of the physical addresses of the allocated host memory.
host_fqe	-4	Host Free Queue Elements must be allocated.
inv_int_holdoff	-5	Interrupt Holdoff timer invalid or not supported.
inv_host_sg	-6	Insufficient host memory allocated.
inv_chan_id	-7	Invalid channel identifier or channel does not exist.
inv_node_id	-8	Invalid SCSI bus node identifier or soft setting not supported.
cant_disable	-9	A device channel cannot be disabled by host command.
inv_offset	-10	Invalid offset value or offset not within buffer.
inv_CCB	-11	Invalid CCB address.
inv_buf_id	-12	Invalid buf_id field value.
no_reset	-13	Adapter should not reset the SCSI bus.
bus_reset	2	Bus reset detected on the SCSI bus
reset_request_denied	3	Host denied the adapter's request to reset the SCSI bus
device_reset	4	Bus Device Reset sent to device on the SCSI bus

# Unaligned Host Memory Buffers

## B.1

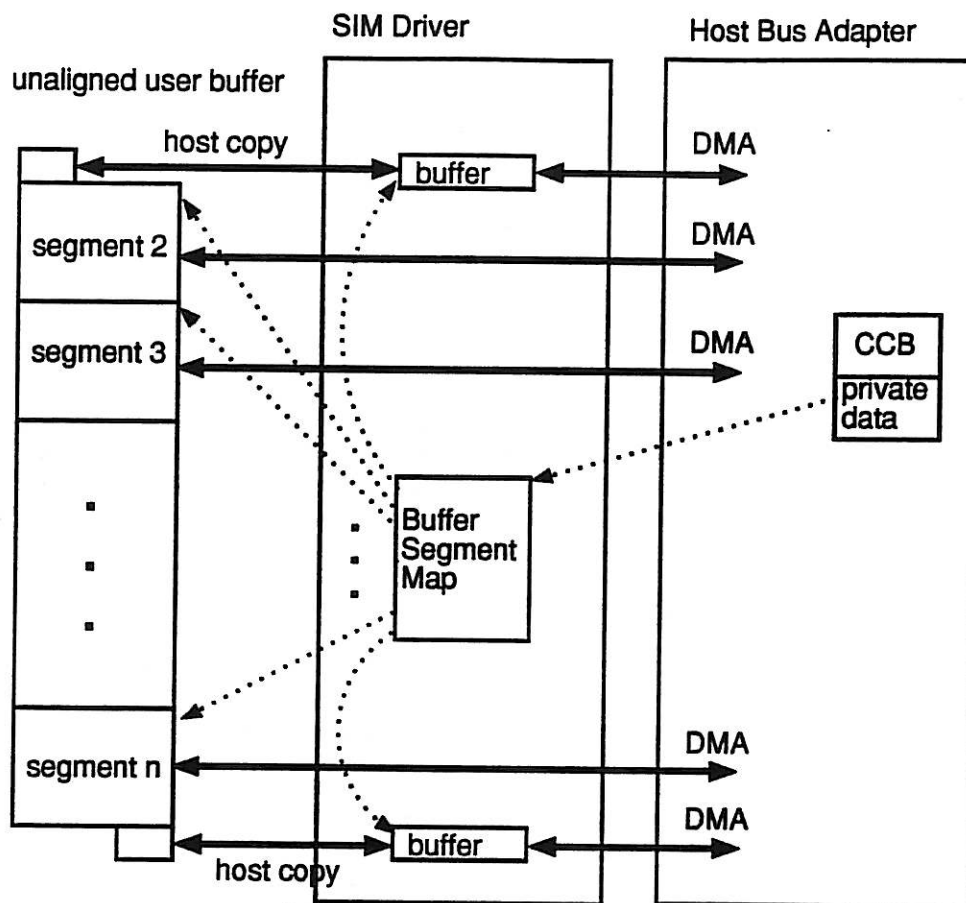
### Unaligned Buffers

Some adapters do not support byte-aligned transfers because of the system bus used. To accommodate host buffers which are not aligned on the boundary specified by the adapter a host assisted solution is required.

When a host detects a buffer is not aligned on the desired boundary, it must allocate a buffer of the minimum alignment size for the start of the buffer. If, in addition, the transfer count indicates that the end of the buffer requires a non-aligned transfer, the host must allocate a second buffer for the end of the buffer. The host should now treat the transfer as if it were a transfer to a scatter-gather buffer. The first scatter-gather entry would define a transfer of the first few unaligned bytes to the first host allocated (alignment) buffer. The last scatter-gather entry would define a transfer of the last few unaligned bytes to the second host allocated (alignment) buffer. The middle entry of the scatter-gather list would define a direct transfer to the aligned portion of the user buffer.

Figure B-1 illustrates a BSM with two buffers allocated for the unaligned portions of the user buffer. The figure also illustrates the DMAs that the adapter would perform and the copying required by the host.





**Figure B-1 DMA of unaligned host buffers**

Note that scatter-gather lists may contain many unaligned buffer segments and that an alignment buffer would have to be allocated for each unaligned portion.

### **B.1.1 Writing unaligned host buffers**

The host is required to construct a buffer segment map (BSM) for unaligned buffer writes such that the entries in the BSM direct the adapter to write the unaligned bytes to special aligned buffers. The special buffers are privately allocated by the SIM driver before the I/O is queued to the adapter. Upon command completion, the host is required to copy the appropriate bytes from the alignment buffers to the appropriate locations in the unaligned buffers. The adapter positions the bytes such that the first byte of the transfer is in byte zero of the buffer.

### **B.1.2 Reading unaligned host buffers**

The host is required to construct a buffer segment map (BSM) for unaligned buffer reads such that the entries in the BSM direct the adapter to read the unaligned bytes from special aligned buffers. The special buffers are privately allocated by the SIM driver before the I/O is queued to the adapter. In addition, before the I/O command is placed on the command queue to the adapter, the host is required to copy the appropriate unaligned bytes from the user buffer to the special alignment buffers and to position the unaligned bytes such that the first byte of the transfer is located in byte zero of the alignment buffer. Note that for reads, the unaligned bytes at the end of the buffer do not require a special buffer, since these bytes start on an aligned boundary (the assumption being the adapter can read the entire transfer unit at the end of the buffer, but only transfer a few bytes).

## Bus Reset and Bus Device Reset

When a SCSI bus is reset or a SCSI bus device is reset, the outstanding I/O for that bus or bus device must be returned to the sender. This appendix presents the sequence of events to ensure that all the I/O is returned before additional I/O is started. The expected information exchanged between the host and the adapter as well as the expected processing steps are presented.

### C.1

#### Adapter detected bus reset

The following sections detail the anticipated steps performed by the adapter and SIM driver during an adapter detected SCSI bus reset. Figure C-1 illustrates the command/response information exchanged between the host and the adapter.

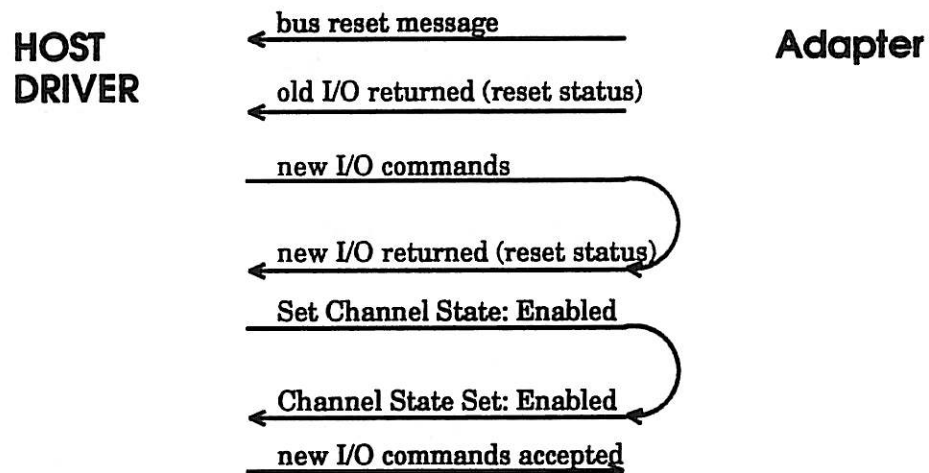


Figure C-1 Adapter/Driver reset information exchanged

#### C.1.1 Adapter reset processing steps

- The SCSI channel detects a bus reset and constructs an unsolicited response message and places it on the response queue to the host. The response message informs the host that a bus reset occurred and that the channel is now in the reset or disabled state.
- When a channel transitions to the reset state, all outstanding channel I/O is returned on the response queue (ADRQ) with the status set to bus reset.
- While a channel is in the reset state, the adapter continues to accept new I/O commands from the command (DACQ) queue. The channel simply sets the I/O status of the new commands to bus reset and returns them on the response queue. This continues until an *Set Channel State: Enabled* command is received on the command queue.

