

Working Draft American National Standard

Project T10/1760-D

Revision 13
16 November 2007

Information technology - Serial Attached SCSI - 2 (SAS-2)

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (International Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.

Permission is granted to members of INCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of INCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

T10 Technical Editor: Robert C Elliott
Hewlett-Packard Corporation
MC 140801
PO Box 692000
Houston, TX 77269-2000
USA

Telephone: 281-518-5037
Email: elliott@hp.com

Reference number
ISO/IEC 14776-152:200x
ANSI INCITS ***-200x

Points of Contact

International Committee for Information Technology Standards (INCITS) T10 Technical Committee

T10 Chair

John B. Lohmeyer
LSI Corporation
4420 ArrowsWest Drive
Colorado Springs, CO 80907-3444
USA

Telephone: (719) 533-7560
Email: lohmeyer@t10.org

T10 Web Site: <http://www.t10.org>

T10 E-mail reflector:

Server: majordomo@t10.org
To subscribe send e-mail with 'subscribe' in message body
To unsubscribe send e-mail with 'unsubscribe' in message body

T10 Vice-Chair

George O. Penokie
IBM Corporation
MS: 2C6
3605 Highway 52 N
Rochester, MN 55901
USA

Telephone: (507) 253-5208
Email: gop@us.ibm.com

INCITS Secretariat

Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

Telephone: 202-737-8888
Web site: <http://www.incits.org>
Email: incits@itic.org

Information Technology Industry Council

Web site: <http://www.itic.org>

Document Distribution

INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site: <http://www.techstreet.com/incits.html>
Telephone: (734) 302-7801 or (800) 699-9277

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704
USA

Web site: <http://global.ihs.com>
Telephone: (303) 397-7956 or (303) 792-2181 or (800) 854-7179

American National Standard
for Information Technology

Serial Attached SCSI - 2 (SAS-2)

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

This standard specifies the functional requirements for the Serial Attached SCSI (SAS) physical interconnect, which is compatible with the Serial ATA physical interconnect. It also specifies three transport protocols, one to transport SCSI commands, another to transport Serial ATA commands to multiple SATA devices, and a third to support interface management. This standard is intended to be used in conjunction with SCSI and ATA command set standards.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, notice of one or more claims has been received.

By publication of this standard, no position is taken with respect to the validity of this claim or of any rights in connection therewith. The known patent holder(s) has (have), however, filed a statement of willingness to grant a license under these rights on reasonable and nondiscriminatory terms and conditions to applicants desiring to obtain such a license. Details may be obtained from the publisher.

No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by

**American National Standards Institute
11 W. 42nd Street, New York, New York 10036**

Copyright © 2007 by Information Technology Industry Council (ITI). All rights reserved.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission of ITI, 1250 Eye Street NW, Suite 200, Washington, DC 20005.

Printed in the United States of America

Contents

	Page
1 Scope	1
2 Normative references	3
2.1 Normative references	3
2.2 Approved references	3
2.3 References under development	3
2.4 Other references	4
3 Definitions, symbols, abbreviations, keywords, and conventions	6
3.1 Definitions	6
3.2 Symbols and abbreviations	22
3.3 Keywords	25
3.4 Editorial conventions	26
3.5 Class diagram and object diagram conventions	27
3.6 State machine conventions	33
3.6.1 State machine conventions overview	33
3.6.2 Transitions	33
3.6.3 Messages, requests, indications, confirmations, responses, and event notifications	34
3.6.4 State machine counters, timers, and variables	34
3.6.5 State machine arguments	34
3.7 Bit and byte ordering	35
3.8 Notation for procedures and functions	35
4 General	36
4.1 Architecture	36
4.1.1 Architecture overview	36
4.1.2 Physical links and phys	37
4.1.3 Logical links	42
4.1.4 Ports (narrow ports and wide ports)	42
4.1.5 Application clients and device servers	45
4.1.6 SAS devices	46
4.1.7 Expander devices	47
4.1.8 Service delivery subsystem	48
4.1.9 Domains	48
4.1.10 Expander device topologies	51
4.1.10.1 Expander device topology overview	51
4.1.10.2 Expander device topologies	51
4.1.11 Pathways	53
4.1.12 Connections	53
4.1.13 Broadcasts	55
4.2 Names and identifiers	57
4.2.1 Names and identifiers overview	57
4.2.2 NAA IEEE Registered format identifier	58
4.2.3 Hashed SAS addresses	59
4.2.4 Device names and expander device SAS addresses	59
4.2.5 Device name for SATA devices with world wide names	60
4.2.6 Port names	60
4.2.7 Port identifiers and SAS port SAS addresses	60
4.2.8 Phy identifiers	61
4.3 State machines	61
4.3.1 State machine overview	61
4.3.2 Transmit data path	62
4.3.3 Receive data path	67
4.3.4 State machines and SAS Device, SAS Port, and SAS Phy classes	70

