December 3, 1990
To : John Lohmeyer, Chairman, X3T9.2
From : Gary R. Stephens
IBM Corporation
D67E/060-1
9000 S. Rita Rd.
Tucson, Az 85719
(602) 799-2246

Subject: Small Computer Fiber Interface (SCFI)

This document begins a new era for SCSI to incorporate function suitable for operation in a Fiber Channel environment and to eliminate some artifacts. This interface is dubbed Small Computer Fiber Interface (SCFI or Scruffy, per Dal Allan). For the present, this document functions as the Fiber Channel FC-4 document mapping SCFI to the Fiber Channel. Extensions, corrections, and deletions have been made to provide the new capabilities of the Fiber Channel. All provisions for SCSI-3-style parallel bus operations are removed; this is the first major artifact.

This document assumes that all parallel bus operations are a mapping of Fiber Channel operations onto a parallel bus. This should result in minimal function in a converter device between Fiber Channel and parallel busses. At present, the information packets will fit into the Fiber Channel frames if data burst lengths are limited to 2048 bytes maximum and the total frame data field length is limited to 2112 bytes for all interface logical elements.

An extensive set of deletions, changes, and additions have been made to the glossary in Section 3. Terminology is used consistently in all sections based on the new terms. All terms used beyond section 3 are defined in this glossary, at first use, or in a secondary glossary at the beginning of a later section. A special glossary has been added for selected Fiber Channel terms.

Section 4 has the Fiber Channel and parallel bus options selection menu. A description of the Fiber Channel options deemed appropriate to SCFI is included. All parallel bus characteristics are referred to SCSI-3 , X3.131-199x, Sections 1-4, 5.1 and 5.3.

Sections 5 and 6 define the boundary between the physical and logical layers. Section 5 deals strictly with the interface protocols. This section provides the SCFI Fiber Channel exchange protocols. Exchange protocols are then named in the logical layer as the correct transport protocol to perform each SCFI operation.

Section 6 contains the logical system description, the layout of commands, status definition, information packets, messages, and actions resulting from certain conditions.

Common commands are in Section 7. Sections 8-xx and some appendices will be updated from SCSI-3. All commands will clearly identify mandatory and optional function. Optional commands will identify mandatory function when implemented.

Gary R. Stephens

X3T9.2/90-132
Revision 03

draft proposed
American National Standard
for Information Systems -

SMALL COMPUTER FIBER INTERFACE
SCFI

December 3, 1990

Secretariat

Computer and Business Equipment Manufacturers Association

Abstract: This standard defines mechanical, optical, and functional requirements for the Small Computer Fiber Interface (SCFI). SCFI permits computers to attach to each other and to intelligent devices. SCFI simplifies connecting computers and intelligent device classes including rigid disks, flexible disks, magnetic tape devices, printers, optical disks, and scanners.

SCFI uses information packets to transfer commands, command parameter data, command response data, logical block data, messages, and status. SCFI uses the Fiber Channel as its primary physical transport layer. SCFI uses the SCSI-3 parallel bus.

SCFI establishes one logical level of interpretation for information packets transfers on either physical transport layer. The Fiber Channel requires major changes in the SCSI-3 information transfer protocol. The parallel interface applies these same changes which greatly simplify parallel bus processing.

SCFI streamlines processing, reduces choices, and adds new functions over SCSI-3. All information transfers in SCFI are independent of the physical bus. SCFI defines a method for managing multiple paths between computers and attached devices on both parallel and serial busses.

This is a draft standard proposed American National Standard of Accredited Standards Committee X3. As such, this is not a complete standard. The X3T9 Technical Committee may modify this document as a result of comments received during the X3 approval process for standards.

POINTS OF CONTACT:

John B. Lohmeyer (X3T9.2 Chair)
NCR Corporation
3718 N. Rock Road
Wichita, KS 67226
(316) 636-8703

I. Dal Allan (X3T9.2 Vice-Chair)
ENDL
14426 Black Walnut Court
Saratoga, CA 95070
(408) 867-6630

This is the initial draft of the proposed standard through Section 7, and Appendix A. This document is the result of a project proposal to create the Small Computer Fiber Interface.

This document has not been approved by the X3T9.2 committee and had not been reviewed by them, as of December 3, 1990.

The current editorial assignments are:

Gary R. Stephens (Acting Technical Editor)
IBM Corporation
9000 S. Rita Rd
Tucson, Az 85744
(602) 799-2246

TABLE OF CONTENTS

Part 1. SCFI BASICS

Section 1. Scope 1-1

Section 2. Referenced Standards and Organizations 2-1
2.1 ANSI Standards 2-1
2.2 Other Standards Organizations 2-1
2.3 Other Organizations 2-1

Section 3. Glossary and Conventions 3-1
3.1 Glossary of Essential SCFI Terms 3-1
3.2 Glossary of Essential Fiber Channel Terms 3-12
3.3 Editorial Conventions 3-16
3.4 Relationship of Essential SCFI and Fiber Channel Terms 3-17

Section 4. Physical Characteristics 4-1
4.1 Serial Interface Physical Description 4-1
4.1.1 Point-to-Point Topology (Optional) 4-3
4.1.2 Single InBand Address Topology 4-3
4.1.2.1 Point-to-Point Operation 4-3
4.1.2.2 String Operation (Optional) 4-5
4.2 Parallel Interface Physical Description (Optional) 4-8

Section 5. SCFI Information Transfer 5-1
5.1 Serial Bus Phases 5-1
5.1.1 BUS FREE Phase 5-2
5.1.2 Information Transfer Phase 5-3
5.1.3 NEXUS TRANSFER Phase 5-3
5.2 Serial Interface Bus Events/Conditions 5-3
5.2.1 Serial Reset Event 5-4
5.2.2 Serial Unexpected Disconnect Event 5-4
5.2.3 Serial Unsuccessful Information Packet Transfer Condition 5-4
5.2.4 Serial Unexpected Disconnect Condition 5-4
5.3 Serial Bus Phase Sequences 5-4
5.3.1 Serial Bus Phase Order 5-5
5.3.2 Signal Restriction Between Phases for Serial Busses 5-5
5.4 Serial Exchange Protocol 5-5
5.4.1 SCFI Serial Frames 5-6
5.4.2 SCFI Serial Frame Sequences 5-7
5.4.3 SCFI Serial Exchanges 5-11
5.4.4 SCFI Logical Operations Characterization 5-11
5.4.5 SCFI Serial Exchange Protocol - Normal Operation 5-11
5.4.5.1 Simple I/O Process (Mandatory) 5-12
5.4.5.2 Complex I/O Processes (Optional) 5-12
5.4.5.3 Operation Types 5-13
5.4.6 SCFI Exchange Protocol - Abnormal Sequences 5-16
5.4.7 SCFI Serial Exchange Protocol State Diagrams 5-19
5.4.8 Resending an Unacknowledged Information Packet 5-24
5.4.8.1 Initiating Controller 5-24

5.4.8.2 Target Controller 5-24
5.4.9 Target Controller Termination of an I/O Process 5-24
5.4.10 Initiating Controller Termination of an I/O Process 5-24
5.4.11 ???? 5-24
5.5 Parallel Interface Bus Phases (Optional) 5-25
5.5.1 SCFI versus SCSI-3 Operating Mode 5-25
5.5.2 Parallel BUS FREE Phase 5-26
5.5.3 Parallel ARBITRATION Phase 5-26
5.5.4 Parallel SELECTION Phase 5-26
5.5.5 Parallel SELECTION Time-out Procedure 5-27
5.5.6 Parallel Information Transfer Phase 5-27
5.5.6.1 Parallel Asynchronous Information Transfer 5-28
5.5.6.2 Parallel Synchronous Data Transfer 5-28
5.5.6.3 Parallel Wide Data Transfer 5-28
5.5.7 Parallel NEXUS TRANSFER Phase 5-28
5.6 Parallel Interface Bus Events (Optional) 5-28
5.6.1 Parallel Unexpected Disconnect Event 5-28
5.6.2 Attention Event 5-29
5.6.3 Parallel Reset Event 5-29
5.6.4 Parallel Unexpected Disconnect Condition 5-30
5.6.5 Parallel Attention Condition 5-30
5.6.6 Parallel Reset Condition 5-30
5.7 Parallel Bus Phase Sequences (Optional) 5-30
5.7.1 Parallel Bus Phase Order 5-30
5.7.2 Parallel Signal Restriction Between Phases 5-30

Part 2. SCFI Logical Operations

Section 6. SCFI Logical Operations 6-1
6.1 Logical System Description 6-2
6.1.1 Logical System Definition 6-2
6.1.2 I/O Processes 6-7
6.1.2.1 General Description of Logical Operations 6-7
6.1.2.2 I/O Process Parameter Requirements (Mandatory) 6-9
6.1.2.3 I/O Process Sense Data (Mandatory) 6-10
6.1.3 Programmable Operating Definition (Parallel) (Optional) 6-10
6.1.4 SCFI Addresses 6-11
6.1.4.1 SCFI Address for a Port (Mandatory) 6-11
6.1.4.2 Logical Units (Mandatory) 6-11
6.1.4.3 Target Routines (Optional) 6-11
6.1.5 Queued I/O Processes (Mandatory) 6-12
6.1.5.1 Untagged I/O Processes (Mandatory) 6-12
6.1.5.2 Tagged I/O Processes (Optional) 6-13
6.1.5.3 Concurrent I/O Processes Execution 6-16
6.1.6 Logical Unit Reservation Termination (Mandatory) 6-16
6.1.7 Assignment Termination (Optional) 6-16
6.1.8 Path Group Termination (Optional) 6-16
6.2 Commands 6-16
6.2.1 Command Set Implementation Requirements (Mandatory) 6-17
6.2.2 Reserved (Mandatory) 6-18
6.2.3 Command Descriptor Block 6-18
6.2.4 Operation Code 6-19
6.2.5 Logical Block Address 6-20
6.2.6 Transfer Length 6-21

6.2.7 Command Parameter Data Length 6-21
6.2.8 Command Response Data Length 6-22
6.2.9 Control Field 6-22
6.2.9.1 Link Bit (Mandatory) 6-22
6.2.9.2 Flag Bit (Mandatory) 6-23
6.3 Status (Mandatory) 6-23
6.3.1 Status Codes (Mandatory) 6-24
6.3.2 Status Code Reporting Priority (Mandatory) 6-25
6.4 Contingent Allegiance Condition (Mandatory) 6-26
6.5 Extended Contingent Allegiance Condition (Optional) 6-27
6.6 Asynchronous Event Notification (Mandatory) 6-28
6.7 Information Packet Structure 6-30
6.7.1 Interface Control Fields (Mandatory) 6-30
6.7.1.1 Interface Control Fields (Mandatory) 6-31
6.7.1.2 Information Packet Serial Encapsulation 6-34
6.7.2 Interface Logical Elements (Mandatory) 6-36
6.7.2.1 Message Interface Logical Element 6-37
6.7.2.2 Command Descriptor Block Interface Logical Element 6-37
6.7.2.3 Command Parameter Data Interface Logical Element 6-37
6.7.2.4 Command Response Data Interface Logical Element 6-37
6.7.2.5 Logical Block Data Interface Logical Element 6-38
6.7.2.6 Status Interface Logical Element 6-38
6.7.2.7 Autosense Interface Logical Element (Optional) 6-38
6.7.3 Information Packet Layout (Mandatory) 6-38
6.7.3.1 Serial Bus 6-38
6.7.3.2 Parallel Bus (Optional) 6-39
6.8 Messages (Mandatory) 6-39
6.8.1 ABORT 6-44
6.8.2 ABORT TAG 6-44
6.8.3 BUS DEVICE RESET 6-44
6.8.4 CLEAR QUEUE 6-45
6.8.5 COMMAND COMPLETE 6-45
6.8.6 EXTENDED MESSAGE REJECT 6-45
6.8.7 INITIATE RECOVERY 6-46
6.8.8 INVALID BUS PHASE DETECTED (Parallel) 6-47
6.8.9 INVALID INFORMATION PACKET 6-47
6.8.10 LINKED COMMAND COMPLETE 6-48
6.8.11 LINKED COMMAND COMPLETE (WITH FLAG) 6-48
6.8.12 MODIFY DATA POSITION 6-48
6.8.13 PARITY ERROR (Parallel) 6-49
6.8.14 RELEASE RECOVERY 6-50
6.8.15 RESEND PREVIOUS INFORMATION PACKET 6-50
6.8.16 SYNCHRONOUS DATA TRANSFER REQUEST (Parallel) 6-50
6.8.17 SYNCHRONOUS PACKET TRANSFER REQUEST 6-52
6.8.18 TERMINATE I/O PROCESS 6-54
6.8.19 TRANSFER READY 6-55
6.9 Command Processing Considerations and Exception Conditions 6-55
6.9.1 Unit Attention Condition 6-55
6.9.2 Incorrect Initiating Controller Connection 6-57
6.9.3 I/O Processes for an Invalid LUN or TRN 6-57
6.9.4 Parameter Rounding 6-58
6.9.5 Unsuccessful I/O Process Termination Condition 6-58
6.9.6 Reset Condition 6-58
6.9.6.1 Reset Condition (Serial) 6-58
6.9.6.2 Reset Condition (Parallel) 6-59

6.9.6.3 Hard Reset Alternative (Parallel) . . . 6-60
6.9.6.4 Soft Reset Alternative (Parallel) . . . 6-60
6.9.7 Unsuccessful Information Packet Transfer Condition (Serial) . . . 6-61
6.9.8 Unexpected Disconnect Condition (Serial) . . . 6-61
6.9.9 Unexpected Disconnect Condition (Parallel) . . . 6-61
6.9.10 Attention Condition (Parallel) . . . 6-61

Part 3. SCFI Common Commands

Section 7. Commands Common to All Device Classes . . . 7-1
7.1 Characteristics of All Device Classes . . . 7-1
7.1.1 Commands Implemented by All Logical Units (Mandatory) . . . 7-1
7.1.1.1 Using the INQUIRY Command . . . 7-1
7.1.1.2 Using the REQUEST SENSE Command . . . 7-1
7.1.1.3 Using the SEND DIAGNOSTIC Command . . . 7-1
7.1.1.4 Using the TEST UNIT READY Command . . . 7-1
7.1.2 Commands Implemented by All Target Routines (Mandatory) . . . 7-2
7.1.2.1 Using the INQUIRY Command . . . 7-2
7.1.2.2 Using the REQUEST SENSE Command . . . 7-2
7.1.3 Commands for All Device Classes . . . 7-2
7.2 Command Descriptions . . . 7-3
7.2.1 SET ICID Command . . . 7-5
7.2.2 REPORT PATH STATUS Command . . . 7-9
7.2.3 ASSIGN Command . . . 7-11
7.2.4 UNASSIGN Command . . . 7-13
7.2.5 CONTROL ACCESS Command . . . 7-15
7.2.6 CHANGE DEFINITION Command (Parallel) . . . 7-19
7.2.7 COMPARE Command . . . 7-23
7.2.8 COPY Command . . . 7-25
7.2.8.1 Copies With Unequal Block Lengths . . . 7-27
7.2.8.2 Errors Detected by the Copy Manager . . . 7-28
7.2.8.3 Errors Detected by a Target Controller . . . 7-28
7.2.8.4 COPY Function Codes 00h and 01h . . . 7-29
7.2.8.5 COPY Function Code 02h . . . 7-30
7.2.8.6 COPY Function Code 03h . . . 7-31
7.2.8.7 COPY Function Code 04h . . . 7-32
7.2.8.8 COPY Function Codes 08h and 09h . . . 7-33
7.2.8.9 COPY Function Code 0Ah . . . 7-35
7.2.8.10 COPY Function Code 0Bh . . . 7-37
7.2.8.11 COPY Function Code 0Ch . . . 7-39
7.2.9 COPY AND VERIFY Command . . . 7-43

Part 4. SCFI Supplemental Material

Appendix A. I/O Process Examples . . . A-1
A.1 Single Command Example . . . A-1
A.2 Disconnect Example . . . A-1
A.3 Linked Commands Example . . . A-2
A.4 Queued I/O Process Example . . . A-2
A.4.1 Typical Sequences for Tagged I/O Processes . . . A-3
A.4.2 Tagged I/O Process Example . . . A-3
A.5 Information Packet Examples . . . A-5

INDEX X-1

FIGURES

3-1. Relationship of SCFI Terms (Part 1 of 9) . . . 3-18
3-2. Relationship of SCFI Terms (Part 2 of 9) . . . 3-19
3-3. Relationship of SCFI Terms (Part 3 of 9) . . . 3-20
3-4. Relationship of SCFI Terms (Part 4 of 9) . . . 3-21
3-5. Relationship of SCFI Terms (Part 5 of 9) . . . 3-22
3-6. Relationship of SCFI Terms (Part 6 of 9) . . . 3-23
3-7. Relationship of SCFI Terms (Part 7 of 9) . . . 3-24
3-8. Relationship of SCFI Terms (Part 8 of 9) . . . 3-25
3-9. Relationship of SCFI Terms (Part 9 of 9) . . . 3-26
4-1. Serial Transport Layer Options . . . 4-2
4-2. Example of Point-to-Point Topology . . . 4-3
4-3. Example of Point-to-Point Operation with Fabric Elements . . . 4-5
4-4. Example of a String with other Fabric . . . 4-7
4-5. Example of a SCFI Port with Ring Fabric Element . . . 4-8
4-6. Parallel Transport Layer Options . . . 4-9
5-1. Serial Interface Phase Sequences . . . 5-5
5-2. Fiber Channel Frame Sequence Link Responses . . . 5-7
5-3. Fiber Channel Frame Sequences for SCFI . . . 5-8
5-4. Sample Fiber Channel Exchanges . . . 5-10
5-5. Characterization of SCFI Operation Types for the Serial Interface . . . 5-14
5-6. Normal SCFI Operation Types Mapped to Fiber Channel . . . 5-15
5-7. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 1 of 3) . . . 5-17
5-8. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 2 of 3) . . . 5-18
5-9. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 3 of 3) . . . 5-19
5-10. SCFI Exchange/Frame Sequence State Diagram (Part 1 of 5) . . . 5-20
5-11. SCFI Exchange/Frame Sequence State Diagram (Part 2 of 5) . . . 5-21
5-12. SCFI Exchange/Frame Sequence State Diagram (Part 3 of 5) . . . 5-22
5-13. SCFI Exchange/Frame Sequence State Diagram (Part 4 of 5) . . . 5-23
5-14. SCFI Exchange/Frame Sequence State Diagram (Part 5 of 5) . . . 5-24
5-15. Parallel Bus Phase Sequences . . . 5-31
6-1. Minimum Logical System Attributes . . . 6-6
6-2. Simplified SCFI System . . . 6-7
A-1. QIOPQ in Order Received by Target Controller . . . A-3
A-2. QIOPQ in Modified by Reordering . . . A-4
A-3. QIOPQ Modified by Head of Queue Tag . . . A-5
A-4. Initial Information Packet Example (Parallel 1 of 3) . . . A-6
A-5. Initial Information Packet Example (Parallel 2 of 3) . . . A-7
A-6. Initial Information Packet Example (Parallel 3 of 3) . . . A-7
A-7. Response Information Packet Example (Parallel 1 of 4) . . . A-8
A-8. Response Information Packet Example (Parallel 2 of 4) . . . A-9
A-9. Response Information Packet Example (Parallel 3 of 4) . . . A-9
A-10. Response Information Packet Example (Parallel 4 of 4) . . . A-9

45

TABLES

5-1. Serial Interface Information Transfer Phase 5-3
5-2. Parallel Interface Information Transfer Phase 5-27
6-1. Typical Command Descriptor Block for Six-Byte Commands 6-18
6-2. Typical Command Descriptor Block for Ten-Byte Commands 6-19
6-3. Typical Command Descriptor Block for Twelve-Byte Commands 6-19
6-4. Operation Code 6-20
6-5. Control Field 6-22
6-6. Status Byte 6-24
6-7. Status Byte Code 6-24
6-8. Status Byte Code Reporting Priority 6-26
6-9. Information Packet Structure_ 6-30
6-10. Interface Control Prefix Fields (Mandatory) 6-31
6-11. Interface Control Suffix Fields (Mandatory) 6-34
6-12. Interface Control Prefix Fields (Serial) 6-34
6-13. Frame Header Field (Serial) 6-35
6-14. Interface Control Suffix Fields (Serial) 6-35
6-15. Interface Logical Element 6-36
6-16. ILE Element Type Codes 6-36
6-17. Information Packet Structure (Serial) 6-38
6-18. Information Packet Structure (Parallel) 6-39
6-19. Message Codes 6-40
6-20. Message Element Format Codes 6-41
6-21. Extended Message Element Format 6-42
6-22. Extended Message Codes 6-43
6-23. EXTENDED MESSAGE REJECT Message Format 6-46
6-24. INVALID BUS PHASE DETECTED Message Format 6-47
6-25. INVALID INFORMATION PACKET Message Format 6-48
6-26. MODIFY DATA POSITION Message Format 6-49
6-27. PARITY ERROR Message Format 6-49
6-28. RESEND PREVIOUS INFORMATION PACKET Message Format 6-50
6-29. SYNCHRONOUS DATA TRANSFER REQUEST Message Format 6-51
6-30. SYNCHRONOUS PACKET TRANSFER REQUEST Message Format 6-52
7-1. Commands for All Device Classes 7-3
7-2. SET ICID Command 7-5
7-3. Command Parameter Data for SET ICID Command 7-7
7-4. REPORT PATH STATUS Command 7-9
7-5. Command Response Data for REPORT PATH STATUS Command 7-9
7-6. ASSIGN Command 7-11
7-7. Command Parameter Data for ASSIGN Command 7-12
7-8. UNASSIGN Command 7-13
7-9. Command Parameter Data for UNASSIGN Command 7-14
7-10. CONTROL ACCESS Command 7-15
7-11. Command Parameter Data for CONTROL ACCESS Command 7-16
7-12. CHANGE DEFINITION Command (Parallel Only) 7-19
7-13. Definition Parameter Values 7-20
7-14. Command Parameter Data for CHANGE DEFINITION Command 7-20
7-15. COMPARE Command 7-23
7-16. COPY Command 7-25
7-17. Pad Bit and Cat Bit Combinations 7-27
7-18. COPY AND VERIFY Command 7-43

Part 1. SCFI BASICS

Section 1. Scope 1-1

Section 2. Referenced Standards and Organizations 2-1
2.1 ANSI Standards 2-1
2.2 Other Standards Organizations 2-1
2.3 Other Organizations 2-1

Section 3. Glossary and Conventions 3-1
3.1 Glossary of Essential SCFI Terms 3-1
3.2 Glossary of Essential Fiber Channel Terms 3-12
3.3 Editorial Conventions 3-16
3.4 Relationship of Essential SCFI and Fiber Channel Terms 3-17

Section 4. Physical Characteristics 4-1
4.1 Serial Interface Physical Description 4-1
4.1.1 Point-to-Point Topology (Optional) 4-3
4.1.2 Single InBand Address Topology 4-3
4.1.2.1 Point-to-Point Operation 4-3
4.1.2.2 String Operation (Optional) 4-5
4.2 Parallel Interface Physical Description (Optional) 4-8

Section 5. SCFI Information Transfer 5-1
5.1 Serial Bus Phases 5-1
5.1.1 BUS FREE Phase 5-2
5.1.2 Information Transfer Phase 5-3
5.1.3 NEXUS TRANSFER Phase 5-3
5.2 Serial Interface Bus Events/Conditions 5-3
5.2.1 Serial Reset Event 5-4
5.2.2 Serial Unexpected Disconnect Event 5-4
5.2.3 Serial Unsuccessful Information Packet Transfer Condition 5-4
5.2.4 Serial Unexpected Disconnect Condition 5-4
5.3 Serial Bus Phase Sequences 5-4
5.3.1 Serial Bus Phase Order 5-5
5.3.2 Signal Restriction Between Phases for Serial Busses 5-5
5.4 Serial Exchange Protocol 5-5
5.4.1 SCFI Serial Frames 5-6
5.4.2 SCFI Serial Frame Sequences 5-7
5.4.3 SCFI Serial Exchanges 5-11
5.4.4 SCFI Logical Operations Characterization 5-11
5.4.5 SCFI Serial Exchange Protocol - Normal Operation 5-11
5.4.5.1 Simple I/O Process (Mandatory) 5-12
5.4.5.2 Complex I/O Processes (Optional) 5-12
5.4.5.3 Operation Types 5-13
5.4.6 SCFI Exchange Protocol - Abnormal Sequences 5-16
5.4.7 SCFI Serial Exchange Protocol State Diagrams 5-19
5.4.8 Resending an Unacknowledged Information Packet 5-24
5.4.8.1 Initiating Controller 5-24
5.4.8.2 Target Controller 5-24
5.4.9 Target Controller Termination of an I/O Process 5-24
5.4.10 Initiating Controller Termination of an I/O Process 5-24
5.4.11 ???? 5-24

5.5 Parallel Interface Bus Phases (Optional) 5-25
5.5.1 SCFI versus SCSI-3 Operating Mode 5-25
5.5.2 Parallel BUS FREE Phase 5-26
5.5.3 Parallel ARBITRATION Phase 5-26
5.5.4 Parallel SELECTION Phase 5-26
5.5.5 Parallel SELECTION Time-out Procedure 5-27
5.5.6 Parallel Information Transfer Phase 5-27
5.5.6.1 Parallel Asynchronous Information Transfer 5-28
5.5.6.2 Parallel Synchronous Data Transfer 5-28
5.5.6.3 Parallel Wide Data Transfer 5-28
5.5.7 Parallel NEXUS TRANSFER Phase 5-28
5.6 Parallel Interface Bus Events (Optional) 5-28
5.6.1 Parallel Unexpected Disconnect Event 5-28
5.6.2 Attention Event 5-29
5.6.3 Parallel Reset Event 5-29
5.6.4 Parallel Unexpected Disconnect Condition 5-30
5.6.5 Parallel Attention Condition 5-30
5.6.6 Parallel Reset Condition 5-30
5.7 Parallel Bus Phase Sequences (Optional) 5-30
5.7.1 Parallel Bus Phase Order 5-30
5.7.2 Parallel Signal Restriction Between Phases 5-30

Section 1. Scope

This American National Standard defines input/output busses for interconnecting computers and peripheral devices using the Small Computer Fiber Interface (SCFI). Although aimed at a serial fiber interface, provision has been added to provide a parallel bus implementation.

SCFI provides commands, support data, and means for transmission. SCFI includes references to the necessary standards and selects a subset of the available options to allow inter-operability of devices meeting this standard. SCFI assures inter-operability only between two SCFI devices having compatible physical transmission layer characteristics. The serial bus definition uses the Fiber Channel standards through FC-3. This document functions as the FC-4 document when operating on the Fiber Channel. The physical parallel bus definition references Sections 1-4, 5.1 and 5.3 of the SCSI-3 standard.

SCFI consists of a selection of I/O busses, logical commands, and responses that operate over a wide range of instantaneous data rates. SCFI introduces information packets which permit all information transfers to occur at maximum rate and eliminates all byte-by-byte processing. SCFI uses information packet transfers on both the serial and parallel busses. The interface essentially supports only buffer-to-buffer transfers. Each information packet is self-identifying which permits correct routing of information packets for each I/O process.

The primary objective of the interface is to provide computers with device type independence within a class of devices. SCFI permits computer users to attach intelligent devices without requiring modifications to hardware or generic software. Generic software is software capable of handling the mandatory functions for a device class. SCFI specifies mandatory function for each device class to make generic software easier to generate. SCFI provides for adding special features and functions through vendor specific fields and codes. Such additions may require software changes to use those features and functions. SCFI provides reserved fields and code values for future standardization.

A second key objective of the standard is to provide a point of incompatibility with SCSI. Fiber Channel operations are, by definition, incompatible. On a parallel bus, properly conforming SCSI-3 devices reject all SCFI protocol extensions. SCFI devices on a parallel bus permit such rejections and allow SCSI-3 devices to continue inter-operation without requiring use of the extensions. SCFI and SCSI-3 devices can operate on the same parallel bus. SCFI and SCSI-3 devices cannot operate with each other unless the SCFI device changes its definition to SCSI-3 mode. SCFI devices, both initiators and targets, attached to parallel both initiating controllers and target controllers, attached to parallel busses, have the option of reverting to the SCSI-3 operating mode. A change in definition of a SCFI device to a SCSI-3 operating mode means the SCSI-3 standard defines all subsequent operations. Refer to the SCSI-3 standard for the SCSI-3 operating mode.

A third key objective of SCFI is to move device-dependent intelligence out to the devices. The command sets allow an attached SCFI device to obtain initialization information from the other attached SCFI devices. Responses identify the SCFI device class, device characteristics, the command set and messages supported, and the changeable parameters. Further requests can determine the readiness of the device to operate, the types of media supported by the device and other pertinent system information. SCFI devices provide parameters required for operation, initialization, or system tuning. The device manages all other parameters.

The new features of SCFI permit transport layer independence. SCFI uses a logical interpretation of and response to information packets. The features consist of new physical layer definitions for a serial protocol and new commands, responses, and messages. SCFI uses reserved information transfer phases in the SCSI-3 to transfer information packets on SCSI-3 parallel busses. SCSI-3 bus length limi-

tations, including arbitration delays, still exist. SCFI can operate with storage-to-storage transfers without the use of a special I/O bus. SCFI does not define this type of information transfer.

For storage devices, the interface uses logical rather than physical addressing for all data blocks. For the direct-access device class, each logical unit can provide information about how many blocks it contains. A logical unit may coincide with all, or part of, a peripheral device, or it may span multiple peripheral devices.

The interface protocol includes provision for the connection of multiple initiating controllers and multiple target controllers. SCFI provides control mechanisms for cooperation between multiple paths when using one or more busses or links in a logical system. The SCSI-3 dual port option is removed.

In SCFI Fiber Channel implementations,

1) fabric contention replaces arbitration.

2) topology and the type of connection between initiating controllers and target controllers determine priority of access to the interface.

3) there is no arbitration time.

In SCFI parallel bus implementations,

1) SCSI-3 provides distributed arbitration.

2) a priority system awards interface control to the highest priority SCFI device that is contending for use of the bus during arbitration.

3) the time to complete arbitration is independent of the number of devices contending for the bus.

Implementations which use other bus structures (e.g., a processor bus) for the transportation layer follow the arbitration rules, access priorities, and timing constraints of that implementation. Such use of SCFI is beyond the scope of this standard.

If a conflict arises between text, tables, and figures, the order of precedence is text, tables, and lastly figures. Not all tables and figures are fully described in text. Tables show data formats and values; figures are illustrative of the text and tables.

Section 2 provides a reference list of prerequisite and corequisite standards.

Section 3 provides a comprehensive glossary of terms used in SCFI.

Section 4 describes physical characteristics and requirements. This consists mainly of references to the controlling physical standards. SCFI limits the choices for each bus type.

The Fiber Channel is available for serial information transfers up to 100 MB/S transfers in one direction. The Fiber Channel permits information transfers up to several kilometers. Extension beyond the maximum length for a single physical link requires a fabric made up of switches and/or repeater devices.

For SCFI parallel busses, there are two electrical alternatives: single-ended and differential. SCFI single-ended and SCFI differential devices do not operate on the same parallel bus. Total cable lengths up to 25 meters are possible using the differential option. A single-ended configuration may have a total cable length of up to 6 meters.

Section 5 describes the protocol for using the selected serial interfaces and parallel busses. The protocol uses the services of the Fiber Channel at the FC-3 and FC-2 levels. SCFI is a level 4 protocol, or FC-4, in the Fiber Channel terminology. Section 5 describes the mandatory requirements for SCFI devices attaching to each interface.

Section 6 provides a logical system description, expanded from SCSI-3. It specifies the SCFI command and status structure. Section 6 describes the method for encapsulating logical functions in information packets. Section 6 also introduces messages and protocols for their use to support mandatory and optional functions and the packet structure.

SCFI classifies commands as mandatory, optional, or vendor specific. SCFI devices implement mandatory common commands and mandatory commands defined for the appropriate device class. SCFI devices may implement optional commands and command functions as well. Within each command, there are mandatory functions which a SCFI device implements; there may be optional functions within some commands. For any command, a SCFI device implements all functions identified as mandatory. Some SCSI-3 options have been made mandatory based on common use.

SCFI devices implement commands and responses that simplify writing self-configuring software. Such software can "discover" necessary device attributes without prior knowledge of specific peripheral device characteristics (such as storage capacity). Many commands also implement a very large logical block address space (2**32 blocks). Some commands implement a smaller logical block address space (2**21 blocks). (Delete These? GRS)

Section 7 specifies those common commands that have a consistent meaning for all device classes.

Sections 8 through 17 describe commands for the following device classes:

- direct-access (e.g., magnetic disk)
- sequential-access (e.g., magnetic tape)
- printer
- processor
- write-once (e.g., optical disk)
- CD-ROM
- scanner
- optical memory
- medium changer
- communications

Commands in each of these sections are unique to the device class, or they have interpretations, fields, or features that are device class specific. For example, several device classes use the WRITE command, but each device class has a different form, parameters or meaning.

Appendices are not part of this standard. Appendices provide extra explanatory material about SCFI. If a conflict arises between the main text and an appendix, the main text takes precedence.

Appendix A contains examples of I/O process management by initiating controllers and target controllers. (New. Extracted from various sections. GRS)

(The following appendices retain their letter names from SCSI-2 for now. GRS)

Appendix D contains information on other standards related to medium types and density codes for flexible disks and magnetic tapes. (incorporate in the mainline text? Section 2. GRS)

Appendix E describes data integrity in command queuing environments. (incorporate in the mainline text? Section 6. GRS)

Appendix F describes normal procedures following a power-on condition. (incorporate in the mainline text? Section 6. GRS)

Appendix I contains the additional sense codes and operation codes in numerical order. (eliminate or incorporate in the mainline text. Section 7. GRS)

Appendix J contains the vendor identification codes as of the date of this document.

An index is found after the last appendix.

Section 2. Referenced Standards and Organizations

2.1 ANSI Standards

ANSI X3.4-1977, American Standard Code for Information Interchange (ASCII),

ANSI X3.131-1990, SCSI-2 (Until SCSI-3 more stable. GRS)

ANSI X3.131-199x, SCSI-3, Only Sections 1 through 5.1 and 5.3

ANSI X3T9.3/9x-xxx Rxx, Fiber Channel FC-3

ANSI X3T9.3/9x-xxx Rxx, Fiber Channel FC-2

ANSI X3T9.3/9x-xxx Rxx, Fiber Channel FC-1

ANSI X3T9.3/9x-xxx Rxx, Fiber Channel FC-0

ANSI X3T9.3/9x-xxx Rxx, Fiber Channel FC-F

2.2 Other Standards Organizations

ECMA-111, SCSI (See INQUIRY command. GRS)

IEC-908, Compact Disc Digital Audio System, 1987

ISO/IEC 10149, Information Technology - Data Interchange on Read-Only 120mm Optical Data Disks (CD-ROM), 1989

ISO IS 9316, (See INQUIRY command. GRS)

ISO IS 3901, International Standard Recording Code

2.3 Other Organizations

The medium catalog numbers in the CD-ROM section (13) are controlled by the Uniform Product Code Council, 8163 Old Yankee Road, Suite J, Dayton, Ohio 45459, U.S.A. and the European Article Number Council, Rue des Colonies, 54- BTE8, 1000 Brussels, Belgium.

Section 3. Glossary and Conventions

3.1 Glossary of Essential SCFI Terms

This section contains a glossary of special terms used in this standard. See additional glossaries in sections 3.2 (merge with 3.1 GRS), 8.4, 9.4, ..., 17.4. A segment of text for any glossary entry prefixed by "(Parallel Interfaces.)" or "(Serial Interfaces.)" applies only to implementations which use that particular transport layer. A segment of text for any glossary entry prefixed by "(Parallel "(Initiating Controllers.)" or "(Target Controllers.)" applies only to SCFI devices when operating in that mode.

A

accept. When related to I/O processes, the process in a target controller which determines that an I/O process may be executed for the sending initiating controller, and if permissible, determines that the I/O process logically correct based on order and content of each interface logical element. Contrast with receipt and execute.

active I/O process. Both an initiating controller and a target controller may have multiple active I/O processes. An active I/O process may be a current I/O process or it may be disconnected. (Initiating Controller.) An I/O process for which a connect has been made. An I/O process becomes active with the initial connection; the same I/O process may be in the queued I/O process queue of the target controller. (Target Controller.) An I/O process which is in execution (not in the queued I/O process queue).

active I/O process queue. A queue in logical elements which contains active I/O processes from the perspective of that logical element. The queue may contain zero or more active I/O processes. The organization of the active I/O process queue is controlled by the logical element.

AEN. Asynchronous event notification.

AIOP. Active I/O process.

AIOPQ. Active I/O process queue.

ASCII byte. A byte whose value is interpreted for graphic presentation according to the ASCII collating sequence.

assigned. An attribute of a target controller function which permits it to accept I/O processes only from certain path groups.

asynchronous event notification. A protocol used by target controllers to notify appropriate initiating controllers of a significant event occurring in the target controller which does not occur while the target controller is executing an active I/O process (e.g., Unit Attention or medium removal). Detection of the event results in either contingent allegiance or extended contingent allegiance.

automatically presented sense data. When contingent allegiance or extended contingent allegiance is established, the target controller notifies the initiating controller by sending a status ILE with a status byte code value of CHECK CONDITION or I/O PROCESS TERMINATED; the target controller prepares sense data appropriate for the error. The target controller may include the sense data as a command response data ILE immediately following the status ILE. If the sense data is transferred in this manner, it is called automatically presented sense data or autosense. If the sense data is not included following the status ILE, the target controller preserves the sense data until the initiating controller

retrieves it or causes it to be deleted without transfer; this is called pending sense data. This term differentiates this data from CDBs, messages, command parameter data, command response data, status and logical block data.

autosense. Automatically presented sense data.

B

bit. See xx bit.

block. A logical block or physical block.

block peripheral device. A peripheral device which is in one of the following device classes: direct access, write once, CD-ROM, or optical memory. A peripheral device in a device class considered as block peripheral device can be a source or destination in a COPY, COMPARE, or COPY AND VERIFY command. See stream peripheral device and other peripheral device.

burst. A contiguous set of bytes associated with a command which represents all or part of the logical block data, the command parameter data, or the command response data. Bursts are variable length. A burst may be part of an information packet. A burst is not part of every information packet.

byte. An 8-bit (octet) construct.

byte string. A contiguous set of ASCII bytes.

C

CA. Contingent allegiance.

CDB. Command descriptor block.

CIOP. Current I/O process.

command. A command descriptor block, and associated command parameter data, sent from an initiating controller to a target controller to cause a logical unit or target routine to perform some action for the initiating controller. A command for an active I/O process with linked commands executes to completion before beginning execution of any later command for the same active I/O process. That is, commands within a set of linked commands for an I/O process are not reordered for execution by the target controller function. s may be

command descriptor block. An organized collection of bytes used to communicate the fixed length portion of commands. This term differentiates this data from messages, command parameter data, command response data, status, autosense, and logical block data.

command parameter data. An organized collection of fields (0 or more in length) provided as an addendum to some command descriptor blocks. This term differentiates this data from messages, CDBs, command response data, status, autosense, and logical block data.

command response data. An organized collection of fields (0 or more in length) provided in response to some commands (e.g., sense data). This term differentiates this data from messages, CDBs, command parameter data, status, autosense, and logical block data.

connect. The initiating controller process which selects an initiator, a target and results in establishing a nexus and an I/O process in a target controller. The connection that results is an initial con-

nection. A connect requires successful information packet transfer. The information packet may not be logically error free.

connection. An initial connection or reconnection. A connection occurs between one initiator and one target on the same SCFI bus. A connection begins when conditions exist between an initiator and a target for information transfer; a connection ends with the next disconnect. (Parallel Interfaces.) At the end of a successful selection. (Serial Interfaces.) When sending or receiving an information packet. The connection for the sender may end before the connection starts for the receiver.

contingent allegiance. A condition in a target controller established when the error is detected and which results in either the target controller sending CHECK CONDITION or I/O PROCESS TERMINATED status for an active I/O process to its initiating controller or in asynchronous event notification. The target controller determines the condition to be of a nature where no recovery procedure requiring exclusive use of the logical unit or target routine is required. The target controller prepares sense data when the error is detected. If the sense data is transferred as autosense, the contingent allegiance is terminated upon successful transfer of the information packet(s) containing the status ILE and the sense data ILE(s).

copy manager. A target controller which has implemented the COPY, COMPARE, and/or the COPY AND VERIFY commands. The target controller becomes a copy manager only on accepting one of these three commands and the corresponding command parameter data. A copy manager implements an initiating controller logical element function more complete than that required for asynchronous event notification.

CPD. Command parameter data.

CRD. Command response data.

current I/O process. An I/O process connected on a SCFI bus. The I/O process may be an active I/O process or it may be a queued I/O process with the ILEs from information packets being routed to the queued I/O process queue in a target controller. (Parallel Interfaces.) Only one I/O process may be current on a single SCFI bus. (Serial Interfaces.) Only one I/O process may be current per fiber at either the initiator end of a link or the target end of a link. Zero or more information packets may be between the link end points for the same or different I/O processes.

D

device class. A set of target controllers all having the same logical model and specified independently in Sections 8 through 17. Mandatory and optional functions are specified including the common commands and functions. All device classes conform to the mandatory requirements of the logical system.

device type. A specific target controller function which implements some or all of the functions for a device class. Each device type implements all mandatory commands and functions for its device class.

disband. A process which breaks up a set of paths established as a path group.

disconnect. (Parallel interfaces.) The action that occurs when a selected SCFI device releases control of the bus, allowing it to go to the BUS FREE phase. (Serial interfaces.) The action that occurs at the end of an information packet transfer or the transmitter stops sending ordered sets during information packet transfer.

E

ECA. Extended contingent allegiance.

establish a path group. A process to form a non-empty set of paths as a group which are treated as equivalent for most I/O process activity.

execute. When applied to an active I/O process, only interface logical elements which have been received and accepted are acted upon by the logical element. Interface logical elements which are received and not accepted result in an appropriate message or a contingent allegiance to report the error. Contrast with receive and accept.

explicitly named path. An explicitly named path exists when the initiating controller successfully completes an active I/O process to transfer the ICID of the initiating controller to a LUN or TRN and the target controller transfers its selected port number to the initiating controller.

extended contingent allegiance. A condition in a target controller established when an error is detected and which results in either the target controller sending CHECK CONDITION or I/O PROCESS TERMINATED status for an active I/O process to an initiating controller or in asynchronous event notification. The target controller determines that the condition is a type where a recovery procedure requires exclusive use of the logical unit or target routine. The target controller prepares sense data when the error is detected. For AEN, the sense data is transferred in the SEND command. For an active I/O process, the sense data may be transferred as autosense. The target controller may preserve the sense data in the target controller for retrieval by a REQUEST SENSE command or deleted by the initiating controller without transfer. A special protocol notifies the initiating controller that the condition exists to differentiate it from contingent allegiance. A protocol notifies the target controller when the initiating controller has completed its implementation of the recovery procedure. The target controller recovery procedure is vendor specific. The initiating controller is not required to follow the recommended recovery procedure. Failure to follow the recommended procedure may have an undesirable effect on the target controller, the target controller function, or attached peripheral devices.

F

field. A set of one or more contiguous bits.

G

grouped. The state of an explicitly named path when it is a member of an established path group.

H

H_C nexus. A nexus which exists between an initiating controller and a target controller. An H_C nexus, or any of its derivatives, may use more than one initiator and one target combination for connections during an active I/O process when using the multiple path option of SCFI.

H_C_L nexus. A nexus which exists between an initiating controller, a target controller, and a logical unit. This relationship replaces the prior H_C_x nexus.

H_C_L_Q nexus. A nexus between an initiating controller, a target controller, a logical unit, and a queue tag. This relationship replaces the prior H_C_x_y nexus.

H_C_R nexus. A nexus which exists between an initiating controller, a target controller, and a target routine. This relationship replaces the prior H_C_x nexus.

H_C_x nexus. A nexus which is either an H_C_L or H_C_R nexus. This relationship replaces the prior H_C_x_y nexus.

H_C_x_y nexus. A nexus which is either an H_C_x or H_C_L_Q nexus. This relationship replaces the prior H_C nexus.

H_I_T_C nexus. An H_C nexus, or one of its derivatives, between an initiating controller and a target controller, which operates in single path mode.

I

I/O process. An I/O process consists of a set of connections consisting of one initial connection and zero or more reconnections. The set of connections relate to the exchange of information packets. The set of connections pertains to a nexus where zero or more commands may be transferred. An I/O process begins with the establishment of a nexus. An I/O process may be either untagged or tagged with a queue tag. When an I/O process contains one or more commands, the active I/O process normally ends with the disconnect following successful transfer of a COMMAND COMPLETE message, if ECA does not exist. When an I/O process consists of only message exchanges, the active I/O process ends with the disconnect at the receiver of the the last message of the exchange. If CA exists, the I/O process ends with the transfer or deletion of the sense data for the contingent allegiance. If ECA exists, the I/O process ends with the disconnect following successful transfer of the RELEASE RECOVERY message from the same initiating controller. An I/O process ends abnormally with the disconnect following execution of an ABORT, an ABORT TAG, a BUS DEVICE RESET, or a CLEAR QUEUE message. An I/O process also ends abnormally when unexpected disconnect occurs. (Parallel Interfaces.) An I/O process ends abnormally with the disconnect following detection of the RESET event.

I/O process queue. An active I/O process queue or a queued I/O process queue. The contents of the I/O process queues in an initiating controller and a target controller may be different.

ICID. Initiating controller ID.

identify function. A process in logical elements used to identify the logical unit or target routine, and optionally a queue tag to establish or reconnect to a correct nexus. The identify function determines whether an information packet establishs a new nexus or whether it is processed as part of an established nexus (i.e., an existing active I/O process or a queued I/O process).

ILE. Interface logical element.

implicitly named path. An implicitly named path exists when the initiating controller has not transferred its ICID to the target controller, but the initiating controller has made at least one connect to the LUN or TRN. The target controller has transferred its selected port number to the initiating controller.

information packet. A set of bytes containing interface control fields and at least one interface logical element.

initial connection. An initial connection is the result of a connect.

initiating controller. A logical element which principally starts I/O processes. Active I/O processes execute using the services of one or more ports in the initiating controller. (Serial Interfaces.) The ports may be for one or more Fiber Channel nodes.

initiating controller ID. An initiating controller ID is an identifier to uniquely name an initiating controller which is communicated to target controllers over the interface as part of the multiple path option.

initiator. An SCFI port operating in initiator mode through which an initiating controller starts and executes I/O processes. An initiator usually attaches to an initiating controller. An initiator uses one port to one SCFI bus.

initiator mode. The current operating mode of a port which permits it to initiate I/O processes at another port. (Parallel interfaces.) An initiator selects another port, which is in target mode, on the same SCFI bus. (Serial interfaces.) An initiator establishes an H_C nexus with another port using a link and possibly fabric.

interface control fields. An organized collection of bytes used to encapsulate interface logical elements into information packets. The transport layer interprets some of these fields to accomplish proper routing of information packets between initiating controllers and target controllers.

interface logical elements. Organized collections of bytes containing self-describing messages, CDBs, command parameter data, command response data, status, or logical block data.

invalid. An illegal, reserved, or unsupported bit, field, code value, bus phase, or protocol sequence.

J

K

L

LBD. Logical block data.

logical block. A unit of data supplied from or requested by an initiating controller usually for a peripheral device. A logical block may be fixed or variable in length.

logical block data. A burst of bytes from or to a logical unit for a logical block. This term differentiates these data from messages, CDBs, command parameter data, command response data, status, and autosense.

logical element. An initiating controller or a target controller.

logical path. A logical path is the set of all paths which have the same ICID and LUN or TRN in the same target controller. The set of paths identify the routes active I/O processes may take between an initiating controller and a logical unit or target routine.

logical system. (Target Controllers.) A logical system is the set of initiating controllers having at least one port attached to at least one port of the target controller and from which a connect has been made. (Initiating Controllers.) A logical system is the set of all logical units and target routines to which a connect has been made. The set of paths available to an I/O process is fixed when each H_C_x_y nexus is established; an H_C nexus uses only the path where the connect was made.

logical unit. An addressable function within a target controller which executes active I/O processes. A logical unit has a name, the logical unit number or LUN, and a set of commands to execute. Often, there is a one-to-one mapping between peripheral devices and logical units, but this is not required. A

logical unit may be part of, all of, composed of more than one peripheral device, or it may have no peripheral device.

logical unit number. The name of a logical unit used during an I/O process to select it to execute an I/O process for an H_C_L or H_C_L_Q nexus.

LSB. Least significant bit.

LU. Logical unit

LUN. Logical unit number.

M

mn**. A mathematical expression representing the number m raised to the n-th power. Both m and n are integer decimal numbers.

message. An organized collection of bytes (1 or more) used for control between an initiating controller and a target controller for control of a nexus. Some message establish conditions which persist beyond the I/O process in which the condition was established. This term differentiates these data from a CDB, command parameter data, command response data, status, autosense, and a logical block.