- When the channel receives an *Set Channel State: Enabled* command it responds with an *Channel State Set: Enabled* message indicating success and subsequently resumes normal processing of subsequent I/O requests.

### C.1.2 SIM Driver reset processing steps

- The host SIM is interrupted, indicating messages are on the response queue.
- For I/O complete response messages, the SIM issues command complete callback, as required.
- When the SIM detects the *bus reset* unsolicited message, it 'schedules' the sending of an *Set Channel State: Enabled* command such that the *Set Channel State: Enabled* command follows any commands in progress (i.e., if the SIM command path was interrupted). In addition, a flag can be set that the SIM command path could check. The flag would direct the SIM command path to return (with bus reset or bus busy status) all new I/O requests directed to the reset channel from the XPT.
- After all response messages are processed, SIM returns from the interrupt.
- At this point, new I/O commands directed to the channel will be processed by the channel, since they would follow the *Set Channel State: Enabled* command; although, the SIM may decide to reject such commands until the *Channel State Set: Enabled* message is received on the response queue.
- When the *Channel State Set: Enabled* message is received, the SIM can issue an asynchronous callback to the XPT or to each peripheral driver as required and resume normal command processing.

## C.2

### Adapter initiated bus reset

It may be necessary for the adapter to initiate a bus reset. The only difference between adapter detected bus reset and adapter initiated bus reset would be the reporting of an error condition, in addition to the reset. The adapter initiates a bus reset by sending a *Bus Reset Request* message to the host.

The host must respond by sending a *Bus Reset Response* command to the adapter over the command queue. There is no explicit response to the *Bus Reset Response* command. The adapter, upon receiving a *Bus Reset Response* command, would signal the SCSI channel to reset the SCSI bus. At this point, the same sequence of information exchanged for adapter detected reset occurs (see C.1).

## C.3

### Adapter detected Bus Device Reset

The adapter places a *Device Disabled* message (see section 6.4) on the ADRQ with the connect ID (path and target ID) in the CCB header set to identify itself. All active and pending I/O is returned to the host with CAM status of 'Bus Device Reset Received' (either 17<sub>16</sub> or a new, T.B.D. code). Any new I/O is returned with CAM status of 'Bus Device Reset Received' until a *Set Device State:Enabled* command (see section 5.2.9) is received for the adapter's connect ID.

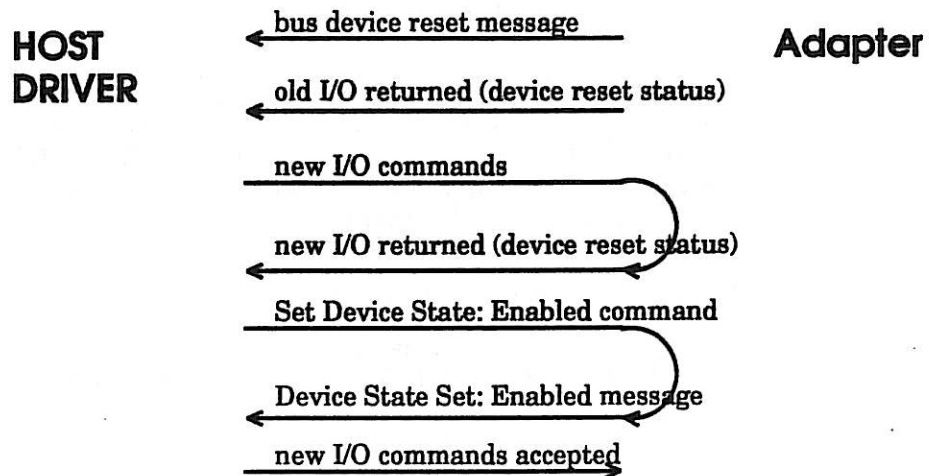
## C.4

### Bus Device Reset

Bus device reset has the same aspects of bus reset in terms of outstanding I/O cleanup. The main difference is the scope for bus device reset is for one device as opposed to all devices on the bus. This implies that a bus device has state information as does a channel. That is, if a bus reset places a channel in a reset state until cleared by the host, then a bus device reset should place a device in a reset state until cleared by the host.

Thus, it seems reasonable to apply the same sequence of events used to process a bus reset to the processing of a bus device reset. The main changes in the event sequence are to change the *bus reset* message to a *Reset SCSI Device* message, to change the *Set Channel*

*State: Enabled* command to a *Set Device State: Enabled* command, and to change the corresponding response to an *Device State Set: Enabled* message. The only other changes would be that the I/O that is returned with device reset status is only the I/O directed to the reset device. Figure C-2 illustrates these changes. Additional details about bus device reset are T.B.D.



**Figure C-2 Adapter/Driver bus device reset information exchanged**

#### **C.4.1 Adapter initiated Bus Device Reset**

The SIMport adapter will issue a Bus Device Reset under certain conditions, such as when an ABORT or ABORT TAG is rejected.

#### **C.4.2 Host initiated Bus Device Reset**

The host may initiate a bus device reset by sending a *Reset SCSI Device* command to the adapter over the command queue. There is no explicit response to the *Reset SCSI Device* command. It takes the form of an immediate command as defined by the CAM XPT/SIM specification. The adapter, upon receiving a *Reset SCSI Device* command, would signal the SCSI channel to reset the SCSI bus device. At this point, the bus device reset sequence of events is similar to a bus reset.

# SIMport on 64-bit architectures

## CAM Command Format

This appendix documents the format of the CAM defined commands supported under SIMport. The fields required or accessed by a SIMport adapter are identified and explained within the context of SIMport. In addition, the use of CAM defined vendor unique fields are defined. CCB format differences for 32 and 64 bit host CPUs are also documented.

### D.1

#### CCB format on systems with 64 bit addresses

This section explains the difference between a CCB formatted under a 32 bit host and a CCB formatted under a 64 bit host. The *ccb\_ptr\_sz* field, specified during initialization, in the SIMport adapter block indicates the word size of the host CPU.

On a 64 bit host the fields in the data structures that are declared to be \* (pointer) types increase from 32 bits to 64 bits in length. Address fields in vendor unique commands are always allocated 64 bits; on a 32 bit host only the lower 32 bits are significant.

The format of a BSD is unchanged. The BSD specifies a 46-bit physical address and a 16-bit byte count field to support host buffer segments. The offsets of fields in the vendor unique CCB change because the size of the CCB header changes; see figure D-1. The offsets of CAM defined CCBs change as well, though they are not illustrated here as use of the C language definitions in the CAM spec will provide the correct definitions. ~~The two undefined bytes and the two bytes of parameters are not part of the header, but are including in the figure for ease of illustration.~~

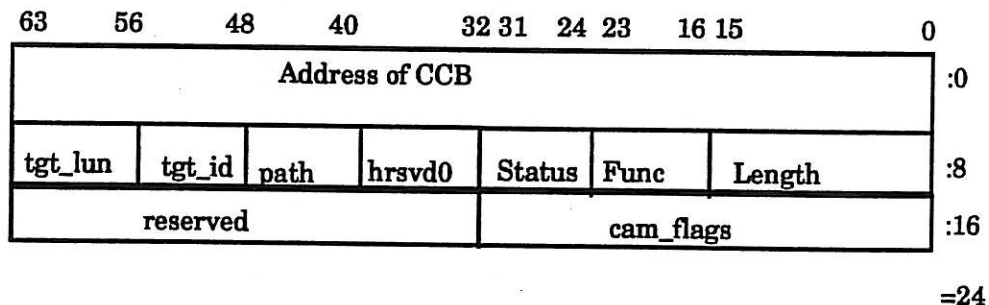


Figure D-1 64 Bit SIMport CCB Header format

# CAM command formats

## E.1

### SIMport supported CAM CCB formats

This section documents the format of the CAM defined CCBs supported under SIMport. The fields required by the adapter are specified and thus these are the only fields required by a non-CAM based host. All commands require that the *Address of this CCB*, *CAM Control Block Length*, *Function code*, and *Connect ID* fields be filled in with the appropriate information.