4.4 Resets	71
4.4.1 Reset overview	71
4.4.2 Hard reset	73
4.4.2.1 Hard reset overview	73
4.4.2.2 Additional hard reset processing by SAS ports	73
4.4.2.3 Additional hard reset processing by expander ports	73
4.5 I_T nexus loss	73
4.6 Expander device model	74
4.6.1 Expander device model overview	74
4.6.2 Expander ports	75
4.6.3 Expander connection manager (ECM)	76
4.6.4 Expander connection router (ECR)	76
4.6.5 Broadcast propagation processor (BPP)	77
4.6.6 Expander device interfaces	77
4.6.6.1 Expander device interface overview	77
4.6.6.2 Expander device interfaces detail	79
4.6.6.3 ECM interface	80
4.6.6.4 ECR interface	81
4.6.6.5 BPP interface	82
4.6.7 Expander device routing	84
4.6.7.1 Routing attributes and routing methods	84
4.6.7.2 Connection request routing	84
4.6.7.3 Expander route table	85
4.6.7.3.1 Expander route table overview	85
4.6.7.3.2 Phy-based expander route table	86
4.6.7.3.3 Expander-based expander route table	87
4.6.8 Expander device reduced functionality	87
4.7 Discover process	88
4.7.1 Discover process overview	88
4.7.2 Discover process in a self-configuring expander device	90
4.7.3 Enabling multiplexing	91
4.8 Configuration subprocess	91
4.8.1 Configuration subprocess overview	91
4.8.2 Allowed topologies	92
4.8.3 Route table optimization	93
4.8.4 Expander route index order	94
4.9 Zoning	99
4.9.1 Zoning overview	99
4.9.2 Zoning expander device requirements	103
4.9.3 Zoning operation	105
4.9.3.1 Zone phy information	105
4.9.3.2 Zone groups	108
4.9.3.3 Zone permission table	108
4.9.3.4 Zoning expander route table	110
4.9.3.5 Source zone group and destination zone group determination	110
4.9.4 Zone phy information and link reset sequences	112
4.9.5 Broadcast processing in a zoning expander device with zoning enabled	114
4.9.6 Zone configuration	115
4.9.6.1 Zone configuration overview	115
4.9.6.2 Lock step	115
4.9.6.3 Load step	116
4.9.6.4 Activate step	116
4.9.6.5 Unlock step	117
4.9.6.6 Zone lock inactivity timer	117
4.9.6.7 Enable a zoning expander device	118
4.10 Phy test functions	118
4.11 Phy events	118