MSB. Most significant bit.

MSG. Message.

multiple path mode. A condition for one I/O process where connections may be made on any path in the established path group where the connect occurred. While operating in multiple path mode, an I/O process may operate in single path status mode or multiple path status mode; the initiating controller establishes the desired operating condition when the connect is made. See definitions for single path status mode and multiple path status mode. For an implicitly named path or an ungrouped path see single path mode.

multiple path status mode. A condition in a path group where any status is transferred on any path in the established path group where a connect is made.

N

name bit. A field containing only one bit. name is replaced with a uniquely identifying phrase.

nexus. A relationship between an initiating controller and a target controller that begins with an initial connection and ends with the completion of the associated I/O process. The relationship may be restricted to operate in single path mode. The relationship may be restricted to a single logical unit or target routine through the identify function. Each initiating controller and target controller may have zero, one, or more nexus established at one time.

O

one. A true signal value, a single bit field with a value of 1b, or a field value numerically equal to 1b.

other peripheral device. A peripheral device which is in one of the following device classes: scanner, medium changer, or graphic arts (2 classes). A peripheral device in a device class considered as 'other' cannot be a source or destination in a COPY, COMPARE, or COPY AND VERIFY command. See stream and block peripheral device definitions.

P

page. Regular, self-describing sets of fields used with some command parameter data and command response data. Each page has a page code.

page code. A field within a page whose value is unique within a command. The page code provides the means to interpret the remaining fields in a page.

parameter. A parameter is the value of a field.

password. An identifier used to permit otherwise unauthorized access to an assigned target controller function.

path. A path is a named link between an initiating controller and a target controller function. At least one connect has been made from the initiating controller to the target controller function. The name consists of: an ICID; an SCFI Address for the an initiator port; the initiating controller port number; the target controller port number; an SCFI Address for the target controller; and, the valid LUN or TRN of the selected logical unit or target routine. A path begins as an implicitly named path and may be modified to an explicitly named path by an initiating controller. Only paths which have been explicitly named participate in established path groups. A path may be grouped or ungrouped.

path group. A path group is a cooperating set of paths in a logical path. The logical path consists a non-empty set of explicitly named paths.

pending sense data. sense data for a contingent allegiance or extenced contingent allegiance not sent to the initiating controller as an autosense ILE when the status for the I/O process was sent. It is held in the target controller until retrieved or deleted by an action from the initiating controller, or by an internal target controller equivalent to a power cycle.

peripheral device. A physical device attaching to an SCFI device. A peripheral device and a SCFI device may be one unit. Examples of peripheral devices are: magnetic disks, an array of magnetic disks, printers, optical disks, magnetic tapes, and initiating controllers. Initiating controllers, as processor device class, are also peripheral devices.

physical block. The minimum recording unit, in bytes, for certain device classes. The physical block may be fixed or variable in length and is device class dependent.

port. A port is the name for the portion of an SCFI device where it attaches to one SCFI bus. An SCFI device may have more than one port. Each port may attach to a different SCFI bus. One port functions in either initiator mode or target mode during a connection. Each port has an SCFI address. (Parallel interfaces.) Initiating controllers translate the SCFI address into a SCFI ID for arbitration.

port number. A unique number for each port assigned by the controlling logical element.

Q

QIOP. Queued I/O process.

QIOPQ. Queued I/O process queue.

queue tag. The parameter associated with an I/O process that uniquely identifies it from other tagged I/O processes for a logical unit from the same initiating controller. The queue tag is not related to the path or paths used for the I/O process; it is related only to the initiating controller and logical unit.

queued I/O process. An I/O process that is in the queued I/O process queue and is not in the active I/O process queue. The state of an I/O process in an initiating controller may be different from the state in the target controller associated with the same I/O process. (Initiating Controller.) An I/O process for which a connect has not been made. (Target Controller.) An untagged or a tagged I/O process which is not active.

queued I/O process queue. A queue in either an initiating controller or a target controller to hold all or part of zero or more I/O processes before acceptance and execution. The I/O processes may be either tagged or untagged.

queueing function. A process in a logical element which transfers or relates ILEs from an information packet to an I/O process in the queued I/O process queue (i.e., not active). The organization of the queued I/O process queue is controlled by the logical element.

R

receive. When applied to I/O processes, the result of transfer of an information packet between two logical elements. An information packet is checked and must be accepted before any action is taken relative to its ILEs. An information packet which is received from an initiating controller not permitted access to a target controller is rejected with an appropriate status. An information packet which is received from an authorized initiaging controller, but which has logical construction or content errors is not accepted; the I/O process is termintated with a message or contingent allegiance as appropriate for the error. Contrast with accept and execute.

reconnect. The act of reviving a nexus to continue an I/O process. The connection may be on the same initiator and target pair as the last connection or on a different pair when using multiple path mode. The action taken to revive the nexus depends on the initiating controller conditions established in the target controller. A reconnect results by successfully establishing the conditions appropriate for the physical bus to transfer an information packet associated with a nexus between an initiator and a target.

reconnection. A reconnection is the result of a reconnect. (Parallel Interfaces.) After a connect, a reconnection exists from the end of a subsequent SELECTION phase by either the initiating controller or the target controller until the next disconnect for an I/O process in an initiating controller. (Serial Interfaces.) After a connect, a reconnection exists while the receiving port is actively receiving an information packet for an I/O process in an initiating controller (i.e., transmission does not constitute a reconnection since the fabric may prevent delivery).

reserved. The term used for bits, fields, code values, and bus phases set aside for future standardization.

S

SCFI. Small Computer Fiber Interface.

SCFI address. The unique address for one port on an SCFI device. The SCFI address is unique on a SCFI bus. Multiple ports on the same SCFI device connected to the same SCFI bus have different SCFI addresses. The form of the SCFI address varies with the physical interface used. The field value assigned to the source and destination fields of the interface control prefix fields. (Parallel interfaces.) The SCFI address is a translation of the SCFI ID for its port. The number of valid SCSI addresses on a SCFI bus is a function of the maximum data bus width of all attached ports. (Serial interfaces.) Neither the SCFI address nor the number of SCFI devices on a serial interface translates to the number of signal lines in the interface.

SCFI bus. (Parallel interfaces.) A SCFI-port and the set of all ports, interconnecting cables, etc., to and through which the port communicates. If a SCFI port has bus signals connecting two external connectors, that portion of the SCFI port is part of the SCFI bus. (Parallel Interfaces.) The maximum number of SCFI ports is 8 times bus width (in bytes, not including parity signals). (Serial interfaces.) The set of all N_ports, links, and fabric elements to and through which a single N_port can communicate.

SCFI device. A device capable of attaching to one or more SCFI busses via one or more ports. The device may allow a port to function in either initiator mode or target mode for any one connection. Each SCFI port is capable of both modes.

SCFI ID. (Parallel interfaces.) The bit-significant representation of the SCFI address referring to one of the data bus signal lines available to the SCFI device, excluding parity lines.

SCFI port. Port.

SCSI. SCSI-3.

SCSI-3. Small Computer System Interface - 3 (X3.xxx-199X).

sense data. Autosense data or command response data formatted according to SCFI rules which identify the reason for contingent allegiance or extended contingent allegiance. The target controller also prepares sense data according to the same rules when a REQUEST SENSE command executes and neither a contingent allegiance nor an extended contingent allegiance condition exists. For contingent allegiance and extended contingent allegiance, Sense data may be held in the target controller and transferred when requested by a REQUEST SENSE command or as an autosense ILE following the status ILE when the error is reported.

signal assertion. (Parallel interfaces.) The act of driving a signal to the true state. (??? GRS)

signal negation. (Parallel interfaces.) The act of driving a signal to the false state or allowing the cable terminators to bias the signal to the false state. (??? GRS)

signal release. (Parallel interfaces.) The act of allowing the cable terminators to bias the signal to the false state. (??? GRS)

single path mode. A condition in a path group for one I/O process where single path status mode is in effect and all connections are made on the path where the connect occurred. An implicitly named path or an ungrouped path is in single path mode since it cannot cooperate with any other path.

single path status mode. An attribute of a path group where status leading to a contingent allegiance or extended contingent allegiance is transferred on the path where the connect occurred. An implicitly named path or an ungrouped path is in single path status mode since it cannot cooperate with any other path.

singular. An attribute of an SCFI command which prohibits its execution if it follows a command in an I/O process which has the Link bit set to 1 or which has the Link bit in its CDB is set to 1.

status. One field sent from a target controller at the completion of each command. Status is also produced to report some conditions which occur in an I/O process when a command is not executing. Most status reports result in ending an I/O process. This term differentiates these data from a message, a CDB, command parameter data, command response data, autosense, and logical block data.

storage device. A peripheral device which is capapable of reading and optionally writing logical blocks to a medium. Some storage devices have the medium prepared so that a write process is not incorporated in the storage device (e.g., CDROM). Most storage devices permit writing at least once; reading may occur multiple times in all storage devices.

stream peripheral device. A peripheral device which is in one of the following device classes: sequential access, printer, processor or communications. A peripheral device in a device class considered as 'stream' can be a source or destination in a COPY, COMPARE, or COPY AND VERIFY command. See block and other peripheral device definitions.

supervisor command. A command which may not be executed by a target controller unless specifically authorized.

T

target. An SCFI port, operating in target mode, thorough which I/O processes pass for execution by a target controller function. A target usually attaches to a target controller. A target uses one port to one SCFI bus.

target controller. A logical element which principally executes I/O processes. Active I/O processes execute using the services of one or more ports available to the target controller in target mode. (Serial Interfaces.) The ports may reside in one or more Fiber Channel nodes.

target controller function. A logical unit or a target routine.

target mode. The current operating mode of a port which permits the port to be selected by another port on the same SCFI bus.

target routine. An addressable function within a target controller which executes active I/O processes. A target routine has a name, a target routine number or TRN, and a command set to execute.

target routine number. The name of a target routine used during an I/O process to select it to execute an I/O process.

third-party. For copy operations, third-party means a copy operation managed by one target controller which is not a source or destination of the copy operation. For multiple path operations, third-party means an assignment made for a path group from another path group.

TR. Target routine.

TRN. Target routine number.

U

unassigned. An attribute of target controller functions which permits the function to respond to I/O processes on any path.

unexpected disconnect. A disconnection that occurs because of a protocol error.

ungrouped. The state of a path when it is not a member of a path group. The path may be an implicitly named path or an explicitly named path.

V

vendor specific. Something (e.g., a bit, field, code value, etc.) this standard identifies, but its use is not defined by this standard and may be used differently in various implementations.

VS. Vendor specific.

W

X

xx. Digits 0-9, except those used as section numbers, in the text of this standard that are not immediately followed by lower-case "b" or "h" are decimal values. Large numbers are not separated by commas or spaces (e.g., 12345; not 12,345 or 12 345).

xxb. Digits 0 and 1 immediately followed by lower-case "b" are binary values.

xxh. Digits 0-9 and the upper-case letters "A"-"F" immediately followed by lower-case "h" are hexadecimal values.

Y

Z

zero. A false signal value or a field with a value of 0b in each bit position.

3.2 Glossary of Essential Fiber Channel Terms

(Section 3.2 is to be incorporated in Section 3.1 in the final standard. GRS)

character. A byte encoded as prescribed for transmission across a link. A 10-bit construct.

class of service. A general definition of the methods by which communication circuits are maintained between communicating N_ports. Class(es) of service supported by an N_port or the fabric is one

determining factor in establishing inter-operability between two N_ports or between N_ports and a fabric.

Class 1 service. An exchange multiplex service. SCFI permits a Class 1 service level. Class 1 service does not reflect the peer-to-peer nature of SCFI since the connection between N_Ports is held until released. That is, the disconnect privilege is not granted in a logical sense. With fabric, the switch connections are set and maintained, whether used or not, until explicitly released. Frames transmitted in the correct sequence at a transmitter arrive in the same sequence at the receiver. Other attributes of Class 2 service are required login, connection and disconnection protocol required, end-to-end confirmation of frame transmission, maximum frame size limitations, dual flow control, and busy and reject link continue and link responses are are not allowed once the connection is made.

Class 2 service. A frame multiplex service of a fabric. SCFI uses a Class 2 service level. Class 2 service most nearly reflects the peer-to-peer nature of SCFI. Frames transmitted in the correct sequence at a transmitter may not arrive in the same sequence at the receiver. Other attributes of Class 2 service are required login, connectionless service, frame multiplexing at the sender or receiver, end-to-end confirmation of frame transmission, maximum frame size limitations, dual flow control, and busy and reject link continue and link responses are allowed.

CRC field. A four-byte field at the end of the frame content field. The CRC field contains information which may help determine if transmission errors occur across a link, the fabric, or passing through any N_port or F_port. This field is normally generated by a hardware element and one or more hardware elements check it at the receivers. The CRC field value is calculated before frame transmission begins. The algorithm for generating the field is specified in the appropriate standard.

D_ID. Destination identifier.

delimiter. An ordered set used as a prefix or suffix for the frame content field. The delimiters are start of frame and end of frame.

destination ID. An identifier unique to the N_port on a SCFI bus in the frame header field of each frame of an exchange which identifies the responder of the exchange.

end of frame. One of several words selected as a trailing delimiter for a frame for transmission across a link. The choice of which word to use in each situation is specified in the exchange protocol.

exchange. A set of one or more frame sequences defined to perform some action for the logical layer of the Fiber channel. Frame sequence selection and the ordering of frame sequences is a function of the logical layer. Once a sequence begins within an exchange, it terminates or is aborted before another sequence begins for the same exchange. An exchange operates between the originator and responder N_ports where the connect was made.

exchange protocol. A set of one or more exchanges defined in the logical layer to carry out its functions. SCFI provides rules to translate each nexus into an exchange protocol. Each exchange protocol is independent of the logical content of the exchanges, frame sequences, and frames. That is, two or more logical functions may use the same exchange protocol. The set of exchange protocols is the FC-4 layer in the Fiber Channel.

F_N_port. A logical component of a fabric element which is responsible for directing frames through the fabric element. An F_N_port may also be the source or destination of frames related to link and fabric control. An F_N_port is an N_port.

F_port. A component of a fabric element which connects to the end of a link. An F_port receives frames from an N_port and directs them into the fabric. An F_port receives frames from the fabric and transmits them to an N_port.

F_port service parameters. A set of parameter specifying the capabilities of an F_port exchanged with its attached N_port or F_port. A Service parameter exchange takes place with an N_port before ULP exchanges are permitted.

fabric. A Fiber channel entity with at least one fabric element. A Fiber Channel implementation does not require a fabric (i.e., point-to-point topology).

fabric element. An optional element of the Fiber Channel which does, at minimum, frame switching or circuit switching. Some fabric elements may perform both functions and may perform the N_port naming function using the F_N_port. A fabric element contains of a minimum of one F_port and an F_N_port. Fabric elements attach to each other via links to provide more sophisticated frame routing and provide more N_ports than a single fabric element is designed to handle.

fabric class of service. The class(es) of service supported by a fabric.

fiber. A single transmission element which transfers frames in a unidirectional manner.

frame. The smallest unit of information transmitted across a link. A frame consists of idle words, a start of frame delimiter, a frame header field, frame content field, an end of frame delimiter, and idle words. A frame, excluding idle words, is from 28(?) to 2140(?) characters long. An N_port sends frames in the order established by the frame sequence used in each exchange protocol. Frames may not arrive at a receiving N_port in the order transmitted, depending on the makeup of the fabric, if present, and the class of service.

frame content field. A set of three fields in a frame: the frame header field, frame data field, and CRC field. The frame content field varies from 20(?) to 2132(?) characters. The frame content field is the principal interface between the physical elements of the Fiber Channel and the logical layers. The frame content field is approximately equivalent to an information packet.

frame data field. A variable length field, always a multiple of 4 bytes, which conveys logical information across a link. A frame data field has a zero minimum length and a maximum length of 2112 bytes. The frame header field has a field designated to identify the number of pad bytes, if any, in a frame data field.

frame header field. A fixed length and fixed format structure, positioned after the start of frame, used by the fabric and N-ports during transmission of frames. The frame header field provides physical addresses for frame routing on the Fiber Channel. The frame header field has other fields which help determine frame sequence errors.

frame sequence. A protocol in the Fiber Channel specifying the types of frames and the order of transmission between N_ports. The Fiber Channel standard specifies rules for determining when each sequence ends.

idle word. An ordered set chosen to represent the BUS FREE state of a fiber (i.e., no frame being transmitted).

layer. A function the Fiber Channel which has a defined interface to other functions. Layers include logical, transport services, transport, physical.

link. A physical interface with two fibers and the connectors at each end.

link commands. A set of frames which initiate an exchange to manage or control the various portions of a Fiber Channel. Some link commands are used at the logical level of SCFI. Link commands are similar to some messages on the parallel bus.

link continue. A set of responses acknowledging receipt of a frame or group of frames. These frames usually relate to ULP exchanges.

link end point. The connector portion of a link which attaches to an N_Port. Connectors at fabric element F_Ports are ignored.

link responses. A set of responses which provide acknowledgement from a responder to the originator of an exchange containing a link command.

login. A procedure for each N_port to identify itself to, or gain its identity from, the fabric or another N_port. An N_port performs a login sequence with each N_port with which it intends to communicate and successfully exchange service parameters. Each N_port exchanges service parameters with its F_port if the fabric exists.

logout. A procedure for removing the service parameter agreement between two N_ports. When the procedure is complete, login occurs before an N_port attempts ULP exchanges.

N_port. A port on the Fiber Channel which attaches to a link. Each N_port attaches, through a link, to exactly one F_port if the fabric exists. An N-port is equivalent to a SCFI port.

N_port class of service. The class(es) of service supported by an N-port.

N_port service parameters. A set of parameter specifying the capabilities of an N_port exchanged with an attached N_port or F_port. N_ports exchange service parameters with an N_port or F_port before attempting to use ULP exchanges. N_port service parameters are similar to the synchronous negotiation agreement and wide negotiation agreement for the parallel busses.

node. A facility composed of one or more N_ports. By definition, a fabric element contains a node since it has an addressable F_N_port. (The naming of N_ports appears related to nodes. Otherwise, no special significance has been attributed to nodes in the Fiber Channel (9/90). GRS)

node identifier. An identifier which uniquely identifies a node on a SCFI bus.

ordered set. A sequence of four characters defined to have meaning to the Fiber Channel transport layer.

originator. The N_port responsible for initiating an exchange. When two N_ports are communicating either may become an originator and start an exchange. The originator of an exchange may be either an initiator or a target.

originator exchange status. A set of fields and values which permits an N_port to monitor an exchange for proper operation (e.g., a violation of a frame sequence). The originator maintains this status for each active exchange.

pad bytes. A variable length field at the end of the frame data field which increases the frame data field to a multiple of 4 bytes. There can be zero to 3 pad bytes per frame data field.

point-to-point topology. A Fiber Channel implementation includes a minimum of two nodes with a minimum of one N-port each and links attaching the N_ports in each node. Each N_port communicates with exactly one other N_port. The destination N_port always receives frames in the order transmitted.

point-to-point operation. A function of the Fiber Channel which logically connects two N_ports with fabric interposed to make them appear connected as in a point-to-point topology.

receiver. A hardware element which accepts signals from a fiber and converts them into characters and some of the characters into bytes (frame content field).

responder. The N_port responsible for accepting an exchange initiated by an originator. When two N_ports are communicating, either may become a responder to an exchange initiated by the other N_port. The responder of an exchange may be either an initiator or a target.

responder exchange status. A set of fields and values which permits an N_port to monitor an exchange for proper operation (e.g., a violation of a frame sequence). This responder maintains this status for each active exchange.

S_ID. Source identifier.

SIBA topology. Single inband address topology.

single inband address topology. A topology option in Fiber Channel which determines all information for routing from the frame header field.

source ID. An identifier unique to the N_port on a SCFI bus in the frame header field of each frame of an exchange which identifies the originator of the exchange.

service parameters. A set of field values specifying the capabilities of an N_port or F_port. Communicating ports exchange service parameters before attempting ULP exchanges. Service parameters are loosely related to a synchronous negotiation agreement and a wide data transfer negotiation agreement for parallel busses.

start of frame. One of several words selected as a leading delimiter for a frame for transmission across a link. The choice of which word to use in each situation is specified in the exchange protocol.

transmitter. A hardware element which encodes bytes into characters or generates ordered sets on command and sends them in a serial fashion on a fiber.

ULP. upper level protocol.

ULP exchanges.. Frames and frame sequences specified primarily for performing the logical functions of an interface.

upper level protocol. The FC-4 operation level in Fiber Channel where speecific interface protocols are mapped to Fiber Channel frame sequences and exchanges.

word. A set of four characters for transmission across a link.

XID. Exchange identifier.

3.3 Editorial Conventions

Certain words and terms used in this standard have specific meanings beyond the normal English meaning. Glossaries define these words and terms (see 3.1 and 3.2, 6.1.1, 8.4, 9.4, ..., 17.4) or the definition appears at first use in the text. Names of signals, phases, messages, commands, statuses, sense keys, additional sense codes, and additional sense code qualifiers are in all upper-case (e.g., REQUEST SENSE).

Normal English capitalization (or lack thereof) is used for other words. Words have the normal technical English definition unless the word or phrase is defined in context or in a glossary, then that definition is used. Some words may have definitions unique to the sections where they are used. This is especially true in Sections 8 through 17. Every effort has been made to limit the number of such instances.

3.4 Relationship of Essential SCFI and Fiber Channel Terms

Some terms, defined in the glossaries of this section, have a relationship to one another as shown pictorially below.

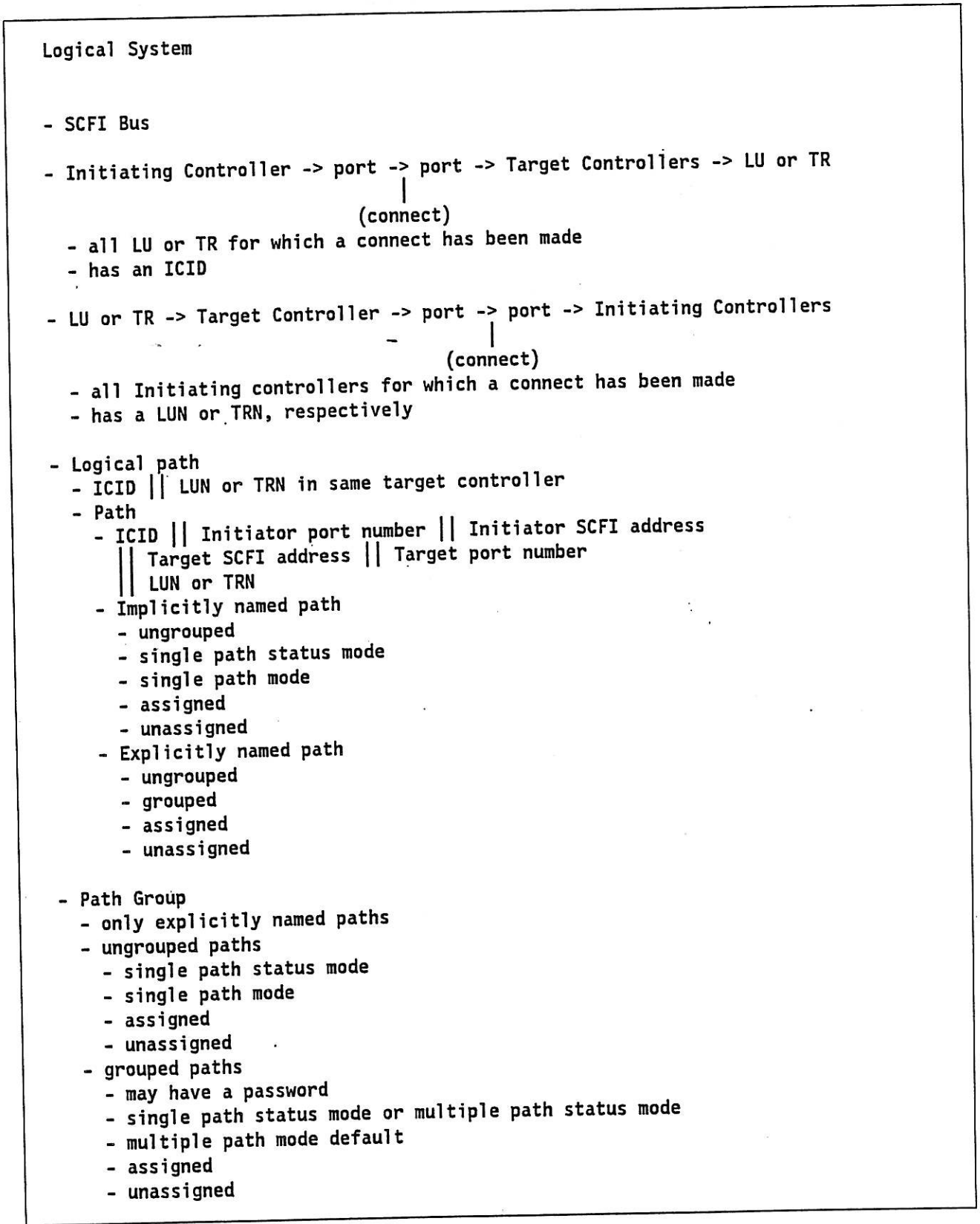


Figure 3-1. Relationship of SCFI Terms (Part 1 of 9)

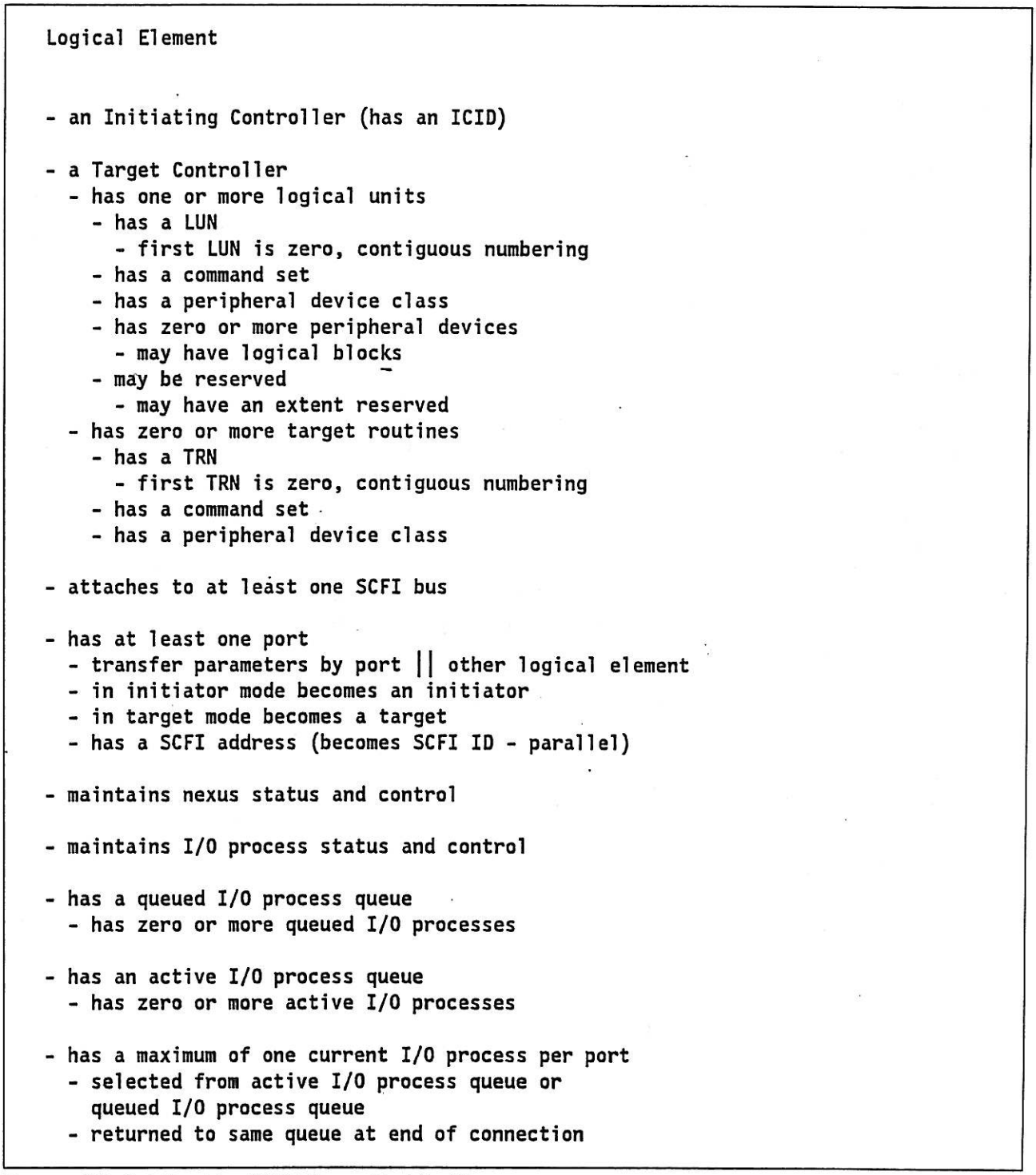


Figure 3-2. Relationship of SCFI Terms (Part 2 of 9)

```
Nexus

-  H_C nexus
   - H_C_x_y nexus
     - H_C_x nexus
       - H_C_L
       - H_C_R
     - H_C_L_Q

- for single path mode replace H_C with H_I_T_C

- I/O Process
  - has zero or more commands
  - made up of one or more information packets
  - may be queued I/O process in queued I/O process queue
    - may have a queue tag
    - may become a current I/O process with any
      connection on the eligible set of paths
      started by an initiating controller
  - may be active I/O process in active I/O Process queue
    - queued I/O processes transfer to active I/O process queue
      for execution
    - may have a queue tag
    - may become a current I/O process with any
      connection on the eligible set of paths

 LEGEND:

   H = Initiating Controller
   C = Target Controller
   I = Initiator
   T = Target
```

Figure 3-3. Relationship of SCFI Terms (Part 3 of 9)

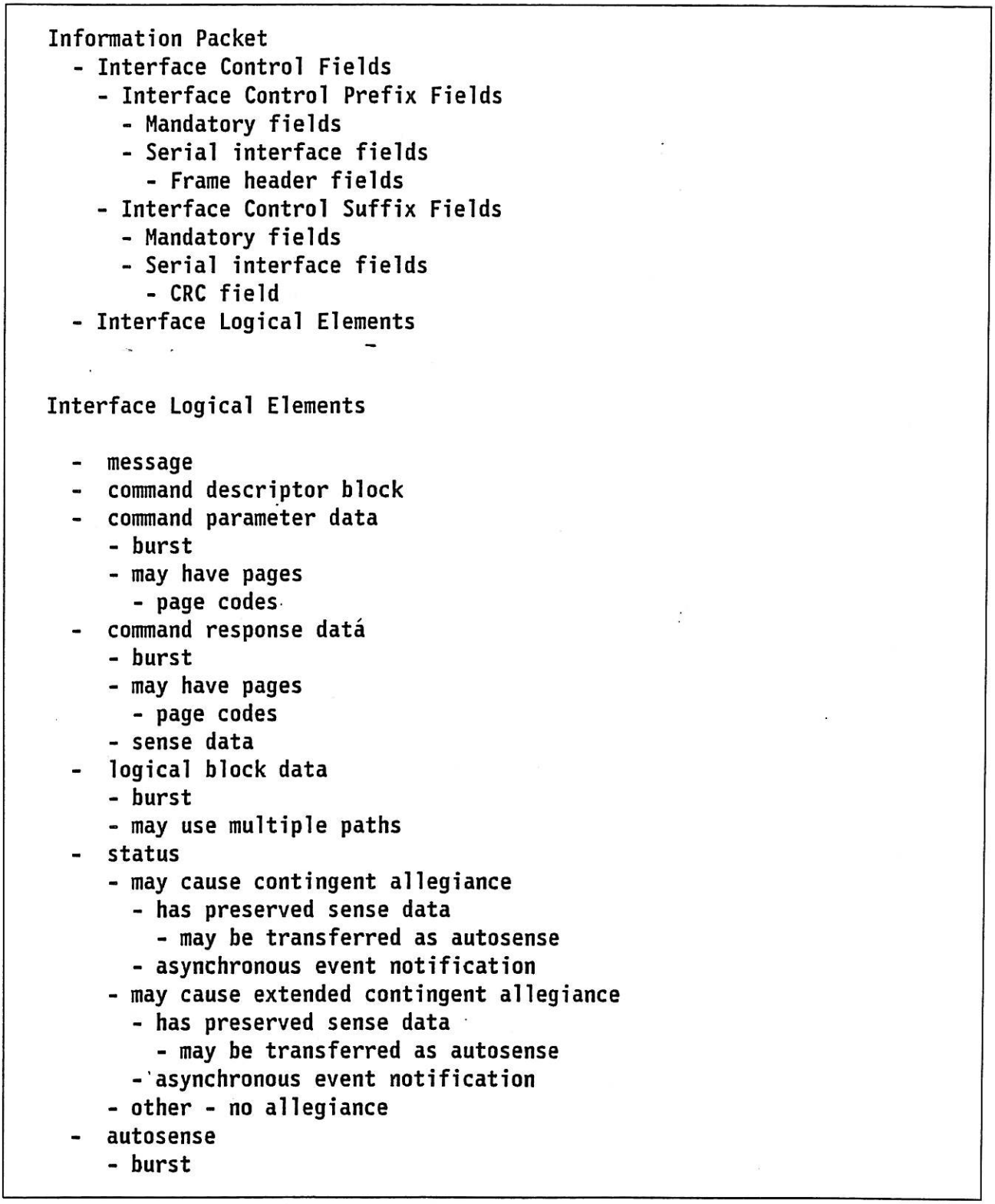


Figure 3-4. Relationship of SCFI Terms (Part 4 of 9)

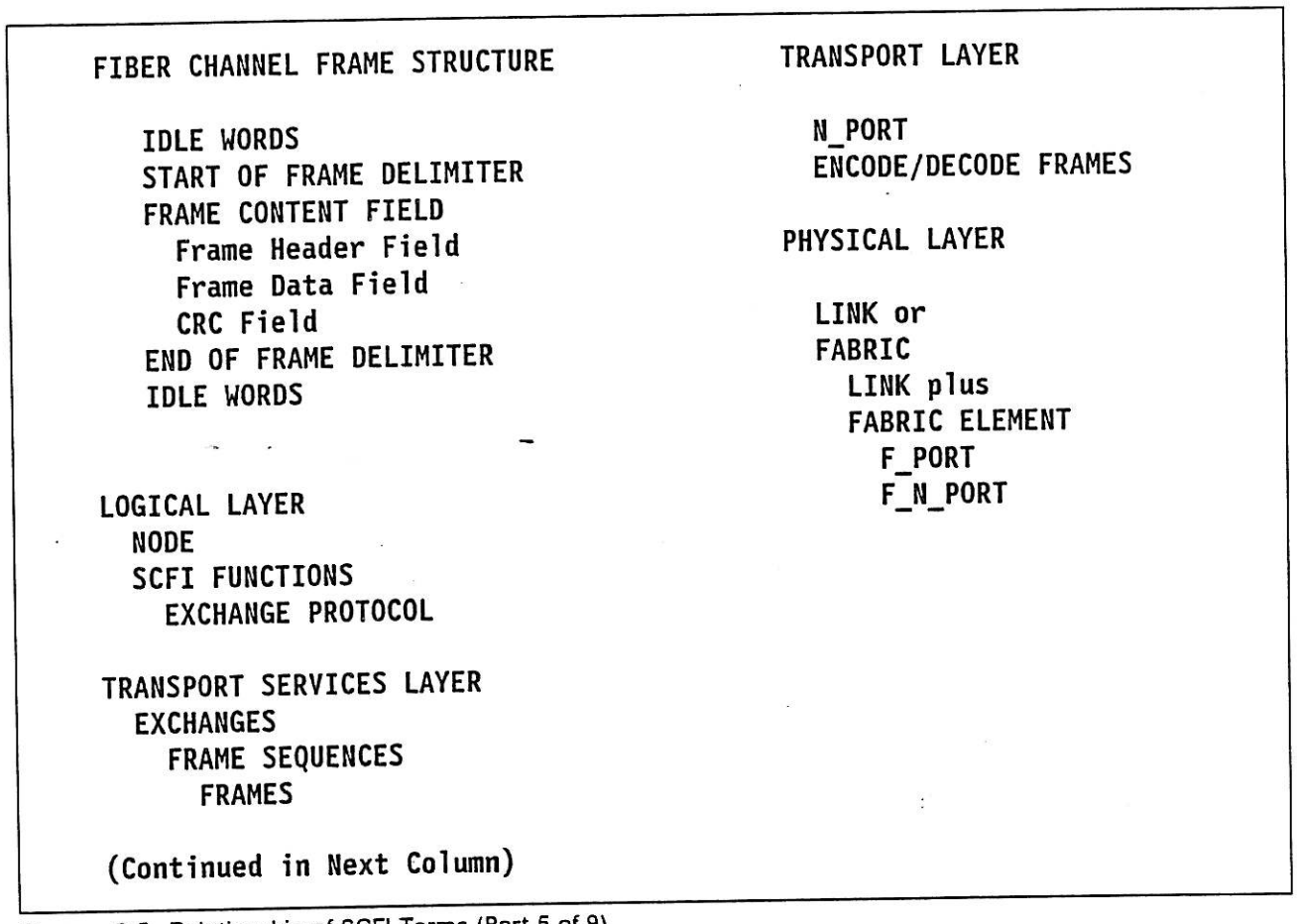


Figure 3-5. Relationship of SCFI Terms (Part 5 of 9)

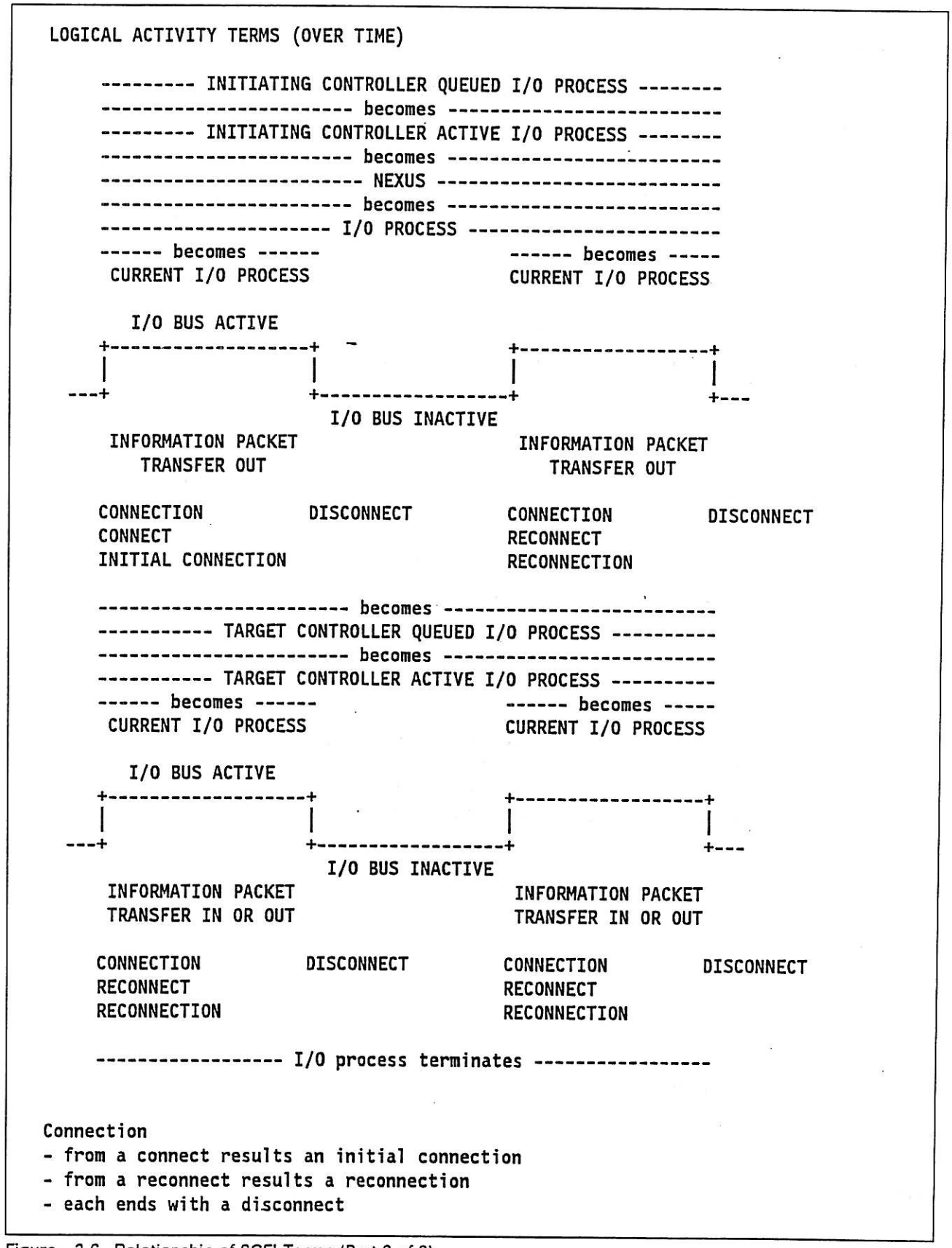
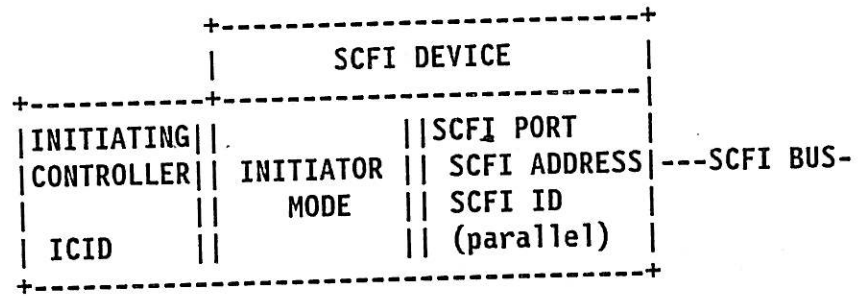


Figure 3-6. Relationship of SCFI Terms (Part 6 of 9)

SCFI DEVICE

Configurations to Transfer an Information Packet
for a Connect for an I/O Process

SCFI Device in Initiator Mode



Fiber Channel Terms

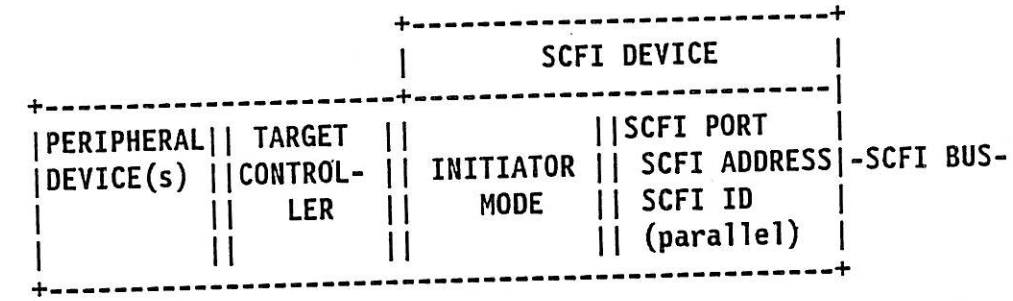
Node Identifier

Transmitter

N_port

N_port Identifier

Link



Fiber Channel Terms

Node Identifier

Transmitter

N_port

N_port Identifier

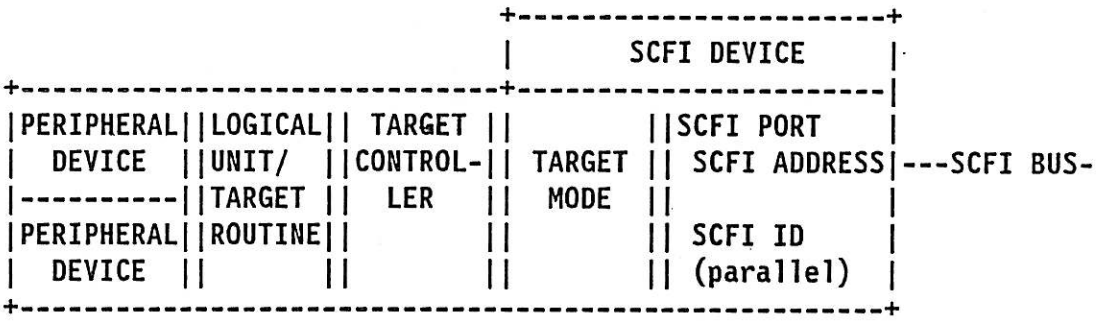
Link

Figure 3-7. Relationship of SCFI Terms (Part 7 of 9)

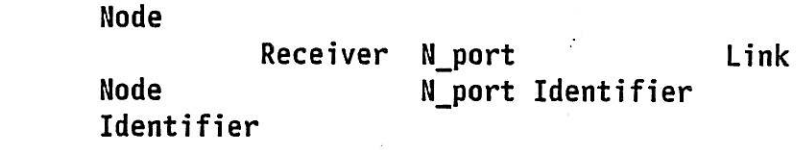
SCFI DEVICE

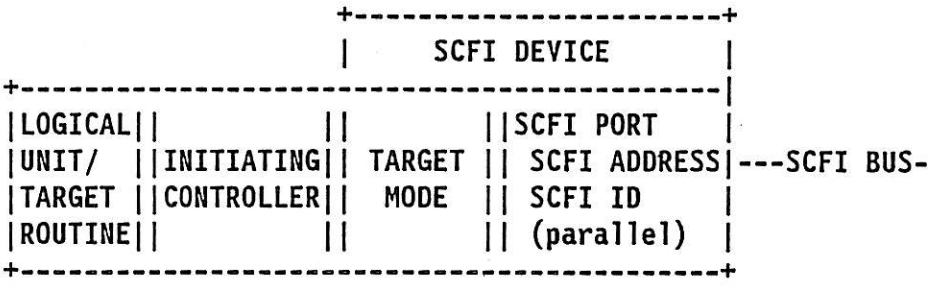
Configurations to Receive an Information Packet
for a Connect for an I/O Process

SCFI Device in Target Mode



Fiber Channel Terms





Fiber Channel Terms

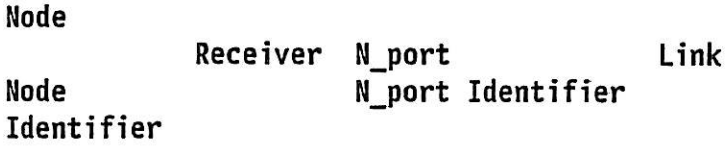


Figure 3-8. Relationship of SCFI Terms (Part 8 of 9)

COMPARISON OF INFORMATION TRANSFER MANAGEMENT

SCSI-3	SCFI (Serial or Parallel)
special phases and interface logical elements (transfer target controlled)	general phases and self-describing interface logical elements information packets transfer initiated by either logical element except the connect

COMPARISON OF BUS ACTIVITIES FOR A COMMAND

CONNECTION 1 (Connect)	CONNECTION 1 (Connect)
MESSAGE OUT phase (w/Attention) message(s) COMMAND phase command descriptor block DATA OUT phase (Opt.) command parameter data (Opt.) MESSAGE IN phase message(s) (Disconnect)	NEXUS TRANSFER phase message(s) command descriptor command parameter data (Opt.) (Disconnect)
CONNECTION 2 (Reconnect)	CONNECTION 2 (Reconnect)
MESSAGE IN phase message(s) DATA IN/OUT phase(s) logical block burst(s) or command response data STATUS phase status MESSAGE IN phase message(s) Disconnect	NEXUS TRANSFER phase message(s) logical block(s) or command response data status message(s) Disconnect

Figure 3-9. Relationship of SCFI Terms (Part 9 of 9)

Section 4. Physical Characteristics

This section specifies the physical transport layers available to SCFI. Each transport layer standard defines the physical requirements for interconnection. For specific details on the Fiber Channel, see the Fiber Channel standards. For specific details on the parallel bus, see the SCSI-3 standard, Sections 1 through 4, Section 5.1 and 5.3. See Section 2, above, for the complete titles and document numbers for these standards.

Serial implementations are incompatible with parallel implementations. Other incompatible combinations exist within these two disjoint interfaces.

A logical system may be constructed completely from fiber or with parallel cables. It may be possible to construct a bridge device to permit interoperability of a logical system compoced of part fiber and part parallel interfaces.

The parallel devices are not considered part of the fiber logical system and the parallel devices are not considered part of the fiber logical system. Rather, the bridge device appears to be a logical element with several logical units and target routines (optional). This logical element may have the attribute of being a multiple class target controller.

Handling the translation of SCFI addresses between parallel interfaces with SCFI IDs and the broader addressing capability of the serial interface is left to the discretion of the bridge device and is not a defined in this standard. (TBD for INQUIRY CRD. GRS)

4.1 Serial Interface Physical Description

SCFI devices attached to a serial interface use a 2-conductor link, called Fiber Channel. The Fiber Channel defines conductors made from copper wire or optical fiber strands. Within the optical fiber option, there are plastic and glass conductors. One conductor, the send conductor, always transfers information packets away from a SCFI device. One conductor, the receive conductor, always transfers information packets into a SCFI device. The Fiber Channel FC-0 standard defines implementations using different types of conductors.