#### E.1.1 Execute SCSI I/O Command

**Table E-1 Execute SCSI I/O Command Field Usage**

Field	Usage
CAM Flags	As defined in the CAM spec, except byte 3 bits are ignored as all data pointers are specified via BSDs in the Private Data area.
Peripheral Diver Pointer	Unused
Next CCB Pointer	Unused, specified in the private data area, see table 5-29.
Request Mapping Information (OSD)	Unused.
Callback on Completion	Unused.
SG List/Data Buffer Pointer	Unused, specified in the private data area, see table 5-29.
Data Transfer Length	As defined in the CAM spec.
Sense Info Buffer Pointer	Unused, specified in the private data area, see table 5-29.
Sense Info Buffer Length	As defined in the CAM spec.
CDB Length	As defined in CAM spec.
Number of Scatter/Gather entries	Unused, specified in the private data area, see table 5-29.
CDB	As defined in CAM spec. If specified as a pointer, then the cdb_BSD field in the Private Data Area contains the Physical Address.
Timeout Value	As defined in CAM spec.
Message Buffer Pointer	Unused, specified in the private data area, see table 5-29. Only used for Target Mode operation.

Message Buffer Length	As specified by the CAM spec. Only used for Target Mode operation.
VU Flags	See table E-2 for definition.
Tag Queue Action	As defined in CAM spec.
Private Data Area	See table 5-29 for definition.

---

**Table E-2 Execute SCSI I/O VU Flags**

---

Bit	Meaning
1-0	Wide Transfer Mode 00 = No Change 01 = 8 bit transfers 10 = 16 bit transfers 11 = 32 bit transfers
2	1-LUN field is SCSI target routine number
3-15	Reserved (Should Be Zero)

---

### **E.1.2 Release SIM Queue Command**

CAM Flags (OSD) - None defined.

### **E.1.3 Abort SCSI I/O Command**

CAM Flags (OSD) - None defined.

CCB to be Aborted Pointer - physical address of CCB to be aborted. This field has been extended to be 64 bits.

### **E.1.4 Reset SCSI Bus Command**

CAM Flags (OSD) - None defined.

### **E.1.5 Reset SCSI Device Command**

CAM Flags (OSD) - None defined.

### **E.1.6 Terminate SCSI I/O Process Command**

CAM Flags (OSD) - None defined.

CCB to be Aborted Pointer - physical address of CCB to be aborted. This field has been extended to be 64 bits.

### **E.1.7 Set ASYNC Callback Command**

---

**Table E-3 Set ASYNC Callback Command Fields**

---

Field	Usage
CAM Flags (OSD)	Unused



Asynchronous Event Enables	None defined.
Vendor Unique	Only the following are acted upon:
CAM defined	SCSI AEN
Asynchronous Callback Pointer	Unused by the adapter.
Peripheral Driver Buffer Pointer	Unused.
Size of Allocated Peripheral Buffer	As specified in the CAM spec.
Pdrv_BSD_0	BSD for start of buffer.
Pdrv_BSD_1	BSD for remaining part of buffer if the buffer <del>crosses a page boundary</del> is not physically contiguous.

### E.1.8 Path Inquiry Command

CAM Flags - None defined.

OSD Usage - None defined.

### E.1.9 Enable LUN Command

**Table E-4 Enable LUN Command Fields**

Field	Usage
CAM Flags (OSD)	As defined in CAM spec, except byte 3 bits are ignored as all data pointers are specified via BSDs in the Private Data area.
Group 6 Vendor Unique CDB Length	As specified in CAM spec.
Group 7 Vendor Unique CDB Length	As specified in CAM spec.
Pointer to Target CCB List	Unused
Number of Target CCBs	Unused
Address of first CCB Queue Carrier	SIMport defined field that replaces the previous two and contains the physical address of the first queue carrier in a linked list of queue carriers for the CCBs to be used.



## Host-Adapter polling mode

This appendix explains how a SIM driver can poll adapter registers in lieu of interrupts. It typically is used during system initialization; although, polling mode can be used during normal adapter operation.

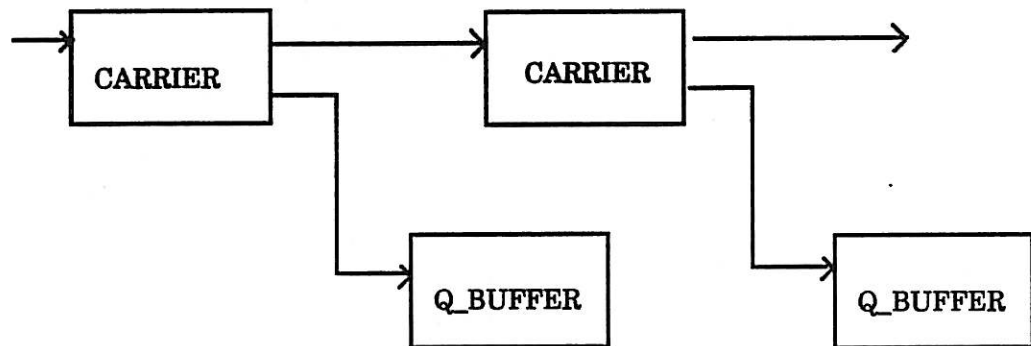
The method consists of the adapter setting bits in the Adapter Status Register (ASR) and the host reading or polling the ASR. The ASR contains three types of mutually exclusive information. The ASR has one bit, the UNIN bit, that indicates that the adapter is in the uninitialized state and no errors have been detected. In this state, the host must complete adapter initialization by writing the ABBR. Another ASR bit is the ASIC bit. The ASIC bit, when set, indicates that a queue element has been placed on the ADRQ and no errors have been detected. The host should examine the ADRQ for messages from the adapter. If neither of these bits are set, then the third type of information is reported in the ASR, adapter errors. All adapter errors reported in the ASR result in the transition of the adapter to an uninitialized error state. In this state, the host is required to reset and re-initialize the adapter. To facilitate the host in detecting an adapter ASR reported error, the adapter is required to clear the ASIC bit, when setting any other bit in the ASR. Thus subsequent to adapter initialization, either the ASIC bit in the ASR is set or an error bit is set in the ASR. Note that it is possible for queue elements to be placed on the ADRQ and an adapter error be reported in the ASR. Thus, after detecting an ASR reported error, the host should still process the ADRQ.

## G.1

## Queue Structure Overview

Each command or response queue entry consists of a CARRIER and a Q\_BUFFER as shown in figure. The carrier links the information contained in the q\_buffer into a singly linked queue. The q\_buffer contains port specific control information.

Queue entries are access in strict FIFO order. Each queue is accessed by one and only one remover and one and only one inserter. Entries are removed from the queue head; entries are inserted at the queue tail.

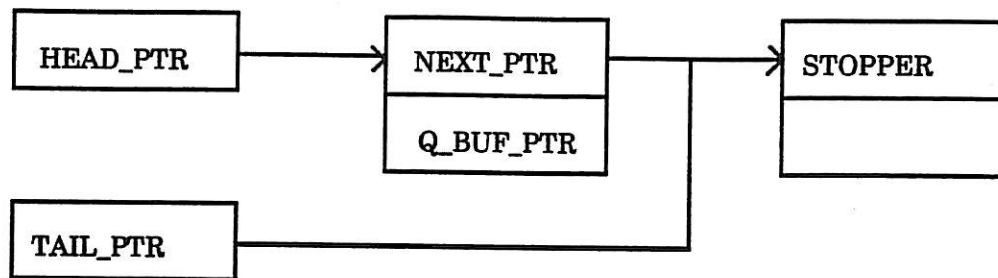


**Figure G-1 Queue Carriers and Queue Buffers**

Carrier and q\_buffer structures are separated to permit independent reuse. This separation is a performance consideration which results from the queue access patterns. Carriers may be physically located in host memory or in memory local to the adapter. Q\_buffers must be located in host memory.

## G.1.1 Queue Mechanics

A queue consists of a queue header, some number of carrier-q\_buffer entries, and a termination carrier or "stopper". The queue header consists of a pointer to the first queue entry carrier (HEAD\_PTR) and a pointer to the stopper (TAIL\_PTR). Each carrier contains a pointer to the next carrier in the queue (NEXT\_PTR) and a q\_buffer pointer (Q\_BUF\_PTR). A stopper is always present; the HEAD\_PTR and TAIL\_PTR are never null. The stopper carrier NEXT\_PTR is simply distinguishable.