5 Physical layer	123
5.1 Physical layer overview	123
5.2 Passive interconnect	123
5.2.1 SATA connectors and cable assemblies	123
5.2.2 SAS connectors and cables	123
5.2.3 Connectors	128
5.2.3.1 Connectors overview	128
5.2.3.2 SAS internal connectors	129
5.2.3.2.1 SAS Drive connectors	129
5.2.3.2.1.1 SAS Drive plug connector	129
5.2.3.2.1.2 SAS Drive cable receptacle connector	129
5.2.3.2.1.3 SAS Drive backplane receptacle connector	130
5.2.3.2.1.4 SAS Drive connector pin assignments	131
5.2.3.2.2 SAS 4i connectors	133
5.2.3.2.2.1 SAS 4i cable receptacle connector	133
5.2.3.2.2.2 SAS 4i plug connector	133
5.2.3.2.2.3 SAS 4i connector pin assignments	134
5.2.3.2.3 Mini SAS 4i connectors	136
5.2.3.2.3.1 Mini SAS 4i cable plug connector	136
5.2.3.2.3.2 Mini SAS 4i receptacle connector	136
5.2.3.2.3.3 Mini SAS 4i connector pin assignments	137
5.2.3.3 SAS external connectors	139
5.2.3.3.1 SAS 4x connectors	139
5.2.3.3.1.1 SAS 4x cable plug connector	139
5.2.3.3.1.2 SAS 4x receptacle connector	139
5.2.3.3.1.3 SAS 4x connector pin assignments	141
5.2.3.3.2 Mini SAS 4x connectors	141
5.2.3.3.2.1 Mini SAS 4x cable plug connector	141
5.2.3.3.2.2 Mini SAS 4x receptacle connector	143
5.2.3.3.2.3 Mini SAS 4x connector pin assignments	146
5.2.4 Cable assemblies	146
5.2.4.1 SAS internal cable assemblies	146
5.2.4.1.1 SAS Drive cable assemblies	146
5.2.4.1.2 SAS internal symmetric cable assemblies	147
5.2.4.1.2.1 SAS internal symmetric cable assemblies overview	147
5.2.4.1.2.2 SAS internal symmetric cable assembly - SAS 4i	148
5.2.4.1.2.3 SAS internal symmetric cable assembly - Mini SAS 4i	149
5.2.4.1.2.4 SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i with vendor-specific sidebands	150
5.2.4.1.2.5 SAS internal symmetric cable assembly - SAS 4i controller to Mini SAS 4i backplane with SGPIO	152
5.2.4.1.2.6 SAS internal symmetric cable assembly - Mini SAS 4i controller to SAS 4i backplane with SGPIO	153
5.2.4.1.3 SAS internal fanout cable assemblies	154
5.2.4.1.3.1 SAS internal fanout cable assemblies overview	154
5.2.4.1.3.2 SAS internal controller-based fanout cable assemblies	155
5.2.4.1.3.3 SAS internal backplane-based fanout cable assemblies	157
5.2.4.2 SAS external cable assemblies	158
5.2.4.2.1 SAS external cable assemblies overview	158
5.2.4.2.2 SAS external cable assembly - SAS 4x	159
5.2.4.2.3 SAS external cable assembly - Mini SAS 4x	160
5.2.4.2.4 SAS external cable assembly - SAS 4x to Mini SAS 4x	162
5.2.5 Backplanes	162
5.2.6 Cable assembly and backplane specifications	163
5.3 Transmitter and receiver device electrical characteristics	168
5.3.1 Compliance points	168
5.3.2 Test loads	175