All fiber channel operations are conducted through primitive signals sent on the conductors or through the exchange of frames. Frames have a specified format in the Fiber Channel. Frames may be variable in length with up to 2112 bytes in the frame data field of frame type 2. For inter-operability at a the frame transfer level, all SCFI devices and any associated fabric shall be capable of accepting frame type 2 with a frame data field length of a maximum of 2112 bytes.

Fiber channel defines two topologies. Either of these may be considered for SCFI implementations. The topologies are point-to-point and single inband address. Implementations using different topologies do not operate together. Within each topology there are classes of service. It is possible to design N_ports and fabric to permit multiple classes of service. SCFI devices and any associated fabric shall be capable of Class 1 and Class 2 service. Refer to the appropriate standard.

There is no interlock mechanism between sender and receiver for the physical transfer of a frame. For each frame received, there is, generally, an acknowledgement on the send conductor. With fabric, several frames from different sources may be on the link, no mechanism exists to acknowledge receipt of a frame on a byte-by-byte basis. Frame acknowledgement is used for transmission confirmation and whether the transfer included physical errors. Physical transport layers do not detect logical errors in the frame content.

SCFI identifies six mutually exclusive options in Figure 4-1 on page 4-2. Point-to-Point with Class 1 or Class 2 service or Single InBand Addressing (SIBA) with Class 1 or Class 2 service are the topology options for SCFI. Class 3 service shall not be implemented for SCFI in either topology. Fabric implementations may include strings. The functions of a fabric element attaching to a string (either end) are defined in the Fiber Channel fabric requirements. A ring fabric element operating in string mode shall be required for each N_port attached to the string.

For the SIBA topology, it may be possible to implement fabric elements which permit an intermix of the conductor types and transmission rates. Such fabric elements shall contain frame buffers to permit speed matching between dissimilar transmission rates.

	N_PORT CONDUCTOR TYPE				
TOPOLOGY	Copper	Optical		String	
		Plastic	Glass	Fabric Element	Fabric
Point-to-Point					
Frame Data Field (Max)	2112	2112	2112	N/A	N/A
Frame Buffer	Yes	Yes	Yes	N/A	N/A
Class 1	Yes	Yes	Yes	N/A	N/A
Class 2	Yes	Yes	Yes	N/A	N/A
SIBA					
Fabric					
Frame Data Field (Max)	2112	2112	2112	N/A	2112
Frame Buffer	Yes	Yes	Yes	N/A	Yes
Class 1	Yes	Yes	Yes	N/A	Yes
Class 2	Yes	Yes	Yes	N/A	Yes
String (Optional)					
Frame Data Field (Max)	2112	2112	2112	2112	2112
Frame Buffer	Yes	Yes	Yes	Yes	Yes
Class 1	Yes	Yes	Yes	Yes	Yes
Class 2	Yes	Yes	Yes	Yes	Yes

Figure 4-1. Serial Transport Layer Options

NOTE: Appropriate adapters may exist, with functions outside the scope of this standard, which would permit inter-operability between different conductor types. The most likely implementation is a fabric element capable of handling different conductor types and transmission rates.

4.1.1 Point-to-Point Topology (Optional)

A point-to-point topology exists when one conductor is outbound from the initiator to a target and the other is inbound to the initiator from the target. The two conductors are called a circuit. No other devices share the conductors. The send conductor of the initiator connects to the receive conductor of the target. The send conductor of the target connects to the receive conductor of the initiator. Figure 4-2 on page 4-3 shows an implementation of the point-to-point topology. Class 1 and Class 2 service behave identically for point-to-point topology.

Figure 4-2 provides one initiator for each target, and a link attaches them. In this implementation, every information packet received by an SCFI device has only one source. Every information packet sent has only one destination. That is, all links are dedicated to exactly two N_Ports each.

NOTE: This is not a likely to be a common SCFI implementation, except with high performance devices or for some other system integration reason.

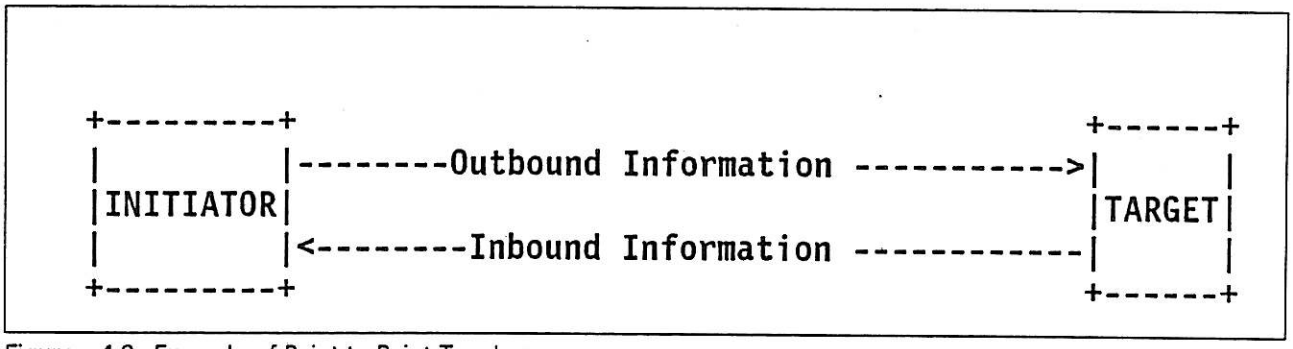


Figure 4-2. Example of Point-to-Point Topology

4.1.2 Single InBand Address Topology

4.1.2.1 Point-to-Point Operation

The most likely implementation of a SCFI bus consists of one or more fabric elements between the initiator and the target. Fabric definitions are part of the applicable standards. Initiators and targets attach to a fabric element. Fabric elements may attach to other fabric elements to provide addressability to a large set of SCFI devices. The fabric may also increase the effective distance between initiator and target.

N_ports condition the fabric elements to make a connection appear point-to-point (See Figure 4-2 above). The conditions are established at the time the connection is made. This is called Class 1 service. Once an N_port has Class 1 service established, the circuit is not shared with any other N_port and other N_ports shall not be permitted to establish Class 1 service with the two connected N_ports.

Multiple initiators and targets may attach to the fabric elements. The routes through the fabric adjust dynamically to connect an initiator and a target based on the clall of service. The mechanism for adjusting the fabric is part of the applicable standards. For SCFI, a Single InBand Address in each frame is sufficient to cause proper routing of the frame and establish the desired calss of service.

In Class 2 service, any two consecutively transmitted frames may be sent to the same or different N_ports. The actual route through the fabric for two frames to the same N_port may be different. This can cause the frames to arrive at the N_port out the sequence transmitted by the sender. This condi-

tion is not an error. The receiving N_port shall reassemble the frames into the original sequence, if required, before operating on them.

SCFI optionally supports dedicated connections (Class 1 service). Class 1 service makes connections for one or more frames. No other N_port can send to a receiving N_port until the connection is removed by explicitly by one of the N_ports participating in the connection. That is, the circuit between two N_ports is dedicated to servicing the connected N_ports.

In Figure 4-3 on page 4-5, an initiating controller can send information packets on each send conductor to a target controller. At the same time, it receives information packets on its two receive conductors. The number of available ports and the complexity of the fabric determines the number of concurrent connections possible. The maximum in this example is six concurrent connections (Class 2 service). The links between fabric elements do not add to the level of concurrent connections. For class 1 service, the maximum number of concurrent connections is three.

In Figure 4-3 on page 4-5, a sender may route each information packet to a different destination. A destination may receive information packets from more than one source. This is similar to parallel bus operation where each arbitration phase may result in connections between various initiators and targets connected to an SCFI bus. This is information packet multiplexing. It is a standard feature of the Fiber Channel.

NOTE: It is technically possible for a node with 2 N_ports to carry out a complete exchange protocol between its two ports.

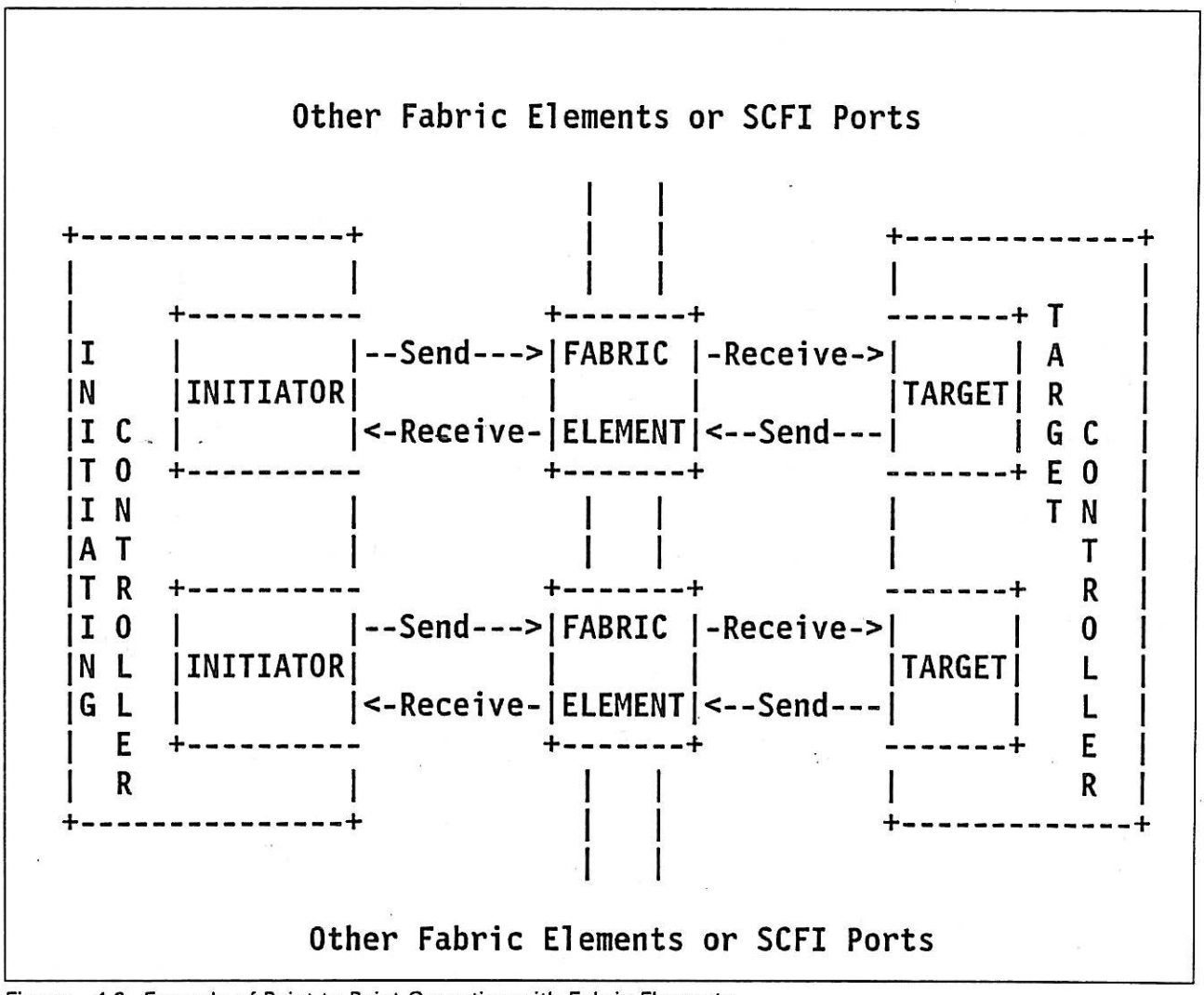


Figure 4-3. Example of Point-to-Point Operation with Fabric Elements

4.1.2.2 String Operation (Optional)

(Much of this text can be removed when Fiber Channel fills in its description of strings in the Fiber Channel Fabric Requirements document. GRS)

The Fiber Channel fabric document specifies optional ring fabric elements for implementing strings. SCFI specifies strings as an option in the SIBA topology. A string may exist as a closed fabric between the SCFI devcies attached to the ring fabric elements, or it may be included with other fabric.

A string looks similar to the parallel daisy-chained bus implementation, but it is not. Strings have one additional requirement for each SCFI port, one performance implication, and one availability implication.

A string forms a loop from the send conductor of one SCFI device port to its receive conductor. The loop is accomplished by attaching the N_port of each SCFI device to a ring fabric element. A ring fabric element is an implementation of a Fiber Channel fabric element with special frame routing routing rules.

See Figure 4-4 on page 4-7. The conductors pass through a ring fabric element for each attached SCFI port. See Figure 4-5 on page 4-8.

The effective bandwidth of a string is less than that of point-to-point operation.

The string becomes inoperative when any ring fabric element or connecting link becomes inoperative. That is, frame forwarding fails when a ring fabric element fails, a user disconnects a link between ring fabric elements, or the link breaks between ring fabric elements.

The receive conductor of a ring fabric element from the previous ring fabric element on the string, in Figure 4-5 on page 4-8, routes frames to the local SCFI port or to the send conductor of the ring fabric element for the next ring fabric element on the string. The receive conductor of the ring fabric element from the next device on the string, in Figure 4-5 on page 4-8, routes frames to the local SCFI port or to the send conductor for the ring fabric element to the previous ring fabric element on the string. The local SCFI port has its send conductor attached to the ring fabric element on a third F_port. The ring fabric element always routes a frame from the local N_port on the send conductor of the ring fabric element to the next ring fabric element on the string Each F_port on a ring fabric element has mandatory flow directions and decision points as specified above and shown in Figure 4-5 on page 4-8.

NOTE: A ring fabric element, with an option, may act as a point-to-point connection to the local SCFI device conductors of the ring fabric element. This potentially permits one implementation of the SCFI port in SCFI devices for either type of operation. This also permits a SCFI ring fabric element to be a separate component in the configuration rather than making two models of the SCFI device. If the ring fabric element has a pass-through mode, a user can add a SCFI device to or remove it from the string without disrupting string operation.

If the ring fabric element is not an expensive item, it may be part of a SCFI device. The ring fabric element becomes transparent in point-to-point mode and active in string operation.

After a sender places a frame on its send conductor from the local N_port, each ring fabric element on the string, in turn, receives the frame. It checks to see if the address is for its local SCFI port. If it is not for the local port, the ring fabric element places the frame on its send conductor to the next/previous ring fabric element on the string.

The sender eventually receives an improperly addressed information packet with the source ID equal to its own N_port ID (i.e., a loop). Since the sender has a ring fabric element port, endless cycling of mis-addressed frames could reduce string bandwidth. A simple rule in the ring fabric element prevents frame cycling. If a frame at the F_port for the previous ring fabric element has either its source or destination ID field identifying the port, the ring fabric element routes the frame to the local N-port. The initiating controller or target controller carries out any necessary error recovery.

For frames for the local SCFI port, the ring fabric element routes the frame to the receive conductor of the local SCFI port. The SCFI port will then send an acknowledgement, as appropriate, on its send conductor, addressed to the sender of the original frame. The ring fabric element always routes this frame on its send conductor to the next ring fabric element on the string.

With a string, fabric may separate ring fabric elements. See Figure 4-4 on page 4-7. A connection to the F_port for the string is always 2-conductors.

A string with a switched fabric element operating with Class 2 service attached can route frames into the string and out of the string to open up the string to initiators and target controllers outside the string on regular N-ports. Thus, a switched fabric element attached to a string located remote from other N_ports on the fabric can provide access to external SCFI initiating controllers and target controllers. The bandwidth of the string devices may be slower than those operating outside the string.

NOTE: A medium changer and associated devices can be located remote from the general processing area for disaster recovery or offsite backup and recovery operations. An archive installation can also be located remote from an active processing area to reduce operational complexity.

An initiator on the string in Figure 4-4 sends a frame on its send conductor. Each ring fabric element receives, checks, and forwards frames to the next ring fabric element. When it reaches its destination or it completes only one loop around the string, it is removed from the string.

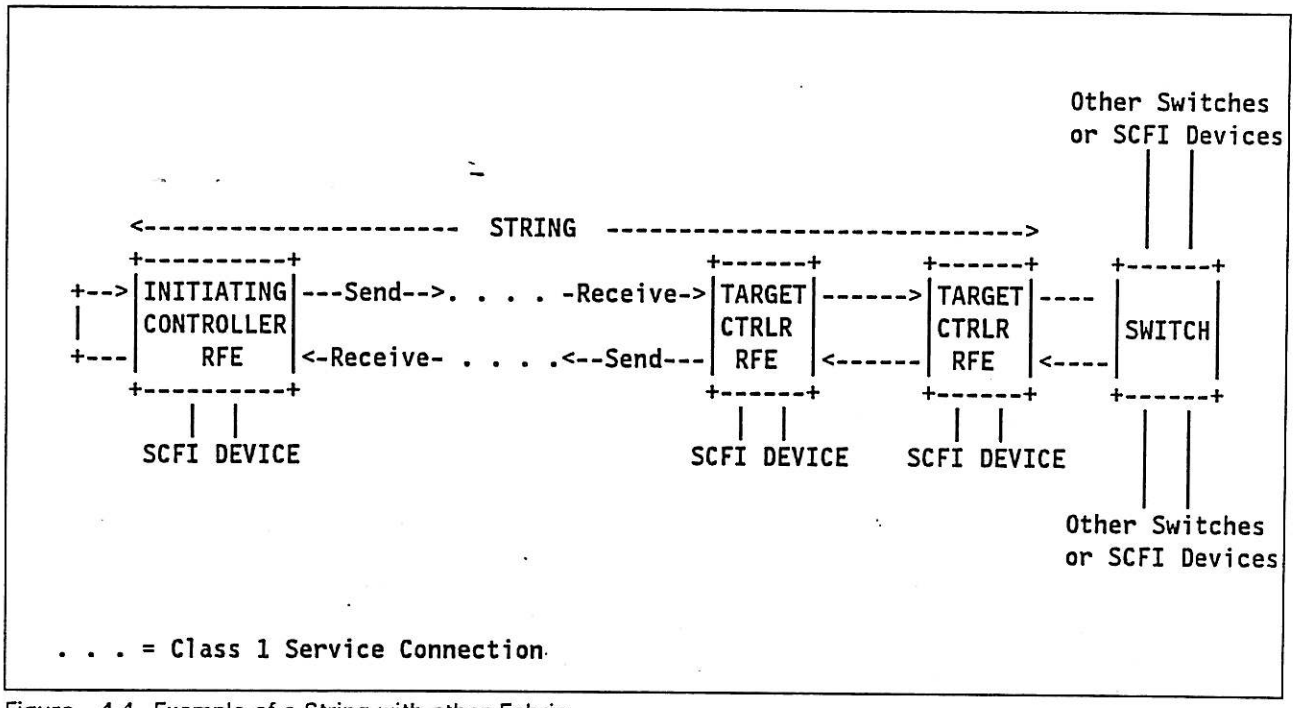


Figure 4-4. Example of a String with other Fabric

If the SCFI port, represented as an initiator in Figure 4-5 on page 4-8 as part of the string, must attach to a ring fabric element. It ring fabric element forwards frames between any two SCFI ports on the string (i.e., normal SCFI peer-to-peer operation) the same as any other SCFI device on the string. The link length, possibly enhanced by fabric elements, permits widely dispersed SCFI devices on the string.

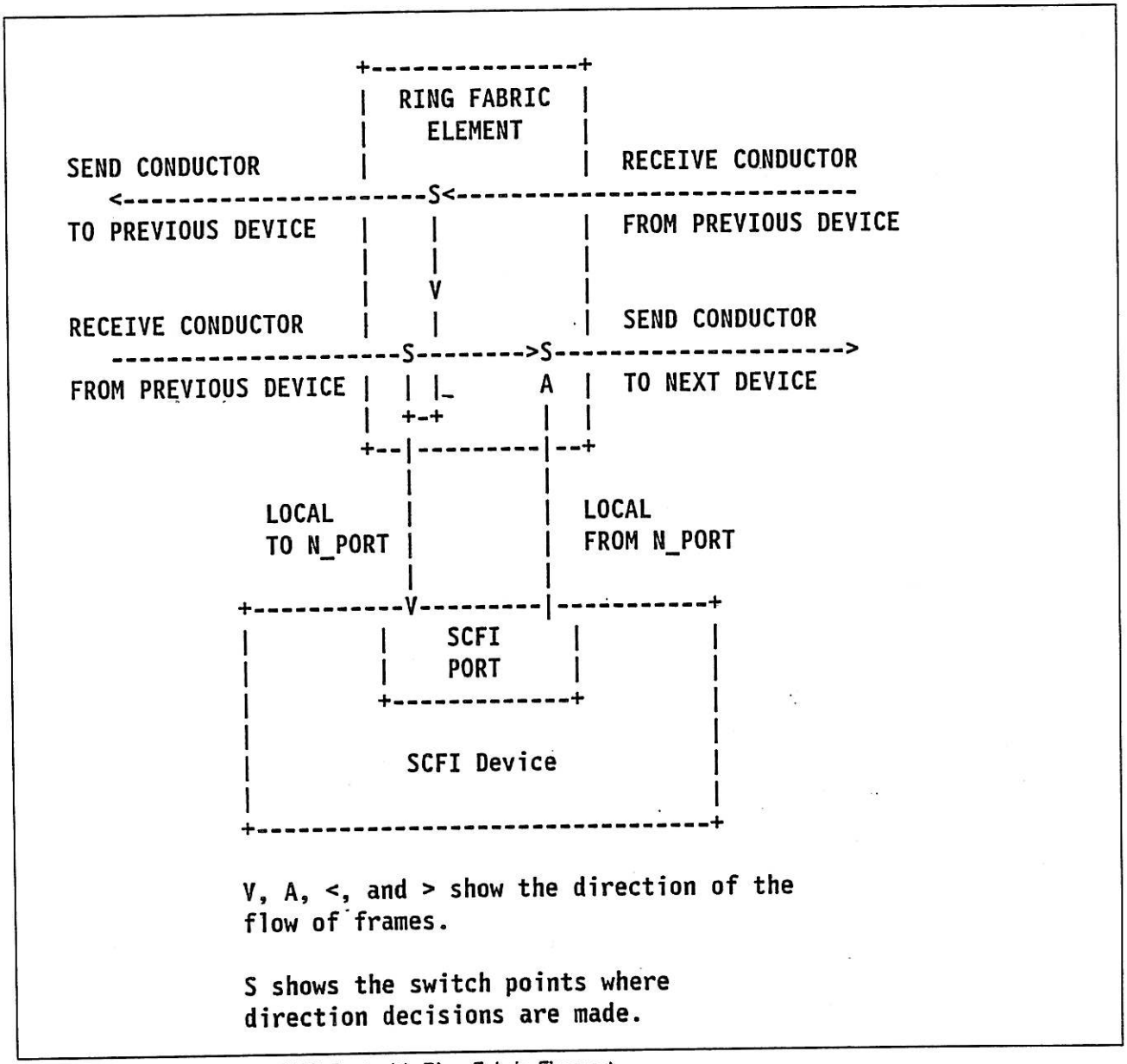


Figure 4-5. Example of a SCFI Port with Ring Fabric Element

4.2 Parallel Interface Physical Description (Optional)

The implementer has a selection of busses, shown in Figure 4-6 on page 4-9, from the SCSI-3 standard. The "X" identifies valid SCFI options.

Single ended and differential implementations are mutually exclusive. SCFI does not support intermixing 8-bit, 16-bit, and 32-bit devices. All SCFI devices which are principally initiating controllers shall supply terminator power as required for termination option A2.

Electrical Type	Connector Type			
	Shielded		Non-Shielded	
Single Ended (6 M)	A1	A2	A1	A2
8-Bit (A Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
16-Bit (P Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
32-Bit (Q Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
Bus Width Intermix				
Termination A1	-	-	-	-
Termination A2	-	-	-	-
Differential (25 M)				
8-Bit (A Cable)				
5 MHZ	-	-	-	-
Fast Synchronous	X	-	X	-
16-Bit (P Cable)				
5 MHZ	-	-	-	-
Fast Synchronous	X	-	X	-
32-Bit (Q Cable)				
5 MHZ	-	-	-	-
Fast Synchronous	X	-	X	-
Bus Width Intermix				
5 MHZ	-	-	-	-
Fast Synchronous	-	-	-	-

Figure 4-6. Parallel Transport Layer Options

Section 5. SCFI Information Transfer

SCFI information transfer is divided into two parts:

1) serial interface in Sections 5.1 through 5.4.

2) parallel interface in Sections 5.5 through 5.8.

In SCFI, the nexus transfer phase is used for all information transfers across a SCFI bus. Interface logical elements (bursts for some) are formed into self-describing units and placed into an information packet of variable length. The information packet is transferred using the nexus transfer phase. SCFI requires that all interface logical element lengths in a single information packet be declared before information packet transfer. Interpretation of the data in the information packet shall not be permitted which halts the transfer of the information packet on the interface. That is, SCFI permits no byte-by-byte processing.

For the parallel bus, the initial information packet transfers are in asynchronous data transfer mode. SCFI devices shall not have an interlock have an interlock mechanism to identify individual bytes for protocol or parity errors as they are transferred. Only the actual transfer time may be slower than in synchronous data transfer mode. Parity errors are handled by retransmitting the information packet. For serial interface implementations, all transfers are in the equivalent of synchronous data transfer mode.

The information transfer phases defined in SCSI-3 may be used as an alternate mode of operation for SCFI devices operating a parallel bus. After detecting a SCSI-3 device on parallel bus, a SCFI device may change its definition to the SCSI-3 operating mode. If such a change occurs, the SCSI-3 standard shall shall control all protocol between the SCSI-3 operating mode level devices.

5.1 Serial Bus Phases

The SCFI serial bus includes two distinct phases:

- BUS FREE phase
- NEXUS TRANSFER phase

The SCFI bus, as determined on a fiber at a SCFI port, can never be in more than one phase at a time. There is no direct signal interlock between SCFI ports on a link.

An active serial interface consists of a continuous stream of encoded words transmitted along one conductor between a source and a destination. Since there are two conductors for each SCFI port, the other conductor is transmitting a continuous stream of encoded byte values from a destination to the source.

The receiving port may not be aware that it is about to receive an information packet. The sending SCFI port may not be aware of the present state of the destination SCFI port. The protocol assumes delivery of information packets and recovers when that assumption proves incorrect. That is, information packet delivery is not guaranteed at the time of transmission in some implementations. The exchange protocol provides for positive responses from the opposite end of the link at some time after transmission. To prevent endless acknowledgement, the last information packet declares an end to the exchange protocol.

For point-to-point topology, the source has complete control of when an information packet is placed into this stream of encoded bytes. The destination also has complete control of when an information packet is placed into the stream of encoded bytes on the reverse conductor.

F_ports control the time when an information packet is placed into its send conductors. The F_port must first detect the BUS FREE phase for the appropriate conductor. When the BUS FREE phase is detected, the F_port then places a pending information packet into its send conductor.

Because of the two independent conductors in the serial interface for an SCFI port, the discussions below apply independently to each conductor.

For each information packet successfully received, the receiving SCFI port shall acknowledge receipt by sending an acknowledge information packet on the reverse conductor as defined in the exchange protocol. For certain exchange protocol, this acknowledgement is transmitted by the SCFI port hardware. Although permitted in Fiber Channel frame sequences, SCFI shall not cause acknowledgements to be suppressed. This is consistent with SFCI parallel bus operations which give positive confirmation for bus activities (except hard reset).

For each information packet which contains a logical error, but which is otherwise received error free, the receiving SCFI port shall send an acknowledge information packet on the reverse conductor according to the rules of the exchange protocol. The logical error is reported using another portion of the exchange protocol. The receiving SCFI port shall not process this information packet further and shall not retain it.

For each information packet received with a physical error, such as a CRC error, the receiving SCFI port shall not acknowledge receipt of the information packet. The port shall not process this information packet further and shall not retain it.

5.1.1 BUS FREE Phase

The BUS FREE phase indicates that no SCFI device is actively using a fiber and that it is available. SCFI devices shall detect the BUS FREE phase on a receive conductor when the idle encoded word is detected on two consecutive word boundaries. The send conductor is in the BUS FREE state unless the N_port is transmitting a frame.

SCFI ports normally do not expect BUS FREE phase to begin except after one of the following occurrences:

(1) after the transfer of a complete information packet in the nexus transfer phase.

(2) ....

On detecting a the BUS FREE phase at any other time, the receiving SCFI port shall manage this condition as an unexpected disconnect event.

If a receiving SCFI port detects the BUS FREE phase at any other time, the sending SCFI port may not be aware that an error has occurred in the transmission of the information packet. The information packet is not acknowledged. (FC-1 idle injection rule? GRS) On detecting a missing information packet, the sending SCFI port shall manage this condition as an unsuccessful information packet transfer condition.

5.1.2 Information Transfer Phase

The nexus transfer phase is known as the information transfer phase because it is used to transfer data and control information via the SCFI bus. The actual content of the information is beyond the scope of this section. Table 5-1 on page 5-3 identifies the only phase of a functional serial bus.

Two outbound frames may be transmitted back-to-back if there is a pause containing at least the minimum number of idle words between the information packets as specified by the Fiber Channel standard and such transmission is permitted by the exchange protocol.

Two inbound phases may be processed back-to-back if there is a pause containing at least the minimum number of idle words between the information packets as specified by the Fiber Channel standard. The two frames may have no relation to each other.

Table 5-1. Serial Interface Information Transfer Phase

Phase Name	Direction Of Transfer
NEXUS TRANSFER	Initiator to target; Initiator from target

5.1.3 NEXUS TRANSFER Phase

The nexus transfer phase is a term that encompasses information transfer between two logical elements. Only the relative location of the sender in an I/O process determines which direction is implied. The words "in" and "out" show the information packet flow directions with respect to an initiating controller. "Out" means sending from the initiating controller. "In" means sending to an initiating controller.

The nexus transfer phase is performed as a synchronous data transfer. Synchronous data transfer is the only mode of operation for the serial SCFI port. The offset between source and destination is logically controlled by the number of information packets outstanding, rather than the number of bytes, as for the parallel interface.

The NEXUS TRANSFER phase allows a logical element to send information packet(s) to other logical elements. Synchronous packet transfer negotiation is defined. The sending logical element shall abide by the current parameters of a synchronous packet transfer negotiation.

The sender shall insert an information packet on its send conductor when it has an information packet pending and the idle word has been detected for a minimum number of times since the last information packet transfer or final initialization of the SCFI port. (Refer to the Fiber Channel standard.)

5.2 Serial Interface Bus Events/Conditions

The SCFI bus has two asynchronous events: the reset event (Fabric purge? GRS); and and the unexpected disconnect event. Each event establishs a condition which causes the logical element to perform certain actions.

5.2.1 Serial Reset Event

(This may not exist on the Fiber Channel; Primitive Signals? GRS)

The reset event is used to clear immediately all frames from the SCFI bus. This event shall take precedence over all other phases and events. Any SCFI device may create the reset event by broadcasting a reset information packet. (??? GRS) (See the controlling serial standard.)

On detection the reset event, all SCFI devices shall perform an an unexpected disconnect on their send conductors if they are actively sending an information packet. The BUS FREE phase always follows the reset event.

(See Section 6.? for the effect of the Reset event on logical operations. GRS)

5.2.2 Serial Unexpected Disconnect Event

SCFI devices normally expect BUS FREE phase to begin on its receive fiber after one of the following occurs:

(1) after a reset event is detected.

(2) after the transfer of one or more information packets in the nexus transfer phase.

All other occurrences of a BUS FREE state cause the SCFI device to establish an Unexpected Disconnect Condition for the logical element. The receiving SCFI device shall handle this as an unexpected disconnect condition.

5.2.3 Serial Unsuccessful Information Packet Transfer Condition

(Section 6 material. GRS)

For an unsuccessful information packet transfer condition, the sending SCFI port:

1) may attempt to retransmit the information packet in error up to a limit determined by the sending SCFI port.

2) if operating in target mode for the I/O process, shall terminate the I/O process after the specified number of retries by clearing all pending data and preparing sense data.

3) if operating in initiator mode for the I/O process, shall abort the I/O process. The target controller prepares sense data. It is recommended that the initiator then request sense data from the target.

5.2.4 Serial Unexpected Disconnect Condition

(Section 6 material. GRS)

5.3 Serial Bus Phase Sequences

5.3.1 Serial Bus Phase Order

The order phases are used on the SCFI bus follows a prescribed sequence.

The reset event can abort a nexus transfer phase, and it is always followed by the BUS FREE phase. Each nexus transfer phase shall be followed by the BUS FREE phase.

If the bus is not in the BUS FREE phase after the reset event, further operation on that SCFI port shall not be attempted.

Figure 5-1 shows the allowable phase sequences. The normal progression is from the BUS FREE phase to one of the nexus transfer phases for each conductor. The nexus transfer phase used is determined by whether the SCFI port is in initiator mode or target mode. For initiator mode, the phase used is NEXUS TRANSFER OUT. For target mode, the phase is NEXUS TRANSFER IN. Each nexus transfer phase shall be followed by the BUS FREE phase.

5.3.2 Signal Restriction Between Phases for Serial Busses

When the SCFI bus is between two information transfer phases, the sending SCFI port shall wait until a specified minimum number of idle words have been transferred on the conductor before attempting to send another information packet. See Figure 5-1. (Refer to the Fiber Channel standard.) The exchange protocol logically may restrict the transmission of another information packet for the same exchange.

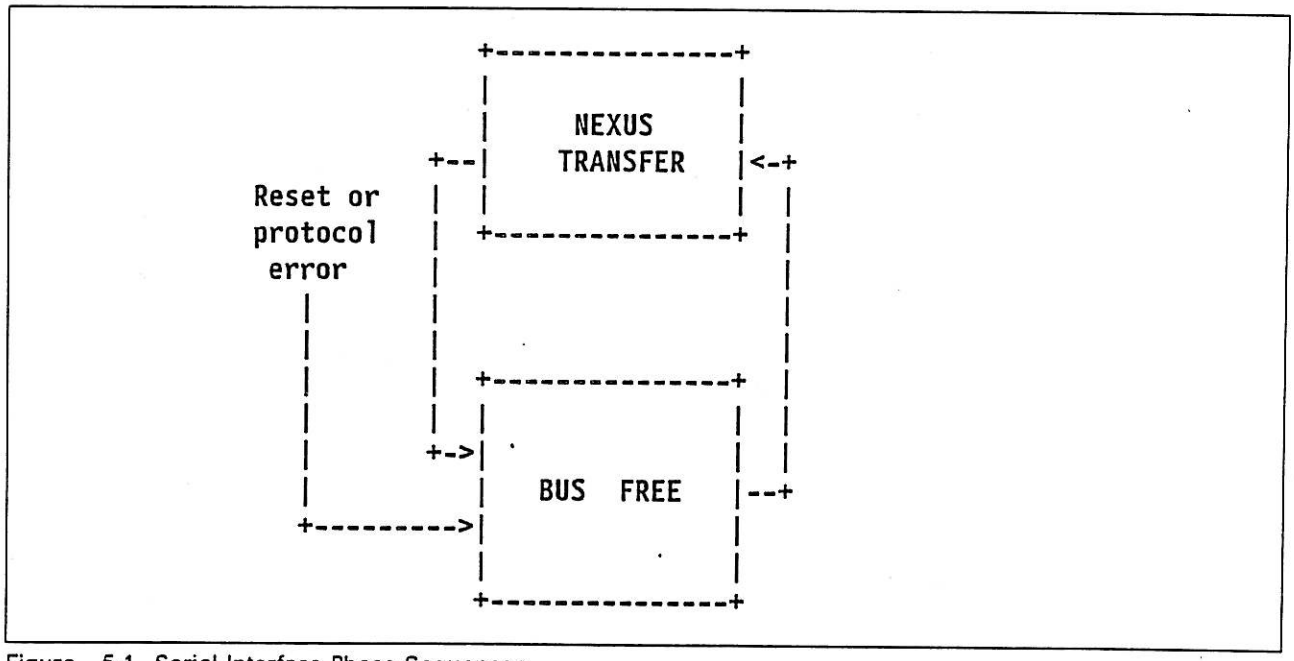


Figure 5-1. Serial Interface Phase Sequences

5.4 Serial Exchange Protocol

The serial exchange protocol contains the valid sequences of exchanges for the Fiber Channel to carrying out SCFI operations. This section specifies how SCFI uses Fiber Channel and does not present a description of the full capabilities of Fiber Channel. The exchange protocol is composed of four elements:

- Frames
- Frame Sequences
- Exchanges
- Exchange Sequences

Frames have a specific layout required to perform operations. There are three frame types. Each has a frame header field and frame content. The distinguishing characteristic, from a transmission standpoint, is the length of the frame data field. Frame type 0 has a zero length frame data field. Frame type 1 has a 32-byte frame data field. Frame type 2 has a variable length data field from 0 to 2112 bytes in multiples of 4 bytes. The use of the various frames as Fiber Channel commands and responses forms the transport mechanism to carry out SCFI operations

5.4.1 SCFI Serial Frames

Frame type 0 contains two link commands and responses for use by SCFI. Only one of these is actually available to the SCFI device, the N_Port Ready, or P_RDY, frame. The P_RDY frame is a link response. The F_Port Ready, or F_RDY, frame is part of an exchange of frames between an N_Port and its nearest neighbor on the fabric, an F_Port. The response paces frame transfers into and out of the fabric, but it is not made available to the SCFI device. The other two frames of frame type 0 are not used. These frames are not described.

Frame type 1 is also used for other types of link commands and responses. All of the frames in this frame type, used by SCFI, are made available to the SCFI device. The following frames of frame type 1 are used:

- Establish Service (ESTS)
- Remove Service (RMVS)
- Establish Exchange (ESTX)
- Abort Exchange (ABTX)
- N_Port Accept (P_ACC)
- Fabric Accept (F_ACC)
- N_Port Reject (P_RJT)
- Fabric Reject (F_RJT)
- N_Port Busy (P_BSY)
- Fabric Busy (F_BSY)

Frame type 2 is used for all of the logical operations of SCFI. The frames are frames used in ULP exchanges. These frames carry the logical content of SCFI. Frame types 0 and 1 perform essential Fiber Channel operations and provide acknowledgements for frame type 2 transmission. The frames used in the frame type 2 group are:

- Write Device Data (WLD)

Fiber Channel uses these frames from frame type groups 0, 1, and 2 to form frame sequences. The Fiber Channel specifies the exact sequence the frames shall be transmitted, the normal response, and the abnormal responses. The frame sequences are the building blocks for conducting logical operations on the Fiber Channel.

Link responses are frames exclusively used to respond to receipt of other frames. Figure 5-2 on page 5-7 specifies the link responses and the commands (link or ULP) with which they are used. The letters for each response are used in Figure 5-3 on page 5-8 to show which responses are used with each Fiber Channel command.

Fiber Channel Frame Sequence Link Responses (FC, Clause 9)

No	Link Response	Frame Type	XCH	Link/ULP Exchanges Command(s)
a	N_Port Accept (P_ACC)	1	E	1, 2, 3, 4
b	Fabric Accept (F_ACC)	1	E	1
c	N_Port Reject (P_RJT)	1	E	1, 2, 3, 4, 5, 6, 7, 8
d	Fabric Reject (F_RJT)	1	E	1
e	N_Port Busy (P_BSY)	1	E	1, 2, 3, 4, 5, 6, 7, 8
f	Fabric Busy (F_BSY)	1	E	1, 2, 3, 4, 5, 6, 7, 8
g	N_Port Ready (P_RDY)	0	-	5, 6, 7, 8

LEGEND:

XCH = Exchange
E = Exchange ends
- = Exchange continues (except last)

Figure 5-2. Fiber Channel Frame Sequence Link Responses

5.4.2 SCFI Serial Frame Sequences

Figure 5-3 on page 5-8 specifies the frame sequences, the frame types used, the expected responses, the abnormal or error responses, and the interlocked or streaming characteristics of each frame sequence.

In Figure 5-3 on page 5-8, frame sequences 1-3 are reserved for Fiber Channel initialization and occur independent of SCFI operations. SCFI operations may not start until certain of these initialization sequences successfully complete. Frame sequence 4, Abort Exchange, is used for events which require an unexpected disconnect by a SCFI device for an active exchange. The Abort Exchange sequence shall only be used when normal SCFI logical operations can not perform the task. Frame sequences 5-8 are used in various combinations to perform SCFI operations. Frame sequence 9 is used only for operations which do not require a response (e.g., BUS DEVICE RESET message). Frame sequences shall only be used as described in this section (5.4).

A set of sequences, logically arranged to some purpose, bounded by the same exchange ID (XID) in each frame is called an exchange. Examples of exchanges are a set of frame sequences to perform a REQUEST SENSE command and to write data to a logical unit. Figure 5-4 on page 5-10. shows examples of frame sequences formed into exchanges. Three versions of the same WRITE command are shown.

The example on the left side shows a very simple single exchange. In sequence A, the initiator establishes a nexus and sends the write command in an information packet in the frame data field of the WLD frame (the frame is acknowledged by the receiver with a P_RDY link response); the target controller responds that it is ready to transfer through its WLD response to the first frame of the exchange (the frame is acknowledged by the receiver with a P_RDY link response). There is no sequence B for this example. In sequence C, The initiator then begins to stream bursts of logical block data to the target controller; the target responds with a P_RDY link response to each frame received. When the

last frame of logical block data is transmitted, the initiating controller signals "no more data" to the target controller. Sequence (D) begins with a WLD frame from the initiating controller which requires a WLD response from the target controller. The response frame in sequence D contains status, intidating successful completion, and the appropriate command completion message. This entire exchange must be conducted on the path where the connect was made because it consists of a single exchange.

Fiber Channel Frame Sequences for SCFI (Fiber Channel, Clause 9)

No	First Frame of Sequence		Frame Type	Seq Type	I/S	Expected Response	Abnormal Response(s)
1	Establish Service	(ESTS)	1	A	Ii	a,b	c,d,e,f
2	Remove Service	(RMVS)	1	A	Ii	a	c,e,f
3	Establish Exchange	(ESTX)	1	A	Ii	a	c,e,f
4	Abort Exchange	(ABTX)	1	A	Ii	a	c,e,f
5	Write Device 1 Frame	(WLD)	2	C	Ir	g+5 from 5	c,e,f
			2	C	Ir	g from 6	c,e,f
6	Read Device 1 Frame	(WLD)	2	C	Ir	g+5	c,e,f
7	Write Device Data	(WLD)	2	C	Ii,So	g	c,e,f
8	Read Device Data	(WLD)	2	C	Ii,Si	g	c,e,f

LEGEND:

I/S = Interlocked/Streaming Attributes

From x - Indicates which frame started the sequence
Starting at 5, the frames exchanged are 5,g,5,g
Used with command parameter data (e.g., mode select CPD)
Starting at 6, the frames exchanged are 6,g,5,g
Used with command response data (e.g., sense data CRD)

Ii - Interlocked - Sequence does not end until response received for last frame Sequence Ends with Response

Ir - Interlocked - Sequence does not end until response received only one frame Sequence Ends with Response

Si - Commands may be streamed from the destination (same exchange)

So - Commands may be streamed from the originator (same exchange)

\+ - P_RDY response plus another frame exchange in the opposite direction

Expected Response = Link Response or ULP Exchange Expected

Abnormal Response = Link Response or ULP Exchange-Unexpected
Responses with letters from Figure 5-2 on page 5-7

Figure 5-3. Fiber Channel Frame Sequences for SCFI

The middle example is similar, except that the target controller responds in Sequence A that it is not ready to begin transfer immediately. The initiating controller puts the nexus in suspense until the target controller, through Sequence B, indicates that it is ready to begin transfer. The remainder of the exchange is identical to the left example. This entire exchange must be conducted on the path where the connect was made because it consists of a single exchange (i.e., single path mode).

The example on the right uses how multiple exchanges to perform the same operation. Although not required, the two exchanges may be processed on different paths if multiple path mode is allowed for the I/O process. Sequence A is similar to the other two examples. The target controller ends the exchange with a "not ready to transfer" indication, as in the middle example, and ends the exchange. The target may now reconnect on any path in the path group where the connect was made to continue the I/O process. Since the remainder of the protocol is a single exchange, it is restricted to the chosen path.

Sample Fiber Channel Exchanges (Fiber Channel, Clause 12)

SEQ	Write Immediate Init. Targ.	Write Deferred Init. Targ.	Write 2 Exchanges Init. Targ.
A)	WLD(x1)-se --> <-- P_RDY(x1)	WLD(x1)-se --> <-- P_RDY(x1)	WLD(x1)-se --> <-- P_RDY(x1)
B)	<-- WLD(x1)-se P_RDY(x1) -->	<-- WLD(x1)-se P_RDY(x1) -->	<-- WLD(x1)-xe P_RDY(x1) -->
C)	n/a	<-- WLD(x1) P_RDY(x1)-se -->	<-- WLD(x2) P_RDY(x2)-se -->
D)	WLD(x1) --> WLD(x1) --> <-- P_RDY(x1) . . . WLD(x1)-se --> <-- P_RDY(x1) <-- P_RDY(x1)	WLD(x1) --> WLD(x1) --> <-- P_RDY(x1) . . . WLD(x1)-se --> <-- P_RDY(x1) <-- P_RDY(x1)	WLD(x2) --> WLD(x2) --> <-- P_RDY(x2) . . . WLD(x2)-se --> <-- P_RDY(x2) <-- P_RDY(x2)
E)	WLD(x1)-se --> <-- P_RDY(x1)	WLD(x1)-se --> <-- P_RDY(x1)	WLD(x2)-se --> <-- P_RDY(x2)
F)	<-- WLD(x1)-xe P_RDY(x1) -->	<-- WLD(x1)-xe P_RDY(x1) -->	<-- WLD(x2)-xe P_RDY(x2) -->

LEGEND:

se - sequence end
xe - sequence end, exchange end
(xn)- exchange number in example
n/a - not applicable

Figure 5-4. Sample Fiber Channel Exchanges

As a fourth modification, the WLD frame from the target controller and one or more WLD sequences in Sequences B and C make up individual exchanges. This permits each subset of WLD frames to be transferred from the initiating controller on different paths to the target controller. In this example, the initiating controller also would has the choice of path on which to receive the final status for the command.

5.4.3 SCFI Serial Exchanges

SCFI permits two very different modes of operation which affect the selection of exchanges to perform the same operation. The two modes are single path mode and multiple path mode. The conditions established between an initiating controller and a target controller determine which mode is used for a nexus. The mode, in turn, affects the choice of exchanges to accomplish the operation. SCFI specifies a sets of exchanges to use with I/O processes for each mode of operation.

A set of exchanges to accomplish the logical function of an interface is called an exchange protocol. Frames form frame sequences; frame sequences form exchanges; exchanges form exchange protocols. SCFI has an exchange protocol for single path mode; and, one for multiple path mode, where appropriate.

The exchange protocol defines, from a Fiber Channel viewpoint, how errors are handled at the physical level. Some functions, such as automatically resending of a frame, may be accomplished in the Fiber Channel transport layers. Some errors require the SCFI logical layer to interpret the events and frame content and perform the necessary actions. One such action is aborting exchanges. Aborting an exchange is not the same as aborting an I/O process. The I/O process may continue with new exchanges as specified in the exchange protocol.

The following sections specify the combinations of frame sequences used in SCFI exchanges, combinations of exchanges to transport the logical functions, and the overall exchange protocol including error recovery for Fiber Channel problems. The exchange protocol identifies ULP exchange sequences for SCFI. Each I/O process, and specifically command execution, will use combinations of these exchange sequences. For each nexus, it shall be possible to map the required function into the defined sequences. Other sequences may be possible, but they are not defined for use in SCFI.

5.4.4 SCFI Logical Operations Characterization

Figure 5-5 on page 5-14 shows the valid combinations of I/O processes with less than or equal to one command per information packet. That is, I/O processes with linked commands are made up of a repetition of single command operations at the Fiber Channel. Each valid combination is assigned an operation type number. All SCFI logical operations can be mapped into one cell of the table. Some cells represent error conditions which interrupt the expected flow of exchange(s) and frame sequences (e.g., CHECK CONDITION status). The rules governing multiple commands per information packet are given later in this section.

Initiating controllers have only five different ways of starting or executing I/O processes which contain at most one command per information packet. Target controllers have up to eight ways to respond, including errors. Most I/O processes will follow the expected path. However, the mapping of SCFI is not complete until the error situations are considered.

The two primary error situations which alter the sequences are aborting an I/O process and terminating an I/O process. These two actions alter the normal flow of exchanges and frame sequences. In addition, single path mode and multiple path mode sequences must be covered.

Cells not numbered in Figure 5-5 on page 5-14. have no defined sequence in SCFI. Numbers in cells identify the SCFI Exchange which shall be used. Only operations in the numbered cells shall be used.

5.4.5 SCFI Serial Exchange Protocol - Normal Operation

Figure 5-5 on page 5-14 identifies six operation types for SCFI. These operations are defined for simple I/O processes.