**Figure G-2 Queue Structure**

Entries are removed from the queue head. To removed an entry, the remover executes the following code fragment:

```

#define STOPPER

void remove_entry (head_ptr  : *carrier;
                  car_ptr   : *carrier;
                  q_b_ptr   : *q_buffer)
/*
 *remove_entry removes an entry from the queue head
 */
/* Returns:   head_ptr   contains a pointer to the updated queue
 *              head.
 *              car_ptr   contains a pointer to the removed
 *              carrier or the stopper value.
 *              q_b_ptr   contains a pointer to the removed
 *              q_buffer. Valid only if car_ptr
 *              contains a pointer to a carrier and not
 *              the stopper value.
 */
{
    *car_ptr = head_ptr;          /* return carrier or stopper */
    if (car_ptr->next_ptr != STOPPER)
    {
        *q_b_ptr = car_ptr->q_buf_ptr; /* return q_buffer */
        *head_ptr = car_ptr->next_ptr; /* update head link */
    };
};
  
```

Entries are inserted into the queue by adding a new stopper and transforming the old stopper into an active entry. To insert an entry, the inserter executes the following (simplified) code fragment:

```
#define STOPPER

void insert_entry (tail_ptr : *carrier;
                  car_ptr : *carrier;
                  q_b_ptr : *q_buffer)
/* insert_entry inserts an entry at the queue tail.
 *
 * Inputs:      car_ptr      contains a pointer to the new carrier.
 *              q_b_ptr      contains a pointer to the q_buffer to be
 *                          inserted.
 *
 * Returns:     tail_ptr      contains a pointer to the updated queue
 *                          tail.
 */
{
    car_ptr->next_ptr = STOPPER; /* carrier becomes STOPPER */
    tail_ptr->q_buf_ptr = q_b_ptr; /* link q_buffer into old STOPPER */
    tail_ptr->next_ptr = car_ptr; /* link in new STOPPER */
    *tail_ptr = car_ptr; /* update tail link */
}
```

Note that while the queue is shared by the inserter and remover, the inserter access only the TAIL\_PTR and the remover accesses on the HEAD\_PTR. This single access pattern has two significant benefits.

- The two pointers need not be in the same frame of reference. For example, the HEAD\_PTR could be a virtual address while the TAIL\_PTR was a physical address.
- After port initialization, the inserter and remover use separate, private, pointer copies. The adapter should maintain command queue HEAD\_PTRs and response queue TAIL\_PTR in local adapter memory. This directly reduces the adapter cost of updating the tail link (by bringing the link closer to the adapter) and indirectly reduces the CPU-I/O cache contention (by reducing the shared cache blocks).

Carriers and q\_buffers are separate to reduce memory-to-memory copies. If the carrier and q\_buffer were combined into a single queue entry structure, the inserter would have to copy the full q\_buffer contents into the (old) stopper entry. With distinct carriers and q\_buffers, the inserter need only update the q\_buffer pointer in the (old) stopper carrier. This is a particular advantage for command q\_buffers which are reused either on an internal command queue or on the response queue. The q\_buffer contents may be updated in place, e.g. only the STATUS need be changed. The adapter access of main memory are minimized.

## G.1.2 Addressing

Carrier and q\_buffer structures are addressed virtually by the port driver and physically by the adapter. Each structure contains a stored TOKEN field which contains the address of that structure in the port driver's virtual address space.

The token is initialized by the port driver when the structure is allocated. The stored token becomes a pointer to the structure when the adapter queues the structure to the port driver. The token field is readonly to the adapter; the adapter copies, but never alters, the token. Furthermore, the adapter needs no knowledge of the port driver's interpretation of the token.

This dual addressing scheme and token manipulation leads to some additional steps and variations in the simple `INSERT_ENTRY` procedure presented in Section G.1.1. There is no change to the `REMOVE_ENTRY` procedure. The inserter always constructs the linkage pointers to be in the remover's address space. The two queue insertion variants are presented in Section G.3

## G.2

### Carrier Structure

Each carrier is a naturally aligned hexaword as shown below.

63		0
	NEXT_PTR	A
	Q_BUF_PTR	A+8
	Q_BUF_TOKEN	A+16
	CAR_TOKEN	A+24

Figure G-3 Carrier Format

Table G-1 Carrier Structure

Quadword	Name	Description
A	NEXT_PTR	Pointer to next carrier or stopper indicator.
A+8	Q_BUF_PTR	Pointer to Q_BUFFER. Q_BUF_PTR is UNDEFINED when the carrier is a stopper.
A+16	Q_BUF_TOKEN	Q_BUFFER port driver address. Q_BUF_TOKEN is UNDEFINED when the carrier is a stopper or when the carrier is an entry on an adapter-driver queue.
A+24	CAR_TOKEN	Carrier port driver address.

The NEXT\_PTR and Q\_BUF\_PTR fields are interpreted by the adapter only if the field is known to contain a physical address. Fields containing driver (virtual) address are never interpreted by the adapter. The format of pointer fields containing physical address is shown below.

63	48	47		5	4		1	0
Reserved	PA<47:5>						Reserved	S

Figure G-4 Carrier Physical Address Pointer Fields

**Table G-2 Carrier Physical Address Pointer Fields**

Bits	Name	Description
<0>	S	Stopper indicator. For NEXT_PTR fields, used to determine whether or not the carrier is a stopper. The stopper indicator value depends on the type of queue; see below. SBZ for Q_BUF_PTR fields.
<4:1>		Reserved
<47:5>	PA	Bits <47:5> of the physical address.
<63:48>		Reserved

The value of NEXT\_PTR<0> which indicates a stopper depends on the type of queue. Non-stopper entries on driver-adapter queues have NEXT\_PTR<0> = 1; stopper entries have NEXT\_PTR<0>=0. Non-stopper entries on adapter-driver queues have NEXT\_PTR<0>=0; stopper entries have NEXT\_PTR<0>=1. This permits both the port driver and the adapter to detect the stopper by a simple bit test rather than a full 64 bit compare. This also permits the adapter to reuse a carrier as a stopper without modifying the carrier.

## G.3

### Queue Insertion Procedures

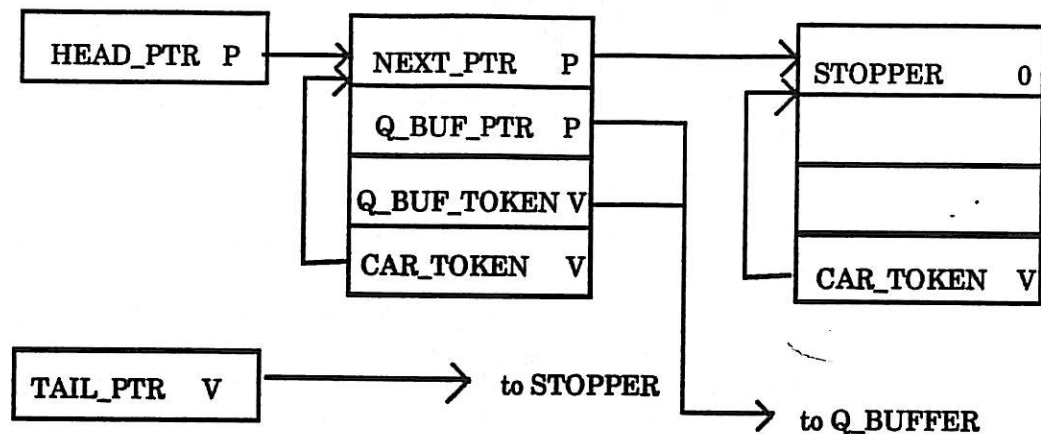
This section contains the detailed queue insertion procedures. Two variants are presented: driver-adapter, and adapter-driver. The differences between the variants are due to different write synchronization and visibility properties.

The procedures allow memory reads and writes to be reordered in the absence of explicit instructions to the contrary. There is no implicit ordering between different entities (processors and I/O adapters). The ONLY way for two entities to communicate reliably is for one entity to write a new value to a share location, and for another entity eventually to read that location and get the new value. All implementations guarantee that a written value will eventually become available to the other entity (no total starvation of writes), but there is no time bound on how long the value may be delayed in transit.

There is a data operation ordering primitive called a Memory Barrier or MB. MB orders the memory access (reads and writes) of a single entity and guarantees that all prior reads and writes are completed before any following reads and writes are initiated. Local I/O space accesses, e.g. device register accesses, are not implicit MBs.