5.3.2.1 Test loads overview.....	175
5.3.2.2 Zero-length test load	176
5.3.2.3 TCTF test load.....	177
5.3.2.4 Low-loss TCTF test load	181
5.3.3 General electrical characteristics	182
5.3.4 Transmitter and receiver device transients	186
5.3.5 Eye masks	186
5.3.5.1 Eye masks overview.....	186
5.3.5.2 Transmitter device eye mask	187
5.3.5.3 Receiver device eye mask	187
5.3.5.4 Receiver device jitter tolerance eye mask.....	188
5.3.6 Transmitter device characteristics	189
5.3.6.1 Transmitter device characteristics overview.....	189
5.3.6.2 Transmitter device signal output characteristics as measured with the zero-length test load...	190
5.3.6.3 Transmitter device signal output characteristics as measured with each test load	190
5.3.6.4 Transmitter device maximum jitter	192
5.3.6.5 Transmitter device signal output levels for OOB signals	192
5.3.7 Receiver device characteristics	193
5.3.7.1 Receiver device characteristics overview.....	193
5.3.7.2 Delivered signal (receiver device signal tolerance) characteristics	194
5.3.7.3 Maximum delivered jitter	196
5.3.7.4 Receiver device jitter tolerance	196
5.3.8 Spread spectrum clocking (SSC).....	197
5.3.8.1 Spread spectrum clocking overview.....	197
5.3.8.2 Transmitter SSC modulation	199
5.3.8.3 Receiver SSC modulation tolerance	200
5.3.8.4 Expander device center-spreading tolerance buffer.....	200
5.3.9 Non-tracking clock architecture.....	201
5.4 READY LED signal electrical characteristics.....	201
6 Phy layer	203
6.1 Phy layer overview	203
6.2 8b10b coding	203
6.2.1 8b10b coding overview	203
6.2.2 8b10b coding introduction.....	203
6.2.3 8b10b coding notation conventions	203
6.3 Character encoding and decoding.....	204
6.3.1 Introduction	204
6.3.2 Transmission order	204
6.3.3 Data and control characters.....	205
6.3.4 Encoding characters in the transmitter	211
6.3.5 Decoding characters in the receiver	212
6.4 Dwords, primitives, data dwords, and invalid dwords.....	212
6.5 Bit order	212
6.6 Out of band (OOB) signals	214
6.6.1 OOB signals overview.....	214
6.6.2 Transmitting OOB signals.....	216
6.6.3 Receiving OOB signals	218
6.6.4 Transmitting the SATA port selection signal.....	220
6.7 Phy reset sequences.....	220
6.7.1 Phy reset sequences overview	220
6.7.2 SATA phy reset sequence	221
6.7.2.1 SATA OOB sequence	221
6.7.2.2 SATA speed negotiation sequence	222
6.7.3 SAS to SATA phy reset sequence	222
6.7.4 SAS to SAS phy reset sequence	224
6.7.4.1 SAS OOB sequence.....	224