5.4.5.1 Simple I/O Process (Mandatory)

A simple I/O process is one where the initiating controller sends only one command per I/O process, or sends only one command at a time for a linked set of commands. That is, only one command is transferred to the target controller at a time. When status and a command completion message are received for a command with the link bit set to one, the initiating controller sends an information packet to the targetr controller with the next, or last, command of the I/O process.

For example, consider an I/O process with five write commands linked together to write five different logical blocks, each 512 bytes each. The initiating controller constructs a simple I/O process by placing each command and its logical block data into five separate information packets. The initiating controller transfers the first information packet. When status and the command completion message are received, the initiating controller sends the second information packet. The process is repeated until all five information packets and their responses have been successfully transferred. The I/O process terminates.

Figure 5-5 on page 5-14 defines the characteristics of simple I/O Processes.

5.4.5.2 Complex I/O Processes (Optional)

By contrast, a complex I/O process involves queuing multiple information packets at the target controller which contain all or more than one command for an I/O process. The characteristics of SCFI operation types define segments of a complex I/O process. The segment boundaries shall be arranged so that each segment fits one of the operation types defined in Figure 5-6 on page 5-15. The initiating controller shall not send more than one segment of a complex I/O process at at time to a target controller.

For example, a complex I/O process might consist of a set of linked commands which write twenty-five logical blocks to twenty-five disjoint logical blocks for the logical unit. This requires twenty-five commands, one for each logical block. The logical block data, say 512 bytes per logical block, is intermingled with the commands.

The initiating controller could send nine information packets, with three command and the corresponding logical block data in the first eight information packets and only one command plus its logical block data in the ninth information packet. Alternatively, the initiating controller could place each command and its logical block data in a separate information packet.

The status and message ILE lengths, plus the possibility of an error which would return autosense data indicates that about six commands, or two information packets make up a segment. The initiating controller then partitions the I/O process into five segments of two information packets each. The last segment has only one information packet.

The target controller can provide an information packet for each command as it is completed or it may collect the status and message ILEs into one information packet and transmit it after the last command of each segment. The initiating controller then sends the next segment to the target controller when it determintes that the segment is complete.

Complex I/O processes are broken into two or more segments as follows:

1. For any operation type selected, the total information transfer shall follow the exact rules of the operation type (e.g., 2 linked read commands or 2 linked write commands, but not one of each linked together Also two reads for greater than the maximum frame length each cannot use operation Op2).
2. For linked commands which do not fit rule 1, the commands shall be segmented so that each segment meets rule 1.

3. At the end of each successfully completed segment, the initiating controller transfers an information packet to start the next segment.

4. If the target controller detects an invalid segment it shall process the segment up through the last command or message interface logical element and then terminate the I/O process. The initiating controller may start a new I/O process at the point of termination. The original I/O process shall not be continued.

5.4.5.3 Operation Types

Each operation type is composed of one or more exchanges. Initiating controllers and target controllers shall follow the prescribed sequences of exchanges and the frame sequences within each exchange.

Each operation type is described more fully in Figure 5-6 on page 5-15. The number of exchanges is defined for normal operation completion. The sequences of exchanges and the frame sequences are given with the frame sequence numbers being taken from Figure 5-3 on page 5-8. Abnormal sequences are identified in Figure 5-7 on page 5-17, Figure 5-8 on page 5-18, and Figure 5-9 on page 5-19. Complete state diagrams are given in Figure 5-10 on page 5-20 through Figure 5-14 on page 5-24.

For Figure 5-6 on page 5-15, multiple path mode is allowed for two operation types: Write Data-MP and Read Data-MP. The initiating controller and target controller must negotiate this before or at the start of the I/O process. Certain conditions must be present at the start of the I/O process to successfully operate in multiple path mode. If all conditions are correct and the initiator indicates that it is prepared to operate in multiple path mode for the I/O process, the target chooses whether to perform multiple path operation and on which paths in the appropriate path group. The target may choose to operate in single path mode or it may appear that way to the initiating controller because of the path selection algorithm in the target controller.

The minimum number of exchanges for a large data transfer type of operation can be estimated fairly closely by assuming that frames carry close to a full payload. The payload bytes can be figured as (maximum frame lentgth) bytes per frame plus the messages, status, and CDBs. Command parameter data, command response data and logical block data are treated equally when calculating the minimum number of frames. The "n" in the table represents this count of exchanges; the intent is to show that maximum burst size agreements can affect the total number of exchanges for an operation. In some instances, the payload could exceed 2048 bytes, but implementers may choose a well known byte count for their maximum burst length (e.g., 2048).

Characterization of SCFI Operation Types for the Serial Interface with Simple I/O Processes

Initiator Outbound Frame Content	Target Inbound Frame Content: Payload Bytes	NONE	Stat. Only	MSGs Only	Stat. + MSGs	Stat. + MSGs. + CRD. Sense	LBD or CRD + Stat. + MSGs	CRD + Stat. + MSGs	LBD + Stat. + MSGs
		<= 2048*	<= 2048*	<= 2048*	<= 2048*	<= 2048*	<= 2048*	> 2048*	> 2048*
Messages Only	<= 2048*	1	2	2	-	-	-	-	-
Messages + CDB	<= 2048*	-	2	2	2	2	2	5	5,6
Messages + CDB + CPD or LBD	<= 2048*	-	2	2	2	2	-	-	-
Messages + CDB + CPD	> 2048*	-	3	3	3	3	-	-	-
Messages + CDB + LBD	> 2048*	-	3,4	3,4	3,4	3,4	-	-	-

LEGEND:

Stat - Status
Payload Bytes - Total bytes for packet less than x bytes, excluding Messages and Status
* - May be shorter based on maximum implemented frame data content

Figure 5-5. Characterization of SCFI Operation Types for the Serial Interface

The syntax used in Figure 5-6 on page 5-15 for describing each operation follows. For each operation type, one exchange is described on one line. Each exchange takes a separate line (See operation Op4). Within a single exchange, the selection of frame sequences is taken only from Figure 5-3 on page 5-8. The various frame sequences within an exchange are separated by a comma. A frame sequence in parentheses is optional (See operation Op3); the response to the previous sequence tells both the initiating controller and the target controller whether to send/expect the optional sequence. A frame sequence bounded by "<" and ">*" indicates a sequence which may be repeated multiple times (See operation Op4). The suffix "*" means that an entire exchange is repeated as many times as necessary (e.g., a long read command of 100000 bytes).

There are three suffixes possible with the frame sequence number: out; in; and *. The suffix "out" means a frame sequence started by the initiating controller. The suffix "in" means a frame sequence started by the target controller. The "in" and "out" suffixes are always adjacent to the frame sequence number (e.g., 7out*; not 7*out). The suffix "*" means a streamed sequence of the frame sequence type.

To show these syntax rules in context, take the exchange "<8in,7out*>**" from operation Op4. The correct interpretation is: a set of exchanges (<...>**) each of which consists of two frame sequences (8 and 7, Figure 5-3 on page 5-8); frame sequence 8 is initiated by a target controller (in); frame sequence 7 is initiated by an initiating controller (out); frame sequence 7 is a streamed sequence (*).

Normal SCFI Operation_Types Mapped to Fiber Channel

Operation No	Name	Mult.Path Mode	Normal # Exch.	Exchange/Frame Seq. Type and Order
Op1	Abort - Logical	NO	1	9out
Op2	Small R/W Data	NO	1	5out
Op3	Write Data - SP	NO	1	5out(,8in),7out*,6out
Op4	Write Data - MP	YES	2+n	5out <8in,7out*>** 6out
Op5	Read Data - SP	NO	1	8out,8in*
Op6	Read Data - MP	YES	1+n	8out 8in*

LEGEND:

R/W = Read/Write
SP = Single path mode
MP = Multiple path mode

Figure 5-6. Normal SCFI Operation Types Mapped to Fiber Channel

Operation Op1 is used for the messages which abort or terminate an I/O process or queued I/O processes. Operation Op2 is used for commands like REQUEST SENSE which have only a small amount of data (< = 256 bytes) to satisfy ALL versions of the command. Operations Op3 and Op4 are used for commands like FORMAT where the various lists may be significantly longer than 2048 bytes. Operations Op3 and Op4 are also used for WRITE commands with a transfer byte count which can vary widely. Operations Op5 and Op6 are used for READ commands with a transfer byte count which can vary widely.

NOTE: Operations Op3 through Op6 leave the determination of multiple path operation in the control of the target controller. A sequence similar to Op5 or Op6 could be defined for write operations where the initiating controller makes the multiple path choices for each exchange.

5.4.6 SCFI Exchange Protocol - Abnormal Sequences

Figure 5-7 on page 5-17, Figure 5-8 on page 5-18, and Figure 5-9 on page 5-19 identify the abnormal sequences for each of the six operations. An operation can be aborted or terminated at the choosing of the initiating controller. All operations must permit this at each reasonable opportunity. The selection of exchanges and frame sequences with each exchange aids this process. The various possible sequences are separated by "|" on the same set of lines or by a blank line between two groups of possible sequences.

The same syntax is used as in Figure 5-6 on page 5-15. This table must show all the possible intercept points and the sequence to be followed once the abort or termination is processed. The suffix "p" is introduced to indicate that only part of the stream of frames was transmitted before the exchange was aborted. This means that a frame sequence of "7out*" is made up of at least two parts: "7outp" (x frames); and "7outp" (n-x frames), where "n" is the expected number of frames in the stream. The frame sequence can be split into n occurrences of the "7outp" partial sequences.

Since the Fiber Channel may have multiple frames in the fabric at one time, the aborting element shall shall accept and discard these frames. The time to abort or terminate an I/O process depends on the fabric delays and the implementation in the target controller. The abort exchange frame sequence, 4, immediately gets the attention of the sending N_port when processed. No additional frames for the exchange are sent. The logical process to abort or terminate usually takes longer. The abort exchange can be thought of as an unexpected BUS FREE event.

Abnormal SCFI Operations Mapped to Fiber Channel (Part 1 of 3)

Operation No	Name	Abnormal # Exch.	Exchange/Frame Seq. Type and Order
Op1	Abort - Logical	1	None
Op2	Small R/W Data	2	5out,4out | 5out,4out 9out | 6out
Op3	Write Data - SP	2	5out,4out | 5out,4out | 5out,8in,4out 9out | 6out | 9out
		-	5out,8in,4out | 5out(,8in),7outp,4out 6out | 9out
			5out(,8in),7outp,4out 6out
			5out(,8in),7out*,9out
			5out(,8in),7out*,6out

LEGEND:

() = optional frame sequence	<...> = exchange
SP = Single path mode	* = streamed sequence
MP = Multiple path mode	** = repeated exchange
in = to initiating controller	out = from initiating controller
p = part of streamed sequence or part of repeated exchanges	<... = aborted exchange
, = frame sequence separator in an exchange	| = separate exchange groups

Figure 5-7. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 1 of 3)

Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 2 of 3)

Operation No	Name	Abnormal # Exch.	Exchange/Frame Seq. Type and Order	
Op4	Write Data - MP	2 to	5out,4out 9out	5out,4out 6out
			5out 9out	5out 6out
			5out 8in,4out 9out	5out 8in,4out 6out
			5out <8in,7outp,4out 9out	5out <8in,7outp,4out 6out
			5out <8in,7out*>p 9out	5out <8in,7out*>p 6out
			5out <8in,7out*>p <8in,4out 9out	5out <8in,7out*>p <8in,4out 6out
			5out <8in,7out*>p <8in,7outp,4out 9out	5out <8in,7out*>p <8in,7outp,4out 6out
			5out <8in,7out*>** 9out	5out <8in,7out*>** 6out

LEGEND:

See Figure 5-7.

Figure 5-8. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 2 of 3)

Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 3 of 3)

Operation No	Name	Abnormal # Exch.	Exchange/Frame Seq. Type and Order	
Op5	Read Data - SP	2 to 2+n	8out,4out	8out,4out
			9out	6out
			8out	8out
			8inp,4out	8inp,4out
			9out	6out
Op6	Read Data - MP	2 to 2+n	8out	8out
			9out	6out
			8out	8out
			8inp	8inp
			9out	6out

LEGEND:

See Figure 5-7 on page 5-17.

Figure 5-9. Abnormal SCFI Operation Types Mapped to Fiber Channel (Part 3 of 3)

5.4.7 SCFI Serial Exchange Protocol State Diagrams

The contents of Figure 5-10 on page 5-20 through Figure 5-14 on page 5-24 summarize the possible exchange combinations and frame sequence combinations allowed in SCFI when operating on the Fiber Channel. This limitation on an otherwise nearly limitless set of combinations permits interoperability of SCFI devices. The exchange protocol is established to retain the SCSI-3 convention of placing control of an I/O process with the target controller. This is consistent with the types of operations requested of target controllers and peripheral devices; certain preprocessing and setup operations must be performed before a target controller is ready to continue with an active I/O process.

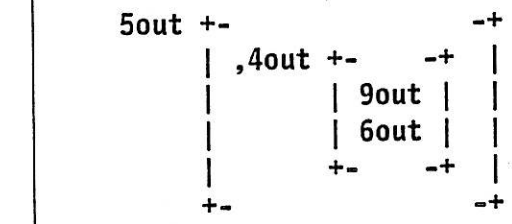
SCFI Exchange/Frame Sequence State Diagram (Part 1 of 5)

Operation Op1 Abort - Logical

NOTES:

1) There is no abort process for an abort.

Operation Op2 Small R/W Data



NOTES:

1) If there is a time lag between 5out and the response, the abort exchange may be processed. Otherwise, the I/O process completes normally.
2) A frame sequence beginning ",4out" is the beginning of a termination sequence which starts with an Abort Exchange frame sequence.

LEGEND:

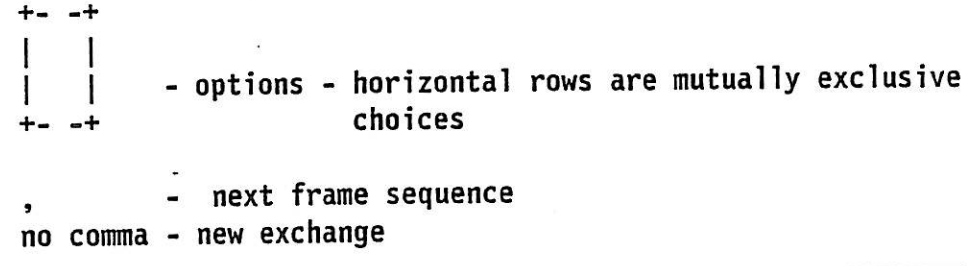
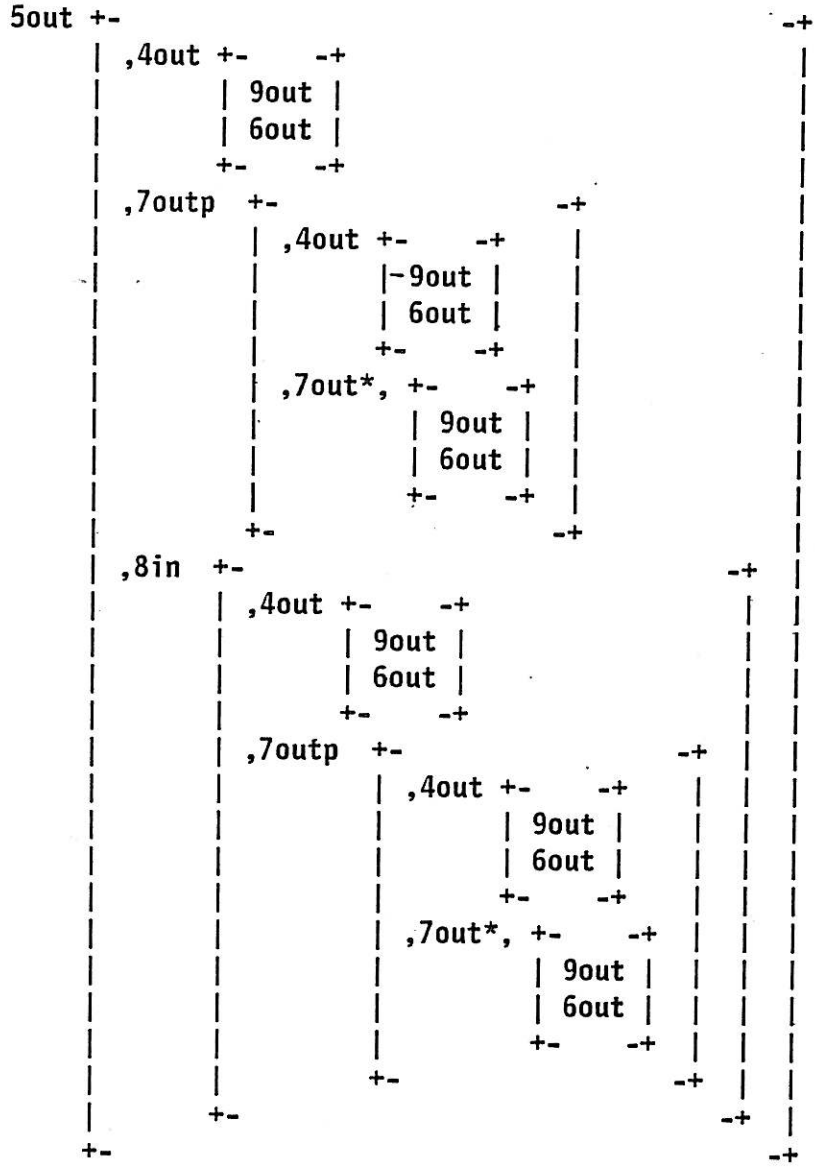


Figure 5-10. SCFI Exchange/Frame Sequence State Diagram (Part 1 of 5)

SCFI Exchange/Frame Sequence State Diagram (Part 2 of 5)

Operation Op3 Write Data - SP



NOTES:

1) The frame sequence beginning ",8in" is optional.
2) A frame sequence beginning ",4out" is the beginning of a termination sequence which starts with an Abort Exchange frame sequence.

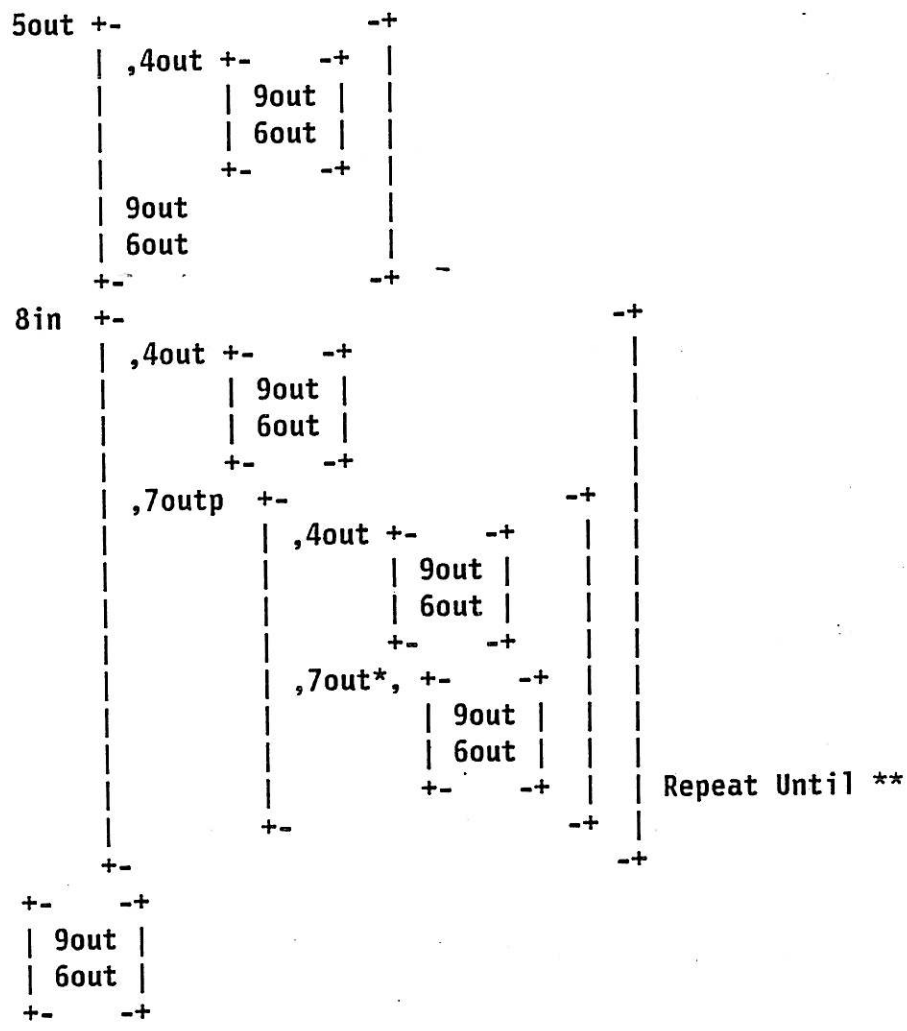
LEGEND:

See Figure 5-10.

Figure 5-11. SCFI Exchange/Frame Sequence State Diagram (Part 2 of 5)

SCFI Exchange/Frame Sequence State Diagram (Part 3 of 5)

Operation Op4 Write Data - MP



NOTES:

1) The exchange beginning "8in" is repeated until "<...>**".
2) A frame sequence beginning ",4out" is the beginning of a termination sequence which starts with an Abort Exchange frame sequence.

LEGEND:

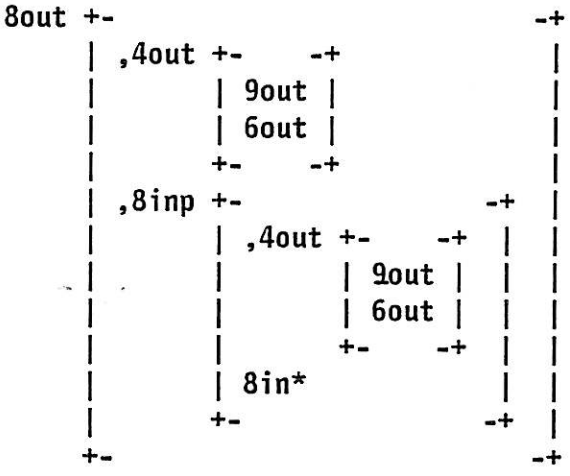
See Figure 5-10 on page 5-20.

Figure 5-12. SCFI Exchange/Frame Sequence State Diagram (Part 3 of 5)

114

SCFI Exchange/Frame Sequence State Diagram (Part 4 of 5)

Operation Op5 Read Data - SP



NOTES:

1) A frame sequence beginning ",4out" is the beginning of a termination sequence which starts with an Abort Exchange frame sequence.

LEGEND:

See Figure 5-10 on page 5-20.

Figure 5-13. SCFI Exchange/Frame Sequence State Diagram (Part 4 of 5)

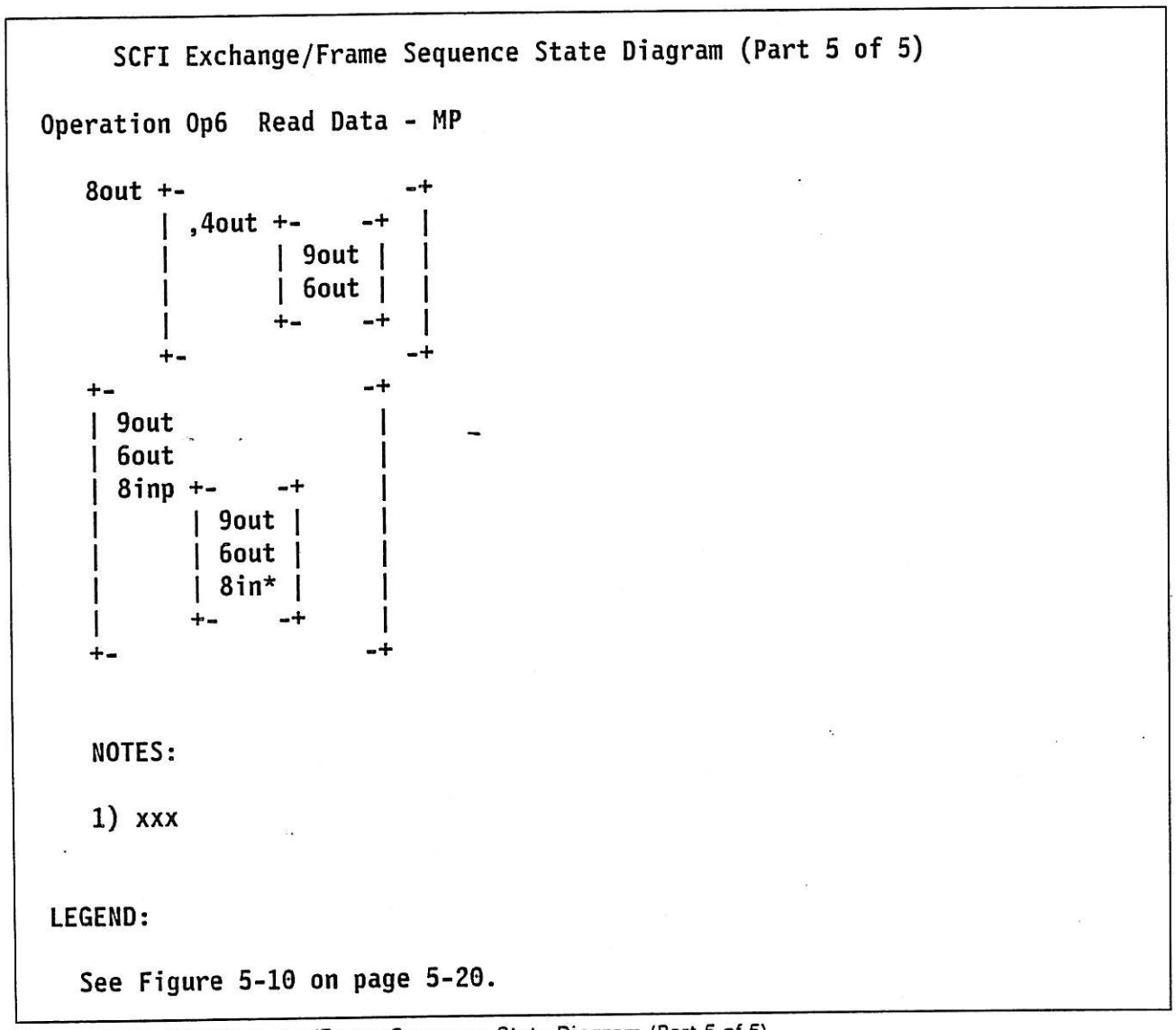


Figure 5-14. SCFI Exchange/Frame Sequence State Diagram (Part 5 of 5)

5.4.8 Resending an Unacknowledged Information Packet

5.4.8.1 Initiating Controller

5.4.8.2 Target Controller

5.4.9 Target Controller Termination of an I/O Process

5.4.10 Initiating Controller Termination of an I/O Process

5.4.11 ????

5.5 Parallel Interface Bus Phases (Optional)

The SCFI architecture includes the distinct phases:

- BUS FREE phase
- ARBITRATION phase
- SELECTION phase
- NEXUS TRANSFER phase

All SCFI devices shall shall implement the phases named above. The default mode of operation shall be asynchronous transfer mode, at maximum DATA BUS width for the parallel interface. Full width transfers result from disallowing intermix of SCFI devices with different DATA BUS widths (Section 4).

A SCFI device shall implement synchronous transfer mode. The SCFI bus can never be in more than one phase at a time.

For a single connection between initiator and target, information packets for various I/O processes can be exchanged. There is no need to distinguish between selection and reselection as for SCSI-3. A reconnection made using the SELECTION phase establishes the physical path between the initiator and target. The need to distinguish between selection and reselection is eliminated just as it is in the serial interface implementation. Information packet transfers are merely routed to a holding buffer for later interpretation. There is no reason to know the precise I/O process involved since each information packet is self-describing in that respect through the information packet prefix fields.

5.5.1 SCFI versus SCSI-3 Operating Mode

The SCFI operating mode and the SCSI-3 operating mode for information transfer are mutually exclusive. It shall be an error to attempt a phase for another operating mode within the connect or a later connection for an I/O process.

If the first information transfer phase of a connect is a nexus transfer phase, the target controller is stating its intent to operate in SCFI mode. If the first information transfer phase of a connect is not a nexus transfer phase, then the target controller is stating its intent to operate in SCSI-3 mode.

Since targets control the selection of information transfer phases, a SCFI target shall initiate a procedure, at power-on or after the reset event, which declares its intent to operate in SCFI mode. The target selects an SCFI address. The first attempt at information transfer on a parallel interface, after power-on reset or after processing the reset condition by a target, shall contain only a request to negotiate synchronous data transfer using a nexus transfer phase.

The selected SCFI or SCSI-3 device shall respond with one of the following:

1) If it does not implement target mode, it does not respond to selection, and the target discovers that it is a SCSI-3 device.

2) If it implements target mode and the nexus transfer phase, the SCFI devices transfer the information packet.

3) If it is a SCFI address which only operates in SCSI-3 operating mode and implements target mode, the SCSI device responds to selection and shall signal ATN on the first assertion of REQ for the reserved phase and shall not assert the ACK signal. If ATN is asserted, the target controller disconnects if it implements only SCFI mode. With disconnection, the selected SCSI-3 device shall release its ATN signal. The target controller has detected that it is possible to operate with that SCFI address using SCSI-3 operating mode. The target controller then may decide to change its definition to SCSI-3

mode to permit operation with the SCSI device or it shall not respond to selection by that SCFI address. If the target controller implements only SCFI mode, the target controller shall not attempt to use SCFI mode with that SCSI device until after a power-on reset, or a reset condition has been processed. The selected SCSI-3 device detects an unexpected disconnect and reacts appropriately according to the SCSI-3 standard. See 5.6.2 for all rules for the use of the ATN signal in SCFI.

5.5.2 Parallel BUS FREE Phase

The BUS FREE phase indicates that no SCFI device is actively using the SCFI bus and that it is available. Sometimes, a target reverts to BUS FREE phase to indicate an error condition that it has no other way to report. This is called as an unexpected disconnect event. SCFI devices shall detect the BUS FREE phase according to the rules specified in SCSI-3 for the BUS FREE phase.

5.5.3 Parallel ARBITRATION Phase

The ARBITRATION phase allows one SCFI device to gain control of the SCFI bus so that it can initiate or resume an I/O process.

The SCFI ID bit is a single bit signal assertion on the DATA BUS that corresponds to the unique SCFI address. All other DATA BUS bits shall be released by the SCFI device. Parity is not valid on the DATA BUS during the ARBITRATION phase. During the ARBITRATION phase, the data bus parity bits may be released or asserted, but shall not be actively driven false.

The procedure for an SCFI device to obtain control of the SCFI bus is the same as specified in SCSI-3.

5.5.4 Parallel SELECTION Phase

The SELECTION phase allows:

1) an initiating controller to select a target controller to initiate some target function (e.g., READ or WRITE command).

2) an initiating controller to reconnect with a target controller after a disconnect.

3) a target controller to reconnect with an initiating controller after a disconnect.

The procedure to perform SELECTION is the same as in SCSI-3 except as follows:

- Selection shall be performed with ATN not asserted since the only phase is the NEXUS TRANSFER phase from the selecting SCFI port to the selected SCFI port.

  The target controller response is to go to the NEXUS TRANSFER phase.

  Note: SCSI-3 requires selection with ATN asserted to force the target controller to enter the MESSAGE OUT phase immediately to acquire the identification data for the I/O process. The target controller response in SCSI-1 was to enter the COMMAND phase, disallow disconnection, and get the identification data from the first CDB. The SCFI implementation is similar to the SCSI-1 sequence which indicated single initiator selection; the target controller enters the NEXUS TRANSFER phase to acquire the identification data for the I/O process.

  (To SCSI-3 add text, now missing about what to do when selection without ATN is attempted. That is, the target controller shall go to the BUS FREE phase and ignore the selection. GRS)

- the single bit selection option shall shall not be implemented in ports when operating in target mode.

5.5.5 Parallel SELECTION Time-out Procedure

The following selection time-out procedure is specified for clearing the SCFI bus if the SCFI port waits a minimum of a selection time-out delay and there has been no BSY signal response from the selected SCFI port.

The SCFI port shall continue asserting the SEL and ATN signals and shall release the DATA BUS. If the SCFI port has not detected the BSY signal to be true after at least a selection abort time plus two deskew delays, the SCFI port shall disconnect. SCFI devices shall ensure that when responding to selection that the selection was still valid within a selection abort time of their assertion of the BSY signal. Failure to comply with this requirement could result in an improper selection.

5.5.6 Parallel Information Transfer Phase

The nexus transfer phase is the information transfer phase for SCFI. The phase is used to transfer all data and control information across the DATA BUS.

The C/D, I/O, and MSG signals are used to identify the physical information transfer phase (see Table 5-2). The target drives these three signals and therefore controls all changes from one phase to another. The initiator can request a NEXUS TRANSFER OUT phase by asserting the ATN signal at certain times (see attention event, section 5.6.2). The target can disconnect by releasing the MSG, C/D, I/O, and BSY signals.

The information transfer phase uses one or more REQ/ACK handshakes to control the information transfer. Each REQ/ACK handshake allows the transfer of one byte of information on the A cable or up to 2 bytes of information on the P cable. During the information transfer phases, the BSY signal shall remain true and the SEL signal shall remain false.

During one information transfer phase, the target shall continuously envelope the REQ/ACK handshake(s) with unchanging C/D, I/O, and MSG signals. An information transfer phase ends when the C/D, I/O, or MSG signals change after the negation of the ACK signal. The time between the end of a phase and the assertion of the REQ signal beginning a new phase is undefined. Assertion of REQ is not required to signal the end of an information transfer phase. It shall be sufficient that when one of the MSG, C/D, or I/O signals changes the information transfer phase is ended.

After signaling the end of an information transfer phase, the target may prepare for a new phase by asserting or negating the C/D, I/O, and MSG signals. These signals may be changed together or individually. They may be changed in any order and may be changed more than once. A new phase does not begin until the REQ signal is asserted for the first byte of the new phase.

An initiator is allowed to anticipate a new phase, based on early information provided by changes in the C/D, I/O, and MSG signals. However, the actual phase is not valid until the REQ signal is asserted at the beginning of the next phase.

Table 5-2. Parallel Interface Information Transfer Phase

MSG	C/D	I/O	Phase Name	Direction Of Transfer
1	0	0	NEXUS TRANSFER	Initiator to target; Initiator from target

Key: 0 = False, 1 = True.
Signal combinations not listed are RESERVED.

5.5.6.1 Parallel Asynchronous Information Transfer

Asynchronous information transfer shall be implemented as described in SCSI-3. It shall be used to perform a synchronous data transfer negotiation.

5.5.6.2 Parallel Synchronous Data Transfer

Synchronous data transfer shall be implemented for use by the nexus transfer phases. It shall be used in these phases if a synchronous data transfer agreement has been established (see 6.e.26). The agreement specifies the REQ/ACK offset and the minimum transfer period. The default agreement is asynchronous data transfer. Synchronous data transfer shall be implemented as specified in SCSI-3.

5.5.6.3 Parallel Wide Data Transfer

Wide data transfer is mandatory if the DATA BUS width exceeds 8 bits, excluding parity. For wide data busses, the ports shall use wide data transfer for each nexus transfer phase. No wide data transfer agreement is required since connection of data busses of unequal widths shall not be supported. The default agreement is the maximum width of the data bus and shall not shall not be changed. Wide data transfers shall implemented as specified in SCSI-3. All SCFI information packet lengths shall be a multiple of four bytes. Pad bytes are specified for inclusion at the end of the information packet. This is consistent with Fiber Channel requirements for the frame data field.

5.5.7 Parallel NEXUS TRANSFER Phase

The NEXUS TRANSFER phase allows a logical element to request that information packet(s) be transferred to a selecteds other logical elememt.

Synchronous data transfer and wide data transfer shall be used during this phase except as specifically exempted.

5.6 Parallel Interface Bus Events (Optional)

The SCFI bus has three asynchronous events: the unexpected disconnect event; the attention event; and the reset event. These events cause the SCFI device to perform certain actions and can alter the phase sequence.

5.6.1 Parallel Unexpected Disconnect Event

If an initiator detects a disconnect the target at any other time than for the conditions below, the target is indicating the unexpected BUS FREE event to the initiator. The target may perform this disconnect independent of the state of the ATN signal. The initiator shall manage this event as an unsuccessful I/O process termination (See 6.j.5.).

SCFI initiators normally expect the BUS FREE phase to begin after one of the following occurs:

(1) after a reset event is detected.

(2) after the transfer of one or more information packets in the nexus transfer phase.

(3) after failure of a target to transfer an information packet using the NEXUS TRANSFER IN phase.

(4) after an unsuccessful selection.

(5) after a selection of a SCSI-3 device (i.e., selection without ATN asserted). The initiating controller may then decide to operate with the target controller in SCSI-3 mode. If so, all subsequent communication follows the SCSI-3 standard.

All other occurrences of a BUS FREE state cause the initiator to establish an Unexpected Disconnect Condition for the Initiating Controller.

5.6.2 Attention Event

The attention event allows the initiator to inform the target during a connection that it has an information packet to send. The target gets this information packet by performing a NEXUS TRANSFER phase.

The initiator generates the attention event by asserting ATN as follows:

a) not during the BUS FREE phase.

b) before the ACK for the first byte of any SCSI-3 information transfer phase to indicate that the initiating controller does not operate in SCSI-3 mode. The initiating controller may then perform one of the following:

- not assert ACK which shall cause the target controller to time out on the phase, abort the I/O process and go to the BUS FREE phase.

  (This needs to be added to SCSI-3 or deleted. GRS)

- respond with ACK, continue to respond with ACK for additional assertions of REQ with data and good parity during the same phase, and when the target controller switches to the MESSAGE OUT phase, send a SCSI-3 ABORT message to the target controller. This provides for compatability with SCSI-3 devices which still support SCSI-1 selection modes.

- during a NEXUS TRANSFER phase to indicate that it has an additional information packet to send during the same connection.

The initiator shall assert the ATN signal at least two deskew delays before negating the ACK signal for the last byte transferred in a bus phase for the attention condition to be honored before transition to the BUS FREE phase.

A target shall respond to each attention event with NEXUS TRANSFER OUT phase as follows:

(1) If the ATN signal becomes true during a NEXUS TRANSFER phase, the target shall enter a new NEXUS TRANSFER phase as the next phase following the current phase (i.e., at the end of the current information packet transfer). The initiator shall continue REQ/ACK handshakes with valid data for the current phase until it detects a phase change.

(2) the target shall not disconnect, except for an unexpected disconnect, before responding to the attention event.

5.6.3 Parallel Reset Event

The reset event is used to clear immediately all SCFI devices from the bus. This event shall take precedence over all other phases and events. Any SCFI device may create the reset event by asserting the RST signal for a minimum of a reset hold time. During the reset event, the state of all SCFI bus signals other than the RST signal is not defined.

All SCFI devices shall release all SCFI bus signals (except the RST signal) within a bus clear delay of the transition of the RST signal to true. The BUS FREE phase always follows the reset event. See reset condition, section 6.j.6, for the effect on logical operations.

5.6.4 Parallel Unexpected Disconnect Condition

(Section 6? GRS)

5.6.5 Parallel Attention Condition

(Section 6? GRS)

5.6.6 Parallel Reset Condition -

(Section 6? GRS)

5.7 Parallel Bus Phase Sequences (Optional)

5.7.1 Parallel Bus Phase Order

The order phases are used on the SCFI bus follows a prescribed sequence.

The reset event aborts any phase, except a BUS FREE phase, and is always followed by the BUS FREE phase. Also, any other phase can be followed by the BUS FREE phase but many such instances are error conditions (see unexpected disconnect condition, 5.6.1).

If the bus is not in the BUS FREE phase after the reset condition, then further operation with the SCFI bus is impossible. Another reset may be attempted.

Figure 5-15 on page 5-31 shows the allowable sequences.

The normal progression is: from the BUS FREE phase to ARBITRATION; from ARBITRATION to SELECTION; from SELECTION to NEXUS TRANSFER; and from NEXUS TRANSFER to BUS FREE.

5.7.2 Parallel Signal Restriction Between Phases

When the SCFI bus is between two information transfer phases, the following restrictions shall apply to the SCFI bus signals:

(1) The BSY, SEL, REQx, and ACKx signals shall not change.

(2) The C/D, I/O, MSG, and DATA BUS signals shall change unless the target is preparing to disconnect. When switching the DATA BUS direction from initiator driving to target driving, the target shall delay driving the DATA BUS by at least a data release delay plus a bus settle delay after asserting the I/O signal and the initiator shall release the DATA BUS no later than a data release delay after the transition of the I/O signal to true. When switching the DATA BUS direction from target driving to initiator driving, the target shall release the DATA BUS no later than a deskew delay after negating the I/O signal.

(3) The ATN and RST signals may change as defined under the descriptions for the parallel attention event and parallel reset event.

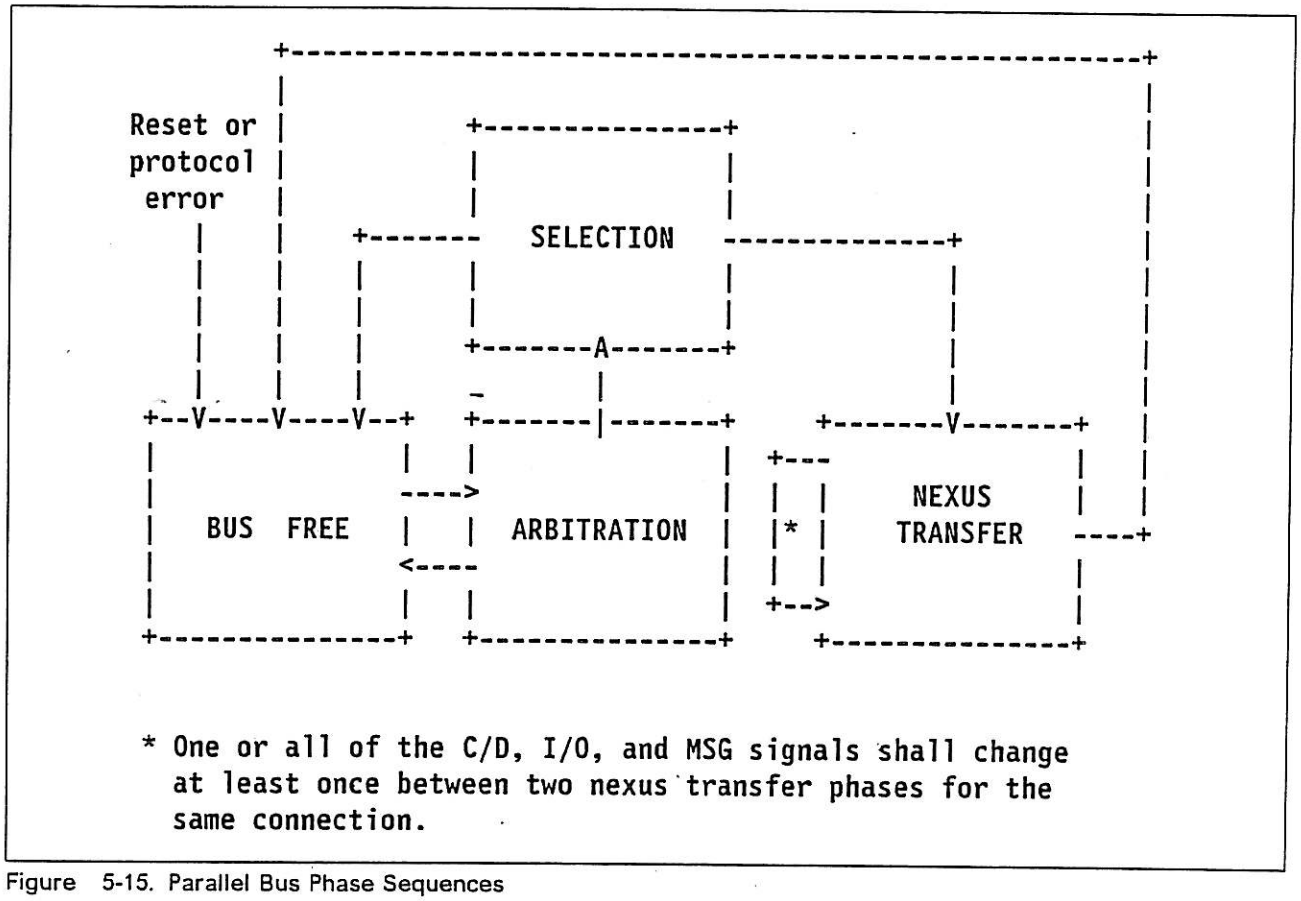


Figure 5-15. Parallel Bus Phase Sequences

Part 2. SCFI Logical Operations

Section 6. SCFI Logical Operations 6-1
6.1 Logical System Description 6-2
6.1.1 Logical System Definition 6-2
6.1.2 I/O Processes 6-7
6.1.2.1 General Description of Logical Operations 6-7
6.1.2.2 I/O Process Parameter Requirements (Mandatory) 6-9
6.1.2.3 I/O Process Sense Data (Mandatory) 6-10
6.1.3 Programmable Operating Definition (Parallel) (Optional) 6-10
6.1.4 SCFI Addresses 6-11
6.1.4.1 SCFI Address for a Port (Mandatory) 6-11
6.1.4.2 Logical Units (Mandatory) 6-11
6.1.4.3 Target Routines (Optional) 6-11
6.1.5 Queued I/O Processes (Mandatory) 6-12
6.1.5.1 Untagged I/O Processes (Mandatory) 6-12
6.1.5.2 Tagged I/O Processes (Optional) 6-13
6.1.5.3 Concurrent I/O Processes Execution 6-16
6.1.6 Logical Unit Reservation Termination (Mandatory) 6-16
6.1.7 Assignment Termination (Optional) 6-16
6.1.8 Path Group Termination (Optional) 6-16
6.2 Commands 6-16
6.2.1 Command Set Implementation Requirements (Mandatory) 6-17
6.2.2 Reserved (Mandatory) 6-18
6.2.3 Command Descriptor Block 6-18
6.2.4 Operation Code 6-19
6.2.5 Logical Block Address 6-20
6.2.6 Transfer Length 6-21
6.2.7 Command Parameter Data Length 6-21
6.2.8 Command Response Data Length 6-22
6.2.9 Control Field 6-22
6.2.9.1 Link Bit (Mandatory) 6-22
6.2.9.2 Flag Bit (Mandatory) 6-23
6.3 Status (Mandatory) 6-23
6.3.1 Status Codes (Mandatory) 6-24
6.3.2 Status Code Reporting Priority (Mandatory) 6-25
6.4 Contingent Allegiance Condition (Mandatory) 6-26
6.5 Extended Contingent Allegiance Condition (Optional) 6-27
6.6 Asynchronous Event Notification (Mandatory) 6-28
6.7 Information Packet Structure 6-30
6.7.1 Interface Control Fields (Mandatory) 6-30
6.7.1.1 Interface Control Fields (Mandatory) 6-31
6.7.1.2 Information Packet Serial Encapsulation 6-34
6.7.2 Interface Logical Elements (Mandatory) 6-36
6.7.2.1 Message Interface Logical Element 6-37
6.7.2.2 Command Descriptor Block Interface Logical Element 6-37
6.7.2.3 Command Parameter Data Interface Logical Element 6-37
6.7.2.4 Command Response Data Interface Logical Element 6-37
6.7.2.5 Logical Block Data Interface Logical Element 6-38
6.7.2.6 Status Interface Logical Element 6-38
6.7.2.7 Autosense Interface Logical Element (Optional) 6-38
6.7.3 Information Packet Layout (Mandatory) 6-38
6.7.3.1 Serial Bus 6-38

6.7.3.2 Parallel Bus (Optional) 6-39
6.8 Messages (Mandatory) 6-39
6.8.1 ABORT 6-44
6.8.2 ABORT TAG 6-44
6.8.3 BUS DEVICE RESET 6-44
6.8.4 CLEAR QUEUE 6-45
6.8.5 COMMAND COMPLETE 6-45
6.8.6 EXTENDED MESSAGE REJECT 6-45
6.8.7 INITIATE RECOVERY 6-46
6.8.8 INVALID BUS PHASE DETECTED (Parallel) 6-47
6.8.9 INVALID INFORMATION PACKET 6-47
6.8.10 LINKED COMMAND COMPLETE 6-48
6.8.11 LINKED COMMAND COMPLETE (WITH FLAG) 6-48
6.8.12 MODIFY DATA POSITION 6-48
6.8.13 PARITY ERROR (Parallel) 6-49
6.8.14 RELEASE RECOVERY 6-50
6.8.15 RESEND PREVIOUS INFORMATION PACKET 6-50
6.8.16 SYNCHRONOUS DATA TRANSFER REQUEST (Parallel) 6-50
6.8.17 SYNCHRONOUS PACKET TRANSFER REQUEST 6-52
6.8.18 TERMINATE I/O PROCESS 6-54
6.8.19 TRANSFER READY 6-55
6.9 Command Processing Considerations and Exception Conditions 6-55
6.9.1 Unit Attention Condition 6-55
6.9.2 Incorrect Initiating Controller Connection 6-57
6.9.3 I/O Processes for an Invalid LUN or TRN 6-57
6.9.4 Parameter Rounding 6-58
6.9.5 Unsuccessful I/O Process Termination Condition 6-58
6.9.6 Reset Condition 6-58
6.9.6.1 Reset Condition (Serial) 6-58
6.9.6.2 Reset Condition (Parallel) 6-59
6.9.6.3 Hard Reset Alternative (Parallel) 6-60
6.9.6.4 Soft Reset Alternative (Parallel) 6-60
6.9.7 Unsuccessful Information Packet Transfer Condition (Serial) 6-61
6.9.8 Unexpected Disconnect Condition (Serial) 6-61
6.9.9 Unexpected Disconnect Condition (Parallel) 6-61
6.9.10 Attention Condition (Parallel) 6-61

Section 6. SCFI Logical Operations

This section is divided into 9 parts:

1) Logical system description in section 6.1;

2) Commands in Section 6.2;

3) Status in Section 6.3;

4) Contingent Allegiance in Section 6.4;

5) Extended Contingent Allegiance in Section 6.5;

6) Asynchronous Event Notification in Section 6.6;

7) Information packet description in section 6.7;

8) Message descriptions in Section 6.8;

9) Exception Conditions in Section 6.9.