#### G.3.1 Driver-Adapter Queues

Entries on the driver-adapter queues are inserted by the port driver and removed by the adapter or channel. The HEAD\_PTR and all NEXT\_PTR links are physical addresses; the TAIL\_PTR is a virtual address. The Q\_BUF\_PTR is a physical address pointer. The stored TOKENs are virtual addresses for use by the adapter, see section G.3.2. Valid carriers have NEXT\_PTR<0> = 1; the stopper carrier has NEXT\_PTR<0>= 0;



**Figure G-5 Driver-Adapter Queue Structure**

To enqueue an entry to the adapter, the port driver executes the following:

```

void driver_insert(tail_ptr; *carrier;
                  car_ptr; *carrier;
                  q_b_ptr; *q_buffer;
                  csr_ptr; *csr_block)
{
    car_ptr->next_ptr = 0;          /* carrier becomes stopper */
                                    /* clear <0> to make stopper */
    car_ptr->car_token = car_ptr; /* token gets VA of carrier */
    q_b_ptr->token = q_b_ptr;      /* token gets VA of q_buffer */
    tail_ptr->q_buf_ptr = to_physical(q_b_ptr);
                                    /* link q_buffer into stopper */
    tail_ptr->q_buf_token = q_b_ptr;
                                    /* Token gets VA of q_buffer */
    mb();                          /* Order writes */
    tail_ptr->next_ptr = to_physical(car_ptr) + 1;
                                    /* link in new stopper */
                                    /* set <0> to make valid */
    mb();                          /* Order writes */
    write_io(csr_ptr->q_c, to_physical(car_ptr)); /* signal adapter */
    *tail_ptr = car_ptr;
}

```

All of the above pointers are assumed to be physical addresses. The `to_physical` function is assumed to return the physical address associated with the specified virtual address. The `csr_ptr->q_c` points to the appropriate command queue insertion register or adapter free queue insertion register. The `write_io` function causes a quadword I/O space write of `to_physical(car_ptr)` to `csr_ptr->q_c`.

The port driver must issue two MBs when enqueueing an entry to the adapter. The first MB ensures that the `q_buffer` contents and `q_buf_ptr` in the (old) stopper carrier will become visible to the adapter before the `NEXT_PTR` becomes valid (`NEXT_PTR<0>=1`). Without this MB, the adapter could access the old `q_buffer` contents. The second MB ensures that the updated valid `NEXT_PTR` will become visible to the adapter before the adapter is notified of the new entry by the `csr_ptr->q_c` write. Without the second MB, the adapter could fail to access the updated carrier contents, observe an invalid `NEXT_PTR`, and infer that the queue was empty.

The port driver write of the queue insertion register serves a dual purpose. First, the write itself notifies the adapter of a newly inserted queue entry. Second, the data written

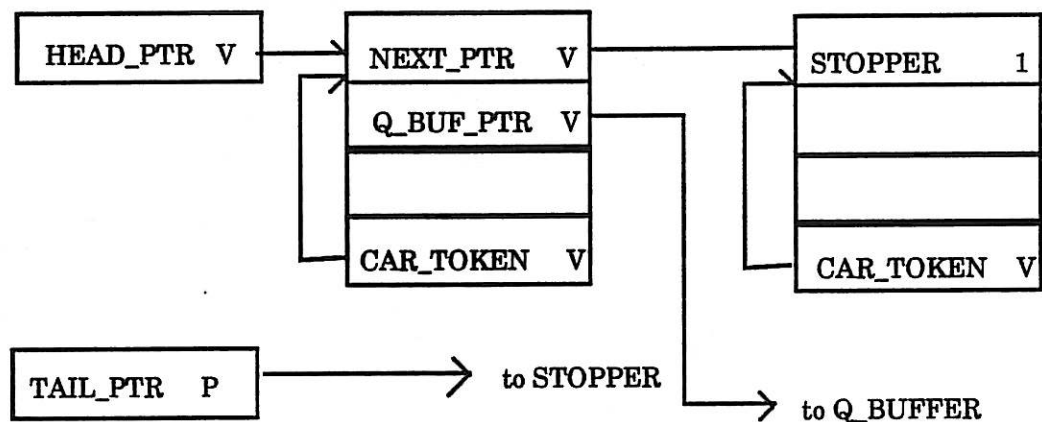


enables the adapter to determine the tail of the queue without actually accessing the stopper carrier. The port driver writes the physical address of the new stopper carrier into the queue insertion register. When the adapter HEAD\_PTR is equal to the register contents, the queue contains only a stopper carrier. Note that while the adapter does not necessarily access the stopper, the stopper is necessary to enable the port driver to insert further queue entries.

### G.3.2 Adapter-Driver Queues

Entries on the adapter-driver queues are inserted by the adapter and removed by the port driver.

The HEAD\_PTR and all NEXT\_PTR links are virtual addresses; the TAIL\_PTR is a physical address. The Q\_BUF\_PTR is a virtual address pointer. The stored TOKENs are used to construct the virtual pointers. Valid carriers have NEXT\_PTR<0>=0; the stopper carrier has NEXT\_PTR<0>=1. This permits the adapter to reuse a carrier as a stopper without modifying the carrier.



**Figure G-6 Adapter-Driver Queue Structure**

To enqueue an entry to the port driver, the adapter executes the following:

```

void adapter_insert(tail_ptr; *carrier;
                   car_ptr; *carrier;
                   q_b_ptr; *q_buffer)
{
    tail_ptr->q_buf_ptr = q_b_ptr->token; /* old stopper gets stored */
                                           /* q_buffer VA */
    mb(); /* order writes */
    tail_ptr->next_ptr = car_ptr->car_token;
                                           /* old stopper gets stored */
                                           /* carrier VA */
    *tail_ptr = car_ptr; /* update tail link with */
                        /* carrier PA */
    request_interrupt(); /* conditionally request */
                        /* completion interrupt */
}
  
```

All the above pointers are physical addresses.

The adapter must issue the equivalent of one MB when enqueueing an entry to the port driver. The MB ensures that the q\_buffer contents and the Q\_BUF\_PTR in the old stopper carrier are visible to the port driver before the NEXT\_PTR becomes valid (NEXT\_PTR<0>=0). If the processor platform and adapter implementations support



atomic hexaword writes, the MB is generated by the adapter between the DMA of the updated `q_buffer` contents and the DMA of the updated carrier.

If an interrupt is requested, a MB must be generated prior to issuing the interrupt request to ensure that all queue entries will be visible to the port driver interrupt service routine. If the processor guarantees that all pending DMA writes have become visible to the processor prior to requesting an interrupt, the MB need not be generated by the adapter.

## **G.4**

### **Free Queues**

The allocation, deallocation, and initialization of free queue entries is the responsibility of the port driver. The exchange of free queue entries between the port driver and the adapter is also controlled by the port driver. Entries may be allocated an/or returned in groups.

The free queue entries are managed internally by the adapter. The adapter may, but does not need to, use queues to manage the carriers and `q_buffers`. The port driver allocates stoppers to enable the adapter to construct queues to manage queue entries. At port initialization the free queue fields in the adapter block each contain a `HEAD_PTR` and `TAIL_PTR` which locate a stopper carrier with `NEXT_PTR < 0` = 0.

## **G.5**

### **Recovering Queue Buffers and Carriers on Adapter Crash**

If the adapter should crash during operation the host will want to recover the queue buffers and queue carriers. The queue carriers and buffers on the Adapter-Driver Queues can be easily recovered by traversing the queue. Queue carriers and buffers on Driver-Adapter queues can be recovered if the host allocates a large block of memory (one per adapter) and divides it up into queue carriers. If the adapter should crash, then the entries on the Adapter-Driver Queues are removed in the normal manner and the carriers are marked as not being on any queue (e.g. setting the `next_ptr` field to 0). Then the host can go through the pool of carriers and correctly distinguish which carriers point to a queue buffer, which are stopper carriers, and which carriers are free. It is necessary to clear the Adapter-Driver Queues first as it otherwise it is not possible to determine which carriers are are stopper carriers as the sense of the stopper bit depends upon which queue it is on.

Another way to determine what the sense of the stopper carrier is if the operating system's virtual addresses do not have all zeros in bits <63:48>. This area is defined to be zero for physical addresses (see table G-2); allowing the sense of the stopper bit to be determined.