6.7.4.2 SAS speed negotiation sequence	226
6.7.4.2.1 SAS speed negotiation sequence overview	226
6.7.4.2.2 SAS speed negotiation sequence timing specifications	227
6.7.4.2.3 Speed negotiation window (SNW) definitions	228
6.7.4.2.3.1 SNW definitions overview	228
6.7.4.2.3.2 SNW-1, SNW-2, and Final-SNW	228
6.7.4.2.3.3 SNW-3	229
6.7.4.2.3.4 Train-SNW	234
6.7.4.2.4 SAS speed negotiation sequence	237
6.7.4.3 Multiplexing sequence	244
6.7.5 Phy reset sequence after devices are attached	245
6.8 SP (phy layer) state machine	246
6.8.1 SP state machine overview	246
6.8.2 SP transmitter and receiver	248
6.8.3 OOB sequence states	250
6.8.3.1 OOB sequence states overview	250
6.8.3.2 SP0:OOB_COMINIT state	251
6.8.3.2.1 State description	251
6.8.3.2.2 Transition SP0:OOB_COMINIT to SP1:OOB_AwaitCOMX	251
6.8.3.2.3 Transition SP0:OOB_COMINIT to SP3:OOB_AwaitCOMINIT_Sent	252
6.8.3.2.4 Transition SP0:OOB_COMINIT to SP4:OOB_COMSAS	252
6.8.3.3 SP1:OOB_AwaitCOMX state	252
6.8.3.3.1 State description	252
6.8.3.3.2 Transition SP1:OOB_AwaitCOMX to SP0:OOB_COMINIT	252
6.8.3.3.3 Transition SP1:OOB_AwaitCOMX to SP4:OOB_COMSAS	252
6.8.3.4 SP2:OOB_NoCOMSASTimeout state	252
6.8.3.4.1 State description	252
6.8.3.4.2 Transition SP2:OOB_NoCOMSASTimeout to SP0:OOB_COMINIT	253
6.8.3.4.3 Transition SP2:OOB_NoCOMSASTimeout to SP4:OOB_COMSAS	253
6.8.3.5 SP3:OOB_AwaitCOMINIT_Sent state	253
6.8.3.5.1 State description	253
6.8.3.5.2 Transition SP3:OOB_AwaitCOMINIT_Sent to SP4:OOB_COMSAS	253
6.8.3.6 SP4:OOB_COMSAS state	253
6.8.3.6.1 State description	253
6.8.3.6.2 Transition SP4:OOB_COMSAS to SP5:OOB_AwaitCOMSAS_Sent	253
6.8.3.6.3 Transition SP4:OOB_COMSAS to SP6:OOB_AwaitNoCOMSAS	253
6.8.3.6.4 Transition SP4:OOB_COMSAS to SP7:OOB_AwaitCOMSAS	254
6.8.3.7 SP5:OOB_AwaitCOMSAS_Sent state	254
6.8.3.7.1 State description	254
6.8.3.7.2 Transition SP5:OOB_AwaitCOMSAS_Sent to SP6:OOB_AwaitNoCOMSAS	254
6.8.3.8 SP6:OOB_AwaitNoCOMSAS state	254
6.8.3.8.1 State description	254
6.8.3.8.2 Transition SP6:OOB_AwaitNoCOMSAS to SP0:OOB_COMINIT	254
6.8.3.8.3 Transition SP6:OOB_AwaitNoCOMSAS to SP8:SAS_Start	254
6.8.3.9 SP7:OOB_AwaitCOMSAS state	254
6.8.3.9.1 State description	254
6.8.3.9.2 Transition SP7:OOB_AwaitCOMSAS to SP2:OOB_NoCOMSASTimeout	254
6.8.3.9.3 Transition SP7:OOB_AwaitCOMSAS to SP6:OOB_AwaitNoCOMSAS	254
6.8.3.9.4 Transition SP7:OOB_AwaitCOMSAS to SP16:SATA_COMWAKE	255
6.8.3.9.5 Transition SP7:OOB_AwaitCOMSAS to SP26:SATA_SpinupHold	255
6.8.4 SAS speed negotiation states	255
6.8.4.1 SAS speed negotiation states overview	255
6.8.4.2 SP8:SAS_Start state	257
6.8.4.2.1 State description	257
6.8.4.2.2 Transition SP8:SAS_Start to SP0:OOB_COMINIT	258
6.8.4.2.3 Transition SP8:SAS_Start to SP1:OOB_AwaitCOMX	258
6.8.4.2.4 Transition SP8:SAS_Start to SP9:SAS_WindowNotSupported	258