All I/O processes are accomplished by transferring information packets composed of interface logical elements. Initiating controllers and target controllers interpret the information logical elements.

Interface logical elements are messages, command descriptor blocks, command parameter data (additional data required for some commands), command response data (data not from logical blocks), logical blocks, autosense, and status.

Initial information packet transfers on a parallel SCFI bus, are in asynchronous data transfer mode. For serial interface implementations, all transfers are in synchronous data transfer mode.

Mandatory requirements for logical operation placed on initiating controllers and target controllers include:

1) All SCFI ports shall implement an active initiator mode and an active target mode (i.e., dynamic switching of the SCFI port mode)..

2) All SCFI ports shall implement synchronous data transfer. (Serial Interfaces.) Synchronous data transfer mode is the only transfer mode. (Parallel Interfaces.) Each SCFI port shall negotiate for and be granted synchronous data transfer mode. All transfers, other than during synchronous negotiation shall be in synchronous data transfer mode.

3) (Parallel Interfaces.) A parallel SCFI bus shall not have an intermix SCFI ports of unequal DATA BUS widths.

4) (Parallel Interfaces.) All SCFI ports which implement a DATA BUS width greater than one byte shall perform all information transfers at full bus width.

5) A SCFI port which normally acts as an initiator, when selected, shall enter target mode. The initiating controller shall declare itself as a processor device class in response to the INQUIRY command.

6) Initiating controllers shall respond correctly to asynchronous event notification.

7) Target controllers shall implement initiator mode on each port to select initiators, to acquire the INQUIRY data to determine its principal initiators, and to support asynchronous event notification; events not occurring in the bounds of an I/O process are reported when they occur.

8) (Parallel Interface.) Initiating controllers and target controllers shall perform a synchronous data transfer negotiation as the first set of information transfers after any event which may make the synchronous negotiation invalid. The default state of a port at power-on shall be asynchronous data transfer mode. The state of a port after a reset event shall be asynchronous data transfer mode.

9) Target controllers shall implement the mandatory commands for both the common commands (Section 7) and for the implemented device class (one of Sections 8 through 17). Within each command, all function defined as mandatory shall be implemented. This provides a minimum functional base for all implementations of a device class.

10) Logical elements shall implement untagged queuing. Tagged queuing is optional.

6.1 Logical System Description

The logical system description gives the SCFI environment to perform device-independent activity using one or more SCFI busses. Figure 6-1 on page 6-6 shows the relationship of several terms from the glossaries in section 3.1 related to the mimimum logical system.

6.1.1 Logical System Definition

A logical system consists eighteen (18) items. Items 1) through 13) are mandatory; items 14) thorugh 18) are optional.

1) a minimum logical system consists of two SCFI ports and one SCFI bus connecting them.

2) each SCFI port is capable of operating in initiator mode (called an initiator) on each SCFI bus.

3) each SCFI port is capable of operating in target mode (called a target) on each SCFI bus.

4) the initiator and target in Items 2 and 3 above, attach to the same SCFI bus and are active in their respective modes during a connection between them (i.e., not the same port); the two ports may reverse modes for each connection.

5) the logical element attaching an SCFI device which principally starts I/O processes is called an initiating controller. An initiating controller controls one or more ports per item 2. The same ports are used for target mode per item 3.

6) the logical element attaching an SCFI device which principally receives and executes I/O processes is called a target controller. A target controller controls one or more ports per item 3. The same ports are used for initiator mode per item 2.

NOTE: The names given to the logical elements attaching an SCFI device do not prevent any SCFI device from using all functions of SCFI. The word "principally" in Items 5 and 6 imply this. Thus, a copy manager, acting principally as a target controller, may act as an initiating controller and use all defined functions of the commands and the logical system to perform a copy operation. This is the peer capability of SCFI as opposed to a master-slave relationship on some other interfaces.

7) each port has an SCFI address unique to the SCFI bus on which it attaches. (Parallel Interfaces.) The SCFI address translates to the physical SCFI ID on the SCFI bus. The SCFI ID may be the same or different for each port when each port attaches to a different SCFI bus.

8) each port assigns a port number by its controlling logical element. The port number is unique within each logical element.

9) each target controller has one or more logical units each identified by a unique logical unit number (LUN). LUNs are contiguous beginning at zero.

10) each target controller has zero or more target routines each identified by a target routine number (TRN). TRNs are contiguous beginning at zero.

11) the extent of a logical system, from a target controller, is the set of all initiating controller/initiator combinations attached, through one or more SCFI busses, to the target controller and from which a connect has been made. For an initiating controller, the extent of a logical system is the set of all logical units and target routines to which a connect has been made on one or more SCFI busses.

12) For initiating controllers having access to a logical unit, access can be restricted to certain extents within some logical unit types, based on device class. Also, access may be temporarily restricted for a complete logical unit. These reservation functions provide the lowest level of restriction to information within a logical unit. There is no equivalent reservation function for target routines.

13) Each initiating controller is assigned an initiating controller ID. An initiating controller ID (ICID) must be unique in a logical system.

An identifier consisting of a ICID, an initiating controller port number, an initiator SCFI address, a target controller SCFI address, a target controller port number, and a LUN or TRN, defines a path when the relationship is established as the result of a connect started by an initiating controller to a logical unit or target routine. This is called an implicitly named path. The initiating controller has the complete name of the path, but the target controller does not. The port numbers are transferred as part of each complete I/O process.

No path exists between a LUN or TRN and an initiating controller unless the LUN or TRN is the object of a connect started by that initiating controller to the LUN or TRN.

An implicitly named path is in an ungrouped state. When a path is in the ungrouped state, each I/O process is limited to operation on the path where the connect was made. This is called single path mode. Each initiator connecting with a target controller port must be considered as attached to a unique initiating controller until the transfer of an ICID from the initiating controller occurs.

14) An initiating controller transfers its ICID using a defined command between the logical elements. The implicitly named path becomes an explicitly named path. An initiating controller connects with and transfers its ICID to each logical unit or target routine with which it needs to define an explicitly named path. The initiating controller is not required to transfer its ICID on all available physical paths between the initiating controller and the LUN or TRN, but only on those paths which it intends to use for additional SCFI functions (Items 15 thorugh 18).

Once paths are completely named, the target controller can distinguish which paths belong to which initiating controllers. The LUN or TRN used with each command must be valid for the target controller. The logical unit need not be ready or installed (e.g., powered off but cabled or not cabled).

An explicitly named path is initially in an ungrouped state. I/O processes are handled in the same manner as an implicitly named path unless additional agreements are reached between the initiating controller and the target controller as defined below.

15) An identifier, consisting of an ICID and either a LUN or TRN in the same target controller, represents one logical path. A logical path consists of a set of one or more paths. A logical path consists of one or more explicitly named paths or exactly one implicitly named path. The paths in a logical path are initially in an ungrouped state. (See Item 13.)

The set of paths available in a logical path from an initiating controller to any one logical unit or target routine may be determined from the results of the INQUIRY command response data provided the command response data is unique to each logical unit or target routine.

16) A set of ungrouped explicitly named paths in a logical path is established as a path group through a command from the initiating controller using any one of the paths in the logical path. This set of paths, a logical path, is called a path group. Establishing a path group is the mechanism for enabling multiple path operations in logical system. Including a path in a path group does not restrict access to the logical unit or target routine from any other path.

When establishing a path group, the initiating controller identifies where status leading to a contingent allegiance or extended contingent allegiance is reported. Any status which does not result in contingent allegiance may be sent over any path in the path group. The choices are single path status mode and multiple path status mode. The condition may be altered by disbanding the group and establishing the group with the alternate choice.

A path may be added to an established group at a later time by a command from the initiating controller on that path.

A path may be removed from an established path group by a command from the initiating controller on that path.

Once a path group is established, the extended functions of assignment (Item 16) and multiple path operation (this Item) may be used.

The inverse of establishing a path group is to disband a path group. A path group may be explicitly disbanded by a command from the initiating controller along any one path in the path group. A path group is implicitly disbanded when the last path is removed from an established path group using a remove path function rather than a disband command.

17) Any logical unit or target routine is initially available to receive I/O processes from any initiating controller attached to the target controller. This use privilege is extended whether the path is explicitly named, implicitly named, and whether for explicitly named paths, the path is grouped or ungrouped.

The two functions of assignment are:

1) assign this LUN or TRN to the path group on which the command was received, or

2) add assignment of this LUN or TRN for another established path group.

The inverse of assign is unassign. Use privileges may be unassigned for any path group to which assignment currently exists from any path for which assignment currently exists. Assignment may be transferred from one initiating controller to another without passing through a state where no assignment exists.

18) Controlled access is the name given to the function which breaks an assignment if some error or failure occurs in an initiating controller which currently has assignment. Breaking assignment may be temporary or permanent, but it must be controlled, as are other functions which can lead to reliability, availability, and data integrity problems.

Assignment permits use privileges only through assigned path groups. Controlled access permits access outside the bounds of assigned path groups. The mechanism to prevent deliberate or accidental loss of assignment protection is the control access function, enabled by a password and checked by the affected target controller.

The control access command has three functions:

1) establish a password in a target controller.

2) general unassign.

3) request temporary unassignment.

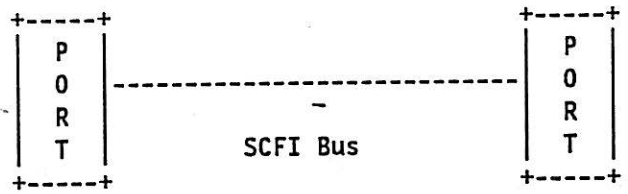
A password is established by an initiating controller having assignment. The password is not reported by a target controller on any path. The target controller checks its established password, if any, against password supplied by the control access command function from an initiating controller not having assignment.

If the target controller has an established password and it matches the password with the command, the control access command and any commands linked to it are executed, if possible. The mechanism by which the unassigned initiating controller acquires the correct password is not defined in SCFI.

Initiating Controller
ICID

Initiating Controller Port Number
SCFI Port in Initiator Mode
Initiator

SCFI Address
SCFI ID (parallel interface)



SCFI ID (parallel interface)
SCFI Address

Target
SCFI Port in Target Mode
Target Controller Port Number

Target Controller
Logical Unit
Reserved/Not Reserved
LUN

Path = ICID ||
Initiating Controller Port Number ||
Initiator SCFI Address ||
Target SCFI Address ||
Target Controller Port Number ||
LUN

Logical path = ICID || LUN
Path implicitly named
Path in the Ungrouped State
Path Group is Not Established
Path Status is Implicitly Named Path

Single path status mode
LUN is Unassigned
No password exists

Figure 6-1. Minimum Logical System Attributes

6.1.2 I/O Processes

Consider the simplified system shown in Figure 6-2 on page 6-7 where an initiating controller and target controller communicate on a SCFI bus to execute an I/O process.

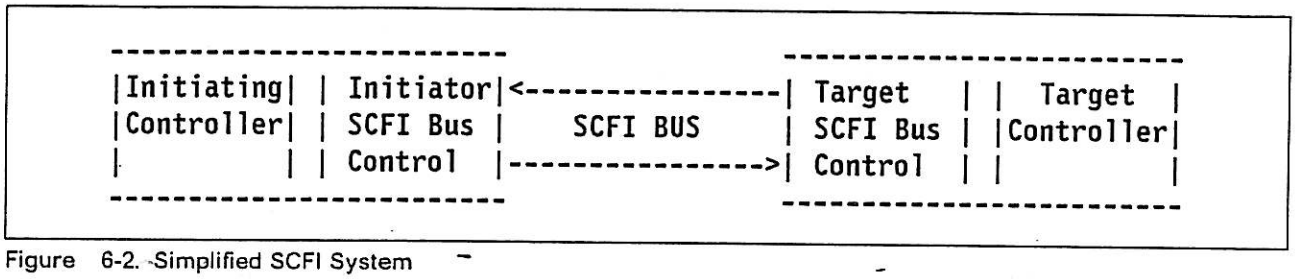


Figure 6-2. Simplified SCFI System

The SCFI architecture requires that both the initiating controller and the target controller involved in an I/O process retain status about its operation and the ports with which it communicates.

6.1.2.1 General Description of Logical Operations

An I/O process consists of a minimum of one information packet. Such an I/O process is concerned only with messages such as BUS DEVICE RESET which do not require a response from the target controller.

Most I/O processes consists of at least two information packets. One information packet is sent from the initiating controller to the target controller to request some function(s) be performed. The target controller responds at completion of the function with an information packet indicating the status of the request. I/O processes which transfer logical blocks may require many information packets.

The form of the request to perform a function is called a command descriptor block. Each command descriptor block is fixed length. There are three lengths specified: 6 bytes, 10 bytes, and 12 bytes. Some commands need to transfer more data than can fit into the command descriptor block. This additional data is called command parameter data. It is variable in length with the length specified in the command descriptor block.

For commands which do not transfer logical block data, the target controller checks the command descriptor block and the command parameter data, if any. If they are correctly formed with valid values, the target controller executes the command. When execution is complete and successful, the target controller forms an information packet containing the status of the command and a command completion message (there are three). The information packet, when received at the initiating controller is checked and the I/O process parameters are updated.

An alternative form of a command has a command descriptor block with no command parameter data, but which requests that the target controller return data, other than logical block data, as command response data. The execution is similar to the command description above.

For commands which transfer logical block data, more than two information packets may be needed. For write-type commands, the initiating controller places the command descriptor block and possibly some of the logical block data in the first information packet and sends it. The target controller checks the command descriptor block as before. If it is correctly formed, the target controller can begin to process the logical block data, if present in the information packet.

Meanwhile, the initiating controller may have filled up additional information packets and sent them, up to the limit agreed upon between the two logical elements. The target controller overlaps receipt of additional information packets with processing of each information packet. The target controller informs the initiating controller as the information packets are processed so that additional packets, if any, can be transferred.

When all data has been transferred, the target controller finishes processing the data as specified by the MODE SELECT parameters and the command descriptor block. The status and command completion message are placed in an information packet and sent to the initiating controller. The I/O process is complete.

The initiating controller can link several commands together to form larger I/O processes. I/O processes with linked commands are treated as a unit by the target controller and the initiating controller.

If, as is occasionally the case, the command descriptor block contains an error, the command is not executed. Any interface logical elements following the command descriptor block are not processed (i.e., treated as not having been received). Additional information packets for the I/O process are not processed. Detecting the error requires the target controller to terminate the I/O process and prepare sense data to identify the error.

The error condition indicated above requires no recovery action which involves the target controller. This condition is called a contingent allegiance condition. If the condition is reported using an autosense information logical element, the condition is immediately cleared with successful transfer of the information packet which transfers the autosense ILE. The I/O process is terminated.

If the target controller does not transfer autosense, the target controller prepares an information packet with status to indicate an error and a command completion message. The information packet, without an autosense ILE is sent to the initiating controller. The I/O process is terminated.

If the target controller sends only status it shall preserve the sense data for possible retrieval by the initiating controller. The initiating controller may form an new I/O process with the first command being a REQUEST SENSE command and send it to the target controller. The target controller sends the sense data in response to the REQUEST SENSE command. The initiating controller may also ignore the status and send some other command which causes the target controller to clear the contingent allegiance condition and the sense data.

If the error is more significant and requires a recovery process involving both the initiating controller and the target controller, the condition is called an extended contingent allegiance. In addition to the status and sense data, a special message informs the initiating controller of this condition. The target controller restricts access to the logical unit until the initiating controller releases it from extended contingent allegiance by sending the target controller another special message.

If an error is detected outside the bounds of an active I/O process, the target controller uses a protocol called asynchronous event notification to transfer appropriate sense data to initiating controllers. Asynchronous event notification may result in extended contingent allegiance as above. The protocol for notification and exit is similar to the preceding example.

There are different kinds of data which are transferred in information packets which are never all transmitted in the same information packet. These data are called interface logical elements. The interface logical elements are command descriptor blocks, status, messages, command parameter data, command response data, autosense, and logical block data.

As described above, a target controller may have multiple ILEs from one or more information packets available for an I/O process from an initiating controller. Additionally, the target controller may permit I/O processes from other initiating controllers to be queued for later execution. This is called untagged

queuing. The depth of the queue is implementation dependent; the minimum is one I/O process allowed in the target controller at a time. The target controller may execute these I/O processes in any order is deems appropriate.

The second level of queuing requires the initiating controller to assign a code or a tag to each of its I/O processes. If the target controller has implemented this level of queuing, the target controller may have several I/O processes for the same initiating controller in the I/O process queues for the same logical unit for the same initiating controller as well as I/O processses from other initiating controllers. This is called tagged queuing.

This section has described, generally, the major elements of SCFI I/O process flow. See section 3.1 for complete definitions of the terms introduced above. Details are covered in the following sections and in the command descriptions. Section 6.1.3 describes interoperability considerations with SCSI-3 logical elements. The specific rules governing execution of I/O processes is discussed in subsequent sections. The remainder of this section describes commands, status, information packet structure, error reporting, messages, and exception handling. Common commands are defined in Section 7. Device class specific commmands are defined in Sections 8 thorugh 17. Some detailed examples are found in Appendix 1.

6.1.2.2 I/O Process Parameter Requirements (Mandatory)

For logical elements, there are several kinds of parameters that each shall maintain:

1) (Parallel Interfaces.) Synchronous transfer negotiation agreement between the ports capable of being used during the I/O process.

2) (Serial Interfaces.) Service parameters of the fabric and each N_port selected for an initial connection.

3) Path names and path group agreements, assignment, and passwords for all logical units and target routines, including the no agreement states of these.

4) A queue of information packet buffer space(s) to be used for holding information packets. The information packets may not be associated with the an I/O process until later. Each information packet may belong to an active I/O process, a queued I/O process, or it may establish a new I/O process. (Parallel Interfaces.) Information packet transfer from the SCFI bus may be directly into an information packet buffer. (Serial Interfaces.) Information packet transfer is made from the SCFI bus into a frame buffer (Fiber Channel physical element). The information packet is then transferred from the frame buffer into a target controller information packet buffer.

5) A queue of I/O processes which are not yet in the active I/O process state and the associated information packets. Support for untagged I/O processes is mandatory; support for tagged I/O processes is optional. The difference in a logical element is the complexity of queue management.

6) A queue of active I/O processes and the state of each I/O process. The state of each active I/O process requires several elements: which information packets have been transferred; the logical order of the information packets; which ILEs have been processed; a status field to hold the status for each I/O process; an indication of the current state of data transfer for each command, if any; an area to receive or prepare sense data for the I/O process should it end with contingent allegiance or extended contingent allegiance; a means to tie the sense data to the correct portion the I/O process.

7) A queue of outgoing informatin logical pending transmission for each active I/O process.

8) A queue of information packets which have been transmitted sufficient to meet the retransmission requirements of the synchronous packet transfer negotiation agreement for each I/O process. These shall be identified to their active I/O process.

As the ILEs for an I/O process are being processed between the initiating controller and the target controller these parameters must be updated. The update for received information packets is made when the information packet is processed; this may not be at the time it is received. The update for information packets sent is made as each information packet is transferred. The state of an I/O process may appear different at the initiating controller and the target controller as a result of the potential difference in time for I/O process updates.

6.1.2.3 I/O Process Sense Data (Mandatory)

For each target controller, the target controller shall maintain the states of each logical element and target routine and maintain the minimum data as required by the device class and the device type implemented. This minimum data is called sense data. The format of the mandatory portion is specified with the REQUEST SENSE command in section 7.

Sense data is to be kept current with the state of the logical unit, target routine, and target controller. As events occur, the target controller shall update the sense data. The target controller shall be prepared to transmit sense data as a response to a REQUEST SENSE command or as autosense for any status leading to contingent allegiance or extended contingent allegiance.

6.1.3 Programmable Operating Definition (Parallel) (Optional)

This section applies only to initiating controllers and target controllers attached via a parallel SCFI bus. There is no alternate operating definition for logical elements attached to a serial SCFI bus.

Some applications require that the operating definition of a logical unit be modified to meet the special requirements of a particular initiator. The program-controlled modification of the operating definition is typically provided to allow operating systems to change the operating definition of a more recently developed targets to one which is more compatible with the operating system. Invoking this function requires that the initiating controller and its initiators comply with the low-level hardware definitions of SCSI-3, including attachment to a parallel SCFI bus.

An initiator may request a list of the operating definitions that the target supports and descriptive text for each operating definition using the INQUIRY command.

The parameters that can be changed by modifying the operating definition of a logical unit include the vendor identification, the device type, the device model, the SCFI compliance level, the SCFI specification level, the command set, and other parameters. The low-level hardware parameters including signal timing and parity definitions cannot be changed by modifying the operating definition. The present operating definition of a logical unit with respect to an initiator can be determined at any time by execution of an INQUIRY command. In some cases, it may be necessary to perform other commands including MODE SENSE and READ CAPACITY.

Each logical unit begins at a particular operating definition. If the logical unit supports the CHANGE DEFINITION command, the present operating definition can be changed to any other operating definition supported by the logical unit. The actual details of the operating definition of a logical unit are vendor-specific. If the operating definition is changed to one that does not include the CHANGE DEFINITION command, the target should continue to accept the CHANGE DEFINITION command.

If an error occurs during execution of a CHANGE DEFINITION command, the original operating definition remains in effect after the command is executed. The new operating definition becomes active only after successful execution of the CHANGE DEFINITION command.

Since new operating definitions may preclude the execution of I/O processes that are already in progress, the target controller shall disconnect to allow completion of any I/O processes that are in progress. (clear QIOPQ also? GRS)

Operating definition changes that may cause conflicts with the normal operation from other initiators shall be indicated to those initiators by generating a unit attention condition for each other initiator. The additional sense code shall be set to CHANGED OPERATING DEFINITION.

6.1.4 SCFI Addresses

There are two levels of addresses within the SCFI architecture: the SC FI address for a port and the logical unit number or target routine number.

6.1.4.1 SCFI Address for a Port (Mandatory)

A SCFI port responds to one SCFI address on the SCFI bus. Generally the logical element provides a means to select one of the 256 available addresses (0 to 255). Only a maximum of 255 other ports may be selected since the selecting port has one SCFI address for itself. Each port on the SCFI bus is assigned an unique address. (Parallel Interfaces.) The maximum SCFI address/SCFI ID combination is limited to 16 unique values in the range 0-15. This address is converted to the SCFI ID during bus arbitration and selection of SCFI devices.

Normally, the SCFI port address is set when the SCFI busses are config ured and it remains static thereafter. Some initiating controllers and target controllers provide vendor specific means to alter this address at other times.

6.1.4.2 Logical Units (Mandatory)

Each target controller shall have a minimum of one logical unit and first or only logical unit number shall be zero. Logical unit numbers shall be contiguous through the maximum number of logical units supported by the target controller. There can be a maximum of 256 logical units. These logical units are usually mapped directly to peripheral devices, but they may be a portion of a peripheral device or may comprise multiple peripheral devices.

An initiating controller can determine whether a target controller implements a logical unit by issuing an INQUIRY command and examining the returned peripheral qualifier and peripheral device type.

Logical unit(s) are defined only for the target role of each logical element. Initiating controllers implement a target role and shall define one or more logical unit numbers as any other principal target controller does.

6.1.4.3 Target Routines (Optional)

An optional feature of the SCFI architecture permits each target controller to have one or more target routines, beginning with target routine number zero. Target routine numbers shall be contiguous through the maximum number of target routines supported by the target controller. There is a maximum of 256 target routines. These target routines are processes that execute directly on the target controller and are not associated with a particular logical unit or peripheral device.

Target routines are principally intended to return information about the target controller. See section 7 for the valid command set for target routines.

6.1.5 Queued I/O Processes (Mandatory)

There are two methods for specifying the type I/O process, untagged (mandatory) and tagged (optional). The nature of the serial interface makes it necessary to be able to receive and store information packets before they are processed. Therefore, as a minimum, untagged queuing is a fundamental requirement for implementations using the serial interface. This is carried over as a requirement for the parallel interface to aid interoperability.

Untagged I/O processes allow a target controller to receive one I/O process from each initiating controller for each logical unit or target routine for each logical unit. Tagged I/O processes allow a target controller to accept multiple I/O processes from each initiating controller for each logical unit. Tagged I/O processes for target routines is not permitted. A target controller may be capable of both types of I/O processes, but only one method may be used at a time per logical unit.

Information packets permit the transfer of more than one ILE per connection for an I/O process. ILEs from one or more information packets may be queued in the target controller just as they may be queued in the initiating controller. That is, the entire initiating controller portion of an I/O process, if permitted by the information packet formation rules, may be transferred to the target controller in one set of information packets and queued in the taraget controller. The initiating controller follows the progress of the I/O process by processing status, logical block data, command response data, autosense, and messages received from the target controller in information packets. If additional information packets are required, the initiating controller sends them to the target controller at the appropriate time.

For implementations supporting only untagged I/O processes, the ILEs for the initiating controllers portion of an entire I/O process may be queued at the target controller. For some I/O processes, only the target controller status and associated messages for each command are required to be returned from the target controller in one or more information packets if there are no errors. If there is an error which causes termination of an I/O process, all of the remaining unprocessed ILEs from the initiating controller for the I/O process are purged from the active I/O process queue.

6.1.5.1 Untagged I/O Processes (Mandatory)

Untagged I/O processes allows a target controller to receive one I/O process from each initiating controller for each logical unit or target routine in the target controller. while one or more I/O processes from other initiating controllers is active. Each queued I/O process forms an H_C_x nexus.

The target controller shall be able to have at least one active I/O process. Other I/O processes may then be held in the queued I/O process queue waiting for activation. Optionally, the target controller may permit other combinations of active I/O processes which might include one active I/O process per logical unit and target routine, or more than one active I/O process per logical unit or target routine.

The specific maximum limits for I/O processes in the target controller are that the target shall not have active or queued more than a total of one I/O process per initiating controller per logical unit or target routine. For a logical system of two initiating controllers and one target controller with two logical units, each initiating controller can have one I/O process in the target controller for each logical unit or target routine. The target controller may not implement a queue depth to actually permit the four I/O processes to be active or queued. (See QUEUE FULL status.)

Only one command for each H_C_x nexus shall be executed at a time and in the order received. If the target controller has no space to store additional ILEs, for an existing active or queued I/O process, the

target controller shall return QUEUE FULL status to for the new information packet. The information packet may be resent at a later time.

An initiating controller may attempt to start or add to an I/O process any time the BUS FREE phase exists. (Parallel Interfaces.) If the disconnect privilege is not granted, the target shall return BUSY status to the new I/O process; the I/O process is not accepted by the target controller.

The H_C_x nexus sufficiently specifies the relationship so that the target controller can reconnect to the initiating controller for the I/O process. It is the responsibility of the initiating controller to assure that only one such I/O process is issued at any time. If the initiating controller attempts to establish a second untagged I/O process, the target controller shall return QUEUE FULL status to the new I/O process. That is, for untagged queuing, the maximum depth for a single initiator/target controller function is one active or queued I/O process. If the initiating controller attempts to send too many information packets for the untagged I/O process, the target controller shall return QUEUE FULL status to the new I/O process. In either case, the initiating controller may attempt to send the information packets at a later time.

6.1.5.2 Tagged I/O Processes (Optional)

Tagged I/O processes allow a target controller to accept multiple I/O processes from the same or different initiating controllers until the logical unit's queued I/O process queue is full. Tagged I/O processes shall not be implemented for target routines. Each I/O process forms an H_C_L_Q nexus.

Only one command for each H_C_L_Q nexus shall be executed at a time and in the order received. If the target controller has no space to store a new information packet for an existing active or queued I/O process, the target controller shall return QUEUE FULL status to for the new information packet. The information packet may be resent at a later time. (Parallel Interfaces.) If the disconnect privilege is not granted for a tagged I/O process the target shall return BUSY status; the I/O process is not accepted by the target controller.

Processing tagged I/O processes may be temporarily disabled internal to the target controller during certain initialization periods or to control internal resource utilization. During these periods the target shall return QUEUE FULL status.

Tagged queuing may also be disabled thourgh the MODE SELECT command on an individual logical unit basis. A target controller that does not support tagged queuing (e.g., not implemented, disabled by the DQue bit in the control mode page) shall respond to any tagged I/O process with an INVALID INFORMATION PACKET message. The I/O process is not accepted by the target controller. The initiating controller may convert the I/O process to an untagged I/O process or it may wait for or attempt to change the state of the state of the target controller for the logical unit.

The queued I/O process queue is setup by the target controller to provide independent support for each logical unit. Target routines shall not accept a tagged I/O process. Initiating controllers may add I/O processes to or delete I/O processes from their portion of the queued I/O process queue. When adding an I/O process, the initiating controller may specify fixed order of execution, allow the target to define the order of execution, or specify that the I/O process is to be executed next. Target controllers shall implement all of these queue management functions, if tagged I/O processes are supported.

The initiating controller uses the interface control prefix fields to specify a unique H_C_L_Q nexus for each I/O process. The H_C_L_Q nexus allows the target controller to reconnect to an initiating controller for a specific I/O process. An initiating controller may have several I/O processes queued or active to the same or different logical units as long as each has a unique nexus. A set of linked commands is a single I/O process, and each information packet is assigned the same queue tag value.

The target controller shall be able to have at least one active I/O process. Other I/O processes may then be held in the queued I/O process queue waiting for activation. Optionally, the target controller may permit other combinations of active I/O processes which might include one active I/O process per logical unit, or more than one active I/O process per logical unit, etc.

The specific maximum limits for I/O processes in the target controller are that the target shall not have active or queued I/O processes with duplicate tags per initiating controller per logical unit or target routine. For a logical system of two initiating controllers and one target controller with two logical units, each initiating controller can have 256 I/O processes in the target controller for each logical unit. The target controller may not implement a queue depth to actually permit the 1024 I/O processes to be active or queued. (See QUEUE FULL status.)

Each tagged I/O process specifies type of I/O process to be associated with the tag. The choices are simple, ordered and head of queue. In the examples that follow, HOQ-n means an I/O process with head of queue tag; Simple-n means an I/O process with a simple queue tag; Ordered-n means an I/O process with an ordered queue tag. In the case of Simple-n I/O processes, no execution order is implied in the example sequences.

For each logical unit when only only simple queued I/O processes are used, the target controller may execute the I/O processes for all initiating controllers in any order within the constraints of the queue management algorithm specified in the control mode page of the MODE SELECT/MODE SENSE commands (see 7.3.3.1).

If ordered queuing is used, the target shall execute the I/O processes in the order received independent of initiating controller. That is, only one ordered I/O process shall be executed per logical unit. All I/O processes received using simple queuing prior to an I/O process received using ordered queuing, regardless of initiating controller, shall be executed before the I/O process with the ordered queue tag. All I/O processes received with an simple queue tag after the last I/O process received with an ordered queue tag, regardless of initiating controller, shall be executed after after termination of that I/O process with the ordered queue tag.

If at least one ordered I/O process is received after one or more simple I/O processes, execute all simple I/O processes before executing the first of a new set of ordered I/O processes. The queue sequence is (Simple-1, ..., Simple-n, Ordered-1, Ordered-2). If simple I/O processes are received after one or more ordered I/O processes, execute all ordered I/O processes in the correct sequence before executing any simple I/O process in the new group. The queue sequence is (Ordered-1,..., Ordered-2, Simple-1, Simple-2). There may be multiple separate groups of simple and ordered I/O processes in the queued I/O process queue. The queue sequence may be (Ordered-x, Simple-x, Ordered-y, Simple-y, ...), where the "-x" and "-y" represent one or more I/O processes. The sequence may also start with Simple-x and then alternate.

An I/O process received using a head of queue tag is placed as the first queued I/O process in the queued I/O process queue. An I/O process received with a head of queue tag shall not suspend any active I/O process for the logical unit. Consecutive I/O processes received with head of queue tag, and placed in the queued I/O process queue before any become active, are executed in a last-in-first-out order.

If a head of queue I/O process is received and simple I/O processes are at the top of the queued I/O process queue, the resulting queue order is (HOQ-1, Simple-1, Simple-2, ...). If a head of queue I/O process is received and ordered I/O processes are at the top of the queued I/O process queue, the resulting queue order is (HOQ-1, Ordered-1, Ordered-2, ...). If a second head of queue I/O process is received, each sequence begins (HOQ-2, HOQ-1, ...).

Unless there is a contingent allegiance or extended contingent allegiance condition for a logical unit, an I/O process received without a queue tag specified while there are any tagged I/O processes in the

I/O process queues for the same initiating controller is an incorrect connection. That is, the initiating controller has failed to manage its queued I/O process queue correctly. An untagged I/O process shall only be directed to an active I/O process which has reported a status leading to contingent allegiance or extended contingent allegiance while other tagged I/O processes are in the queued or active I/O process queues for a logical unit.

If two or more active I/O processes are active as a result of decisions made in the queuing algorithm and an error affects their normal completion, a contingent allegiance or extended contingent allegiance condition shall be generated for all affected I/O proceses. If two I/O processes have pending reports of contingent or extended contingent allegiance to the same initiating controller for the same logical unit, the target controller shall report them serially. That is, a second error shall not be reported to the same initiating controller for the same logical unit while either a contingent allegiance or extended contingent allegiance is pending for the logical unit.

During recovery from the error reported for an I/O process using a queue tag, the target controller returns BUSY status to other initiating controllers for the logical unit while the contingent allegiance or extended contingent allegiance condition exists. During this time all I/O process execution in the active I/O process queue is suspended for the logical unit. No queued I/O processes are added to the active I/O process queue that affect the logical unit. All I/O processes used for recovery operations shall be untagged. During CAna d ECA, an I/O process using an ABORT TAG message is treated as an untagged I/O process even though it appears as an untagged I/O process for identification purposes. The QIOPQ may be modified by removing all or selected I/O processes in the queue as part of the recovery procedure for estended contingent allegiance.

One of two I/O process queue management options may be used after an active I/O process detects an error which results in contingent allegiance or extended contingent allegiance. The error recovery option is specified in the control mode page (see 7.3.3.1) of the MODE SELECT/MODE SENSE comands.

- The first recovery option is to continue execution of I/O processes remaining in the active and queued I/O process queues after the contingent allegiance or extended contingent allegiance condition has cleared.
- The second recovery option clears the active and queued I/O process queues for the logical unit after the contingent allegiance or extended contingent allegiance condition has been cleared. When the queue is cleared because of this recovery option, a unit attention condition shall be generated for all other initiating controllers and the additional sense code shall be set to I/O PROCESSES CLEARED BY ANOTHER INITIATOR. (update Section 7. GRS).

Several messages may be used to clear part or all of the command queue during the recovery process. For contingent allegiance, only one I/O process which includes a command can be executed which may include linked commands. I/O processes containing only messages may be executed without destroying the preserved sense data. (should an initiator be allowed an INITIATE RECOVERY message to convert a CA to ECA at its choosing? GRS)

For extended contingent allegiance, several untagged I/O processes may be executed until the RELEASE RECOVERY message is executed at the target controller. Refer to the ABORT, ABORT TAG, BUS DEVICE RESET, and CLEAR QUEUE messages in this section for details.

Deferred errors are normally related to an I/O process that has already completed. The target controller shall not retain the queue tag value assigned to the original I/O process.

(add/revise for ASSIGN/UNASSIGN. GRS)

The TEST UNIT READY and INQUIRY commands are often sent as head of queue, since the information returned has no effect on the condition of the logical unit.

The RESERVE and RELEASE commands should be sent as an ordered I/O process. Use of the head of queue or simple queue functions with these I/O processes could result in reservation conflicts when other queued I/O processes become active; such I/O processes are purged by the target controller as if they had never been received by reporting RESERVATION CONFLICT status to the initiating controller(s) for each affected I/O process.

6.1.5.3 Concurrent I/O Processes Execution

The various options for processing untagged and tagged processes permit concurrent execution of I/O processes within a target controller function under certain circumstances. The rules for concurrent processing are summarized below

CONCURRENT I/O PROCESS EXECUTION WITHIN A LOGICAL UNIT

I/O PROCESS GROUP	TAG TYPE	MAX NUMBER PERMITTED
Untagged	Treated as SIMPLE	1 per initiating controller up to number of different initiating controllers
Tagged	SIMPLE	Up to number of processes target controller can queue independent of number of initiating controllers
Tagged	ORDERED (fifo)	1
Tagged	HEAD OF QUEUE (lifo)	1

6.1.6 Logical Unit Reservation Termination (Mandatory)

(add condiserations of when reservations are removed. GRS)

6.1.7 Assignment Termination (Optional)

(add condiserations of when assignmane is removed. GRS)

6.1.8 Path Group Termination (Optional)

(add condiserations of when path groups are removed. GRS)

6.2 Commands

By specifying only the functions essential to communicate via this protocol, a wide range of peripheral devices of varying capability can operate in the same environment. Because subsets of the full command set may be implemented, optional functions are noted. A command consists of a command descriptor block and, in some instances, command parameter data.

Singular commands are commands, possibly including command parameter data, which must appear in an I/O process by themselves. That is, there shall be no preceeding command with the link bit set to one and a singular command shall have its link bit set to zero.

Supervisor commands are commands which shall not be executed in an I/O process unless specifically enabled in the interface control prefix fields. Such commands normally affect access to logical units or extents. The interface control prefix fields, if controlled by a supervisory agent in the initiating controller, can indicate when supervisor commands are permitted in an I/O process. Such a supervisory function is optional, but it provides a check on the content of I/O processes which can affect availability and data reliability.

Three terms commonly used with commands are receipt, acceptance, and execution. Receipt of a command is not the same as acceptance of a command. Receipt refers only to the transfer of a command in an information packet. The command may, as part of its I/O process, go to the queued I/O process queue or to the active I/O process queue.

Acceptance of a command means that the I/O process is active, the command is next to be executed, and that all fields are valid for the implementation and the current conditions in the logical unit or target routine permit attempting execution. A received command may be accepted or rejected.

Execution applies only to commands which have been received and accepted. The implemented procedure for the command is performed to the best ability of the target controller. The command may terminate sucessfully or with error.

A command which is received, accepted, executed, and completed sucessfully causes status and a command completion message to be prepared for transmission to the initiating controller in an information packet; the status and command completion message may not be sent until later with other status and messages for the I/O process.

A command which is received, accepted, and, after execution begins, terminates with an error causes status to be prepared, a contingent or extended contingent allegiance to be established, sense data to be prepared, and final messages for the I/O process. The information packet is sent at this time to inform the initiating controller of the error. Sense data may be included in the information packet as autosense data. The final sequence of interface logical elements is: CHECK CONDITION or I/O PROCESS TERMINATED status ILE, autosense in a CRD ILE (optional), INITIATE RECOVERY message ILE (optional), and a COMMAND COMPLETE message ILE. The INITIATE RECOVERY message and the COMMAND COMPLETE message may be combined in a single message ILE. See Sections 6.3, 6.4, 6.5, and 6.6 for discussions of status, contingent allegiance and extended contingent allegiance.

A command which is received and rejected causes status to be prepared, a contingent allegiance to be established, sense data to be prepared, and status with final messages for the I/O process. The information packet is sent at this time to inform the initiating controller of the error. The methods for sending sense data are described above.

6.2.1 Command Set Implementation Requirements (Mandatory)

A device class is a group of devices that implement the mandatory commands and functions in Sections 5 through 7 and the mandatory commands and functions defined in one of the sections 8 through 17.

A device type is a specific implementation within a device class. Each device type may implement different optional features of its device class, but it shall implement the mandatory commands and functions in Sections 5 through 7 and the mandatory commands and functions in the appropriate section for its device class.

6.2.2 Reserved (Mandatory)

Reserved bits, fields, bytes, and code values are set aside for future standardization. Their use and interpretation may be specified by future extensions to this standard. A reserved bit, field, or byte shall be set to zero.

A target controller that detects a reserved bit, field, or byte that i s not zero or detects a reserved code value shall shall reject the command and terminate the I/O process without altering the medium. A contingent allegiance is established with the initiating controller. The sense key shall be set to ILLEGAL REQUEST.

6.2.3 Command Descriptor Block

A command is communicated by sending a command descriptor block (CDB) to a target controller. For several commands, the command descriptor block requires command parameter data (CPD). See the specific commands for detailed information about which commands require command parameter data.

The command descriptor block shall have the operation code as its first byte and a control field as its last byte (6.2.4 and 6.2.9).

Command parameter data may be sent as an interface logical element in the same information packet as the CDB, in a separate information packet, or split across several information packets. (See 6.2.7.)

If there is an invalid parameter in the command descriptor block or the command parameter data, the target controller shall reject the command and terminate the I/O process without altering the medium. A contingent allegiance is established with the initiating controller. The sense key shall be set to ILLEGAL REQUEST.

Table 6-1, Table 6-2 on page 6-19, and Table 6-3 on page 6-19 are typical command layouts. These are examples and do not specify the exact field layout for all commands. See the field layout for each command. General descriptions of the command descriptor block fields are given after the layouts.

Table 6-1. Typical Command Descriptor Block for Six-Byte Commands

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved			(MSB) Logical Block Address (Cont'd)				
2 - 3	Logical Block Address (if required) (LSB)							
4	(MSB) Transfer Length (if required) Command Parameter Data Length (if required) Command Response Data Length (if Required) (LSB)							
5	Control Field							

Table 6-2. Typical Command Descriptor Block for Ten-Byte Commands

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2 - 5	(MSB) Logical Block Address (if required) (LSB)							
6	Reserved							
7 - 8	(MSB) Transfer Length (if required) / Command Parameter Data Length (if required) / Command Response Data Length (if Required) (LSB)							
9	Control Field							

Table 6-3. Typical Command Descriptor Block for Twelve-Byte Commands

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2 - 5	(MSB) Logical Block Address (if required) (LSB)							
6 - 9	(MSB) Transfer Length (if required) / Command Parameter Data Length (if required) / Command Response Data Length (if Required) (LSB)							
10	Reserved							
11	Control Field							

6.2.4 Operation Code

The first byte of all SCFI commands shall contain an operation code as defined in this standard. The operation code definition may change between device classes. Only the common commands, in Section 7. are guaranteed to have the same meaning in all device classes. Operation codes are divided in four categories as follows:

Operation Code Type	Description
M	Mandatory - Commands so designated shall be implemented in order to meet the minimum requirement of this standard.
O	Optional - Commands so designated, if implemented, shall be implemented as defined in this standard.
V	Vendor specific - Operation codes so designated are available for vendor defined commands. See the vendor specifications where compatibility is desired.
R	Reserved - Operation codes so designated shall not be used. They are reserved for future extensions to this standard.

The operation code (Table 6-4) of the command descriptor bl ock has a group code field and a command code field. The three-bit group code field provides for eight groups of command codes. The five-bit command code field provides for thirty-two command codes in each group. Thus, a total of 256 possible operation codes exist. Operation codes are defined in the subsequent sections of this document. Operation codes may be redefined in each device class section. Only the commands in Section 7 are guaranteed to have the same meaning in all device classes.

Table 6-4. Operation Code

Bit / Byte	7	6	5	4	3	2	1	0
0	Group Code			Command Code				

Group Codes:

- Group 0 - six-byte commands (see Table 6-1 on page 6-18)
- Group 1 - ten-byte commands (see Table 6-2 on page 6-19)
- Group 2 - ten-byte commands (see Table 6-2 on page 6-19)
- Group 3 - reserved
- Group 4 - reserved
- Group 5 - twelve-byte commands (see Table 6-3 on page 6-19)
- Group 6 - vendor specific
- Group 7 - vendor specific

6.2.5 Logical Block Address

The logical block address for logical units or within a partition of a logical unit shall begin with block zero and be contiguous up to the last logical block for that logical unit or within that partition. Not all device classes support logical block addressing. See the model at the beginning of each device class section.

For those commands providing logical block addresses: a six-byte command descriptor block contains a 21-bit logical block address; a ten-byte or twelve-byte command descriptor block contains a 32-bit logical block address. Logical block addresses in command parameter data have their length specified for each occurrence. See the specific command descriptions.

(drop duplicate 6 or 10 byte commands in favor of 10 or 12 byte commands? GRS)

6.2.6 Transfer Length

The transfer length field specifies the amount of data to be transferred, usually the number of blocks. Not all commands provide a transfer length field. A target controller shall not attempt to transfer more data than specified by the command descriptor block transfer length value (i.e., block count times block size for fixed block formats). An initiating controller shall not attempt to transfer more data specified by the command descriptor block transfer length value (i.e., block count times block size for fixed block formats).

For several commands the transfer length indicates the requested number of bytes to be sent as defined in the command description. For these commands the transfer length field may be identified by a different name. See the individual command descriptions for further information.

Several commands that use one byte for the transfer length allow up to 256 blocks or bytes of data to be transferred by one command. A transfer length value of 1 to 255 indicates the maximum number of blocks or bytes that shall be transferred. A value of zero indicates 256 blocks or bytes. At least one block or byte is specified for transfer in any such command. Some commands specify that a value of zero means that no bytes or blocks are to be transferred. This shall not be considered as an error.

In commands that use more than 8 bits for the transfer length, a transfer length of zero indicates that no data transfer shall take place. A value of one or greater indicates the maximum number of blocks or bytes that shall be transferred.

Refer to the specific command descriptions for further information.

6.2.7 Command Parameter Data Length

The command parameter data length field is used to specify the exact number of bytes in command parameter data. An initiating controller shall not include, as part of the command, command parameter data which exceeds the length specified in the command descriptor block. Command parameter data may be included as a subsequent interface logical element in the same information packet as the command descriptor block, a separate information packet, or split across multiple information packets. This field is typically used in command descriptor blocks for command parameter data that are sent to a target controller (e.g., mode parameters, diagnostic parameters, log parameters, etc.) but which is not logical block data.

A command parameter data length field of zero indicates that no data shall be transferred. This condition shall not be considered as an error.

A target controller which detects the length of the command parameter data to be different than specified in the command descriptor block shall reject the command, terminate the I/O process with contingent allegiance, and set the sense key to ILLEGAL REQUEST.

Since initiating controllers prepare information packets, if an initiating controller attempts to include command parameter data than is not equal in length to the CDB value, ...(? GRS)

(Section 7+ CDBs may require changes. GRS)

6.2.8 Command Response Data Length

The command response data length field specifies the maximum number of bytes that a target controller may transfer as command response data. The command response data length is used to limit the amount of command response data (e.g., sense data, mode data, log data, diagnostic data, etc.) returned to an initiating controller by a target controller. A command response data length of zero indicates that no data shall be transferred. This condition shall not be considered as an error.

The target controller may place command response data in a single command response data information logical element, or split the command response data over multiple information packets. The total length of the data bytes shall not exceed the command response data length specified in the command descriptor block.

Since target controllers prepare information packets, if a target controller attempts to include command response data than is greater than the length in the CDB value, ...(? GRS)

(Section 7+ CDBs may require changes. GRS)

6.2.9 Control Field

The control field is the last byte of every command descriptor block. The control field is defined in Table 6-5.

Table 6-5. Control Field

Bit Byte	7	6	5	4	3	2	1	0
-	Vendor Specific		Reserved				Flag	Link

Support for the link bit and flag bit is mandatory for target controllers.

Singular commands require that each such commands be in an I/O process by themselves. The link bit shall be zero. Refer to the command descriptions in Section 7. All device class commands support both the link bit and the flag bit.

6.2.9.1 Link Bit (Mandatory)

The link bit is used to continue an I/O process across multiple commands. Status referred to is described in section 6.3. The last or only command of a set of linked commands has its link bit set to zero.

A link bit of one indicates that the initiating controller requests a continuation of the I/O process using the target controller active I/O process queue for the next command or wait for a new command from the initiating c ontroller upon successful transfer of status and messages for the current command. If the link bit is one, and if the command completes successfully, the target controller shall return INTERMEDIATE or INTERMEDIATE-CONDITION MET status and shall then send one of the two messages defined by the flag bit.