# SIMport Data Structures

```

#ifndef SIMPORT_H
#define SIMPORT_H
/*****
** Definition MODULE simport.h
**
** -----
** Abstract:
**
**   {text}
**
** History:
**
**   Modification Date   Person           Description of modification
**   15-JUL-1992        Mike Mackay       Provided to SIMport architect
**   16-JUL-1992        Rich Whalen       Updated Queue Carrier format
**                                           Adapter State Set format
**   24-AUG-1992        Rich Whalen       Corrected typedef syntax,
**                                           utiny changed to u_char
**   26-OCT-1992        Rich Whalen       Miscellaneous changes to insure
**                                           agreement with SIMport architecture
**                                           document
**
**
**
** *****/
**/

```

```

/*****
**  MODULE DATA STRUCTURE:  Miscellaneous Structures
**
**      The following are miscellaneous structures used in the SIMport
**      modules.
**-----
**  DESCRIPTION:
**
**
*****/

/*
** Host Virtual Address
*/
Struct VA
{
    ulong    lo;
    ulong    hi;
};

/*
** Host Physical Address
*/
Struct PA
{
    ulong    lo;
    ulong    hi;
};

/*
** unsigned 64 bit integer type
*/
typedef struct
{
    ulong    lo;
    ulong    hi;
} uxlong;

/*
** Compressed version of Execute SCSI IO (part of the QProcSlot)
*/
struct PackedSCSIIO
{
    ulong    DataXferLen;
    u_char   SenseBuffLen;
    u_char   CDBLen;
    u_char   CDB[12];
    ulong    CmdTimOut;
    ushort   VUFlags;
    u_char   QTag;
};

```

```

/*****
**  MODULE DATA STRUCTURE:  QueueCarrier
**
**      The QueueCarrier is the SIMport defined data structures used
**      to form the interface queues between the host and the adapter.
**-----
**  DESCRIPTION:
**
**      The QueueCarrier is used to pass queue buffers between the
**      host and the adapter. It contains a link field which is used
**      to form a one-way linked list. It contains a pointer to a
**      queue buffer which must be formatted as a CAM or SIMport CCB
**      command. It also contains queue buffer and queue carrier token
**      fields which are host virtual addresses. In addition, for
**      adapter internal use only, the CarPtr field is the physical
**      address of the queue carrier. Note that the CarPtr field is
**      not part of the host resident queue carrier.
*****/

/*
** SIMport queue carrier, with additional field for PA of carrier
**/
struct QueueCarrier
{
    struct PA      NextPtr;
    struct PA      QBufPtr;
    struct VA      QBufToken;
    struct VA      CarToken;
    struct
    {
        u_char      FuncCode;
        u_char      status;
        struct
        {
            bfield   SBZpadding: 14;
            bfield   QueuePriority: 1;
            bfield   ReturnStatusFormat: 1;
        }          flags;
        struct ConnID CID;
    };
};

```

```

/*****
**  MODULE DATA STRUCTURE:  CAMHeader, ConnId
**
**      The CAMHeader is part of a CAM defined CCB command. It defines
**      the fields common to all CCB commands. The ConnId is a field
**      in the CCBHeader used to address a bus, device, and LUN.
**-----
**  DESCRIPTION:
**
**      The CCBAddr is the host virtual address of the CCB. The CCBLen
**      is the length in bytes of the CCB. The function is the
**      function code for the command or message. The CAMStatus is
**      where the command complete status is posted. The CID contains
**      the target address for the command. The CAMFlags is a set of
**      command specific flags. Refer to the CAM specification for
**      additional details.
*****/

/*
** CAM Command Header
*/
struct CAMHeader
{
    /*
    ** note that the CCBAddr and the CCBLen is not required
    ** when the CCBHeader resides in a QProcSlot or an ImmedProcSlot.
    */
    struct VA      CCBAddr;
    ushort        CCBLen;
    u_char        function;
    u_char        CAMStatus;
    struct ConnID  CID;
    ulong         CAMFlags;
};

/*
** CAM Command Header Connect ID
*/
struct ConnID
{
    u_char        rsvd;
    u_char        channel;
    u_char        target;
    u_char        lun;
};

```

```

/*****
**  MODULE DATA STRUCTURE:  SIMport Command/Message Formats
**
**      The following structures define the SIMport command and
**      message formats.  SIMPortBodies is a union of all the SIMport
**      specific command and message structure definitions.
**      Subsequently, are the definition of each SIMport command and
**      message.  Included are parameter definitions where appropriate.
**-----
**  DESCRIPTION:
**
**      The fields in the SIMPortBodies union are the
**      SIMport defined command and message structure definitions.  The
**      union is used in the ScratchCmdBuff structure and in the
**      ImmedProcSlot structure.  Each additional structure defines the
**      CCB body format of a SIMport defined command or message.  Refer
**      to the SIMport specification for additional details.
*****/

/*
** SIMport vendor unique function codes definitions
*/
#define SPO_VU_SET_ADAP_STATE      80
#define SPO_VU_SET_PARAM          81
#define SPO_VU_SET_CHAN_STATE     82
#define SPO_VU_SET_DEV_STATE      83
#define SPO_VU_VER_ADAP_SANITY    84
#define SPO_VU_READ_CNTRS         85
#define SPO_VU_BSD_REQUEST        86
#define SPO_VU_BUS_RESET_REQ      87
#define SPO_VU_UNSol_RESEL        88
#define SPO_VU_CHAN_DISABLED      89
#define SPO_VU_DEV_DISABLED       8A

/*
** SIMport Command Bodies
*/
union SIMPortBodies
{
    SPO_SetAdapStateType      SetAS;
    SPO_AdapStateSetType      ASSet;
    SPO_SetParamType          SetP;
    SPO_ParamSetType          PSet;
    SPO_SetChanStateType      SetCS;
    SPO_ChanStateSetType      CSSet;
    SPO_SetDevStateType       SetDS;
    SPO_DevStateSetType       DSSet;
    SPO_VerAdapSanityType     ASanity;
    SPO_AdapSanityVerType     SanityVer;
    SPO_ReadCntrsType         ReadC;
    SPO_CntrsReadType         CRead;
    SPO_BSDRequestType        BSDReq;
    SPO_BSDRespType           BSDResp;
    SPO_BusResetReqType       BusResetReq;
    SPO_BusResetRespType      BusResetResp;

```

```

        SPO_UnsolReselType      UnResel;
        SPO_ChانDisabledType    ChanDis;
        SPO_DevDisabledType     DevDis;
};
/*
** SIMport Set Adapter State command
*/
typedef struct
{
    u_char    state;
    u_char    SBZ_1;
    ushort    flags;
    u_char    SBZ_2[4];
} SPO_SetAdapStateType;
/*
** SIMport Adapter State Set response message
*/
typedef struct
{
    u_char    state;
    u_char    NHostSg;
    struct
    {
        bfield    IntHoldOff: 1;
        bfield    ChanNode: 1;
        bfield    LinkBSM: 1;
        bfield    BSDReq: 1;
        bfield    TgtProcessor;
        bfield    TgtPhaseCog;
        bfield    unused: 10;
    }
    ushort    NAdapQC;
    u_char    KATime;
    u_char    NFreeQ;
    u_char    NChan;
    u_char    XferAlign;
    u_char    nAENbuf;
    u_char    SBZpadding[5];
    u_char    nodes_on_chan[16];
} SPO_AdapStateSetType;

/*
** values for AdapStateSet status field
*/
#define SPO_SUCCESS      1
#define SPO_INV_COMMAND  0
#define SPO_HOST_MEM     -3
#define SPO_HOSTFQE      -4

/*
** values for the AdapStateSet XferAlign field
*/
#define SPO_XFER_ALIGN_BYTE      0
#define SPO_XFER_ALIGN_WORD      1
#define SPO_XFER_ALIGN_LONGWORD  2
#define SPO_XFER_ALIGN_QUADWORD  3
#define SPO_XFER_ALIGN_OCTAWORD  4

```

```

/*
** SIMport Set Parameter command
*/
typedef struct
{
    u_char          SBZ_1;
    u_char          NHostSg;
    struct
    {
        bfield      EnableCounters: 1;
        bfield      unused: 15;
    } flags;
    ulong           SystemTime;
    ulong           RPTimer;
    ulong           IntHoldOffTimer;
    struct BSD      HostSgBSD;
} SPO_SetParamType;
/*
** SIMport Parameter Set response message
*/
typedef struct
{
} SPO_ParamSetType;

/*
** values for ParamSet status field
*/
#define SPO_SUCCESS          1
#define SPO_INV_COMMAND      0
#define SPO_NOT_DISABLED     -2
#define SPO_INT_HOLD OFF    -5
#define SPO_HOST_SG         -6
/*
** SIMport Set Channel State command
*/
typedef struct
{
    u_char          state;
    u_char          SBZ_1;
    ushort          SBZ_flags;
    u_char          ChanID;
    u_char          NodeID;
    u_char          SBZpadding[2];
} SPO_SetChanStateType;
/*
** SIMport Channel State Set response message
*/
typedef struct
{
    u_char          state;
    u_char          SBZ_1;
    ushort          SBZ_flags;
    u_char          ChanID;
    u_char          NodeID;
    u_char          SBZpadding[2];
} SPO_ChanStateSetType;