6.8.4.2.5 Transition SP8:SAS_Start to SP10:SAS_AwaitALIGN	258
6.8.4.2.6 Transition SP8:SAS_Start to SP27:SAS_Settings	258
6.8.4.3 SP9:SAS_WindowNotSupported state	259
6.8.4.3.1 State description	259
6.8.4.3.2 Transition SP9:SAS_WindowNotSupported to SP14:SAS_Fail	259
6.8.4.4 SP10:SAS_AwaitALIGN state	259
6.8.4.4.1 State description	259
6.8.4.4.2 Transition SP10:SAS_AwaitALIGN to SP0:OOB_COMINIT	259
6.8.4.4.3 Transition SP10:SAS_AwaitALIGN to SP11:SAS_AwaitALIGN1	259
6.8.4.4.4 Transition SP10:SAS_AwaitALIGN to SP12:SAS_AwaitSNW	259
6.8.4.4.5 Transition SP10:SAS_AwaitALIGN to SP14:SAS_Fail	259
6.8.4.5 SP11:SAS_AwaitALIGN1 state	259
6.8.4.5.1 State description	259
6.8.4.5.2 Transition SP11:SAS_AwaitALIGN1 to SP0:OOB_COMINIT	259
6.8.4.5.3 Transition SP11:SAS_AwaitALIGN1 to SP12:SAS_AwaitSNW	260
6.8.4.5.4 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_Fail	260
6.8.4.6 SP12:SAS_AwaitSNW state	260
6.8.4.6.1 State description	260
6.8.4.6.2 Transition SP12:SAS_AwaitSNW to SP0:OOB_COMINIT	260
6.8.4.6.3 Transition SP12:SAS_AwaitSNW to SP13:SAS_Pass	260
6.8.4.7 SP13:SAS_Pass state	260
6.8.4.7.1 State description	260
6.8.4.7.2 Transition SP13:SAS_Pass to SP0:OOB_COMINIT	260
6.8.4.7.3 Transition SP13:SAS_Pass to SP8:SAS_Start	260
6.8.4.7.4 Transition SP13:SAS_Pass to SP15:SAS_PHY_Ready	260
6.8.4.8 SP14:SAS_Fail state	261
6.8.4.8.1 State description	261
6.8.4.8.2 Transition SP14:SAS_Fail to SP1:OOB_AwaitCOMX	261
6.8.4.8.3 Transition SP14:SAS_Fail to SP8:SAS_Start	261
6.8.4.9 SP15:SAS_PHY_Ready state	261
6.8.4.9.1 State description	261
6.8.4.9.2 Transition SP15:SAS_PHY_Ready to SP0:OOB_COMINIT	261
6.8.4.10 SP27:SAS_Settings state	261
6.8.4.10.1 State description	261
6.8.4.10.2 Transition SP27:SAS_Settings to SP0:OOB_COMINIT	262
6.8.4.10.3 Transition SP27:SAS_Settings to SP1:OOB_AwaitCOMX	262
6.8.4.10.4 Transition SP27:SAS_Settings to SP8:SAS_Start	262
6.8.4.10.5 Transition SP27:SAS_Settings to SP28:SAS_TrainSetup	262
6.8.4.11 SP28:SAS_TrainSetup	262
6.8.4.11.1 State description	262
6.8.4.11.2 Transition SP28:SAS_TrainSetup to SP0:OOB_COMINIT	263
6.8.4.11.3 Transition SP28:SAS_TrainSetup to SP29:SAS_Train	263
6.8.4.12 SP29:SAS_Train state	263
6.8.4.12.1 State description	263
6.8.4.12.2 Transition SP29:SAS_Train to SP0:OOB_COMINIT	263
6.8.4.12.3 Transition SP29:SAS_Train to SP1:OOB_AwaitCOMX	263
6.8.4.12.4 Transition SP29:SAS_Train to SP30:SAS_TrainingDone	263
6.8.4.12.5 Transition SP29:SAS_Train to SP28:SAS_TrainSetup	263
6.8.4.13 SP30:SAS_TrainingDone state	263
6.8.4.13.1 State description	263
6.8.4.13.2 Transition SP30:SAS_TrainingDone to SP0:OOB_COMINIT	264
6.8.4.13.3 Transition SP30:SAS_TrainingDone to SP1:OOB_AwaitCOMX	264
6.8.4.13.4 Transition SP30:SAS_TrainingDone to SP28:SAS_TrainSetup	264
6.8.4.13.5 Transition SP30:SAS_TrainingDone to SP15:SAS_PHY_Ready	264
6.8.5 SATA host emulation states	264
6.8.5.1 SATA host emulation states overview	264
6.8.5.2 SP16:SATA_COMWAKE state	266