If an active I/O process is partially in the target controller with the remaining portion in the initiating controller, the target controller shall send required information to the initiating controller in one or more information packets sufficient to establish the completion of all commands, in sequence, up through the present command. The initiating controller shall then transfer one or more information

packets to continue the I/O process. These information packets shall not be placed in the queued I/O process queue.

If the link bit is zero, and if the command completes successfully, the target controller shall send GOOD or CONDITION MET status and the COMMAND COMPLETE message. The I/O process terminates.

If the link bit is zero or one, and if the command completes unsuccessfully, the target controller shall send the COMMAND COMPLETE message. That is, if the comand is received, accepted and execution ends with an error (as opposed to an initiating controller forced termination, see Section 6.3), the I/O process terminates at the completion of contingent allegiance or extended contingent allegiance.

If the link bit is set to one in a singular command, the target controller shall reject the command and terminate the I/O process with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST.

6.2.9.2 Flag Bit (Mandatory)

The flag bit specifies which message the target controller shall return to the initiating controller if the link bit is one and the command completes without error. Status referred to is described in section 6.3.

The flag bit shall be set to zero if the link bit is zero. If link bit is zero and the flag bit is one, the target controller shall reject the command and terminate the I/O process with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST.

If the flag bit is zero and the link bit is one, and if the command completes successfully, the target controller shall send the LINKED COMMAND COMPLETE message. If the flag bit is one and the link bit is one, and if the command completes successfully, the target controller shall send the LINKED COMMAND COMPLETE (WITH FLAG) message.

6.3 Status (Mandatory)

The status byte and its status byte code are specified in Table 6-6 on page 6-24 and Figure "Status Byte Code", respectively. A status byte shall be sent from the target controller to the initiating controller at the completion of each command unless the command is terminated by one of the following events:

- an ABORT message,
- an ABORT TAG message,
- a BUS DEVICE RESET message,
- a CLEAR QUEUE message,
- a hard reset condition, or
- an unexpected disconnect.

In some cases, status transfer may occur prior to transferring a command descriptor block.

Status transfer can lead to a contingent allegiance condition, an extended contingent allegiance condition, or be a report. Each status byte code identifies the type of status that it is.

Table 6-6. Status Byte

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved	Status Byte Code						Reserved

A status byte code value which leads to contingent allegiance or extended contingent allegiance shall cause the target controller to prepare sense data for the initiating controller to help identify the error or problem. See sections 6.4 and 6.5. Sense data is a special form of command response data which may be transferred to the initiating controller in response to a REQUEST SENSE command, during asynchronous event notification, or as autosense data appended behind the status ILE for these selected status byte code values in a command response data interface logical element. See the REQUEST SENSE command in section 7 and the SEND command in section 11 for the format of sense data.

Table 6-7. Status Byte Code

Bits of Status Byte										
7	6	5	4	3	2	1	0	Status	Type	Support
R	0	0	0	0	0	0	R	GOOD	Report	M
R	0	0	0	0	0	0	R	CHECK CONDITION	CA, ECA	M
R	0	0	0	0	1	0	R	CONDITION MET	Report	O
R	0	0	0	1	0	0	R	BUSY	Report	M
R	0	0	1	0	0	0	R	INTERMEDIATE	Report	M
R	0	0	1	0	1	0	R	INTERMEDIATE - CONDITION MET	Report	O
R	0	0	1	1	0	0	R	RESERVATION CONFLICT	Report	M
R	0	1	0	0	0	1	R	I/O PROCESS TERMINATED	CA, ECA	M
R	0	1	0	1	0	0	R	QUEUE FULL	Report	M
R	1	0	1	1	0	0	R	ASSIGNMENT CONFLICT	Report	O
All Other Codes								Reserved		

Key: R - Reserved bit = 0b
CA - Contingent Allegiance
ECA - Extended Contingent Allegiance
M - Mandatory
O - Optional

6.3.1 Status Codes (Mandatory)

The definitions of the status byte codes are given below.

GOOD. This status or CONDITION MET indicates that the target controller has successfully completed a single command I/O process or the last command of a set of linked commands for an I/O process. See INTERMEDIATE status for all commands in a set of linked commands with the link bit set to one. This is a report status from a target controller.

CHECK CONDITION. This status indicates that an error has occurred. This status code requires contingent allegiance or extended contingent allegiance be established in the target controller.

CONDITION MET. This status or INTERMEDIATE-CONDITION MET is returned whenever the requested operation is satisfied for certain commands. (See the SEARCH DATA and PREFETCH commands). The link bit for commands receiving CONDITION MET status shall be zero. This is a report status from a target controller.

BUSY. This status shall be returned whenever a target controller is unable to retain an information packet from an otherwise acceptable initiator (i.e., no reservation conflicts). The recommended initiator recovery action is to issue the command again at a later time. See RESERVATION CONFLICT for information packets transferred by unacceptable initiators. The information packet is not retained by the target controller. This is a report status from a target controller. For initiating controllers, refer to the RESEND PREVIOUS INFORMATION PACKET message.

INTERMEDIATE. This status or INTERMEDIATE-CONDITION MET shall be returned for every successfully completed command in a set of linked commands where the link bit is set to one. If INTERMEDIATE or INTERMEDIATE-CONDITION MET status is not returned for each command with its link bit set to one, the I/O process is terminated. This is a report status from a target controller.

INTERMEDIATE-CONDITION MET. This status is a combination of CONDITION MET and INTERMEDIATE status. (See the SEARCH DATA and PREFETCH commands). The link bit for commands receiving INTERMEDIATE-CONDITION MET status shall be one. This is a report status from a target controller.

RESERVATION CONFLICT. This status shall be returned whenever an initiator attempts to access a logical unit or an extent within a establish an I/O process for a logical unit or an extent within a logical unit that is reserved with a conflicting reservation type. The recommended initiator recovery action is to transfer the information packet again at a later time. An I/O process shall not be established in the target controller. This is a report status from a target controller.

I/O PROCESS TERMINATED. This status shall be returned whenever the target terminates an active I/O process after processing a TERMINATE I/O PROCESS message. This status code requires contingent allegiance or extended contingent allegiance be established in the target controller.

QUEUE FULL. This status is returned when a simple queue, ordered queue, or head of queue I/O process is attempted to be established and the queued I/O process queue is full. This status is also returned for information packets of an established queued I/O process when there is insuficcient space to store more information packets for the I/O process. The information packet is not placed in the queued I/O process queue. This is a report status from a target controller. For initiating controllers, refer to the RESEND PREVIOUS INFORMATION PACKET message.

ASSIGNMENT CONFLICT. This status shall be returned whenever an initiator attempts to access a logical unit that is assigned to another path. An I/O process shall not be established in the target controller. This is a report status from a target controller.

6.3.2 Status Code Reporting Priority (Mandatory)

The priority of reporting status byte codes proceeds from the most restrictive to the least restrictive meanings.

Table 6-8. Status Byte Code Reporting Priority	
STATUS NAME	**PRIORITY**
ASSIGNMENT CONFLICT.	Highest
RESERVATION CONFLICT.	
BUSY.	
QUEUE FULL.	
I/O PROCESS TERMINATED.	
CHECK CONDITION.	
INTERMEDIATE or INTERMEDIATE - CONDITION MET.	
GOOD or CONDITION MET.	Lowest

ASSIGNMENT CONFLICT is the highest priority code since it indicates a condition in the target controller which prevents I/O processes from being established until the assignment is removed by some third party.

RESERVATION CONFLICT is the second highest priority code since it indicates a condition in the target controller which prevents I/O processes from being established until the reservation is removed by some third party.

BUSY and QUEUE FULL indicate that no reservation conflict exists, but that at the present time, the target controller cannot accept additional information packets. BUSY indicates a target controller condition which has resources tied up which prevent checking the queue status. QUEUE FULL merely indicates a lack of available resources to store information packets.

I/O PROCESS TERMINATED and CHECK CONDITION indicate an error in processing some portion of an I/O process. I/O PROCESS terminated indicates forced termination of an I/O process by the initiating controller. CHECK CONDITION indicates a target controller detected error which prevents further processing of the I/O process.

INTERMEDIATE and INTERMEDIATE-CONDITION MET status byte codes are of equal priority since the use of the status is dictated by the command processing rules. Intermediate status indicates a continuing responsibility on the part of the target controller and initiating controller to continue an I/O process containing a set of linked commands.

GOOD and CONDITION MET status byte codes are of equal priority since the use of the status is dictated by the command processing rules. Either status indicates that the I/O process is terminated normally by the target controller.

6.4 Contingent Allegiance Condition (Mandatory)

Implementation of contingent allegiance (CA) is a mandatory. See the glossary, Section 3.1 for a definition. The continginent allegiance condition permits reporting an error to an initiating controller which requires transferring an appropriate status and corresponding sense data. The sense data may be held in the target controller until requested by or deleted by the initiating controller, or it may be included as an autosense ILE following the status ILE at the time the error is reported. A contingent allegiance and its subsequent clearing results in termination of the affected I/O process. Any information packets held in the target controller for the I/O process are deleted.

If contingent allegiance is the reporting method selected for an error, the condition shall exist from detection of an error which causes CHECK CONDITION or I/O PROCESS TERMINATED status or following an unexpected disconnect. The alternative condition is extended contingent allegiance.

Target controllers shall report a maximum of one contingent allegiance or ECA per initiator per logical unit or target routine. If autosense is used for reporting contingent allegiance, the condition is cleared with successful transmission of the information packet. If CA or ECA is outstanding to the initiating controller for the logical unit, the condition is not reported until the prior existing condition is cleared. This is required because the recovery action consists of one or more untagged I/O processes. If multiple CAs or ECAs are reported, the target controller and initiating controller cannot determine which condition to respond to or which condition is being reported.

While the contingent allegiance condition exists the target shall preserve the appropriate sense data not successfully transmitted as autosense. The condition persists until sense data has been retrieved or the initiating controller has taken an action to reset the held sense data. If the target controller selects to report the sense data using autosense (i.e., including the sense data as a CRD ILE with the status) the contingent allegiance condition is cleared with successsful transfer of the information packet since all reporting requirements have been met.

While the contingent allegiance condition exists, if the target is unable to maintain separate sense data, the target shall respond to any other requests for access to the logical unit or target routine from another initiator with a BUSY status. Execution of commands for the logical unit or target routine for which the contingent allegiance condition exists shall be suspended until the contingent allegiance condition is cleared.

If only status is reported to the initiating controller, the contingent allegiance condition shall be preserved for the H_C_x_y nexus on path where it is reported, until it is cleared. The type of nexus and conditions in effect for the nexus, established with the connect, determine which path is used to report the status leading to contingent allegiance.

The contingent allegiance condition shall be cleared upon the generation of a hard reset condition, or by an ABORT message, a BUS DEVICE RESET message, or any subsequent command for the nexus. If the subsequent command is a REQUEST SENSE command, the sense data is reported in response to the command and then the contingent allegiance condition is cleared.

6.5 Extended Contingent Allegiance Condition (Optional)

Implementation of extended contingent allegiance (ECA) is optional. See the glossary, Section 3.1 for a definition. The extended contingent allegiance condition extends a contingent allegiance condition for an H_C_x_y nexus to include a protected period of time for the initiating controller to take appropriate action related to the error reported. An extended contingent allegiance and its subsequent clearing results in termination of the affected I/O process. Any information packets held in the target controller for the I/O process are deleted.

An extended contingent allegiance condition should not be generated for every CHECK CONDITION or I/O PROCESS TERMINATED status that occurs. Simple errors not requiring an extended recovery may be dealt with by using the contingent allegiance protocol.

Target controllers shall report a maximum of one ECA or CA per initiator per logical unit or target routine. If CA or ECA is outstanding to the initiating controller for the logical unit, the condition is not reported until the prior existing condition is cleared. This is required because the recovery action consists of one or more untagged I/O processes. If multiple CAs or ECAs are reported, the target con-

troller and initiating controller cannot determine which condition to respond to or which condition is being reported.

This condition is generated by the target sending an INITIATE RECOVERY message on the same path as and following the CHECK CONDITION or I/O PROCESS TERMINATED status and prior to the COMMAND COMPLETE message. The sense data may be held in the target controller waiting for a REQUEST SENSE command to retrieve it, or the sense data may be included with the status report as a CRD ILE immediately following the status ILE and before the INITIATE RECOVERY message. Retriveing sense data does not end an extended contingent allegiance as it does a contingent allegiance. With either method of reporting sense data, this condition shall be preserved until it is cleared by a BUS DEVICE RESET message, a RELEASE RECOVERY message, or a hard reset condition.

If the target controller needs to generate an extended contingent alle giance condition during an asynchronous event notification (6.6), it shall send an INITIATE RECOVERY message prior to the SEND command. (See Section 6.6.)

While the extended contingent allegiance condition exists, the target shall respond to any other request for access to the logical unit from any other path with BUSY status. Execution of commands for the logical unit for which the extended contingent allegiance condition exists from the active I/O process queue shall be suspended until the extended contingent allegiance condition is cleared. No queued I/O process shall be made active. An extended contingent allegiance condition can be generated for only one H_C_L nexus at a time. The extended contingent allegiance condition is cleared as defined above when a RELEASE RECOVERY message is received from the initiating controller for which the INITIATE RECOVERY message was sent.

During an extended contingent allegiance, only untagged I/O processes from the eligible path shall be executed for the logical unit. If the initiator sends a tagged I/O process, the target controller shall respond with QUEUE FULL status. After the extended contingent allegiance condition is cleared any commands remaining in the command queue shall be processed per the rule in section 7.3.3.1. (cross check with queued I/O processes. GRS)

During the extended contingent allegiance condition, appropriate error recovery sequences may be executed. Such I/O processes can correct data, modify or delete queued commands, perform LOG SENSE commands and obtain diagnostic information. Extended contingent allegiance is recommended for error conditions that may require execution of multiple-step error-recovery procedure without interference from other initiators. The recommended recovery procedure is target controller vendor specific. The initiating controller is not required to follow the recommended recovery procedure; it must only exit by sending a RELEASE RECOVERY message. However, the target controller function may be compromised as a result of not following the recommended procedure.

6.6 Asynchronous Event Notification (Mandatory)

Implementation of asynchronous event notification is mandatory. An asynchronous event is one which occurs in a target controller which affects the operation of the target controller with its initiating controller(s), but which does not occur as a result of an active I/O process. The unit attention condition is an instance where asynchronous event notification shall be used. An example of an asynchronous event is medium removal in a sequential access device, a removable optical device, or a CD_ROM. An SCFI device, which normally functions as a target controller, shall implement initiator mode to notify an initiating controller of an asynchronous event.

Retrieval of the INQUIRY data by target controllers shall be performed prior to the first use of asynchronous event notification with an initiating controller through an initiator.

SCFI devices that respond to an INQUIRY command as a processor device class, may be notified of asynchronous events using this protocol. Such SCFI devices are normally initiators attached to an initiating controller. Asynchronous event notification may not be used with a device that reports a device class other than procressor (e.g., devices executing COPY commands). The INQUIRY command should be issued to logical unit zero of each SCFI device responding to selection.

Each SCFI device that responds as processor device class shall be issued a TEST UNIT READY command to determine that the SCFI device is ready to receive an asynchronous event notification. An SCFI device returning CHECK CONDITION status without autosense is issued a REQUEST SENSE command to clear any pending unit attention condition. Processor device class SCFI devices which return GOOD status when issued a TEST UNIT READY command shall accept a SEND command with an AEN bit of one.

A SEND command with an AEN bit of one is interpreted as an asynchronous event by an initiating controller. The sense data identifying the condition being reported shall be returned as a command parameter data of the the SEND command (see 11.2.2.). The data sent is shown in Table 11-4.

An error condition or unit attention condition shall be reported once per occurrence of the event causing it (See contingent allegiance, Section 6.4 and extended contingent allegiance, Section 6.5). Notification of command-related error conditions shall be sent only to the initiating controller where the connect was made for the I/O process. That is, any path to the initiating controller may be used for asynchronous event notification.

This protocol is not intended to be used while an H_C_L nexus, H_C_R nexus, or H_C_L_Q nexus exists with an initiating controller for an active I/O process. Asynchronous event notification is not intended for use with target routines (i.e., an H_C_R nexus) but its use is not prohibited.

The procedure to identify candidate initiating controllers through their initiators shall be repeated whenever a target controller deems it appropriate or when an event occurs that may invalidate the current information. (See 6.8.16, SYNCHRONOUS DATA TRANSFER REQUEST message, for examples of these events.)

The four principal uses of asynchronous event notification are:

(1) informing a processor device of an error condition encountered after I/O process completion;

(2) informing all processor devices that a newly initialized device is available;

(3) informing all processor devices of other unit attention conditions; and,

(4) informing all processor devices of other asynchronous events.

Other uses of asynchronous event notification are not prohibited.

An example of the first case above is a device that implements a write cache. If the target is unable to write cached data to the medium, it may use an asynchronous event notification to inform the initiating controller of the failure. An extended contingent allegiance condition may also be established during the same H_C_L nexus used for the asynchronous event notification.

An example of the second case above is using the asynchronous event notification protocol to notify initiating controllers that a system resource has become available. The sense key in the sense data sent to the initiating controller shall be set to UNIT ATTENTION.

An example of the third case above is a device that supports removable media. Asynchronous event notification may be used to inform an initiating controller of a not-ready-to-ready transition (medium

changed) or of an operator initiated event (e.g., activating a write protect switch or activating a start or stop switch).

An example of the fourth case above is a sequential-access device performing a REWIND command where the immediate bit had been set to one. Asynchronous event notification may be used to inform an initiator that the beginning of medium has been reached or that the rewind operation failed. Completion of a CD-ROM AUDIO PLAY command started in the immediate mode is another example of this case.

6.7 Information Packet Structure

An information packet is set of self-describing interface logical elements in defined orders surrounded by the appropriate interface control fields necessary to send the logical content of the packet on a physical bus (Table 6-9). The maximum information packet length is 2112 bytes.

Information packets shall be a multiple of 4 bytes in length. Pad bytes are added to permit construction of information packets of such lengths. The value of the pad bytes is not specified in Fiber Channel. The value of pad bytes in SCFI shall be 00h.

An information packet shall contain at least one interface logical element and shall not contain more than 255 interface logical elements.

If an information packet is not a multiple of 4 bytes long or there are zero or greater than 255 interface logical elements in the information packet, the information packet shall be rejected. The target controller returns an information packet with an INVALID INFORMATION PACKET message.

Table 6-9. Information Packet Structure

Bit / Byte	7	6	5	4	3	2	1	0
0 to m	Interface Control Prefix Fields							
m+1 to n	One to 255 Interface Logical Elements							
n+0 to n+3	Interface Control Suffix Fields							

6.7.1 Interface Control Fields (Mandatory)

The interface control fields provide a carrier for the information content of the SCFI logical operations. The control fields vary between the serial interface and the parallel interface. The difference is primarily because of the mode of transfer and the requirements for error detection. Serial interfaces use a cyclic redundancy check field to detect errors in transmission. The parallel interface uses the parity signals and correponding checkers to check each byte as it is received. The serial interface also has additional requirements for the frame structure to permit routing frames through the fabric. The parallel interface control fields are presented first since the serial interface control prefix fields are strictly additions to the parallel interface control fields.

The interface control fields encapsulate the interface logical elements. The identify function for both initiating controllers and target controllers uses the interface control fields to establish the nexus and direct each in managing the I/O process.

6.7.1.1 Interface Control Fields (Mandatory)

The mandatory interface control fields are a subset of the interface control fields for the serial interface. The mandatory interface control fields are used with the parallel interface. That is, a parallel information packet is encapsulated by the additional fields required for the serial interface. The mandatory fields are described first and then the serial fields are added before and after the mandatory interface control fields to form a serial interface information packet.

The reasons for the parallel information packet being of different are: routing of information packets is not present on the parallel interface (possible fabric routing is absent); CRC is not required on the parallel bus since parity is checked on the parallel bus as each information packet is transferred.

Table 6-10. Interface Control Prefix Fields (Mandatory)

Bit / Byte	7	6	5	4	3	2	1	0
0 - 1	(MSB) Packet Length (LSB)							
2	Packet Type							
3	Reserved							
4	LUNTRN Valid	LUNTRN	QNexus	HOQ	OrdSim	DiscPriv	Pad Byte Count	
5	MltPath	SuspMpth	EnbSpvr	Reserved				
6	Initiating Controller Port Number							
7	Reserved							
8	Original Initiator SCFI Address							
9	Reserved							
10	Original Target SCFI Address							
11	Target Controller Port Number							
12	LUNTRNID							
13	Queue Tag							
14-15	Reserved Log. Elem. Seq. No? GRS)							

Table 6-10 shows the interface control prefix fields for the parallel interface. All reserved fields and bits shall be set to zero.

The packet length field indicates the total length of the information packet in bytes, including the interface control prefix fields and the interface control suffix fields. The actual usable maximum packet length is a function of the selected physical interface and may be further restricted by maximum burst length negotiations. This value shall be a multiple of 4 and greater than or equal to 24.

The packet type field identifies the general content of an information packet.

- A packet type code of 00h indicates an information packet to start a new nexus.
- A packet type code of 01h indicates a target controller initiated packet to terminate an I/O process.
- A packet type code of 02h indicates an intermediate information packet from an initiating controller to continue an I/O process established with an information packet of type 00h. A packet type code of 03h indicates an
- A packet type code of 03h indicates an intermediate information packet from a target controller to continue an I/O process established with an information packet of type 00h.
- A packet type code of 04h indicates that a nexus is requested to be established by the initiating controller from a target controller because of asynchronous event notification.
- Packet type code values of 05h through FFh are reserved.

A logical unit/target Valid (LUNTRN Valid) bit of zero specifies that the I/O process is to the target controller and forms an H_C nexus. A LUNTRN Valid bit of one specifies that the information packet is for a logical unit or target routine as specified by the LUNTRN bit and forms an H_C_x_y nexus. Certain information packets cannot be directed to a specific logical unit or target routine. The target controller shall use this bit, when 1b, as a valid field indicator for the LUNTRN, HOQ, OrdSim, LUNTRNID, and Queue Tag fields. This value shall remain unchanged in all information packets for the nexus. An I/O process with a LUNTAR Valid bit of zero shall operate in single path mode.

A logical unit/target (LUNTRN) bit of zero specifies that the I/O process is for a logical unit; an H_C_L or H_C_L Q nexus will be formed depending on the value of the QNexus bit. A LUNTRN bit of one specifies that the I/O process is for a target routine; an H_C_R nexus is formed. The target controller shall ignore this bit when the LUNTRN Valid bit is zero. The logical unit or target routine is identified in the LUNTRNID field. When sent by a target controller during a reconnection, the value of this bit shall be the same as the most recent value received for this I/O process. The value supplied in this field with the connect shall not change during the information packet exchange related to the nexus from the value transferred in the information packet with packet code 00h. The LUNTRN bit is set as appropriate during a connect for asynchronous event notification.

The queue nexus (QNexus) bit when one indicates that the nexus is of the form H_C_L_Q. When the queue nexus bit is zero, the nexus formed will be of the form H_C_L, or H_C_R. The target controller shall use this bit, when 1b, as a valid field indicator for the HOQ, OrdSim, and Queue Tag fields. This value shall remain unchanged in all information packets for the nexus. If tagged queuing is implemented, support of this bit set to one shall be implemented.

A HOQ bit of one specifies that the I/O process is to be placed at the head of the queued I/O process queue for the initiating controller. This field is to be interpreted only when QNexus is 1b. This value shall remain unchanged in all information packets for the nexus. If the QNexus bit is supported, this bit shall be supported.

A OrdSim bit of one specifies ordered queue processing in the target controller. A OrdSim bit of zero specifies that a simple tagged I/O process is requested in the target controller. This field is to be interpreted only when QNexus is 1b. This value shall remain unchanged in all information packets for the nexus. If the QNexus bit is supported, this bit shall be supported.

(Parallel Interfaces.) A disconnect privilege (DiscPriv) bit of one specifies that the target controller is allowed to disconnect. A DiscPriv bit of zero specifies that the target controller shall not disconnect. When sent by a target controller during a reconnection, the value of this bit shall be the same as the most recent value received for this I/O process. (Serial Interfaces.) For serial bus implementations, the value of the disconnect privilege bit shall always be set to 1b.

The pad byte count field indicates the number of pad bytes added at the end of the interface logical elements to construct an information packet which is a multiple of 4 bytes in length.

A multiple path (MltPath) bit of one specifies that the initiating controller considers the path capable of multiple path operation. When the value is zero, the initiating controller considers the path to be in single path mode. The setting has no effect on any other active or queued I/O process in the target controller. This value shall remain unchanged in all information packets for the nexus. The MltPath bit indicates the initiating controller state of the path. The target controller shall confirm that its state is the same. If the target controller state is different, it shall not process the information packet and shall indicate the error by sending an information packet with the INVALID INFORMATION PACKET message. Accepting this bit set to one is optional.

A suspend multiple path (SuspMpth) bit of one specifies that the initiating controller requires the target controller to stop multiple path operation for an I/O process. The effect is to form an H_I_T_C nexus from the H_C nexus. The SuspMpth bit is interpreted for each nexus. The setting has no effect on any other active or queued I/O process in the target controller. If the MltPath bit is set to 0b, the SuspMpth bit is ignored and the target controller shall operate in single path mode for the I/O process. If the MltPath bit is set to 1b and the SuspMpth bit is set to 1b, the target controller shall operate in single path mode for the I/O process. If the MltPath bit is set to 1b and the SuspMpth bit is set to 0b, the target controller may operate in single path mode or multiple path mode for the I/O process. Accepting this bit set to one is optional.

A enable supervisor command (EnbSpvr) bit of one specifies that the initiating controller has granted the target controller the privilege of executing any valid supervisor command received during the I/O process in which the enable supervisor command bit is set to 1b If the bit is set to zero and a supervisor command is present in the I/O process, the target controller shall terminate the I/O process with contingent allegiance and prepare sense data. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to SUPERVISOR COMMANDS NOT ENABLED. (Section 7 update required. GRS) The enable supervisor command bit setting is interpreted for each nexus. The setting has no effect on any other active or queued I/O process in the target controller. Accepting this bit set to one is optional.

The next eight fields identify the path on which the connect was made for the nexus. The LUNTRN bit identifies whether the nexus is directed toward a logical unit or a target routine. These fields shall not change in any information packet of any type transferred for the nexus once established as specified below. These fields along with the LUNTRN Valid, LUNTRN QNexus, HOQ, and OrdSim bits identify the nexus to be reestablished in any later connection on any path for the nexus, and also in the target controller, when operating in a multiple path environment.

The initiating controller port number identifies the internal port number assigned by the initiating controller to the port where the connect was made using an information packet packet type code of 00h. This value shall remain unchanged in all information packets for the nexus.

The original initiator SCFI address identifies the external name of the SCFI device on the initiating controller where the connect was made using an information packet packet type code of 00h. This value shall remain unchanged in all information packets for the nexus.

The original target SCFI address identifies the SCFI device, from the initiating controller's perspective on the selected SCFI bus, information where the connect was made using an information packet packet type code of 00h. This value shall remain unchanged in all information packets for the nexus.

The target port number identifies the physical port number assigned by the target controller to the target SCFI address. This field shall be zero when the packet type code is 00h for the first I/O process between two logical elements. The target controller supplies its unique port number value in the

response information packet. Once the initiating controller receives an information packet for the nexus that contains the target port number, all later information packets sent with packet type code 00h shall have the target port number placed in this field. If the disconnect privilege is not granted to a target controller, the target port number may not be transferred until the information packet with a packet type of 01h is transferred. The target controller may receive multiple information packets during the connect with a target port number value of FFh. If the target controller receives an information packet with packet type code 00h, other than as specified above, and the target port number does not match the internally assigned port number, the target controller shall not process the information packet and shall indicate the error by sending an information packet with the INVALID INFORMATION PACKET message.

The logical unit number target routine number (LUNTRNID) field specifies a logical unit number if the LUNTRN bit is zero. The LUNTRNID field specifies a target routine number if the LUNTRN bit is one. The response to an invalid value in the LUNTRNID field is described in 6.7.3. The target controller shall ignore this field when the LUNTRN Valid bit is zero. This value shall remain unchanged in all information packets for the nexus.

The queue tag field has meaning only when the QNexus bit is set to 1. If the QNexus bit is set to zero, this field shall be set to 00h. If the target controller detects any value, other than 00h, it shall indicate the error by sending an information packet with an INVALID INFORMATION PACKET message. The target controller shall ignore this field when the LUNTRN Valid bit is zero. This value shall remain unchanged in all information packets for the nexus.

Table 6-11. Interface Control Suffix Fields (Mandatory)

Bit / Byte	7	6	5	4	3	2	1	0
none or -n to -1	Zero to 3 Pad Bytes							

Each information packet shall be rounded up to the nearest multiple of 4 bytes by adding from zero to three pad bytes. The value of -n in Table 6-11 may be -1, -2, or -3 if any pad bytes are required. Including more than 3 pad bytes shall indicate the error by sending an information packet with the INVALID INFORMATION PACKET message.

6.7.1.2 Information Packet Serial Encapsulation

Table 6-12. Interface Control Prefix Fields (Serial)

Bit / Byte	7	6	5	4	3	2	1	0
-16 to -1	Frame Header Field							
0 - 15	Interface Control Prefix Fields (Mandatory)							

The format of the serial interface control prefix fields, Table 6-12, is the frame header fields followed by the mandatory interface control prefix fields. The packet length field does not include the frame

header field length. Only the frame header fields are described below. The negative displacement for the frame header keeps all common fields at the same offset for both interfaces.

Table 6-13 shows the Fiber Channel frame header field.

(Add Fiber Channel frame header fields. Changing monthly. GRS)

Table 6-13. Frame Header Field (Serial)

Bit / Byte	7	6	5	4	3	2	1	0
-16 to -13	(MSB) Destination ID (D_ID) (LSB)							
-12 to -9	(MSB) Source ID (S_ID) (LSB)							
-8 to -7	(MSB) Data Structure Type (LSB)							
-6	Reserved		FC Fabric			OrgRsp	LnkCtl	Reserved
-5	Reserved	LstSeq	LstXch	DevHdr	Reserved			Pad Byte Count
-4	Frame Command							
-3	Exchange ID (XID)							
-2 to -1	(MSB) Sequence Count (LSB)							

Each information packet shall be rounded up to the nearest multiple of 4 bytes by adding from zero to three pad bytes. The value of -n in Table 6-14 may be -1, -2, or -3 if any pad bytes are required. Including more than 3 pad bytes shall constitute an invalid information packet.

The serial interface control suffix fields consists of the mandatory interface control suffix fields plus the CRC field. The CRC field is a 4-byte CRC value calculated according to algorithms provided in the Fiber Channel Standard.

Table 6-14. Interface Control Suffix Fields (Serial)

Bit / Byte	7	6	5	4	3	2	1	0
none or -n to -1	Zero to 3 Pad Bytes							
0 - 3	CRC							

6.7.2 Interface Logical Elements (Mandatory)

The interface logical elements are messages, command descriptor blocks, command parameter data, command response data, logical blocks, and status as described in Sections 6 though 17 of this standard. A self-describing set of fields surrounds the interface logical elements to permit the information packet to be correctly identified and checked at the receiving SCFI port.

A self-describing interface logical element is composed of four fields: the element length, the element type, a reserved field, and the logical element bytes. See Table 6-15.

Table 6-15. Interface Logical Element

Bit / Byte	7	6	5	4	3	2	1	0
0 - 1	(MSB) Element Length (LSB)							
2	Element Type							
3	Reserved							
4 - n	Element Bytes							

The element length is a two-byte binary field. The value range is from 4 to 65535. If the element length value is 4, the logical element contains only an element type field and is to be ignored by the receiver of the information packet. The value is 4 plus the number of element bytes included up to the maximum value. The maximum length for any one interface logical element must be limited bu the physical bus requirements for the maximum packet length.

The element type is a one-byte binary field. The valid values are defined in Table 6-16.

Table 6-16. ILE Element Type Codes

Element Type Value	Interface Logical Element
0	Message
1	Command Descriptor Block
2	Command Parameter Data
3	Command Response Data
4	Logical Block Data
5	Status
6	Autosense
7 - 255	Reserved

The reserved field is a one-byte field. The sender shall set the field to zero. If the receiver of an information packet detects a value other than zero it shall not process the logical element or any logical elements which follow in the same information packet.

The logical element bytes field is a 0 to 65531 byte variable length field composed of the information about the element type declared in the element type field. The maximum length which may be used for this field may vary by the physical interface implemented.

An interface logical element with an element length of 4 shall be ignored if all fields are correctly coded. That is, the interface logical element is taking up space, but it does not carry any data useful to the I/O process. This shall not be considered an error. One use could be to fill out information packets to maximum length for transmission, except for pad bytes.

6.7.2.1 Message Interface Logical Element

The message interface logical element includes one or more complete messages and applicable to the physical bus and protocol implemented. A multiple byte message shall not be split into bursts.

The element bytes are filled in with the message bytes in the order which they would be transferred on a single byte wide bus. See sections 6.4 and 6.5 for the message descriptions and their use.

6.7.2.2 Command Descriptor Block Interface Logical Element

The command descriptor block interface logical element is composed of the set of variable length CDBs for commands defined in Sections 7 through 17 of this standard. Command parameter data is transferred in its own interface logical element. A command descriptor block shall not be split into bursts.

The element bytes are filled in with the CDB bytes in the order which they would be transferred on a single byte wide bus.

6.7.2.3 Command Parameter Data Interface Logical Element

The command parameter data interface logical element is a burst of data transferred for a command which is sent as an addendum to the command descriptor block. The CDB identifies the total length of the command parameter data in the parameter length fields, etc.

The element bytes are filled in with the command parameter data bytes in the order which they would be transferred on a single byte wide bus.

(Offset field??? GRS)

6.7.2.4 Command Response Data Interface Logical Element

The command response data interface logical element is a burst of data transferred from a target controller to an initiating controller for a command which is not taken from the logical blocks of the LUN. Sense data and MODE SENSE pages are examples of command response data. The CDB identifies the maximum length of the command response data in the allocation length fields, etc.

The element bytes are filled in with the command response data bytes in the order which they would be transferred on a single byte wide bus.

(Offset field??? GRS)

6.7.2.5 Logical Block Data Interface Logical Element

The logical block interface logical element consists of a burst of data from the logical blocks requested for transfer between an initiating controller and a target controller.

The element bytes are filled in with the bytes from a burst of data for one or more logical blocks in the order which they would be transferred on a single byte wide bus. The element length may vary.

(Offset field??? GRS)

6.7.2.6 Status Interface Logical Element

The status interface logical element is the status as defined in Section 6.8 of this standard. Status shall not be split into bursts.

The element bytes are filled in with status in the order which they would be transferred on a single byte wide bus.

6.7.2.7 Autosense Interface Logical Element (Optional)

The autosense interface logical element is the sense data for an I/O process which terminates with CA or ECA. Autosense shall not be split into bursts.

The element bytes are filled in with sense data in the order which they would be transferred on a single byte wide bus.

6.7.3 Information Packet Layout (Mandatory)

6.7.3.1 Serial Bus

The serial bus information packet layout consists of the following minimum structures: one interface control prefix (serial); one interface logical element immediately following the interface control prefix; and one interface control suffix (serial). See Table 6-17.

The remainder of the information packet, following the one mandatory interface logical element is made up of zero or more interface logical elements, as required to perform the operations for the I/O process.

Table 6-17. Information Packet Structure (Serial)

Bit / Byte	7	6	5	4	3	2	1	0
0 - n	Interface Control Prefix Fields (Serial)							
n+1 to p	One to 255 Interface Logical Elements							
p+0 to p+3	Interface Control Suffix Fields (Serial)							

6.7.3.2 Parallel Bus (Optional)

The parallel bus information packet layout consists of the following minimum structures: one interface control prefix (Parallel); one interface logical element immediately following the interface control prefix; and one interface control suffix (Parallel). See Table 6-18 on page 6-39.

The remainder of the information packet, following the one mandatory interface logical element is made up of zero or more interface logical elements, as required to perform the operations for the I/O process. Certain operations are limited to only one interface logical element in the information packet (e.g., an ABORT message).

Each information packet is completely self-identifying for compatibility with Fiber Channel operations and to permit multiple path operations.

Table 6-18. Information Packet Structure (Parallel)

Bit / Byte	7	6	5	4	3	2	1	0
0 - n	Interface Control Prefix Fields (Mandatory)							
n+1 to p	One to 255 Interface Logical Elements							
p+1 to q	Interface Control Suffix Fields (Mandatory)							

6.8 Messages (Mandatory)

Messages allow communication of information between an initiating controller and target controller not contained in the other information packet contents (e.g., commands). The element bytes of a message interface logical element may be one, two, or multiple bytes in length.

Zero or more messages may be transferred along with other interface logical elements in a single information packet. A multiple-byte message shall not be split between message interface logical elements.

The initiating controller and target controller controller are required to control the content of information packets when it sends a message interface logical element in an information packet for certain messages as identified in Table 6-19 on page 6-40. The "Single Mesasge ILE in IP" column, when "YES" indicates the an information packet containing the message ILE shall be the only ILE in the information packet; no other ILE of any type shall be present in the information packet. The assigned ranges to messages and their lengths is defined in Table 6-20 on page 6-41.

All SCFI devices shall implement the mandatory messages tabulated in the "Init" column of Table 6-19 on page 6-40 for information packet operations. All SCFI devices shall implement the mandatory messages tabulated in the "Targ" column of Table 6-19 on page 6-40 for information packet operations. This is modified by messages unique to the parallel interface. See the "-" prefix on the message name for these messages.

One-byte, Two-byte, and extended message formats are defined. The first byte of the message determines the format as follows:

One-byte messages consist of a single byte where the value of the byte determines which function is to be performed as defined in Table 6-19.

Two byte messages consist of a message code followed by a parameter byte. The message code determines which function is to be performed using the value in the parameter byte.

Table 6-19 (Page 1 of 2). Message Codes

Code	Support Initiator	Support Target	Message Name	Direction		Single Message ILE in IP
06h	O	M	ABORT		Out	Yes
0Dh	O	O	ABORT TAG (Note 1)		Out	Yes
0Ch	O	M	BUS DEVICE RESET		Out	Yes
0Eh	M	M	CLEAR QUEUE		Out	Yes
00h	M	M	COMMAND COMPLETE	In		No
***	M	M	EXTENDED MESSAGE REJECT	In	Out	Yes
0Fh	M	O	INITIATE RECOVERY	In		No
0Fh	M	O	INITIATE RECOVERY (Note 2)		Out	Yes
***	M	M	-INVALID BUS PHASE DETECTED	In	Out	Yes
***	M	M	INVALID INFORMATION PACKET	In	Out	Yes
0Ah	M	M	LINKED COMMAND COMPLETE	In		No
0Bh	M	M	LINKED COMMAND COMPLETE (with Flag)	In		No
***	M	M	MODIFY DATA POSITION	In		No
***	M	M	-PARITY ERROR	In	Out	Yes
10h	M	O	RELEASE RECOVERY		Out	Yes
***	M	M	RESEND PREVIOUS INFORMATION PACKET	In	Out	Yes
***	M	M	-SYNCHRONOUS DATA TRANSFER REQUEST	In	Out	Yes
***	M	M	SYNCHRONOUS PACKET TRANSFER REQUEST	In	Out	Yes
11h	M	M	TERMINATE I/O PROCESS		Out	Yes
??h	M	M	TRANSFER READY	In		Yes
02h- 1Fh			Reserved If Not Listed Above			
20h- 2Fh			Reserved for Two-Byte Messages			
30h- 7Fh			Reserved			
80h- FFh			Reserved			

Table 6-19 (Page 2 of 2). Message Codes

Key: M = Mandatory support, O = Optional support.
Targ = Target Controller
Init = Initiating Controller
ILE = Interface Logical Element
IP = Information Packet
In = Target to initiator, Out = Initiator to target.
Yes = IP has Single Message ILE
No = May be other ILEs in IP
- = Parallel SCFI implementations only.
*** = Extended message Code Value (see Tables Table 6-22 on page 6-43)
80h+ = Codes 80h through FFh are used for IDENTIFY messages.

NOTES:

(1) The ABORT TAG message is required if tagged queing is implemented.

(2) Outbound INITIATE RECOVERY messages are only valid during the extended contingent allegiance protocol.

A value of 01h in the first byte of a message indicates the beginning of a multiple-byte extended message. The minimum number of bytes in an extended message is three. The maximum number of bytes in an extended message is 258 bytes. The extended message codes and the extended message format are shown in Table 6-22 on page 6-43 and Table 6-21 on page 6-42, respectively.

Table 6-20. Message Element Format Codes

Format Code	**Message Element Byte Format**
00h	One-Byte Message (COMMAND COMPLETE)
01h	Extended Messages
02h - 1Fh	One-Byte Messages
20h - 2Fh	Two-Byte Messages
30h - 7Fh	Reserved
80h - FFh	Reserved Messages

The extended message length specifies the length in bytes of the extended message code plus the extended message arguments to follow. Therefore, the total length of the message is equal to the extended message length plus two. A value of zero for the extended message length indicates 256 bytes follow.

Table 6-21. Extended Message Element Format		
Byte	**Value**	**Description**
0	01h	Extended Message Format Code
1	n	Extended Message length
2	-	Extended Message Code (Table 6-22 on page 6-43)
3 - n+1	-	Extended Message Parameters

The extended message codes are listed in Table 6-22 on page 6-43. The extended message arguments are specified with each extended message description. For extended messages, the message code is followed by a variable length set of zero or more parameter bytes. The function to be performed is determined by the value in the Extended Message Code field.

Table 6-22. Extended Message Codes	
Code Value	**Name**
	(Ordered by Extended Message Name)
04h	EXTENDED MESSAGE REJECT
02h	INVALID BUS PHASE DETECTED (Parallel Only)
06h	INVALID INFORMATION PACKET
00h	MODIFY DATA POSITION
03h	PARITY ERROR (Parallel Only)
05h	RESEND PREVIOUS INFORMATION PACKET
01h	SYNCHRONOUS DATA TRANSFER REQUEST (Parallel Only)
07h	SYNCHRONOUS PACKET TRANSFER REQUEST
08h - 7Fh	Reserved
80h - FFh	Reserved
	(Ordered by Extended Message Code)
00h	MODIFY DATA POSITION
01h	SYNCHRONOUS DATA TRANSFER REQUEST (Parallel Only)
02h	INVALID BUS PHASE DETECTED (Parallel Only)
03h	PARITY ERROR (Parallel Only)
04h	EXTENDED MESSAGE REJECT
05h	RESEND PREVIOUS INFORMATION PACKET
06h	INVALID INFORMATION PACKET
07h	SYNCHRONOUS PACKET TRANSFER REQUEST
08h - 7Fh	Reserved
80h - FFh	Reserved

SCFI messages are defined in this section. The requirements for implementation are given in Table 6-19 on page 6-40. Additional messages shall be implemented when certain SCFI functions are implemented. These messages are identified in Table 6-19 on page 6-40 for information packets and in the message descriptions.

IMPLEMENTORS NOTES: The BUS DEVICE RESET, CLEAR QUEUE, ABORT, and ABORT TAG messages provide a means to clear one or more I/O processes before normal termination. The BUS DEVICE RESET message clears all I/O processes for all initiating controllers on all logical units and all target controller routines of the target controller. The CLEAR QUEUE message clears all I/O processes for all initiating controllers on the specified logical unit or target controller routine of the target controller. The ABORT message clears all I/O processes for the selecting initiating controller on the specified logical unit or target controller routine of the target controller. The ABORT TAG message clears an active I/O process or a queued I/O process with a matching queue tag value from the same initiating controller.

6.8.1 ABORT

The ABORT message is sent from the initiating controller to the target controller to clear the active I/O process plus any queued I/O process for the H_C_x nexus. Pending data, status, and queued I/O processes for any other H_C_X nexus shall not be cleared.

The information packet containing an ABORT message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the ABORT message.

If only an H_C nexus has been established, no status or message shall be sent for the current I/O process and no other I/O process shall be affected.

It is not an error to issue this message to an H_C_x nexus that does not have an active or queued I/O process.

Previously established conditions, including MODE SELECT parameters, reservations, and extended contingent allegiance shall not be changed by the ABORT message.

6.8.2 ABORT TAG

The ABORT TAG message shall be implemented if tagged queuing is implemented.

The information packet containing an ABORT TAG message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the ABORT TAG message.

The target controller shall clear the I/O process identified. If the target controller has already started execution of the I/O process, the execution shall be halted. The medium contents may have been modified before the information packet was processed. In either case, any pending status or data for the I/O process shall be cleared and no status or ending message shall be sent to the initiating controller.

Pending status, data, and commands for other active or queued I/O processes shall not be affected. Execution of other I/O processes queued for the H_C_x nexus shall not be aborted. Previously established conditions, including MODE SELECT parameters, reservations, and extended contingent allegiance shall not be changed by the ABORT TAG message.

6.8.3 BUS DEVICE RESET

The BUS DEVICE RESET message is sent from an initiating controller to direct a target controller to clear all I/O processes on that SCFI device. This message forces a reset condition to the selected SCFI device. The target controller shall create a unit attention condition for all initiating controllers (see 6.13).

The information packet containing an BUS DEVICE RESET message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the BUS DEVICE RESET message.

6.8.4 CLEAR QUEUE

The information packet containing an CLEAR QUEUE message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the CLEAR QUEUE message.

The target controller shall perform an action equivalent to receiving a series of ABORT messages from each initiating controller. All I/O processes, from all initiating controllers, in the queue for the specified logical unit or target routine shall be cleared from the queue. All active I/O processes shall be terminated. The medium may have been altered by partially executed commands.

All pending status and data for that logical unit or target routine for all initiating controllers shall be cleared. No status or message shall be sent for any of the I/O processes. A unit attention condition shall be generated for all other initiating controllers with I/O processes that either were active or were queued for that logical unit or target routine. When reporting the unit attention condition the additional sense code shall be set to COMMANDS CLEARED BY ANOTHER INITIATOR. Previously established conditions, including MODE SELECT parameters, reservations, and extended contingent allegiance shall not be changed by the CLEAR QUEUE message.

6.8.5 COMMAND COMPLETE

The COMMAND COMPLETE message is sent from a target controller to an initiating controller to indicate that the execution of an I/O process has completed and that valid status has been sent to the initiating controller. The I/O process may have completed successfully or unsuccessfully as indicated in the status.

The target controller shall consider the message transmission to be successful when an acknowledgement information packet is received for the information packet containing the COMMAND COMPLETE MESSAGE.

6.8.6 EXTENDED MESSAGE REJECT

The EXTENDED MESSAGE REJECT message (Table 6-23 on page 6-46) is sent from either the initiating controller or target controller to indicate that a portion of a message in a message interface logical element in the last information packet was inappropriate or has not been implemented.

The content of the extended message identifies the message group and the message byte within the group. The message also indicates the reason for the rejection. This provides a logical interlock so that the receiver of the message can determine which message byte is in error.

The information packet containing an EXTENDED MESSAGE REJECT message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the EXTENDED MESSAGE REJECT message.

Table 6-23. EXTENDED MESSAGE REJECT Message Format		
Byte	**Value**	**Description**
0	01h	Extended Message Format Code
1	04h	Extended Message length
2	04h	Extended Message Code
3	x	ILE Position
4	x	Position in ILE
5	x	Reject Reason Code

The ILE Position is the ordinal position of the message interface logical element within the information packet containing the error. The ILE Position is a value between 1 to 255. There may be more than one message interface logical element in one information packet.

The Position in ILE is the byte position of the byte in error within the element bytes of the message interface logical element in error.

The Reject Reason Code identifies the reason for sending the message. A code of 00h indicates that the message code in a one-byte, two-byte, or extended message is not implemented. A code of 01h indicates that the message is inappropriate. A code of 02h indicates that the length byte in an extended message is invalid for the message code. A code of 03h indicates that a parameter byte within two-byte and extended messages is invalid. Codes 04h-FFh are reserved.

6.8.7 INITIATE RECOVERY

The INITIATE RECOVERY message shall be implemented if extended contingent allegiance is implemented.

A target controller that supports extended contingent allegiance shall inform the initiating controller it is entering this condition by sending an INITIATE RECOVERY message immediately following a CHECK CONDITION or I/O PROCESS TERMINATED status. The extended contingent allegiance condition remains in effect until terminated as described in 6.12.

If an asynchronous event occurs, the target controller may enter an extended contingent allegiance condition by becoming a temporary initiating controller and sending the INITIATE RECOVERY message followed by the SEND command that is used to perform asynchronous event notification (see 6.6). The successful transmission of this message establishes the extended contingent allegiance condition in the initiating controller which remains in effect until terminated as described in 6.12.

If the target controller notifies multiple initiating controllers of the asynchronous event, it may include the INITIATE RECOVERY message in each notification, depending on the target controllers ability to manage recovery by multiple initiating controllers (e.g., a buffered mode setting of 2 for sequential access devices).