```



```

/*
** values for ChanStateSet status field
*/
#define SPO_SUCCESS 1
#define SPO_INV_COMMAND 0
#define SPO_NOT_ENABLED -1
#define SPO_INV_CHAN_ID -7
#define SPO_INV_NODE_ID -8
/*
** SIMport Set Device State command
*/
typedef struct
{
    u_char state;
    u_char SBZ_1;
    ushort SBZ_flags;
    u_char SBZ_2[4];
} SPO_SetDevStateType;
/*
** SIMport Device State Set response message
*/
typedef struct
{
    u_char state;
    u_char SBZ_1;
    ushort SBZ_flags;
    u_char SBZ_2[4];
} SPO_DevStateSetType;

/*
** values for DevStateSet status field
*/
#define SPO_SUCCESS 1
#define SPO_INV_COMMAND 0
#define SPO_NOT_ENABLED -1
#define SPO_CANT_DISABLE -9
/*
** SIMport Verify Adapter Sanity command
*/
typedef struct
{
} SPO_VerifyAdapSanityType;
/*
** SIMport Adapter Sanity Verified response message
*/
typedef struct
{
} SPO_AdapSanityVerType;
/*
** SIMport Read Counters command
*/
typedef struct
{
    u_char SBZpadding[ 2];
    struct
    {
        bfield ZeroCounters: 1;
    }

```

```

        bfield    unused:15;
    }            flags;
    u_char        SBZ_2[4];

} SPO_ReadCnttrsType;
/*
** SIMport Counters Read response message
*/
typedef struct
{
    u_char        SBZpadding[2];
    ushort        SBZ_flags;
    ulong         TimeSinZero;
    ulong         BusFaults;
    ulong         SCSIComSent;
    ulong         SCSIComRecv;
    ulong         DataSent;
    ulong         DataRecv;
    ulong         SCSIBusResets;
    ulong         BDRsSent;
    ulong         SelTimeouts;
    ulong         ParityErrs;
    ulong         UnsolReselect;
    ulong         BadMessages;
    ulong         MessReject;
    ulong         UnexpDiscon;
    ulong         UnexpPhase;
    ulong         SyncViol;
    ulong         ExpansionSpace[5];
    ulong         ImpSpecific[8];
} SPO_CnttrsReadType;
/*
** Values for Counters Read Status field
*/
#define          SPO_NOT_ENABLED -1

/*
** SIMport BSD Request request message
*/
typedef struct
{
    u_char        BufID;
    u_char        SBZpadding[1];
    ushort        SBZ_flags;
    ulong         offset;
    struct VA      CCB_VA;
} SPO_BSDReqType;
/*
** SIMport BSD Response command
*/
typedef struct
{
    u_char        BufID;
    u_char        SBZpadding[1];
    ushort        SBZ_flags;
    ulong         offset;
    struct VA      CCB_VA;
}

```

```

        struct BSD                      BSDBlock[12];
    } SPO_BSDRespType;

/*
** values for BSDReq status field
*/
#define SPO_SUCCESS          1
#define SPO_INV_COMMAND      0
#define SPO_INV_OFFSET      -10
#define SPO_INV_CCB          -11
#define SPO_INV_BUF_ID      -12

/*
** values for BSDReq and BSDReqp BufID field
*/
#define SPO_BUFID_DATA       0
#define SPO_BUFID_CDB        1
#define SPO_BUFID_SENSE      2
#define SPO_BUFID_MESSAGE    3

/*
** SIMport Bus Reset Request request message
*/
typedef struct
{
    u_char          reason;
    u_char          TargetID;
    u_char          SBZ_flags[2];
    u_char          SBZ_padding[4];
    struct PA       CCB;
} SPO_BusResetReqType;

/*
** values for the BusResetReq reason field
*/
#define SPO_REJECT_BDR       1
#define SPO_PHASE_ERROR      2
#define SPO_DATA_OUT         3
#define SPO_RSVD_PHASE       4
#define SPO_NO_MSG_OUT       5

/*
** SIMport Bus Reset Response command
*/
typedef struct
{
} SPO_BusResetRespType;

/*
** values for BusResetResp status field
*/
#define SPO_SUCCESS          1
#define SPO_INV_COMMAND      0
#define SPO_NO_RESET         -13

/*
** SIMport Unsolicited Reselection message
*/
typedef struct
{

```

```

        u_char                SBZpadding[6];
    } SPO_UnsolReselType;

/*
** values for UnsolResel status field
*/
#define          SPO_UNSol_RESEL          1
/*
** SIMport Unsolicited Channel Disable message
*/
typedef struct
{
    u_char                SBZpadding[6];
} SPO_ChanDisabledType;

/*
** values for ChanDisabled status field
*/
#define          SPO_BUS_RESET            2
#define          SPO_RESET_REJECT        3
/*
** SIMport Unsolicited Device Disable message
*/
typedef struct
{
    u_char                SBZpadding[6];
} DevDisabledType;

/*
** values for DevDisabled status field
*/
#define          SPO_DEVICE_RESET        4

```

```

/*****
**
**  MODULE DATA STRUCTURE:  PrivateDataArea
**
**    The Private Data Area is part of a CAM defined Execute SCSI I/O
**    CCB command. It is used to pass SIMport specific information
**    about the SCSI I/O command.
**-----
**  DESCRIPTION:
**
**    The PrivateDataArea is part of an Execute SCSI I/O command.
**    The format of the PrivateDataArea is specified by SIMport. It
**    is passed to the adapter as part of the body of an Execute
**    SCSI I/O command. It exists in the adapter as part of the
**    QProcSlot (QPS) data structure. Refer to the SIMport
**    specification for addition details.
*****/

/*
** The SIMport Private Data Area of Exec SCSI IO CCB (part of QPS)
**/
struct PrivateDataArea
{
    u_char          SCSIStatus;
    u_char          SenseRes;
    u_char          SBZFiller[2];
    ulong           DataRes;
    struct BSD      StartDataBSD;
    struct BSD      EndDataBSD;
    struct BSD      CDBBSD;
    struct BSD      SenBSD;
    struct BSD      MsgBSD;
    struct PA       CCBNextPtr;
};

```

```

/*****
**  MODULE DATA STRUCTURE:  SPO_AdapState
**
**      The SPO_AdapState is a state variable which maintains the
**      current state of the adapter.
**-----
**  DESCRIPTION:
**
**      The primary (or steady) adapter states are UNINITIALIZED,
**      DISABLED, and ENABLED. During adapter initialization and reset
**      the following intermediate states are used SPO_WAIT_ABBR and
**      SPO_NEW_ABBR.
*****/

```

```

u_char    SPO_AdapState;

```

```

#define     SPO_UNINITIALIZED    0
#define     SPO_WAIT_ABBR        1
#define     SPO_NEW_ABBR         2
#define     SPO_DISABLED         3
#define     SPO_ENABLED          4

```

```

/*****
**  MODULE DATA STRUCTURE:  AdapBlock
**
**      The AdapBlock is an adapter resident copy of the host resident
**      SIMport defined Adapter Block.
**-----
**  DESCRIPTION:
**
**      The AdapBlock contains initialization parameters and the links
**      for the SIMport interface queues between the adapter and the
**      host.
**      QBSIZE is the number of bytes in a SIMport queue buffer.
**      CCBPtrSize, is the number of bytes in a SIMport host pointer
**      field.
**      The SIMportQueue structures contain the head and tail queue
**      carrier addresses for the queues.
*****/

```

```

struct AdapBlock
{
    ushort          QBSIZE;
    u_char          CCBPtrSize;
    u_char          SBZFiller[5];
    struct SIMportQueue  DACQ;
    struct SIMportQueue  ADRQ;
    struct SIMportQueue  DAFQ;
    struct SIMportQueue  ADFQ;
};

struct SIMPortQueue
{
    struct PA      Head;
    struct PA      Tail;
};

```

```

/*****
**  MODULE DATA STRUCTURE:  ASRBitNumbers
**
**      The ASRBitNumbers defines the names for each bit in the ASR
**      register.
**-----
**  DESCRIPTION:
**
**      Note that the ASR register is a 64 bit register. Currently
**      SIMport does not define bits 32-63. For additional details
**      about the ASR register, refer to the SIMport specification.
*****/

```

```

/*
** The following define the symbols for the bit numbers in the ASR
** register.
*/