6.8.5.2.1 State description	266
6.8.5.2.2 Transition SP16:SATA_COMWAKE to SP17:SATA_AwaitCOMWAKE	266
6.8.5.3 SP17:SATA_AwaitCOMWAKE state	266
6.8.5.3.1 State description	266
6.8.5.3.2 Transition SP17:SATA_AwaitCOMWAKE to SP0:OOB_COMINIT	266
6.8.5.3.3 Transition SP17:SATA_AwaitCOMWAKE to SP18:SATA_AwaitNoCOMWAKE	266
6.8.5.4 SP18:SATA_AwaitNoCOMWAKE state	266
6.8.5.4.1 State description	266
6.8.5.4.2 Transition SP18:SATA_AwaitNoCOMWAKE to SP0:OOB_COMINIT	266
6.8.5.4.3 Transition SP18:SATA_AwaitNoCOMWAKE to SP19:SATA_AwaitALIGN	266
6.8.5.5 SP19:SATA_AwaitALIGN state	266
6.8.5.5.1 State description	266
6.8.5.5.2 Transition SP19:SATA_AwaitALIGN to SP0:OOB_COMINIT	266
6.8.5.5.3 Transition SP19:SATA_AwaitALIGN to SP20:SATA_AdjustSpeed	267
6.8.5.6 SP20:SATA_AdjustSpeed state	267
6.8.5.6.1 State description	267
6.8.5.6.2 Transition SP20:SATA_AdjustSpeed to SP0:OOB_COMINIT	267
6.8.5.6.3 Transition SP20:SATA_AdjustSpeed to SP21:SATA_TransmitALIGN	267
6.8.5.7 SP21:SATA_TransmitALIGN state	267
6.8.5.7.1 State description	267
6.8.5.7.2 Transition SP21:SATA_TransmitALIGN to SP0:OOB_COMINIT	267
6.8.5.7.3 Transition SP21:SATA_TransmitALIGN to SP22:SATA_PHY_Ready	267
6.8.5.8 SP22:SATA_PHY_Ready state	267
6.8.5.8.1 State description	267
6.8.5.8.2 Transition SP22:SATA_PHY_Ready to SP0:OOB_COMINIT	268
6.8.5.8.3 Transition SP22:SATA_PHY_Ready to SP23:SATA_PM_Partial	268
6.8.5.8.4 Transition SP22:SATA_PHY_Ready to SP24:SATA_PM_Slumber	268
6.8.5.9 SP23:SATA_PM_Partial state	268
6.8.5.9.1 State description	268
6.8.5.9.2 Transition SP23:SATA_PM_Partial to SP0:OOB_COMINIT	268
6.8.5.9.3 Transition SP23:SATA_PM_Partial to SP16:SATA_COMWAKE	268
6.8.5.9.4 Transition SP23:SATA_PM_Partial to SP18:SATA_AwaitNoCOMWAKE	268
6.8.5.10 SP24:SATA_PM_Slumber state	268
6.8.5.10.1 State description	268
6.8.5.10.2 Transition SP24:SATA_PM_Slumber to SP0:OOB_COMINIT	268
6.8.5.10.3 Transition SP24:SATA_PM_Slumber to SP16:SATA_COMWAKE	269
6.8.5.10.4 Transition SP24:SATA_PM_Slumber to SP18:SATA_AwaitNoCOMWAKE	269
6.8.6 SATA port selector state SP25:SATA_PortSel	269
6.8.6.1 State description	269
6.8.6.2 Transition SP25:SATA_PortSel to SP1:OOB_AwaitCOMX	269
6.8.7 SATA spinup hold state SP26:SATA_SpinupHold	270
6.8.7.1 State description	270
6.8.7.2 Transition SP26:SATA_SpinupHold to SP0:OOB_COMINIT	270
6.9 SP_DWS (phy layer dword synchronization) state machine	270
6.9.1 SP_DWS state machine overview	270
6.9.2 SP_DWS receiver	272
6.9.3 SP_DWS0:AcquireSync state	273
6.9.3.1 State description	273
6.9.3.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1	273
6.9.4 SP_DWS1:Valid1 state	273
6.9.4.1 State description	273
6.9.4.2 Transition SP_DWS1:Valid1 to SP_DWS0:AcquireSync	274
6.9.4.3 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2	274
6.9.5 SP_DWS2:Valid2 state	274
6.9.5.1 State description	274
6.9.5.2 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync	274
6.9.5.3 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired	274