When this message is sent outbound, the initiating controller is indicating that it is entering extended contingent allegiance with the target controller as a result of a asynchronous event notification that contained a message interface logical element with the INITIATE RECOVERY message. The information packet containing an INITIATE RECOVERY message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the INITIATE RECOVERY message.

6.8.8 INVALID BUS PHASE DETECTED (Parallel)

Table 6-24. INVALID BUS PHASE DETECTED Message Format

Byte	Value	Description							
0	01h	Extended Message Format Code							
1	02h	Estended Message length							
2	02h	Extended Message Code							
3	x	7 0b	6 0b	5 0b	4 0b	3 0b	2 Msg	1 C/D	0 I/O

The INVALID BUS PHASE DETECTED message (-- Table 'S6IPBD' unknown --) is sent from either logical element to the other to inform it that an illegal or reserved bus phase was detected on a parallel interface.

The information packet containing an INVALID BUS PHASE DETECTED message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the INVALID BUS PHASE DETECTED message.

Table 6-24 defines the format for the INVALID BUS PHASE DETECTED message.

The Msg bit shall contain 1 if the phase where the phase error was detected has the MSG signal set to true. Otherwise, the Msg bit shall be set to zero.

The C/D bit shall contain 1 if the phase where the phase error was detected has the C/D signal set to true. Otherwise, the C/D bit shall be set to zero.

The I/O bit shall contain 1 if the phase where the phase error was detected has the I/O signal set to true. Otherwise, the I/O bit shall be set to zero.

The three bits identify the phase code which was detected in error.

6.8.9 INVALID INFORMATION PACKET

The INVALID INFORMATION PACKET message (Table 6-25 on page 6-48) is sent from either the initiating controller or target controller in an information packet to indicate that all or a portion of an information packet transferred was invalid. The format of the information packet does not conform to the rules for information packet construction. Errors in the content of the element bytes of interface logical elements is not pointed to using this message; see sense data.

The information packet containing an INVALID INFORMATION PACKET message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the INVALID INFORMATION PACKET message.

Table 6-25. INVALID INFORMATION PACKET Message Format		
Byte	**Value**	**Description**
0	01h	Extended Message Format Code
1	05h	Extended Message length
2	06h	Extended Message Code
3	x	ILE Position
4	x	Reason Code
5 - 6	xxxxh	(MSB) Relative Position (LSB)

The ILE Position is the ordinal position of the interface logical element group within the information packet containing the error. The ILE Position is a value between 1 to 255. There may be more than one message interface logical element in each information packet. A value of 0 in ILE Position indicates that the information control prefix or suffix fields are in error.

The Reason Code identifies the reason for sending the message. A code of 00h indicates that the interface logical element type in a descriptor is invalid. A code of 01h indicates that the interface logical element length field is invalid. Codes 02h-FFh are reserved.

The Relative Position identifies the byte position of the field or bit in error. The exact bit position is not provided. The value gives the relative position of the byte to the beginning of the ILE when the ILE Number value is 1 to 255. The value gives the relative position of the byte to the beginning of the information packet when the ILE Number value is zero.

6.8.10 LINKED COMMAND COMPLETE

The LINKED COMMAND COMPLETE message is sent from a target controller to an initiating controller to indicate that the execution of a linked command has completed and that status has been sent. The next command may come from the active I/O process queue as part on an information packet already transferred or it comes from the initiating controller in a new information packet.

6.8.11 LINKED COMMAND COMPLETE (WITH FLAG)

The LINKED COMMAND COMPLETE (WITH FLAG) message is sent from a target controller to an initiating controller to indicate that the execution of a linked command (with the flag bit set to one) has completed and that status has been sent. The next command may come from the active I/O process queue as part on an information packet already transferred or it comes from the initiating controller in a new information packet.

6.8.12 MODIFY DATA POSITION

Table 6-26. MODIFY DATA POSITION Message Format		
Byte	**Value**	**Description**
0	01h	Extended Message Format Code
1	05h	Extended Message length
2	00h	Extended Message Code
3 - 6	xxxxxxxxh	(MSB) Change in Relative Position (LSB)

The MODIFY DATA POSITION message (Table 6-26) requests that the signed argument be added (two's complement) to the value of the current logical block position in the initiating controller. This message shall control or adjust the logical block data transfer position for a current I/O process. If the change in relative position added to the current initiating controller results in a value less than zero or greater than the total length for the command, the initiating controller shall ....

6.8.13 PARITY ERROR (Parallel)

Table 6-27. PARITY ERROR Message Format									
Byte	**Value**	**Description**							
0	01h	Extended Message Format Code							
1	02h	Estended Message length							
2	03h	Extended Message Code							
3	x	7 0b	6 0b	5 0b	4 0b	3 0b	2 Msg	1 C/D	0 I/O

The PARITY ERROR message is sent from either logical element to the other to inform it that a parity error was detected during an information packet transfer on a parallel interface.

The information packet containing an ABORT TAG message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the ABORT TAG message. Table 6-27. defines the format for the parity error message.

The Msg bit shall contain 1 if the invalid phase detected has the MSG signal set to true. Otherwise, the Msg bit shall be set to zero.

The C/D bit shall contain 1 if the invalid phase detected has the C/D signal set to true. Otherwise, the C/D bit shall be set to zero.

The I/O bit shall contain 1 if the invalid phase detected has the I/O signal set to true. Otherwise, the I/O bit shall be set to zero.

The three bits identify the phase where the parity error was detected.

6.8.14 RELEASE RECOVERY

The RELEASE RECOVERY message shall be implemented if extended contingent allegiance is implemented.

The RELEASE RECOVERY message is sent from an initiating controller to a target controller to terminate an extended contingent allegiance condition previously established by an INITIATE RECOVERY message. The extended contingent allegiance condition ends on successful processing of the RELEASE RECOVERY message.

The information packet containing an RELEASE RECOVERY message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the RELEASE_RECOVERY message.

If a RELEASE RECOVERY message is received by a target controller that implements extended contingent allegiance when no extended contingent allegiance condition is active, the message shall not be rejected. The message shall be ignored.

6.8.15 RESEND PREVIOUS INFORMATION PACKET

Table 6-28. RESEND PREVIOUS INFORMATION PACKET Message Format

Byte	Value	Description
0	01h	Extended Message Format Code
1	02h	Extended Message length
2	05h	Extended Message Code
3	xxh	Previous Packet Number

The RESEND PREVIOUS INFORMATION PACKET message (Table 6-28) is sent by an initiating controller or a target controller to indicate that a previous information packet must be resent. The synchronous packet transfer request negotiation has produced an agreement for the maximum number of packets to be retained by both the initiating controller and the target controller.

A value of zero in the previous packet number field causes the device to resend the last information packet transferred. Values of 1 through 255 represent the 2nd to 256th most recent packets transmitted. It shall be an error for a device to transmit a value in this field which is larger than the limit negotiated using the SYNCHRONOUS PACKET TRANSFER REQUEST message (6.8.17).

The information packet containing an RELEASE RECOVERY message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the RELEASE RECOVERY message.

6.8.16 SYNCHRONOUS DATA TRANSFER REQUEST (Parallel)

Table 6-29. SYNCHRONOUS DATA TRANSFER REQUEST Message Format		
Byte	**Value**	**Description**
0	01h	Extended Message Format Code
1	03h	Extended Message length
2	01h	Extended Message Code
3	xxh	Transfer Period (m times 4 nanoseconds)
4	xxh	REQ/ACK Offset

(Rework like SPTR. GRS)

A SYNCHRONOUS DATA TRANSFER REQUEST (SDTR) message (Table 6-29) exchange shall be initiated by an SCFI device whenever a previously-arranged data transfer agreement may have become invalid. The agreement becomes invalid after any condition which may leave the data transfer agreement in an indeterminate state such as:

(1) after a hard reset condition

(2) after a BUS DEVICE RESET message and

(3) after a power cycle.

The information packet containing an SYNCHRONOUS DATA TRANSFER REQUEST message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the SYNCHRONOUS DATA TRANSFER REQUEST message.

The SDTR message exchange establishes the permissible transfer periods and the REQ/ACK offsets for all logical units and target routines on the two devices. This agreement only applies to the information transfer phases.

The transfer period is the minimum time allowed between leading edges of successive REQ pulses and of successive ACK pulses to meet the device requirements for successful reception of data.

The REQ/ACK offset is the maximum number of REQ pulses allowed to be outstanding before the leading edge of its corresponding ACK pulse is received at the target controller. This value is chosen to prevent overflow conditions in the device's reception buffer and offset counter. A REQ/ACK offset value of zero shall indicate asynchronous data transfer mode; a value of FFh shall indicate unlimited REQ/ACK offset.

The originating device (the device that sends the first of the pair of SDTR messages) sets its values according to the rules above to permit it to receive data successfully. If the responding device can also receive data successfully with these values (or smaller transfer periods or larger REQ/ACK offsets or both), it returns the same values in its SDTR message. If it requires a larger transfer period, a smaller REQ/ACK offset, or both to receive data successfully, it substitutes values in its SDTR message as required, returning unchanged any value not required to be changed. Each device when transmitting data shall respect the limits set by the other's SDTR message, but it is permitted to transfer data with larger transfer periods, smaller REQ/ACK offsets, or both than specified in the other's SDTR message. The successful completion of an exchange of SDTR messages implies an agreement, retainec by both the initiating controller and target controller, as follows:

Responding Device SDTR response	Implied Agreement
(1) Non-zero REQ/ACK offset	Each device transmits data with a transfer period equal to or greater than and a REQ/ACK offset equal to or less than the values received in the other device's SDTR message.
(2) REQ/ACK offset equal to zero	Asynchronous transfer
(3) Unexpected Disconnect	Asynchronous transfer
(4) Parity Error	Asynchronous transfer

If a logical element recognizes that negotiation is required, it asserts it sends a SDTR message to begin the negotiating process. The other logical element shall respond with the proper SDTR message. If an abno rmal condition prevents the target controller from returning an appropriate response, both devices shall go to asynchronous data transfer mode for data transfers between the two devices.

Following the second logical element resoponse of (1) above, the implied agreement for synchronous operation shall be considered to be negated by both the initiating controller and the target controller if the first logical element sends as the first message a MESSAGE PARITY ERROR (thrid message of sequence). In this case, both devices shall go to asynchronous data transfer mode for data transfers between the two devices. For the PARITY ERROR case, the implied agreement shall be reinstated if a re-transmittal of the second of a pair of messages is successful. After a vendor-specific number of retry attempts (greater than zero), if the second logical element receives a PARITY ERROR message, it shall terminate the retry activity. The initiating controller shall accept such action as aborting the negotiation, and both devices shall go to asynchronous data transfer mode for data transfers between the two devices. Following an initiating controller's responding SDTR message, an impli ed agreement for synchronous operation shall not be considered to exist until the target controller leaves the MESSAGE OUT phase, indicating that the target controller has accepted the negotiation.

The implied synchronous agreement shall remain in effect until a BUS DEVICE RESET message is received, until a hard reset condition occurs, or until one of the two SCFI devices elects to modify the agreement. The default data transfer mode is asynchronous data transfer mode. The default data transfer mode is entered at power on, after a BUS DEVICE RESET message, or after a hard reset condition.

6.8.17 SYNCHRONOUS PACKET TRANSFER REQUEST

Table 6-30. SYNCHRONOUS PACKET TRANSFER REQUEST Message Format

Byte	Value	Description
0	01h	Extended Message Format Code
1	03h	Extended Message length
2	07h	Extended Message Code
3	00h	Reserved
4	xxh	Packet Offset Count

A SYNCHRONOUS PACKET TRANSFER REQUEST (SPTR) message (Table 6-30) exchange shall be initiated by an SCFI device whenever a previously arranged packet transfer agreement may have become

invalid. Each SCFI device shall be able to retain at minimum one information packet per I/O process for retransmission (i.e., the last information packet successfully transferred in each direction).

The information packet containing an SYNCHRONOUS PACKET TRANSFER REQUEST message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the SYNCHRONOUS PACKET TRANSFER REQUEST message.

In the absence of an explicit agreement for value larger than one, the implied agreement shall be 1 for both initiating controllers and target controllers. At the end of an exchange, both the initiating controller and target controller agree to maintain, for retransmitting only, a minimum number of information packets (greater than or equal to one).

The packet offset count represents n+1 information packets that the initiating controller or target controller declares that it can retain at maximum or has agreed upon with another SCFI device. A maximum of 256 information packets may be retained per I/O process at the end of an SPTR negotiation. For initiating controllers which implement the MODIFY DATA POSITION message, there may be an additional requirement to retain additional information packets.

The synchronous packet transfer agreement becomes invalid after any condition which may leave the packet transfer agreement in an indeterminate state such as:

(1) after a hard reset condition

(2) after a BUS DEVICE RESET message and

(3) after a power cycle.

In addition, an SCFI device may initiate an SPTR message exchange whenever it is appropriate to negotiate a new packet transfer agreement.

The SPTR message exchange establishes the minimum number of information packets that the initiating controller and target controller shall maintain to permit retransmitting in case of physical or logical errors in information packets. This agreement only applies to information packets. See the RESENT PREVIOUS INFORMATION PACKET message.

The originating device (the device that sends the first of the pair of SDTR messages) sets its values according to its capabilities to retain information packets on a per I/O process basis. If the responding device can also retain information packets at this level, it returns the same values in its SPTR message. If it requires a smaller retention count, it substitutes a lower value in its SPTR message. The originating device analyzes the response and may initiate another SPTR message with the same or a still lower value. The negotiation shall end when both devices exchange the same value or the value reaches 1, whichever occurs first.

Each device shall retain the number of information packets for retransmitting, as agreed upon in the most recent SPTR exchange or one, whichever is larger. If there is any error in the negotiation process, the agreement shall be 1 after the I/O process.

6.8.18 TERMINATE I/O PROCESS

The TERMINATE I/O PROCESS message is sent from the initiating controller to the target controller to advise the target controller to terminate the current I/O process without corrupting the medium. Upon successful receipt of this message the target controller shall terminate the current I/O process as soon as possible and return I/O PROCESS TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and qualifier shall be set to I/O PROCESS TERMINATED. The TERMINATE I/O PROCESS message shall not affect pending status, data, and commands for other queued or active I/O processes. However, continued execution and status of other I/O processes queued for the H_C_x nexus or I_T_x nexus may be affected by the queue error recovery option specified in the control mode page (see 7.3.3.1).

e information packet containing an TERMINATE I/O PROCESS message shall contain only one interface logical element, a message interface logical element. The message interface logical element shall contain only one message, the TERMINATE I/O PROCESS message.

If the I/O process that is being terminated has a data transfer associated with it, the valid bit in the sense data shall be set to one and the information field shall be set as follows: (1) If the command descriptor block specifies an allocation length or parameter list length in bytes, the information field shall be set to the difference (residue) between the transfer length and the number of bytes successfully transferred. (2) If the command descriptor block specifies a transfer length field, the information field shall be set as defined in the REQUEST SENSE command (see 7.2.14).

If the I/O process being terminated has no data transfer associated with it the target controller shall set the valid bit in the sense data to zero and terminate the I/O process with I/O PROCESS TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and qualifier shall be set to I/O PROCESS TERMINATED.

When any error condition is detected while terminating an I/O process the target controller shall ignore the TERMINATE I/O PROCESS message and terminate the I/O process with the appropriate error status and sense data for the error condition.

If the target controller completes all processing for a command (i.e., all data has been read, written, or processed) and a TERMINATE I/O PROCESS message is received before the I/O process is terminated, the target controller shall ignore the TERMINATE I/O PROCESS message and terminate the I/O process in the normal manner.

If the target controller receives a TERMINATE I/O PROCESS message before the command descriptor block is transferred or the message is issued to an H_C_x nexus that does not have an active or queued I/O process, the target controller shall set the valid bit in the sense data to zero and terminate the I/O process with I/O PROCESS TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and qualifier shall be set to I/O PROCESS TERMINATED.

If the affected I/O process is in the command queue (an H_C_x nexus for untagged queuing or an H_C_L_Q nexus for tagged queuing) and has not started execution, the target controller shall record the event with the queued I/O process and wait until the command is in the active queue (started executing) then terminate the I/O process. The target controller shall terminate the I/O process with I/O PROCESS TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and qualifier shall be set to I/O PROCESS TERMINATED. The valid bit shall be set to zero.

IMPLEMENTORS NOTE: The TERMINATE I/O PROCESS message provides a means for the initiating controller to request the target controller to reduce the transfer length of the current command to the amount that has already been transferred. The initiating controller can use the sense data to determine the actual number of bytes or blocks that have been transferred. This message is normally

used by the initiating controller to stop a lengthy read, write, or verify operation when a higher priority command is available to be executed. It is up to the initiating controller to complete the terminated command at a later time, if required.

6.8.19 TRANSFER READY

The TRANSFER READY message is sent from a target controller to an initiating controller to indicate that the execution of an active I/O process has been suspended pending receipt of additional information packets. The message is used to direct the initiating controller to continue sending information packets in the correct sequence to the target controller on the path where the message is received.

This message is also the enabling mechanism for multiple path operations for logical block data being sent from an initiating controller to a target controller. The target controller sends an information packet containing this message on the next path where it wants information packet transferred when logical block data is involved. No equivalent message is needed when read operations are performed since the target controller picks the path it wants to use and sends the information packets containing logical block data.

6.9 Command Processing Considerations and Exception Conditions

The following sections describe some exception conditions and errors associated with command processing and the sequencing of commands.

6.9.1 Unit Attention Condition

The target shall generate a unit attention condition for each initiator on each valid logical unit whenever the target has been reset by a BUS DEVICE RESET message, a hard reset condition, or by a power-on reset.

The target shall also generate a unit attention condition on the affected logical unit(s) for each initiating controller whenever one of the following events occurs:

(1) A removable medium may have been changed.

(2) The mode parameters in effect for this initiating controller have been changed by another initiating controller.

(3) The version or level of microcode has been changed.

(4) Tagged commands queued for this initiating controller were cleared by another initiating controller.

(5) INQUIRY data has been changed.

(6) The mode parameters in effect for this initiating controller have been restored from nonvolatile memory.

(7) A change in the condition of a synchronized spindle.

(8) Any other event occurs that requires the attention of the host system.

The unit attention condition shall persist on the logical unit for each initiating controller until that initiating controller clears the condition as described in the following paragraphs.

If the target controller has no active I/O process for an initiating controller, the unit attention condition shall be reported using asynchronous event notification to each initiating controller. If each SEND command completes with GOOD status, this report clears the unit attention condition for each initiating controller for which AEN has been reported.

The remainder of this discussion relates to events which occur in a target controller from detection of the event until the target controller reports the unit attention using AEN. This dicsussion describes actions which take place in this interval of time. If the following procedures are used, the unit attention has been reported and AEN is not required (unit attention is reported only once per event per initiating controller).

If an INQUIRY command is received from an initiating controller to a logical unit with a pending unit attention condition (before the target generates the AEN), the target shall perform the INQUIRY command and shall not clear the unit attention condition. If the INQUIRY command is received after the target has generated the AEN, the unit attention condition on the logical unit has been cleared; and the target shall perform the INQUIRY command.

If a command is received while preparing a unit attention AEN, other than the INQUIRY command, the target controller shall terminate the command with a contingent allegiance condition with sense key of UNIT ATTENTION. If the sense data is supplied as autosense, the contingent allegiance is cleared and the unit attention condition is also cleared.

If autosense is not supplied with the CHECK CONDITION status, if any other command is received after the target has generated the contingent allegiance condition for a pending unit attention condition, the unit attention condition on the logical unit shall be cleared. If no other unit attention condition is pending the target shall perform the command. If another unit attention condition is pending the target shall not perform the command and shall generate another contingent allegiance condition.

If a REQUEST SENSE command is received from an initiator with a pending unit attention condition (before the target generates the AEN or establishes a contingent allegiance condition), the target controller shall report any pending sense data and preserve the unit attention condition on the logical unit.

If the target has already generated the contingent allegiance condition for the unit attention condition, the target shall report the unit attention condition and preserve any other pending sense data for a subsequent contingent allegiance or AEN.

If an initiator issues a command other than INQUIRY or REQUEST SENSE while a unit attention condition exists for that initiator (prior to reporting it with AEN or generating the contingent allegiance condition for the unit attention condition), the target shall not perform the command and shall report CHECK CONDITION status unless a higher priority status as defined by the target controller is also pending (e.g., BUSY or RESERVATION CONFLICT).

If after generating the contingent allegiance condition for a pending unit attention condition, the next command received from that initiator on the logical unit is not REQUEST SENSE, then that command shall be performed and the unit attention condition shall be cleared for that initiator on the logical unit and the sense data is lost (see 6.6).

IMPLEMENTORS NOTES:

(1) Targets may queue unit attention conditions on logical units. After the first unit attention condition is cleared, another unit attention condition may exist (e.g., a power on condition followed by a microcode change condition). The initiating controller can clear all pending unit attention conditions by repeatedly sending the REQUEST SENSE command until a sense key other than UNIT ATTENTION is returned by the target.

(2) See 6.9.3 for requirements concerning selection of an invalid logical unit.

6.9.2 Incorrect Initiating Controller Connection

An incorrect initiating controller connection occurs on an initial connection when an initiator:

(1) attempts to establish an H_C_L_Q nexus when an H_C_L nexus already exists from a previous connection for the same H, C, and L values, or

(2) attempts to establish an H_C_L nexus when an H_C_L_Q nexus already exists unless there is a contingent allegiance or extended contingent allegiance condition present for the logical unit or target routine.

A target controller that detects an incorrect initiator connection shall abort terminate all I/O processes for the initiating controller with contingent allegiance. The sense key shall be set to ABORTED COMMAND and the additional sense code shall be set to OVERLAPPED I/O PROCESSES ATTEMPTED. (Section 7 Update. GRS) Other sense data appropriate for the condition is included in the sense data (e.g., number of blocks not processed). If the target controller requires a significant recovery action, it may use extended contingent allegiance rather than contingent allegiance when reporting the error.

6.9.3 I/O Processes for an Invalid LUN or TRN

The target controller response to an I/O process directed to an invalid LUN or TRN is described in the following paragraphs.

The LUN oir TRN may not be valid because:

(1) the target controller does not support the logical unit number. In response to an INQUIRY command the target controller shall return the INQUIRY data with the peripheral qualifier set to the value required in Table 7-16. In response to any other command except REQUEST SENSE the target controller shall terminate the command with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED.

(2) the target controller supports the logical unit, but the peripheral device is not currently attached to the target controller. In response to an INQUIRY command the target controller shall return the INQUIRY data with the peripheral qualifier set to the value required in Table 7-16. In response to a REQUEST SENSE command the target controller shall return the sense data with a sense key of NO SENSE to indicate that the LUN or TRN is supported. In response to any other command the target shall terminate the command with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT OPERATIONAL. (Update section 7. GRS) (path group and assignment considerations? GRS

(3) the target controller supports the logical unit and the peripheral device is attached, but not operational. In response to an INQUIRY command the target shall return the INQUIRY data with the peripheral qualifier set to the value required in Table 7-16. In response to a REQUEST SENSE command the target controller shall return the sense data with a sense key of NO SENSE to indicate that the LUN or TRN is supported. In response to any other command the target shall terminate the command with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT OPERATIONAL. (path group and assignment considerations? GRS

(4) the target supports the logical unit but is incapable of determining if the peripheral device is attached or is not operational when it is not ready. In response to an INQUIRY command the target shall return the INQUIRY data with the peripheral qualifier set to the value required in Table 7-16. In

response to a REQUEST SENSE command In response to a REQUEST SENSE command the target controller shall return the sense data with a sense key of NO SENSE to indicate that the LUN or TRN is supported. In response to any other command the target shall terminate the command with contingent allegiance. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT OPERATIONAL. (path group and assignment considerations? GRS

6.9.4 Parameter Rounding

range of values. target controllers may choose to implement only selected values from this range. When the target controller receives a value that it does not support, it either rejects the command (CHECK CONDITION status with ILLEGAL REQUEST sense key) or it rounds the value received to a supported value. The target controller shall reject unsupported values unless rounding is permitted in the description of the parameter.

Rounding of parameter values, when permitted, shall be performed as follows: A target controller that receives a parameter value that is not an exact supported value shall adjust the value to one that it supports and shall return CHECK CONDITION status with a sense key of RECOVERED ERROR. The additional sense code shall be set to ROUNDED PARAMETER. The initiating controller is responsible to issue an appropriate command to learn what value the target controller has selected.

IMPLEMENTORS NOTE: Generally, the target should adjust maximum-value fields down to the next lower supported value than the one specified by the initiating controller. Minimum-value fields should be rounded up to the next higher supported value than the one specified by the initiating controller. In some cases, the type of rounding (up or down) is explicitly specified in the description of the parameter.

6.9.5 Unsuccessful I/O Process Termination Condition

Upon detection of an unsuccessful I/O process termination condition, the target shall terminate the I/O process by clearing all pending data, establish a contintent allegiance, and prepare sense data that may be retrieved by a REQUEST SENSE command. (any AEN considerations? GRS)

When an initiator detects an unexpected disconnect, it is recommended that a REQUEST SENSE command be attempted to obtain valid sense data that is available.

6.9.6 Reset Condition

(Update. GRS)

6.9.6.1 Reset Condition (Serial)

The effect of the reset condition on I/O processes which have not completed, path groups, assignments, reservations, and SCFI device operating modes is determined in the following section.

SCFI devices shall:

(1) Attempt to complete any I/O processes which have not completed and that were fully identified

(2) Preserve all SCFI device path groups, path group status, assignments, passwords and reservations, as appropriate

(3) Preserve any SCFI device operating modes (MODE SELECT, PREVENT/ALLOW MEDIUM REMOVAL commands, etc.)

(4) Preserve all the information required to continue normal dispatching of I/O processes queued prior to the reset condition.

The SCFI bus may be reset with minimum disruption to the operation of the logical system. To ensure proper operation the following conditions shall be met:

(1) An initiator shall not consider an I/O process to be fully identified until it receive an acknowledgement information packet from the target.

(2) A target shall consider an I/O process to be fully identified when the target detects no error in the information packet containing the identify information

(3) If an initiator attempts a connection with a logical unit for which there already is an active I/O process with the same queue tag (if any) for the same initiator, the target shall clear the original I/O process and perform the new I/O process.

(4) If a target attempts a reconnection with an initiator to continue an I/O process for which the initiator has no saved pointers, the initiator shall abort that unknown I/O process by sending the ABORT or ABORT TAG message, depending on whether the I/O process is a tagged I/O process. The message is contained in an information packet.

(5) An initiator shall consider an I/O process to be completed when no error is detected in the transfer of an information packet containing the COMMAND COMPLETE message.

(6) A target shall consider an I/O process to be completed when it detects the acknowledge information packet for the information packet containing the COMMAND COMPLETE message.

(7) When using information packets, an initiator shall not transmit an acknowledge information packet for an information packet until it has actually saved the pointers for the current I/O process.

(8) A target shall consider the pointers saved when it detects the acknowledgement packet for each information packet.

(9) If the reset condition occurs before the acknowledgement information packet is received for an information packet containing a data interface logical element, the target shall terminate the I/O process with CHECK CONDITION status. The target shall set the sense key to ABORTED COMMAND. This is necessary because the target cannot determine whether the data pointer has actually been saved. If the reset condition occurs during transfer of an information packet, the packet is retransmitted and the I/O process continues.

6.9.6.2 Reset Condition (Parallel)

The effect of the Reset condition on I/O processes which have not completed, SCFI device reservations, and SCFI device operating modes is determined by whether the SCFI device has implemented the hard reset alternative or the soft reset alternative (one of which shall be implemented) as defined in 6.9.6.2 and 6.9.6.5. The hard and soft reset alternatives are mutually exclusive within a system. A facility for targets to report which reset alternative is implemented is provided in the SftRe bit of the INQUIRY data (7.2.5).

IMPLEMENTORS NOTE: Environmental conditions (e.g., static discharge) may generate brief glitches on the RST signal. It is recommended that SCFI devices not react to these glitches. The manner of rejecting glitches is vendor specific. The bus clear delay following a RST signal transition to true is measured from the original transition of the RST signal, not from the time that the signal has been confirmed. This limits the time to confirm the RST signal to a maximum of a bus clear delay.

6.9.6.3 Hard Reset Alternative (Parallel)

NOTE: The hard reset alternative is left in SCFI for backward compatibility to SCSI-3. It is recommended that all SCFI devices implement the soft reset alternative. Soft reset is mutually exclusive with the hard reset alternative in a logical system in SCSI-3.

SCFI devices that implement the hard reset alternative, upon detection of the reset condition, shall:

(1) Clear all I/O processes including queued I/O processes.

(2) Release all SCFI device reservations.

(3) Return any SCFI device operating modes to their appropriate initial conditions, similar to those conditions that would be found after a normal power-on reset. MODE SELECT conditions shall be restored to their last saved values if saved values have been established. MODE SELECT conditions for which no values have been saved shall be returned to their default values.

(4) Unit attention condition shall be set (See 6.xx).

It is recommended that following a reset to selection time after a hard reset condition ends, SCFI targets be able to respond with appropriate status and sense data to the TEST UNIT READY, INQUIRY, and REQUEST SENSE commands.

6.9.6.4 Soft Reset Alternative (Parallel)

SCFI devices that implement the soft reset alternative, upon detection of the reset condition, shall:

(1) Attempt to complete any I/O processes which have not completed and that were fully identified

(2) Preserve all SCFI device reservations

(3) Preserve any SCFI device operating modes (MODE SELECT, PREVENT/ALLOW MEDIUM REMOVAL commands, etc.)

(4) Preserve all the information required to continue normal dispatching of I/O processes queued prior to the reset condition.

The soft reset alternative allows an initiator to reset the SCFI bus with minimum disruption to the operation of other initiators in a multiple initiator system. To ensure proper operation the following conditions shall be met:

(1) An initiator shall not consider an I/O process to be fully identified until a) when not using information packets, the IDENTIFY message (and queue tag message, if any) is sent to the target and the target responds by changing to any other information transfer phase and requests that at least one byte be transferred. b) when using information packets, during the next connection with the target for the same I/O process, the target provides no response in the information packet to indicate an error in the last information packet received.

(2) A target shall consider an I/O process to be fully identified when during the next connection with the same initiator for the same I/O process, the target reports no error in the last information packet received.

(3) If an initiator attempts a connection with a logical unit for which there already is an active I/O process with the same queue tag (if any) for the same initiator, the target shall clear the original I/O process and perform the new I/O process.

(4) An initiator shall consider an I/O process to be completed when no error is detected in the transfer of an information packet containing the COMMAND COMPLETE message.

(5) A target shall consider an I/O process to be completed when it detects the end of the information packet transfer with the ATN signal false.

(6) An initiator shall not negate the ACK signal for an information packet until it has actually saved the active pointers for the current I/O process. The saved pointers must be updated before the initiator responds to a connection for another information packet for the same I/O process.

(7) A target shall consider the data pointer saved when it detects the transition of the ACK signal to false for each information packet.

(8) If the reset condition occurs during the transfer of an information packet, the target shall terminate the I/O process with CHECK CONDITION status. The target shall set the sense key to ABORTED COMMAND. This is necessary because the target cannot determine whether the data pointer has actually been saved.

NOTE: If the ATN signal is true in conditions (6) or (8), the target would normally switch to MESSAGE OUT phase or NEXUS TRANSFER OUT phase and attempt to transfer a message byte or an information packet. If the reset condition occurs before the target successfully receives the message, it may assume that the initiator has not successfully received the COMMAND COMPLETE message or the SAVE DATA POINTER message. In the case of COMMAND COMPLETE message, the target may reselect the initiator and attempt to send the COMMAND COMPLETE message again. In the case of the SAVE DATA POINTER message, the target may reselect the initiator and terminate the I/O process as described in condition (9).

6.9.7 Unsuccessful Information Packet Transfer Condition (Serial)

For an unsuccessful information packet transfer condition, the sending SCFI port:

1) may attempt to retransmit the information packet in error up to a limit determined by the sending SCFI port.

2) if operating in target mode for the I/O process, shall terminate the I/O process after the specified number of retries by clearing all pending data and preparing sense data.

3) if operating in initiator mode for the I/O process, shall abort the I/O process. The target controller prepares sense data. It is recommended that the initiator then request sense data from the target.

6.9.8 Unexpected Disconnect Condition (Serial)

TBD

6.9.9 Unexpected Disconnect Condition (Parallel)

TBD

6.9.10 Attention Condition (Parallel)

TBD

Part 3. SCFI Common Commands

Section 7. Commands Common to All Device Classes 7-1
7.1 Characteristics of All Device Classes 7-1
7.1.1 Commands Implemented by All Logical Units (Mandatory) 7-1
7.1.1.1 Using the INQUIRY Command 7-1
7.1.1.2 Using the REQUEST SENSE Command 7-1
7.1.1.3 Using the SEND DIAGNOSTIC Command 7-1
7.1.1.4 Using the TEST UNIT READY Command 7-1
7.1.2 Commands Implemented by All Target Routines (Mandatory) 7-2
7.1.2.1 Using the INQUIRY Command 7-2
7.1.2.2 Using the REQUEST SENSE Command 7-2
7.1.3 Commands for All Device Classes 7-2
7.2 Command Descriptions 7-3
7.2.1 SET ICID Command 7-5
7.2.2 REPORT PATH STATUS Command 7-9
7.2.3 ASSIGN Command 7-11
7.2.4 UNASSIGN Command 7-13
7.2.5 CONTROL ACCESS Command 7-15
7.2.6 CHANGE DEFINITION Command (Parallel) 7-19
7.2.7 COMPARE Command 7-23
7.2.8 COPY Command 7-25
7.2.8.1 Copies With Unequal Block Lengths 7-27
7.2.8.2 Errors Detected by the Copy Manager 7-28
7.2.8.3 Errors Detected by a Target Controller 7-28
7.2.8.4 COPY Function Codes 00h and 01h 7-29
7.2.8.5 COPY Function Code 02h 7-30
7.2.8.6 COPY Function Code 03h 7-31
7.2.8.7 COPY Function Code 04h 7-32
7.2.8.8 COPY Function Codes 08h and 09h 7-33
7.2.8.9 COPY Function Code 0Ah 7-35
7.2.8.10 COPY Function Code 0Bh 7-37
7.2.8.11 COPY Function Code 0Ch 7-39
7.2.9 COPY AND VERIFY Command 7-43

Section 7. Commands Common to All Device Classes

This section describes all commands which are common to all device classes. Several of the commands are mandatory for logical units ands target routines. The remainder are optional unless they are designated as mandatory in the appropriate device class section.

7.1 Characteristics of All Device Classes

This section describes some of the general characteristics expected of most SCFI devices. Section 6 of this standard also contains information pertaining to all device classes.

7.1.1 Commands Implemented by All Logical Units (Mandatory)

This standard defines four commands that all logical units shall implement: INQUIRY, REQUEST SENSE, SEND DIAGNOSTIC, and TEST UNIT READY. These commands are used to configure the SCFI busses, to test logical units, and to return important information concerning errors and exception conditions.

7.1.1.1 Using the INQUIRY Command

The INQUIRY command may be used by a system to determine the configuration of the SCFI bus. Logical Units respond with information that includes their type and standard level and may include vendor identification, model number and other useful information. It is recommended that target controllers be capable of returning this information (or whatever part of it that is available) upon completing power-on initialization. A logical unit may take longer to get certain portions of this information, especially if it retrieves the information from the medium.

7.1.1.2 Using the REQUEST SENSE Command

Whenever a contingent allegiance or extended contingent allegiance condition is established for a logical unit, the initiating controller that received the error should issue a REQUEST SENSE command to receive the sense data describing what caused the condition unless the sense data is provides as autosense data with the status. If the initiating controller issues some other command and autosense data has not been provided, the sense data is lost.

7.1.1.3 Using the SEND DIAGNOSTIC Command

The SEND DIAGNOSTIC command provides a means to request the logical unit to perform a self test. While the test is logical uint specific, the means of requesting the test is standardized and the response is simply GOOD status if all is well or CHECK CONDITION status if the test fails.

The SEND DIAGNOSTIC command also provides other powerful features when used in conjunction with the RECEIVE DIAGNOSTIC RESULTS command, but this capability is optional.

7.1.1.4 Using the TEST UNIT READY Command

The TEST UNIT READY command is useful in that it allows an initiating controller to poll a logical unit until it is ready without the need to allocate space for returned data. It is especially useful to check status of logical units with removable media. Logical units are expected to respond promptly to indicate the current status (i.e., a logical unit should avoid lengthy disconnections in an attempt to respond with GOOD status).

7.1.2 Commands Implemented by All Target Routines (Mandatory)

This standard defines two commands that all target routines shall implement: INQUIRY and REQUEST SENSE. These commands are used to test target routines, and to return important information concerning errors and exception conditions.

7.1.2.1 Using the INQUIRY Command

The INQUIRY command may be used by a system to determine the configuration of the SCFI bus. Target controllers respond with information that includes their type and standard level and may include vendor identification, model number and other useful information. It is recommended that target controllers be capable of returning this information (or whatever part of it that is available) upon completing power-on initialization. A target routine may take longer to get certain portions of this information, especially if it retrieves the information from the medium.

7.1.2.2 Using the REQUEST SENSE Command

Whenever a contingent allegiance or extended contingent allegiance condition is established for a target routine, the initiating controller that received the error should issue a REQUEST SENSE command to receive the sense data describing what caused the condition unless the sense data is provides as autosense data with the status. If the initiating controller issues some other command and autosense data has not been provided, the sense data is lost.

7.1.3 Commands for All Device Classes

The operation codes for commands that apply to all logical units and target routines for all device classes are listed in Table 7-1 on page 7-3. Commands identified as mandatory shall be implemented for logical units. Commands identified as mandatory for target routines shall be implemented if target routines are implemented.

When a command is implemented, whether mandatory or optional, all functions identified as mandatory within the command shall be implemented.

Table 7-1. Commands for All Device Classes

Command Name	Operation Code	Logical Unit Support	Target Routine Support	Section
SET ICID	xxh	O	R	7.2.1
REPORT PATH STATUS	xxh	O	R	7.2.2
ASSIGN	xxh	O	R	7.2.3
UNASSIGN	xxh	O	R	7.2.4
CONTROL ACCESS	xxh	O	R	7.2.5
CHANGE DEFINITION (Parallel Only)	40h	O	R	7.2.6
COMPARE	39h	O	R	7.2.7
COPY	18h	O	R	7.2.8
COPY AND VERIFY	3Ah	O	R	7.2.9
INQUIRY	12h	M	M	-- Heading 'INQUIRE' un
LOG SELECT	4Ch	O	R	-- Heading 'LOGSEL' un
LOG SENSE	4Dh	O	R	-- Heading 'LOGSNS' un
MODE SELECT	55h	Z	R	-- Heading 'MODESEL' u
MODE SENSE	5Ah	Z	R	-- Heading 'MODESNS'
READ BUFFER	3Ch	O	R	-- Heading 'RDBUF' unk
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	R	-- Heading 'RECVDR' un
REQUEST SENSE	03h	M	M	-- Heading 'REQSNS' un
SEND DIAGNOSTIC	1Dh	M	R	-- Heading 'SNDDIAG' u
TEST UNIT READY	00h	M	R	-- Heading 'TUR' unkno
WRITE BUFFER	3Bh	O	R	-- Heading 'WTBUF' unk

Key:

M = Command implementation is mandatory.
O = Command implementation is optional.
R = Reserved
Z = Command implementation is device class specific. See the appropriate device class section

7.2 Command Descriptions

The commands listed in Table 7-1 are defined in this section.

7.2.1 SET ICID Command

(move to correct position. GRS)

Table 7-2. SET ICID Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2	Reserved		NamePath	AddPath	RmvPath	EstabPG	SnglStus	DsbndPth
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7 - 8	(MSB) Command Parameter Data Length (LSB)							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: YES
- Supervisor Command: NO
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The SET ICID command (Table 7-2) has five functions: to explicitly name the initiating controller using this path on which a connect occurred; to add a path to an established path group; to remove a path from an established path group; to establish a path group; to disband a path group. This command is valid only for logical units. If explicitly named path groups are supported, this command and all CDB options shall be supported.

A path shall be explicitly named before it can be included in an established path group. The set of paths consisting of the same ICID and LUN in a target controller is called a path group. Naming the paths in a path group does not established the path group. If an established path group is required, it shall be explicitly established using this command with the correct option specified. Establishing a path group is the mechanism for enabling multiple path operations in logical system and allowing assignment for multiple paths. Including a path in a path group does not restrict access to the logical unit from any other path.

When the name path (NamePath) bit is set to one, the command is used to name the path on which the connect is made. Command parameter data supplies the ICID of the initiating controller. The interface control prefix fields contain the remaining information to completely name the path to a LUN. The LUN must be valid for the target controller. The path remains in an ungrouped state. Explicitly naming a path to a logical unit or target routine does not alter access privileges (i.e., assignment). If the NamePath bit is set to one, and the RmvePath, the EstabPth, or DsbndPth bit is set to one, the

command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... The SnglPath bit shall be ignored. The command parameter data length shall be set to 16. See the command parameter description below.

When the name path (NamePath) bit is set to one and the path already has been given a name, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

When the name path (NamePath) bit is set to zero, the path name function shall not be invoked in the target controller.

A path may be added to an established group at a later time by a command from the initiating controller on the path to be added to the path group. When the add path (AddPath) bit is set to one, the named path which has been previously named or is named with this command, if the NamePath bit is set to one, shall be added to an established path group of the same name. The attributes of the path group shall be transferred to the new path. The path enters the grouped state. Adding a path to a path group causes the assignment status of the path group to be transferred to the new path. Adding a path to If the AddPath bit is set to one and the RmvePath bit, the EstabPth bit, or the DsbndPth bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... The SnglPath bit shall be ignored. The command parameter data length shall be set to 16. If the ICID name does not match an established path group, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... If the path does not have a name assigned and the AddPath bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

A path may be removed from an established path group by a command from the initiating controller on the path to be removed from the path group. When the remove path (RmvePath) bit is set to one, the named path which has been previously named, shall be deleted from the established path group of the same name. The attributes of the path group shall be removed from the path. The path enters the ungrouped state. The path retains the name provided through the add path function. Removing a path from a path group removes assignment for that path if assignment exists for the path group. A path group is implicitly disbanded when the last path is removed from an established path group using the RmvePath function rather than a disband command. The logical path remains as it did before the path group was established. If the RmvePath bit is set to one and the AddPath bit, the EstabPth bit, or the DsbndPth bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... The SnglPath bit shall be ignored. The command parameter data length shall be set to 16. If the ICID name does not match an established path group, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... If the path does not have a name assigned and the RmvePath bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

A set of named paths shall be established as a path group when the establish path (EstabPth) bit is set to one. The value of the single path status (SnglStus) bit is used in conjunction with the EstabPth bit as specified below. The status reporting attribute can only be changed by disbanding the path group and reestablishing it with the desired attribute. Each path in the newly established path group enters the grouped state. Establishing a path group for a logical unit or target routine does not alter access privileges. If the EstabPth bit is set to one and the AddPath bit, the RmvePath bit, or the DsbndPth bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... The command parameter data length shall be set to 16. If the ICID name does not match the name of one or more paths in an ungrouped state, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

If the EstabPth bit is set to one and the single path status (SnglStus) bit is set to one, the path group is established with the attribute that any contingent allegiance or extended contingent allegiance condi-

tion shall be reported on the path where the connect was made for each I/O process initiated in the path group. This is called single path status mode. Any report status may be returned on any valid path in the established path group.

If the EstabPth bit is set to one and the single path status (SnglStus) bit is set to zero, the path group is established with the attribute that any contingent allegiance or extended contingent allegiance condition may be reported on any path in the established path group. This is called multiple path status mode. Any report status may be returned on any valid path in the established path group.

The inverse of establishing a path group is to disband a path group. A path group shall be explicitly disbanded when the disband path group (DsbndPth) bit is set to one and the command is received from the initiating controller along any one path in the path group. Each path enters the ungrouped state. Each path retains the name provided through the add path function. Disbanding a a path group removes assignment for that path if assignment exists for the path group. If the DsbndPth bit is set to one and the AddPath bit, the RmvePath bit, or the EstabPth bit is set to one, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... The SnglPath bit shall be ignored. The command parameter data length shall be set to 16. If the ICID name does not match the name of an established path group, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

Table 7-3. Command Parameter Data for SET ICID Command

Bit Byte	7	6	5	4	3	2	1	0
0 - 11	(MSB) ICID Name (LSB)							
12	Reserved							
13	Reserved							
14	Reserved							
15	Reserved							

The ICID Name field of the SET ICID command consists of a 12 byte left justified ASCII value which uniquely names the initiating controller responsible for the initiator making the connect for the SET ICID command. The ICID Name is not an attribute of the initiator used to send the SET ICID command; it is an attribute of the controller of the initiator. An ICID Name shorter than 12 bytes shall be padded to the LSB with ASCII blanks (20h). If the ICID Name field is set to all 20h (i.e., all blanks), the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

NOTE: The ICID Name value can be a manufacturer ID concatenated with the serial number of the computer system. That is, all initiators in the same computer system use the same initiating controller ID so that path grouping of several paths is possible. This also permits multiple path operations.

7.2.2 REPORT PATH STATUS Command

(move to correct position. GRS)

Table 7-4. REPORT PATH STATUS Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: YES
- Supervisor Command: NO
- Command Parameter Data: NONE
- Command Response Data: YES
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The REPORT PATH STATUS command (Table 7-4) reports the status of the path on which the connect was made relative to the path group naming level and status of a path group. The command response data provides the target controller status of the path and path group. There are no options specified in the CDB. If explicitly named path groups are supported, this command and all CDB options shall be supported. The command response data is defined in Table 7-5.

Table 7-5. Command Response Data for REPORT PATH STATUS Command

Bit / Byte	7	6	5	4	3	2	1	0
0 - 11	(MSB) ICID Name (LSB)							
12	ValICID	ImplPath	PathOthr	Ungrpd	Grouped	Reserved		
13	Reserved							
14	Reserved							
15	Reserved							

If the Valid ICID (ValICID) bit is set to one then the ICID name field contains a valid ICID name from the target controller. A value of zero for the ValICID bit means that the ICID Name field shall be ignored. When the PathOthr, Ungrpd, or Grouped is set to one, the ValICID bit shall be set to one. In this instance, if the target controller returns command response data with the ValICID bit is set to zero, ... If the ImplPath bit is set to one and the ValICID bit is set to one in the command response data,.... When the ValICID bit is set to one, the ICIC Name field contains the ICID value transferred by a SET ICIC command which was successfully executed.

The state of the path may be any one of the following:

1) If the Implicitly named path (ImplPath) bit is set to one, no explicit ICID has been received from the initiating controller to any LUN or TRN on this target controller using this path. An I/O process received on a path in this state shall operate in single path mode.

2) If the Path to Other LUNs (PathOthr) bit is set to one, an explicitly named path to at least one LUN or TRN on this path, other than the one selected exists. This is functionally equivalent to status 1), but it imparts additional information to the initiating controller. An I/O process received on a path in this state shall operate in single path mode.

3) If the Ungrouped (Ungrpd) bit is set to one, an explicitly named path to the selected LUN or TRN exists for this path, but it is not currently part of an established path group. An I/O process received on a path in this state operates in single path mode.

4) If the Grouped bit is set to one, an explicitly named path to the selected LUN or TRN exists and is currently established in the grouped state. The path group can consist of one or more paths. An I/O process received on this path may respond on any path in this path group unless single path status is in effect or multiple path operation has been temporarily suspended for an I/O process.

The ImplPath, PathOthr, Ungrpd, and Grouped bits are mutually exclusive. If the target returns command response data with more than one of these bits set to one, ...

7.2.3 ASSIGN Command

(move to correct position. GRS)

Table 7-6. ASSIGN Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7 - 8	(MSB) Command Parameter Data Length (LSB)							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: YES
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The purpose of the ASSIGN command (Table 7-6) is to act as the logical equivalent of switches or manual cable changes to restrict access to logical units or target routines. Physical reconfigurations may be impractical or impossible in a logical system to effect the desired access restrictions. Both initiating controllers and target controllers must keep track of assignment. Both must communicate on the appropriate paths and the target controller must be prepared to reject attempts for access from unassigned paths. If explicitly named path groups are supported, this command and all CDB options shall be supported.

The ASSIGN command specifies the path group on which a logical unit or target routine may operate. Any logical unit or target routine is initially available to receive I/O processes from any initiating controller attached to the target controller. This use privilege is extended whether the path is explicitly named, implicitly named, and whether for explicitly named paths, the path is grouped or ungrouped. Two or more established path groups may share use privileges to the exclusion of other paths or path groups. An implicitly named path may hold assignment for itself.

Assignment to one path group or to one implicitly named path means that no initiating controller on any path not in the assigned path group or the single assigned implicitly named path shall gain access to the logical unit or target routine unless temporary assignment has been granted through the CONTROL ACCESS command. Any path holding assignment through an established path group may add assignment of other established path groups. The mechanism by which an initiating controller

obtains the path group name required for adding assignment of other path groups is not defined in SCFI.