```

```

#define SPO_ASR_0      0
#define SPO_ASR_1      1
#define SPO_ASR_ADSE   2
#define SPO_ASR_AMSE   3
#define SPO_ASR_AAC     4
#define SPO_ASR_ASTE   5
#define SPO_ASR_6      6
#define SPO_ASR_7      7
#define SPO_ASR_ASIC    8
#define SPO_ASR_UNIN    9
#define SPO_ASR_AME    31

```

```

/*****
**  MODULE DATA STRUCTURE:  AFPReg
**
**      The AFPReg structure defines the format of the Adapter Failing
**      Parameter Register.
**-----
**  DESCRIPTION:
**
**      The ErrorNum field is extracted from the ErrorCode data
**      structure. The remaining fields are passed to the SPO_Error
**      routine in the EParamBlock.
*****/

```

```

struct AFPReg
{
    u_char      ErrorNum;
    u_char      param1;
    u_char      param2;
    u_char      param3;
    ulong       param4;
};

```



```

/*****
**  MODULE DATA STRUCTURE:  BSD
**
**      The BSD (Buffer Segment Descriptor) defines either a host
**      memory segment or a BSM.
**-----
**  DESCRIPTION:
**
**      The BSD is a data structure used to contain the physical
**      address and size of a contiguous segment of host memory. The
**      BPType flag indicates whether the format of the memory segment
**      is that of a host memory segment or of a SIMport Buffer
**      Segment Map (BSM) . A BSM is a list of BSDs and is used to
**      address a host buffer that consists of multiple host memory
**      segments. The ByteCount is the number of bytes in the segment
**      or in the BSM. The remaining fields contain the host address
**      bits of the memory segment or of the BSM.
*****/

/*
** SIMport Buffer Segment Descriptor
**/
struct BSD
{
    union
    {
        ulong      BP;          /* access BufPtrLo as 32 bit entity */
        struct
        {
            bfield  BPType: 2;    /* access BufPtrLo type bits only */
            bfield  BPBits: 30;   /* access BufptrLo ptr bits only */
            b;
        }
        BufPtrLo;    /* Lower 32 bits of Buf_Ptr field */
        ushort      BufPtrHi;    /* Upper 16 bits of the Buf_Ptr field */
        ushort      ByteCount;   /* Number of bytes in the segment or BSM */
    };
};

```

```

/*****
**  MODULE DATA STRUCTURE:  BSM
**
**      A BSM (Buffer Segment Map) is a SIMport data structured which
**      is used to define a host memory buffer which consists of
**      multiple segments and thus requires multiple BSDs to define.
**-----
**  DESCRIPTION:
**
**      The BSM is used to define a physically discontinuous host
**      memory buffer. The NextBSMBSD field is a link to the next BSM,
**      if required. The NumEntries field is a count of the number of
**      valid BSDs in the BSDBlock. The TotalCount field contains the
**      total byte count that this BSM maps. The BufferOffset field
**      contains the buffer offset from the start of the buffer that
**      this BSM maps. The BSDBlock field contains a BSD data
**      structure for each physically contiguous host memory segment.
*****/
**/

/*
** SIMport Buffer Segment Map
**/
struct BSM
{
    struct BSD    NextBSMBSD;        /* link for linked list of BSMs */
    ushort       NumEntries;         /* number of BSDs in BSDBlock */
    u_char       SBZpadding[6];
    ulong        TotalCount;         /* total byte count for this BSM */
    ulong        BufferOffset;        /* buffer offset for this BSM */
    struct BSD    BSDBlock[17];      /* list of buffer segment BSDs */
};

#endif

```

---

# Index

## A

- Abort SCSI I/O, 5-21, E-2
- Adapter
  - Fatal Errors, 3-6
- Adapter Block, H-14
- Adapter diagnostics, 4-1
- Adapter Features, 5-5
- Adapter Node Id, 5-7
- Adapter reset time, 4-1
- Adapter State Set Response, H-6
- Adapter Target Id, 5-7
- Adapter Timeout Period, 4-2
- ASIC Bit, 3-6
- Async Event Notification, 5-22, 5-23, 6-1
- autosense, 5-20

## B

- BSD, 5-13, 5-19, 5-23, 10-1, H-16
  - 64 bit format, D-1
- BSD Request, H-9
- BSD Response, H-9
- BSM, H-17
- Bus Reset Request, H-10
- Bus Reset Response, H-10

## C

- CAM Header, H-4
- CARRIER
  - format, 3-3
- CCB, 2-2, 2-3, 5-2
- ccb\_ptr\_size, 3-2, D-1
- Channel State Set Response, H-7
- Command Abort, 5-20, 5-21
- Connect ID, H-4
- Counters Read Response, H-9

## D

- Device State Set Response, H-8
- Diagnostics, 4-1

## Disabled State

- Adapter, 2-3, 4-2, 4-4, 5-3, 5-17
- Channel, 4-5, 5-3, 5-17, 5-21, 6-2
- Device, 4-5, 5-22, 6-3, C-2

## E

## Enabled State

- Adapter, 2-3, 4-2, 4-4, 5-5
- Enable Counters, 5-7
- Enable LUN, 5-25, 7-1, E-3
- Execute SCSI IO
  - Private Data Area, H-12
- Execute SCSI I/O, 2-4, 5-1, 5-13, 5-18, E-1

## F

- freeze counter, 5-21
- Free Queue Entries, 5-4
- Frozen Queues, 5-21

## H

- host resident buffers, 5-18

## I

- Immediate CCB commands, 2-3, 2-4
- Interrupt Enable Bit, 3-5, 4-1
- Interrupt Holdoff Timer, 2-5, 3-8, 5-5, 5-6

## M

- Maintenance and Performance counters, 5-7, 5-13
- Maintenance Initialize bit, 2-2, 3-5, 4-1
- Memory Barrier, G-5

## O

- Operating with Host Interrupts Disabled, 3-6

## P

- Parameter Set Response, H-7

## Q

- Queued CCB commands, 2-4
- Queued Commands, 2-4
  - Maximum number, 5-4
- Queues
  - Carriers and Buffers, G-1
  - Carrier Contents, G-4
  - Stopper Carriers
    - Adapter-Driver Queues, G-7
    - Driver-Adapter Queues, G-5
  - Structure, G-2
  - Valid Carriers
    - Adapter-Driver Queues, G-7
    - Driver-Adapter Queues, G-5
- Queue Carrier, 2-2, 2-4, H-3
  - Structure, 3-3

## R

- Read Counters Command, H-8
- Release SIM Queue Command, 5-21
- Request Sense, 5-20
- Reset Adapter Command, 2-2
  - completion time, 4-1
- Reset Pending Timer, 5-6, 5-16
- Reset SCSI Bus, 5-21, E-2
  - Request Timer, 5-6
- Reset SCSI Device, 5-22, C-2, C-3, E-2

## S

- Set Adapter State Command, H-6
- Set Channel State Command, H-7
- Set Device State Command, H-8
- Set Parameter Command, H-7
- SIM Queues, 2-4, 5-1
  - dispatch order, 2-4
  - Insertion order, 3-4
- Single Interrupt Support, 3-6, 9-1
- Status Values
  - bus reset, 6-3
  - cant\_disable, 5-10
  - device reset, 6-3
  - host\_fqe, 5-5
  - host\_mem, 5-5
  - inv\_buf\_id, 5-15
  - inv\_CCB, 5-15
  - inv\_chan\_id, 5-9
  - inv\_command, 5-5, 5-7, 5-9, 5-10, 5-12, 5-15, 5-17
  - inv\_host\_sg, 5-7
  - inv\_int\_holdoff, 5-7
  - inv\_node\_id, 5-9
  - inv\_offset, 5-15
  - not\_disabled, 5-7
  - not\_enabled, 5-9, 5-10, 5-12
  - no\_reset, 5-17

- Reset Request Denied, 6-3
  - success, 5-5, 5-7, 5-9, 5-10, 5-12, 5-15, 5-17
- system time, 5-6

## T

- Tagged Queuing, 5-20
- Target mode support, 5-5, 7-1
  - Enable LUN Command, 5-25
- Terminate SCSI I/O, 5-22, 5-23, E-2
- Timers
  - adapter time-out, 4-2
  - interrupt holdoff, 2-5, 3-8, 5-5
  - interval, 2-5
  - Reset Pending, 5-6, 5-16
  - SCSI command time-out, 5-20
- time of day, 5-6
- Transfer Alignment, 5-4, B-1

## U

- UNIN bit, 3-6
- Unsolicited Channel Disable, H-11
- Unsolicited Device Disable, H-11
- Unsolicited Reselection, H-10

## V

- Vendor Unique Function Codes, H-5