A logical unit or target routine may be assigned to multiple established path groups. An implicitly named path or an ungrouped path can gain assignment for itself, but it shall not be capable of adding assignment for any other paths or path groups.

The two functions of assignment are:

1) If the command parameter data length value is zero, assign this LUN or TRN to the path group on which the command was received. It shall not be an error to receive this command on a path which currently has assignment.

2) If the command parameter data length is 16 and the ICID name field matches the name of another established path group, and the path on which the command is received also has assignment, the target controller shall add assignment of the LUN or TRN to the other established path group. It shall not be an error to receive this command on a path which currently has assignment and which requests assignment for another established path group for which assignment currently exists. If the ASSIGN command is received on a path for which assignment does not exist and requests assignment for another established path group, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... If the ICID Name field does not match the ICID name of another established path group in the target controller, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

The command parameter data is defined in Table 7-7.

Table 7-7. Command Parameter Data for ASSIGN Command

Bit / Byte	7	6	5	4	3	2	1	0
0 - 11	(MSB) ICID Name (LSB)							
12	Reserved							
13	Reserved							
14	Reserved							
15	Reserved							

The ICID Name field of the ASSIGN command consists of a 12 byte left justified ASCII value which uniquely names the path group for which assignment is to be made. See the SET ICID command for the construction of the ICID Name field. If the ICID Name field is set to all 20h (i.e., all blanks), the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

7.2.4 UNASSIGN Command

(move to correct position. GRS)

Table 7-8. UNASSIGN Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7 - 8	(MSB) Command Parameter Data Length (LSB)							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: YES
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The UNASSIGN command (Table 7-8) is used to remove a path group from the set of assigned path groups. If explicitly named path groups are supported, this command and all CDB options shall be supported.

Use privileges may be unassigned for any path group to which assignment currently exists from any path for which assignment currently exists. The mechanism by which an initiating controller obtains the path group name required for removing assignment of other path groups is not defined in SCFI. If the ASSIGN command is received on a path for which assignment does not exist, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

The two functions of unassignment are:

1) If the command parameter data length value is zero, unassign this LUN or TRN to for the established path group on which the command was received.

2) If the command parameter data length is 16 and the ICID name field matches the name of another established path group which has assignment, the target controller shall remove assignment of the LUN or TRN for the other established path group. If the ICID Name field does not match the ICID name of another established path group in the target controller, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

The command parameter data is defined in Table 7-9.

Table 7-9. Command Parameter Data for UNASSIGN Command

Bit / Byte	7	6	5	4	3	2	1	0
0 - 11	(MSB) ICID Name (LSB)							
12	Reserved							
13	Reserved							
14	Reserved							
15	Reserved							

The ICID Name field of the UNASSIGN command consists of a 12 byte left justified ASCII value which uniquely names the path group for which assignment is to be made. See the SET ICID command for the construction of the ICID Name field. If the ICID Name field is set to all 20h (i.e., all blanks), the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

7.2.5 CONTROL ACCESS Command

(move to correct position. GRS)

Table 7-10. CONTROL ACCESS Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (XXh)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7 - 8	(MSB) Command Parameter Data Length (LSB)							
9	Control Field							

Command Attributes:

- Bypass Assignment: YES with Correct Password
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: YES
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The CONTROL ACCESS command (Table 7-10) transfers a password on an assigned path to the target controller for a path group; or,permits general unassign from an assigned path group; or, allows an unassigned path having the correct password access to a logical unit or target routine for one I/O process. Breaking assignment may be temporary or permanent, but it shall be controlled. If explicitly named path groups are supported, this command and all CDB options shall be supported.

Controlled access permits access outside the bounds of assigned path groups. The CONTROL ACCESS command provides a mechanism to prevent deliberate or accidental loss of assignment protection is the control access function, enabled by a password, and checked by the affected target controller.

A password is established by an initiating controller having assignment. The password is not reported by a target controller on any path. The target controller checks its established password, if any, against password supplied by the control access command function from an initiating controller not having assignment.

If the target controller has an established password and it matches the password with the command, the control access command and any commands linked to it are executed, if possible. The mechanism by which the unassigned initiating controller acquires the correct password is not defined in SCFI.

The command parameter data length value shall be 16.

Table 7-11. Command Parameter Data for CONTROL ACCESS Command

Bit / Byte	7	6	5	4	3	2	1	0
0 - 11	(MSB) ICID Name (LSB)							
12	EstabPwd	GnlUnasn	ReqTAsgn	Reserved				
13	Reserved							
14	Reserved							
15	Reserved							

The ICID Name field of the UNASSIGN command consists of a 12 byte left justified ASCII value which uniquely names the path group for which assignment is to be made. See the SET ICID command for the construction of the ICID Name field. If the ICID Name field is set to all 20h (i.e., all blanks), the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to ....

The control access command has three functions:

1) if the establish password (EstabPwd) bit is set to one, establish a password in a target controller. The command must be received on a path currently holding assignment. Only one password, the first received, shall be the password for the established path group. If a password exists and the password in the command parameter data is not equal to the established password, terminate the I/O process with contingent allegiance. The sense key shall be .... and the additional sense shall be ....

2) if the general unassign (GnlUnasn) bit is set to one, and the the CONTROL ACCESS command is received on a path that does not have assignment, the command shall be rejected with a contingent allegiance with the sense key set to .... and the additional sense set to .... Otherwise, if no password has been set or the password supplied matches the password in the target, the target controller removes assignment for all paths in any path group having assignment when the command was processed. The result is that the logical unit or target routine.

3) if the request temporary assignment (ReqTAsgn) bit is set to one, and the control access command is received from a path that does not have assignment and the password matches the password in the target controller, the CONTROL ACCESS command and any commands linked to it are executed to the extent possible. A status which would lead to contingent allegiance or extended contingent allegiance on the unassigned path is not permitted, since that would grant the unassigned path permission to start a second I/O process (first to retrieve the sense data and then link additional commands based on the results of the sense data analysis. The contingent allegiance or extended contingent allegiance is made with an assigned path whether functional or not. Asynchronous event notification may be used to report the condition on the alternate path.

NOTES:

1) When a request for temporary assignment is granted, the issuing initiating controller may link to an assign command which will grant assignment to the once unassigned initiating controller. The initiating controller can then use normal I/O processes.

2) An I/O process containing a valid control access command requesting temporary assignment, a control access command performing a general unassign, and an assign command for the new initiating controller breaks all old assignments and transfers assignment of the LUN to the new initiating controller. This operation permits continued operation without loss of the function provided by the logical unit. Some I/O processes may be aborted.

7.2.6 CHANGE DEFINITION Command (Parallel)

Table 7-12. CHANGE DEFINITION Command (Parallel Only)

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (40h)							
1	Reserved							
2	Reserved							Save
3	Definition Parameter							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Command Parameter Data Length							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: NO
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

Use of this command is restricted to initiating controllers and target controllers attached to a parallel SCFI bus. This command is retained to permit interoperability with SCSI-3 target controllers only on a logical system with a parallel SCFI bus.

The CHANGE DEFINITION command (Table 7-12) modifies the operating definition of the selected logical unit or target controller with respect to commands from the selecting initiating controller or for all initiating controllers. Definitions supported by a target controller are returned in the implemented operating definition page (see 7.3.4.3). It is permissible for a SCFI, SCSI-3, or SCSI-2 target controller that has its definition changed to an SCSI-1 device to accept a CHANGE DEFINITION command.

A save control bit (Save) of zero indicates that the target controller shall not save the operating definition. A Save bit of one indicates that the target controller shall save the operating definition to non-volatile memory.

The definition parameter field is defined in Table 7-13 on page 7-20.

Table 7-13. Definition Parameter Values

Value	Description
00h	Use Current Operating Definition
01h	SCSI-1 Operating Definition
02h	CCS Operating Definition
03h	SCSI-2 Operating Definition
04h	SCSI-3 Operating Definition
05h	SCFI Operating Definition
06h-7Fh	Reserved
80h-FFh	Vendor Specific

The parameter data length field specifies the length in bytes of the command parameter data. A command parameter data length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. Command parameter data lengths greater than zero indicate the number of bytes that shall be transferred.

The command parameter data is vendor specific (See -- Table 'S7CDCPD' unknown --).

Table 7-14. Command Parameter Data for CHANGE DEFINITION Command

Bit Byte	7	6	5	4	3	2	1	0
0 - n (n < 255)	Vendor Specific							

The CHANGE DEFINITION command causes one of the operating definition modifications listed below:

(1) Change the operating definition of a logical unit relative to the initiating controller that issued the command. In this case, the target controller is capable of maintaining an unique operating definition for each logical unit relative to each initiating controller on the SCFI busses.

(2) Change the operating definition of the target controller relative to the initiating controller that issued the command. In this case, the target controller is capable of maintaining an unique operating definition, for each initiating controller on the SCFI busses, that applies to all logical units of the target controller.

(3) The operating definition of a logical unit relative to all initiating controllers on the SCFI busses. In this case, the target controller is capable of maintaining an unique operating definition for each logical unit relative to all initiating controllers in the system.

(4) The operating definition of the target controller relative to all initiating controllers in the system. In this case, the target controller is capable of maintaining only one operating definition.

The operating definition is modified after successful completion of the command. A target controller shall consider the command successfully completed when it detects the assertion of the ACK signal for the COMMAND COMPLETE message. The initiating controller should verify the new operating definition by issuing an INQUIRY command requesting the implemented operating definition page (see Table 7-75).

2/6

If the CHANGE DEFINITION command is not executed successfully for any reason, the operating definition shall remain the same as it was before the CHANGE DEFINITION command was attempted. If it is impossible to return to the previous operating definition, an unit attention condition shall be generated by the target controller.

After a power-on condition or a hard RESET condition, the target controller shall set its initial operating definition to the last saved value, if saving is implemented, or its default value, if saving is not implemented.

IMPLEMENTORS NOTE:

(1) The command parameter data may be used to specify a password to validate an operating definition change.

(2) The current operating definition parameter values establish operating definitions compatible with the appropriate SCSI or SCFI standard. Vendor specific values are available for those applications where more complex operation definition changes are required.

(3) This standard does not provide a direct means to determine which of the above four methods has been implemented by the target controller. An indirect means of determining which method is implemented exists in that the target controller is required to inform affected initiating controllers of operating definition changes via the unit attention condition.

(4) Cases (3) and (4), above, may result in incompatibilities if there are other initiating controllers on the SCFI busses operated below the SCSI-2 level.

(5) The method of changing the operating definition is implementation dependent. Some implementations may require the target controller's operating mode be re-initialized as if a power-up or hard-reset had occurred. Other implementations may modify only those operating definitions that are affected by the CHANGE DEFINITION command.

(6) The present operating definition of the target controller may always be interrogated through the INQUIRY command. When an SCSI-3 or SCSI-2 target controller has its operating definition changed to CCS or SCSI-1, certain changes are needed to promote compatibility with pre-existing SCSI-1 initiating controllers. The recommended changes are as follows:

(a) The target controller should not initiate selections to other SCSI devices to determine if any initiating controllers support AEN. The target controller should assume that none are capable of receiving AEN and not issue an AEN.

(b) The target controller should not generate extended contigent allegiance conditions by issuing an INITIATE RECOVERY message.

(c) If a REQUEST SENSE command with an allocation length of zero is received, the target controller should return four bytes of sense data.

(d) If an INQUIRY command is received, the returned data should have appropriate values in the ANSI version and response data format fields. The features supported bits should be zero.

(e) A change in the operating definition may change the vendor identifier, the device type, the device model, the SCSI implementation level, the command set, and any other operating characteristics.

7.2.7 COMPARE Command

Table 7-15. COMPARE Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (39h)							
1	Reserved							Pad
2	Reserved							
3 - 5	(MSB) Command Parameter Data Length (LSB)							
6	Reserved							
7	Reserved							
8	Reserved							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: NO
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The COMPARE command (Table 7-15) provides the means to compare data from one logical unit with another or the same logical unit in a manner similar to the COPY command.

This command functions in the same manner as the COPY command, except that the data from the source is compared on a byte-by-byte basis with the data from the destination. The command parameter data transferred to the target controller is the same as for the COPY command. This command parameter data contains the information to identify the logical units involved in the comparison and the length of the comparison. (See 7.2.8, "COPY Command" on page 7-25 about the COPY command and the command parameter data.)

If the comparison is unsuccessful, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to MISCOMPARE. The remaining fields in the sense data shall be set as documented in the COPY command.

7.2.8 COPY Command

Table 7-16. COPY Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (18h)							
1	Reserved							Pad
2 - 4	(MSB) Command Parameter Data Length (LSB)							
5	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: NO
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The COPY command (Table 7-16) provides a means to copy data from one logical unit to another or the same logical unit. The logical unit that receives and performs the COPY command is called the copy manager. The copy manager is responsible for copying data from a logical unit (source device) to a logical unit (destination device). These logical units may reside on different SCFI devices or the same SCFI device (in fact all three may be the same logical unit). Some SCFI devices that implement this command may not support copies to or from another SCFI device, or may not support third party copies (i.e., both the source and the destination logical units reside on other SCFI devices).

The pad bit is used in conjunction with the Cat bit in the segment descriptors to define what action should be taken when a segment of the copy does not fit exactly into an integer number of destination blocks. See 7.2.8.1, "Copies With Unequal Block Lengths" on page 7-27 for valid combinations and meanings.

The command parameter data length field specifies the length in bytes of the parameters that shall be sent during the DATA OUT phase of the command. A command parameter data length of zero indicates that no data shall be transferred. This condition shall not be considered as an error.

The COPY command parameter data (Table 7-6) begins with a four-byte header that contains the COPY function code and priority. Following the header is one or more segment descriptors.

Table 7-6: COPY Command Parameter Data

Bit / Byte	7	6	5	4	3	2	1	0
0	COPY Function Code					Priority		
1	Vendor Specific							
2	Reserved							
3	Reserved							
4-mm	Segment Descriptor(s)							

0 - xx	Segment Descriptor 0 (See specific table for length.)
. . .	
0 - xx	Segment Descriptor n (See specific table for length.)

The COPY function code field defines a specific format for the segment descriptors. The COPY function codes are defined in Table 7-7. A target controller need not support all function codes for its device type.

The priority field of the COPY command parameter data establishes the relative priority of this COPY command to other commands being executed by the same target controller. All other commands are assumed to have a priority of 1. Priority 000b is the highest priority with increasing values indicating lower priorities.

The segment descriptor formats are determined by the COPY function code. The segment descriptor format used for block devices (i.e., write-once, CD- ROM, optical-memory, and direct-access devices) shall be the same. The segment descriptor format used for stream devices (i.e., printer, processor, communications, and sequential-access devices), shall be the same. Thus a copy operation from a write-once device to a printer device uses the same segment descriptor format as a copy operation from a direct-access device to a sequential-access device (see Table 7-7). The segment descriptor formats are described in Tables 7-8 through 7-11. A maximum of 256 segment descriptors are permitted. The segment descriptors are identified by ascending numbers beginning with zero.

Table 7-7: COPY Function Codes

Peripheral Device Class		COPY Function Codes	Segment Descriptor Table	
Source	Destination			Comments
Block Devices (Device class 0,4,5,7)	Stream Devices (Device class 1,2,3,9)	00h, 08h	7-8	
Stream Devices (Device class 1,3,9)	Block Devices (Device class 0,4,5,7)	01h, 09h	7-8	(Note 3)
Block Devices (Device class 0,4,5,7)	Block Devices (Device class 0,4,5,7)	02h, 0Ah	7-9	(Note 3)
Stream Devices (Device class 1,3,9)	Stream Devices (Device class 1,2,3,9)	03h, 0Bh	7-10	
Sequential-Access (Device type 1)	Sequential-Access (Device type 1)	04H, 0Ch	7-11	Image Copy

NOTES:
(1) COPY function codes 05h - 07h and 0Dh - 0Fh are reserved.
(2) COPY function codes 10h - 1Fh are vendor specific.
(3) When using the COMPARE command the destination block device may be a CD-ROM device or an optical-memory device that uses read-only media.

7.2.8.1 Copies With Unequal Block Lengths

When copying data between two devices with unequal block lengths, it is possible for the last source block to not completely fill the last destination block for one or more segments in the COPY command. Two optional bits are defined to specify the actions of the copy manager in this circumstance. The Pad bit (in the command descriptor block) and the Cat bit (in each applicable segment descriptor) are defined in Table 7-17.

Table 7-17. Pad Bit and Cat Bit Combinations

Pad	Cat	Copy Manager Action
0	0	On inexact segments, it is vendor specific whether the COPY manager rejects the COPY command with CHECK CONDITION status and ILLEGAL REQUEST sense key, the COPY manager writes or accepts short blocks (variable-block mode on sequential-access devices), or the COPY manager adds pad bytes (00h) to the destination block or strips pad bytes from the source block.
1	0	On inexact segments, the COPY manager shall add pad bytes (00h) to the destination block to completely fill the block or it shall strip pad bytes from the source block, always stopping at the end of a complete block.
xb	1	The COPY manager shall always write or read complete blocks. On inexact segments, the remainder of the block contains data from the next segment. This code is not valid in the last segment of the COPY command.

IMPLEMENTORS NOTE: Use of pad bytes is intended to assist in managing COPY commands between devices of different block lengths where partial-block residues may occur. The initiating

controller which issued the COPY command is responsible for management of these pad areas (i.e., remembering where they are). One possible method is to write the COPY command parameter data information to the destination medium prior to issuing the COPY command for backup and to read this information prior to issuing the COPY command for restore.

7.2.8.2 Errors Detected by the Copy Manager

Two classes of exception conditions may occur during execution of a COPY command. The first class consists of those exception conditions detected by the copy manager These conditions include parity errors while transferring the COPY command and status byte, invalid parameters in the COPY command, invalid segment descriptors, and inability of the SCFI device controlling the COPY functions to continue operating. In the event of such an exception condition, the SCFI device managing the COPY shall:

(1) Terminate the COPY command with CHECK CONDITION status.

(2) The valid bit in the sense data shall be set to one. The segment number shall contain the number of the segment descriptor being processed at the time the exception condition is detected. The sense key shall contain the sense key code describing the exception condition (i.e., not COPY ABORTED). The information field shall contain the difference between the number of blocks field in the segment descriptor being processed at the time of the failure and the number of blocks successfully copied. This number is the residue of unprocessed blocks remaining for the segment descriptor.

7.2.8.3 Errors Detected by a Target Controller

The second class of errors consists of exception conditions detected by the SCFI device transferring data at the request of the copy manager. The copy manager detects exception conditions by receiving CHECK CONDITION status from one of the SCFI devices it is managing. It then shall recover the sense data associated with the exception condition.

The copy manager may also be the source or destination SCFI device (or both). It shall distinguish between a failure of the management of the COPY and a failure of the data transfer being requested. It shall then create the appropriate sense data internally.

After recovering the sense data associated with the detected error, the SCFI copy manager shall:

(1) Terminate the COPY command with CHECK CONDITION status.

(2) The valid bit in the sense data shall be set to one. The segment number shall contain the number of the segment descriptor being processed at the time the exception condition is detected. The sense key shall be set to COPY ABORTED. The information field shall contain the difference between the number of blocks field in the segment descriptor being processed at the time of the failure and the number of blocks successfully copied. This number is the residue of unprocessed blocks remaining for the segment descriptor.

The first byte of the command-specific information field shall specify the starting byte number, relative to the first byte of sense data, of an area that contains (unchanged) the source logical unit's status byte and sense data. A zero value indicates that no status byte or sense data is being returned for the source logical unit.

The second byte of the command-specific information field shall specify the starting byte number, relative to the first byte of sense data, of an area that contains (unchanged) the destination logical unit's status byte and sense data. A zero value indicates that no status byte or sense data is being returned for the destination logical unit.

7.2.8.4 COPY Function Codes 00h and 01h

The format for the segment descriptors for COPY transfers between block and stream devices is specified in Table 7-8. This format is required for COPY function codes 00h or 01h. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-8: Segment Descriptor for COPY Function Codes 00h and 01h

Bit / Byte	7	6	5	4	3	2	1	0
0	Source Address			Reserved	Cat	Source LUN		
1	Destination Address			Reserved		Destination LUN		
2	(MSB)							
3	Stream Device Block Length							(LSB)
4	(MSB)							
7	Block Device Number of Blocks							(LSB)
8	(MSB)							
11	Block Device Logical Block Address							(LSB)

The source address and source LUN fields specify the SCFI bus ID and logical unit of the device to copy the data from for this segment of the COPY command. The destination address and destination LUN fields specify the SCFI bus ID and logical unit to copy the data to for this segment of the COPY command. Some SCFI devices may not support third-party COPY in which the copying SCFI device is not the source or destination device. Some SCFI devices only support COPY within the SCFI device and not to other SCFI devices. If an unsupported COPY operation is requested, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN PARAMETER DATA (see 7.2.3.1).

A catenate (Cat) bit (optional) of one indicates that the COPY manager shall catenate the last source block of a segment with the first source block of the next segment if the last source block does not end exactly at the end of the destination block. The definition of a cat bit of zero depends on the setting of the pad bit in the command descriptor block (see 7.2.8.1, "Copies With Unequal Block Lengths" on page 7-27).

The stream device block-length field specifies the block length to be used on the stream device logical unit during this segment of the COPY command. If the SCFI device managing the COPY knows this block length is not supported, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN PARAMETER DATA . If the block length is found to be invalid while executing a read or write operation to the stream device, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to COPY ABORTED (see 7.2.3.2).

The block device number of blocks field specifies the number of blocks in the current segment to be copied. A value of zero indicates that no blocks shall be transferred in this segment.

The block device logical block address field specifies the starting logical block address on the logical unit for this segment.

7.2.8.5 COPY Function Code 02h

The format for the segment descriptors for COPY transfers among block devices is specified in Table 7-9. This format is required for COPY function code 02h. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-9: Segment Descriptor for COPY Function Code 02h

Bit / Byte	7	6	5	4	3	2	1	0
0	Source Address			DC	Cat	Source LUN		
1	Destination Address			Reserved		Destination LUN		
2	Reserved							
3	Reserved							
4	(MSB) Stream Device Block Length							
7								(LSB)
8	(MSB) Block Device Number of Blocks							
11								(LSB)
12	(MSB) Block Device Logical Block Address							
15								(LSB)

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

A destination count (DC) bit of zero indicates that the number of blocks field refers to the source logical unit. A DC bit of one indicates that the number of blocks field refers to the destination logical unit.

The number of blocks field specifies the number of blocks to be transferred to or from (depending on the DC bit) the block device during this segment. A value of zero indicates that no blocks shall be transferred.

The source logical block address field specifies the starting logical block address on the source block device.

The destination logical block address field specifies the starting logical block address on the destination block device.

7.2.8.6 COPY Function Code 03h

The format for the segment descriptors for COPY transfers among stream devices is specified by Table 7-10. This format is required for COPY function code 03h. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-10: Segment Descriptor for COPY Function Code 03h

Bit / Byte	7	6	5	4	3	2	1	0
0	Source Address			DC	Cat	Source LUN		
1	Destination Address			Reserved		Destination LUN		
2	Reserved							
3	Reserved							
4	(MSB) Source Block Length							
5								(LSB)
6	(MSB) Destination Block Length							
7								(LSB)
8	(MSB) Number of Blocks							
11								(LSB)

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

A destination count (DC) bit of zero indicates that the number of blocks field refers to the source logical unit. A DC bit of one indicates that the number of blocks field refers to the destination logical unit.

The source block length field specifies the block-length of the source device for this segment of the COPY. A zero in this field indicates variable block-length. For non-zero values, this field shall match the logical unit's actual block-length.

If block-length mismatches are detected prior to the beginning of the read operation by the SCFI device managing the COPY, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER DATA (see 7.2.3.1).

If the mismatches are detected during the read operation by the COPY manager, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to COPY ABORTED (see 7.2.3.2). and the additional sense code shall be set to INVALID FIELD IN PARAMETER DATA.

The destination block-length field specifies the block length to be used on the destination logical unit during the COPY. Destination block length mismatches are handled in an analogous manner as source block length mismatches.

The number of blocks field specifies the number of blocks to be transferred to or from (depending on the DC bit) the device during this segment. A value of zero indicates that no blocks shall be transferred.

7.2.8.7 COPY Function Code 04h

The format for the segment descriptors for image COPY transfers between sequential-access devices is specified in Table 7-11. This format is required for COPY function code 04h. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-11: Segment Descriptor for COPY Function Code 04h

Bit / Byte	7	6	5	4	3	2	1	0
0	Source Address			Reserved			Source LUN	
1	Destination Address			Reserved			Destination LUN	
2	Count							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Vendor Specific							
11								

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

The image mode COPY command copies an exact image of the source device medium to the destination device medium, beginning at their current positions. The copy function terminates when the source device:

(1) encounters an end-of-partition as defined by the source device

(2) encounters an end-of-data as defined by the source device (i.e., BLANK CHECK sense key)

(3) has copied the number of consecutive filemarks specified in the count field from the source device to the destination device

(4) has copied the number of consecutive setmarks specified in the count field from the source device to the destination device, if the RSmk bit in the device configuration page (see 9.3.3.1) is one.

A count field of zero indicates that the COPY command shall not terminate due to any number of consecutive filemarks or setmarks. Other error or exception conditions (e.g., early-warning end-of-partition on the destination device) may cause the COPY command to terminate prior to completion. In such cases, it is not possible to calculate a residue, so the information field in the sense data shall be set to zero.

7.2.8.8 COPY Function Codes 08h and 09h

The format for the segment descriptors for COPY transfers between block and stream devices is specified in Table 7-8. This format is required for COPY function codes 08h or 09h. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

SCFI devices may not support third-party COPY in which the copying SCFI device is not the source or destination device. Some SCFI devices only support COPY within the SCFI device and not to other SCFI devices. If an unsupported COPY operation is requested, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN PARAMETER DATA (see 7.2.3.1).

Table 7-x8: Segment Descriptor for COPY Function Codes 08h and 09h

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				Cat	Reserved		
1	Reserved							
2	(MSB) Stream Device Block Length							
3	(LSB)							
4	(MSB) Block Device Number of Blocks							
7	(LSB)							
8	(MSB) Block Device Logical Block Address							
11	(LSB)							
12	SICIDVal	DICIDVal	SInqVal	DInqVal	Reserved			
13	(MSB) Source ICID Name							
24	(LSB)							
25	Reserved							
26	Source SCFI Address							
27	Source LUN							
28	(MSB) Destination ICID Name							
39	(LSB)							
40	Reserved							
41	Destination SCFI Address							
42	Destination LUN							
43	(MSB) Source Inquiry Data							
78	(LSB)							
79	(MSB) Destination Inquiry Data							
114	(LSB)							
115	Reserved							

A catenate (Cat) bit (optional) of one indicates that the COPY manager shall catenate the last source block of a segment with the first source block of the next segment if the last source block does not end exactly at the end of the destination block. The definition of a cat bit of zero depends on the setting of

the pad bit in the command descriptor block (see 7.2.8.1, "Copies With Unequal Block Lengths" on page 7-27).

The stream device block-length field specifies the block length to be used on the stream device logical unit during this segment of the COPY command. If the SCFI device managing the COPY knows this block length is not supported, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN PARAMETER DATA . If the block length is found to be invalid while executing a read or write operation to the stream device, the command shall be terminated with CHECK CONDITION status and the sense key shall be set to COPY ABORTED (see 7.2.3.2).

The block device number of blocks field specifies the number of blocks in the current segment to be copied. A value of zero indicates that no blocks shall be transferred in this segment.

The block device logical block address field specifies the starting logical block address on the logical unit for this segment.

The source ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected source device is the correct one.

The source SCFI address and source LUN fields specify the SCFI bus ID and logical unit of the device to copy the data from for this segment of the COPY command.

The destination ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected destination device is the correct one.

The destination address and destination LUN fields specify the SCFI bus ID and logical unit to copy the data to for this segment of the COPY command. Some

The source INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the source matches the I/O process information.

The destination INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the destination matches the I/O process information.

7.2.8.9 COPY Function Code 0Ah

The format for the segment descriptors for COPY transfers among block devices is specified in Table 7-9. This format is required for COPY function code 0Ah. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-x9: Segment Descriptor for COPY Function Code 0Ah

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved			DC	Cat	Reserved		
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB) Source Device Block Length							
7								(LSB)
8	(MSB) Block Device Number of Blocks							
11								(LSB)
12	(MSB) Destination Device Logical Block Address							
15								(LSB)
16	SICIDVal	DICIDVal	SInqVal	DInqVal	Reserved			
17	(MSB) Source ICID Name							
28								(LSB)
29	Reserved							
30	Source SCFI Address							
31	Source LUN							
32	(MSB) Destination ICID Name							
43								(LSB)
44	Reserved							
45	Destination SCFI Address							
46	Destination LUN							
47	(MSB) Source Inquiry Data							
82								(LSB)
83	(MSB) Destination Inquiry Data							
118								(LSB)
119	Reserved							

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

A destination count (DC) bit of zero indicates that the number of blocks field refers to the source logical unit. A DC bit of one indicates that the number of blocks field refers to the destination logical unit.

The source logical block address field specifies the starting logical block address on the source block device.

The number of blocks field specifies the number of blocks to be transferred to or from (depending on the DC bit) the block device during this segment. A value of zero indicates that no blocks shall be transferred.

The destination logical block address field specifies the starting logical block address on the destination block device.

The source ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected source device is the correct one.

The source SCFI address and source LUN fields specify the SCFI bus ID and logical unit of the device to copy the data from for this segment of the COPY command.

The destination ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected destination device is the correct one.

The destination address and destination LUN fields specify the SCFI bus ID and logical unit to copy the data to for this segment of the COPY command. Some

The source INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the source matches the I/O process information.

The destination INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the destination matches the I/O process information.

7.2.8.10 COPY Function Code 0Bh

The format for the segment descriptors for COPY transfers among stream devices is specified by Table 7-10. This format is required for COPY function code 0Bh. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-x10: Segment Descriptor for COPY Function Code 0Bh

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved			DC	Cat	Reserved		
1	Reserved							
2	Reserved							
3	Reserved							
4	(MSB) Source Block Length							
5	(LSB)							
6	(MSB) Destination Block Length							
7	(LSB)							
8	(MSB) Number of Blocks							
11	(LSB)							
12	SICIDVal	DICIDVal	SInqVal	DInqVal	Reserved			
13	(MSB) Source ICID Name							
24	(LSB)							
25	Reserved							
26	Source SCFI Address							
27	Source LUN							
28	(MSB) Destination ICID Name							
39	(LSB)							
40	Reserved							
41	Destination SCFI Address							
42	Destination LUN							
43	(MSB) Source Inquiry Data							
78	(LSB)							
79	(MSB) Destination Inquiry Data							
114	(LSB)							
115	Reserved							

228

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

A destination count (DC) bit of zero indicates that the number of blocks field refers to the source logical unit. A DC bit of one indicates that the number of blocks field refers to the destination logical unit.

The source block length field specifies the block-length of the source device for this segment of the COPY. A zero in this field indicates variable block-length. For non-zero values, this field shall match the logical unit's actual block-length.

If block-length mismatches are detected prior to the beginning of the read operation by the SCFI device managing the COPY, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID FIELD IN PARAMETER DATA (see 7.2.3.1).

If the mismatches are detected during the read operation by the COPY manager, the command shall be terminated with CHECK CONDITION status. The sense key shall be set to COPY ABORTED (see 7.2.3.2). and the additional sense code shall be set to INVALID FIELD IN PARAMETER DATA.

The destination block-length field specifies the block length to be used on the destination logical unit during the COPY. Destination block length mismatches are handled in an analogous manner as source block length mismatches.

The number of blocks field specifies the number of blocks to be transferred to or from (depending on the DC bit) the device during this segment. A value of zero indicates that no blocks shall be transferred.

The source ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected source device is the correct one.

The source SCFI address and source LUN fields specify the SCFI bus ID and logical unit of the device to copy the data from for this segment of the COPY command.

The destination ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected destination device is the correct one.

The destination address and destination LUN fields specify the SCFI bus ID and logical unit to copy the data to for this segment of the COPY command. Some

The source INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the source matches the I/O process information.

The destination INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the destination matches the I/O process information.

7.2.8.11 COPY Function Code 0Ch

The format for the segment descriptors for image COPY transfers between sequential-access devices is specified in Table 7-11. This format is required for COPY function code 0Ch. The segment descriptor may be repeated up to 256 times within the command parameter data length specified in the command descriptor block.

Table 7-x11: Segment Descriptor for COPY Function Code 0Ch

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Count							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Vendor Specific							
11								
12	SICIDVal	DICIDVal	SInqVal	DInqVal	Reserved			
13	(MSB) Source ICID Name							
24								(LSB)
25	Reserved							
26	Source SCFI Address							
27	Source LUN							
28	(MSB) Destination ICID Name							
39								(LSB)
40	Reserved							
41	Destination SCFI Address							
42	Destination LUN							
43	(MSB) Source Inquiry Data							
78								(LSB)
79	(MSB) Destination Inquiry Data							
114								(LSB)
115	Reserved							

See 7.2.3.3 for definitions of the source address, the source LUN, the destination address, the destination LUN, and CAT fields.

The image mode COPY command copies an exact image of the source device medium to the destination device medium, beginning at their current positions. The copy function terminates when the source device:

(1) encounters an end-of-partition as defined by the source device

(2) encounters an end-of-data as defined by the source device (i.e., BLANK CHECK sense key)

(3) has copied the number of consecutive filemarks specified in the count field from the source device to the destination device

(4) has copied the number of consecutive setmarks specified in the count field from the source device to the destination device, if the RSmk bit in the device configuration page (see 9.3.3.1) is one.

A count field of zero indicates that the COPY command shall not terminate due to any number of consecutive filemarks or setmarks. Other error or exception conditions (e.g., early-warning end-of-partition on the destination device) may cause the COPY command to terminate prior to completion. In such cases, it is not possible to calculate a residue, so the information field in the sense data shall be set to zero.

The source ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected source device is the correct one.

The source SCFI address and source LUN fields specify the SCFI bus ID and logical unit of the device to copy the data from for this segment of the COPY command.

The destination ICID name field is provided when the SICIDVal bit is set to one. The copy manager may use the REPORT PATH STATUS command to verify the selected destination device is the correct one.

The destination address and destination LUN fields specify the SCFI bus ID and logical unit to copy the data to for this segment of the COPY command. Some

The source INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the source matches the I/O process information.

The destination INQUIRY data is provided to further permit the copy manager to verify that the LUN being defined as the destination matches the I/O process information.

7.2.9 COPY AND VERIFY Command

Table 7-18. COPY AND VERIFY Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code (3Ah)							
1	Reserved						BytChk	Pad
2	Reserved							
3 - 5	(MSB) Command Parameter Data Length (LSB)							
6	Reserved							
7	Reserved							
8	Reserved							
9	Control Field							

Command Attributes:

- Bypass Assignment: NO
- Bypass Reservation: NO
- Singular Command: NO
- Supervisor Command: NO
- Command Parameter Data: YES
- Command Response Data: NONE
- Logical Unit Status: Valid LUN, Other status N/A
- Target Routine Status: N/A

The COPY AND VERIFY command (Table 7-18) performs the same function as the COPY command, except that a verification of the data written to the destination logical unit is performed after the data is written. The command parameter data transferred to the target controller is the same as for the COPY command. This command parameter data contains the information to identify the logical units involved in the copy and the length of the copy. See 7.2.8, "COPY Command" on page 7-25 for additional information about the COPY command.

A byte check (BytChk) bit of zero causes a medium verification to be performed with no data comparison. A BytChk bit of one causes a byte-by-byte compare of data written on the destination medium and the data transferred from the source medium. If the compare is unsuccessful for any reason, the copy manager shall return CHECK CONDITION status with the sense key set to MISCOMPARE. The remaining fields in the sense data shall be set as documented in the COPY command.

The pad bit has the same function and meaning as in the COPY command. See 7.2.8.1, "Copies With Unequal Block Lengths" on page 7-27.

All command parameter data is as described in 7.2.8, "COPY Command" on page 7-25.

Part 4. SCFI Supplemental Material

Appendix A. I/O Process Examples A-1
A.1 Single Command Example A-1
A.2 Disconnect Example A-1
A.3 Linked Commands Example A-2
A.4 Queued I/O Process Example A-2
A.4.1 Typical Sequences for Tagged I/O Processes A-3
A.4.2 Tagged I/O Process Example A-3
A.5 Information Packet Examples A-5

Appendix A. I/O Process Examples

The following sections give examples of typical command processing in the SCFI environment. Multiple path operation is not described in these examples.

A.1 Single Command Example

An untagged I/O process containing one READ command is used in this section to illustrate a simple I/O process on the SCFI bus. This example does not include error or exception conditions. This example also assumes the parallel bus is being used. The initiator does not grant the target the privilege of disconnecting from the SCFI bus.

During the SELECTION phase, the initiator causes the attention condition inform the target that the initiator wishes to send an information packet. The target enters the NEXUS TRANSFER OUT phase and transfers an information packet from the initiator.

The first interface control prefix fields inform the target controller which logical unit is to be used. At this point, an H_I_T_C_L nexus has been established for the I/O process.

The first interface logical element in the information packet is a command descriptor block for the READ command. The target interprets the command. The target prepares an information packet with the data read from the logical unit. The target then switches to the NEXUS TRANSFER IN phase and transmits the information packet containing the data. This information packet also contains, after the data interface logical element, interface logical elements for status and messages. The message interface logical element contains the COMMAND COMPLETE message. After successfully sending the information packet, the target disconnects.

A.2 Disconnect Example

In the above single command example, the length of time necessary to obtain the data may require a time-consuming physical positioning operation. In order to improve bus performance, the target may be permitted to disconnect from the initiator, allowing other I/O processes to use the SCFI bus. In this case, the initiator grants the target the privilege of disconnecting from the SCFI bus.

After the target receives the information packet with the READ command, the target automatically disconnects from the SCFI bus. This is not an unexpected disconnect condition since the disconnect privilege was granted to the target.

After the target retrieves the requested data from the peripheral device it prepares an information packet with a logical block data interface logical element and then arbitrates using the same SCFI bus on which the connect was made. This information packet also contains, after the data, interface logical elements for status and messages. The message interface logical element contains the COMMAND COMPLETE message.

Upon winning arbitration, it selects the initiator and sends the information packet using the NEXUS TRANSFER IN phase. This revives the H_I_T_C_L nexus so that the initiator can process the information packet. After successfully sending the information packet, the target disconnects.

If the target wishes to disconnect after transferring part of the data (e.g., while crossing a cylinder boundary), it may do so by placing only part of the data in the information packet above and omitting

the status and message interface logical elements following the data. After successfully sending the information packet, the target disconnects. The initiator retains the current logical block data position.

One or more additional information packets containing the message and data interface logical elements is prepared and sent using the procedure described for the first information packet. Each information packet processed causes the initiator to update the logical block data position.

The last information packet may contain data, but it must contain the status and message interface logical elements.

On those occasions when an error or exception condition occurs and the target elects to repeat the information transfer, the target may repeat the some or all of the logical block data transfer by preparing an information packet containing a MODIFY DATA POINTERS message in the message interface logical element before the data element. At the end of the information packet processing, the logical block data position is updated.

A.3 Linked Commands Example

The initiating controller and target target controller prepare information packets with the interface logical elements in the correct order.

An I/O process may contain multiple commands "linked" together. Upon completing a linked command successfully, the target automatically proceeds to the next linked command for the I/O process. All commands in a series of linked commands are addressed to the same nexus and are part of a single I/O process.

The commands are not entirely independent. When using the relative address bit (see 8.1.10). the address of the last logical block accessed by one of the commands is available in the target controller to the next command. Thus one can search for a particular data pattern using a SEARCH DATA command and then read the logical block containing the data pattern with a READ command linked to the SEARCH DATA command. One can also read a logical block at a specified displacement from the block containing the data pattern.

A LINKED COMMAND COMPLETE or LINKED COMMAND COMPLETE (WITH FLAG) message is sent from the target to the initiator to indicate that a linked command completed. The initiator then updates the I/O process status so that subsequent transfers from the target controller reference the next command of the series. Command processing of linked and single commands is similar except that relative addressing is permitted in linked commands.

For example, a successful completion of a SEARCH DATA EQUAL command causes the target to continue with the linked READ command from the initiator. If the relative address bit in the READ command has been set to one, and the address field of the READ command is set to zero, the target transfers the successfully searched block to the initiator.

A.4 Queued I/O Process Example

This example of tagged I/O processes considers the execution of a number of commands each for a different I/O process. After each command, the state of the queued I/O process queue kept in the target controller is shown to indicate the function actually performed on the queue by the target controller. All examples shown in this section assume that multiple path mode is not being used. These examples also assume a parallel interface.

240

A.4.1 Typical Sequences for Tagged I/O Processes

An I/O process with tags uses the following sequences for normal execution. The initiator first arbitrates for the SCFI bus, and after successfully obtaining the SCFI bus, selects the appropriate SCFI device. The ATN signal is asserted during the SELECTION phase to indicate that a NEXUS TRANSFER OUT phase is required. The target then transfers the information packet and disconnects. The target controller places the command, identified by the H_C_L_Q nexus, at the appropriate place in the queued I/O process queue (e.g., by LUN by initiating controller).

When the target moves I/O processes to the active I/O process queue for execution, the target controller prepares the information packet(s) to transfer the appropriate data. The target begins an ARBITRATION phase and, upon winning, enters the SELECTION phase. After successful selection, the target sends the information packet to the initiator. The initiator uses the interface control prefix fields to identify the correct I/O process. ~transfer. The information packet contains GOOD status and a COMMAND COMPLETE message. This terminates the I/O process at the target controller. The initiating controller interprets the information packet, handles any logical block or command response data and termintates the active I/O process.

A.4.2 Tagged I/O Process Example

An example of the execution of five queued I/O processes demonstrates how tagged I/O processes operate. All tagged I/O processes are from one initiator to a single logical unit of a single target. The five I/O processes are defined in Figure A-1. The target is a direct-access device. At the time the I/O processes are being considered for execution, it is assumed that the actuator is in position to access logical block 10000.

Commands in Order Received by Target Controller

Queued I/O Process Queue

Command	Init	Queue Tag Type	Queue Tag Value	Logical Block Address	Transfer Length	Status
READ	1-t0	SIMPLE	01h	10000	1000	Queued
READ	1-t1	SIMPLE	02h	100	1	Queued
READ	1-t2	ORDERED	03h	1000	1000	Queued
READ	1-t3	SIMPLE	04h	10000	1	Queued
READ	1-t4	SIMPLE	05h	2000	1000	Queued

Figure A-1. QIOPQ in Order Received by Target Controller

An optimizing algorithm might require that blocks close to the actuator position be the first blocks accessed, followed by those increasingly farther from the actuator position. However, the I/O process with queue tag 03h has an ordered queue tag. All simple queue tag I/O processes received before the ordered queue tag must be executed first, while all I/O processes with simple tags transferred after the ordered queue tag I/O process must be executed after the ordered queue tag I/O process.

If a target supports an optimizing algorithm, the actual order in which the I/O processes are executed may be as shown in Figure A-2 on page A-4.

24/

Commands in Order of Execution

Queued I/O Process Queue

Command	Init	Queue Tag Type	Queue Tag Value	Logical Block Address	Transfer Length	Status
READ	1-t0	SIMPLE	01h	10000	1000	Queued
READ	1-t1	SIMPLE	02h	100	1	Queued
READ	1-t2	ORDERED	03h	1000	1000	Queued
READ	1-t4	SIMPLE	05h	2000	1000	Queued
READ	1-t3	SIMPLE	04h	10000	1	Queued

Figure A-2. QIOPQ in Modified by Reordering

I/O processes with queue tag values 01h and 02h are executed in the order received since the actuator is already in position to execute I/O process 01h. I/O process 02h must be executed before I/O process 04h or 05h because the ordered I/O process 03h was transmitted after I/O processes 01h and 02h but before I/O processes 04h and 05h. I/O process 03h is then executed after I/O process 02h. I/O process 05h is executed next because the actuator is in position to access block 2000. I/O process 04h is executed last.

As an example of the operation of the head of queue tag I/O process, consider that a new I/O process, identified by a head of queue tag of 08h, is transmitted to the target while the ordered I/O process 03h is being executed. I/O processes 01h and 02h have completed. I/O process 03h continues execution, but the new head of queue tat I/O process is placed in the queue for execution next. In this case, the queued I/O process queue for execution after ordered I/O process 03h completes would appear as shown in Figure A-3 on page A-5.

Modified by Head of Queue Tag

Active I/O Process Queue

Command	Init	Queue Tag Type	Queue Tag Value	Logical Block Address	Transfer Length	Status
READ	1-t2	ORDERED	03h	1000	1000	Active

Queued I/O Process Queue

Command	Init	Queue Tag Type	Queue Tag Value	Logical Block Address	Transfer Length	Status
READ	1-t5	HOQ	08h	0	8	Queued
READ	1-t4	SIMPLE	05h	2000	1000	Queued
READ	1-t3	SIMPLE	04h	10000	1	Queued

Figure A-3. QIOPQ Modified by Head of Queue Tag

To obtain maximum performance gains using tagged I/O processes requires careful implementation of the queuing algorithms in the target controller. In addition, initiating controllers should allow a maximum number of simple I/O processes to be executed with a minimum number of ordered I/O processes. RESERVE and RELEASE commands, SET LIMITS commands, and appropriate software locking conventions should be used to guarantee the proper relationship between the commands executed and the data stored on the peripheral devices. These conventions are not defined by this standard.

A.5 Information Packet Examples

The following two information packets comprise a complete I/O process to cause a rewind command to be executed for a sequential access device. The first information packet is a request for a sequential device to perform a rewind operation.

The total exchange requires 56 bytes to be transferred. Each information packet is 28 bytes long. The complete I/O process requires only two information transfer phases. For parallel interfaces, the target controller would normally be granted permission to disconnect. For serial interfaces, the disconnect privilege is mandatory.

Interface Control Prefix Fields (Parallel)								
Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	Packet Length 0000001Ch							(LSB)
2	Packet Type 00H (Initiate I/O Process)							
3	Reserved 00h							
4	LUNTRN Valid 1b	LUNTRN 0b	QNexus 0b	HOQ 0b	OrdSim 0b	Disc Priv 1b	Pad Byte Count 10b	
5	Mlt Path 0b	Susp Mpth 0b	Enb Spvr 0b	Reserved 000000b				
6	Initiating Controller Port Number 00h							
7	Reserved 00h							
8	Original Initiator SCFI Address 07h							
9	Reserved 00h							
10	Original Target SCFI Address 01h							
11	Target Controller Port Number 04h							
12	LUNTRNID 00h							
13	Queue tag 00h							
14-15	Reserved 00000000h							

Figure A-4. Initial Information Packet Example (Parallel 1 of 3)

Byte	CDB Interface Logical Element
0	
	--- Element Length ---
1	000Ah
2	Element Type
	01h
3	Reserved
	001h
4-9	Element Bytes
	01h (Sequential Rewind)
	00h
	00h
	00h
	00h
	00h

Figure A-5. Initial Information Packet Example (Parallel 2 of 3)

Byte	Interface Control Suffix (Parallel)
-2	
	--- Pad Bytes ---
-1	0000h

Figure A-6. Initial Information Packet Example (Parallel 3 of 3)

The sequential device disconnects, interprets the information packet and carries out the rewind operation. It then sends the following information packet during a reconnection. The information packet is 28 bytes long and includes the interface control prefix fields, the status interface logical element and the COMMAND complete message interface logical element.

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	Packet Length 0000001Ch							(LSB)
2	Packet Type 01h (Terminate I/O Process)							
3	Reserved 00h							
4	LUNTRN Valid 1b	LUNTRN 0b	QNexus 0b	HOQ 0b	OrdSim 0b	Disc Priv 1b	Pad Byte Count 10b	
5	Mlt Path 0b	Susp Mpth 0b	Enb Spvr 0b	Reserved 000000b				
6	Initiating Controller Port Number 00h (Internal Port Number)							
7	Reserved 00h							
8	Original Initiator SCFI Address 07h							
9	Reserved 00h							
10	Original Target SCFI Address 01h							
11	Target Controller Port Number 04h (Internal Port Number)							
12	LUNTRNID 00h							
13	Queue tag 00h							
14-15	Reserved 00000000h							

Figure A-7. Response Information Packet Example (Parallel 1 of 4)

Byte	Status Interface Logical Element
0	
	Element Length
1	0005h
2	Element Type 05h
3	Reserved 00h
4	Element Bytes 00h (Good Status)

Figure A-8. Response Information Packet Example (Parallel 2 of 4)

Byte	Message Interface Logical Element
0	
	Element Length
1	0005h
2	Element Type 00h
3	Reserved 00h
4	Element Bytes 00h (Command Complete)

Figure A-9. Response Information Packet Example (Parallel 3 of 4)

Byte	Interface Control Suffix (Parallel)
-2	
	Pad Bytes
-1	0000h

Figure A-10. Response Information Packet Example (Parallel 4 of 4)

END OF SUPPLEMENTAL MATERIAL

INDEX

END OF DOCUMENT X3T9.2/90-132 R03