6.9.6 SP_DWS3:SyncAcquired state	274
6.9.6.1 State description	274
6.9.6.2 Transition SP_DWS3:SyncAcquired to SP_DWS0:AcquireSync	274
6.9.6.3 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1	274
6.9.7 SP_DWS4:Lost1 state	274
6.9.7.1 State description	274
6.9.7.2 Transition SP_DWS4:Lost1 to SP_DWS0:AcquireSync	274
6.9.7.3 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered	275
6.9.7.4 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2	275
6.9.8 SP_DWS5:Lost1Recovered state	275
6.9.8.1 State description	275
6.9.8.2 Transition SP_DWS5:Lost1Recovered to SP_DWS0:AcquireSync	275
6.9.8.3 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired	275
6.9.8.4 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2	275
6.9.9 SP_DWS6:Lost2 state	275
6.9.9.1 State description	275
6.9.9.2 Transition SP_DWS6:Lost2 to SP_DWS0:AcquireSync	275
6.9.9.3 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered	275
6.9.9.4 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3	275
6.9.10 SP_DWS7:Lost2Recovered state	275
6.9.10.1 State description	275
6.9.10.2 Transition SP_DWS7:Lost2Recovered to SP_DWS0:AcquireSync	276
6.9.10.3 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1	276
6.9.10.4 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3	276
6.9.11 SP_DWS8:Lost3 state	276
6.9.11.1 State description	276
6.9.11.2 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync	276
6.9.11.3 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered	276
6.9.12 SP_DWS9:Lost3Recovered state	276
6.9.12.1 State description	276
6.9.12.2 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync	276
6.9.12.3 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2	276
6.10 Multiplexing	276
6.11 Spin-up	278
7 Link layer	279
7.1 Link layer overview	279
7.2 Primitives	279
7.2.1 Primitives overview	279
7.2.2 Primitive summary	280
7.2.3 Primitive encodings	285
7.2.4 Primitive sequences	290
7.2.4.1 Primitive sequences overview	290
7.2.4.2 Single primitive sequence	290
7.2.4.3 Repeated primitive sequence	290
7.2.4.4 Continued primitive sequence	291
7.2.4.5 Extended primitive sequence	291
7.2.4.6 Triple primitive sequence	291
7.2.4.7 Redundant primitive sequence	292
7.2.5 Primitives not specific to type of connections	293
7.2.5.1 AIP (Arbitration in progress)	293
7.2.5.2 ALIGN	294
7.2.5.3 BREAK	295
7.2.5.4 BREAK_REPLY	295
7.2.5.5 BROADCAST	295
7.2.5.6 CLOSE	295
7.2.5.7 EOAF (End of address frame)	296

7.2.5.8 ERROR	296
7.2.5.9 HARD_RESET	296
7.2.5.10 MUX (Multiplex)	296
7.2.5.11 NOTIFY	297
7.2.5.11.1 NOTIFY overview	297
7.2.5.11.2 NOTIFY (ENABLE SPINUP).....	297
7.2.5.11.3 NOTIFY (POWER LOSS EXPECTED).....	298
7.2.5.12 OPEN_ACCEPT	299
7.2.5.13 OPEN_REJECT	299
7.2.5.14 SOAF (Start of address frame).....	302
7.2.5.15 TRAIN.....	302
7.2.5.16 TRAIN_DONE	302
7.2.6 Primitives used only inside SSP and SMP connections	302
7.2.6.1 ACK (Acknowledge)	302
7.2.6.2 CREDIT_BLOCKED	302
7.2.6.3 DONE	302
7.2.6.4 EOF (End of frame)	303
7.2.6.5 NAK (Negative acknowledgement)	303
7.2.6.6 RRDY (Receiver ready).....	303
7.2.6.7 SOF (Start of frame)	304
7.2.7 Primitives used only inside STP connections and on SATA physical links	304
7.2.7.1 SATA_ERROR.....	304
7.2.7.2 SATA_PMACK, SATA_PMNAK, SATA_PMREQ_P, and SATA_PMREQ_S (Power management acknowledgements and requests)	304
7.2.7.3 SATA_HOLD and SATA_HOLDA (Hold and hold acknowledge).....	304
7.2.7.4 SATA_R_RDY and SATA_X_RDY (Receiver ready and transmitter ready).....	304
7.2.7.5 Other primitives used inside STP connections and on SATA physical links	304
7.3 Physical link rate tolerance management.....	305
7.3.1 Physical link rate tolerance management overview	305
7.3.2 Phys originating dwords	306
7.3.3 Expander phys forwarding dwords.....	306
7.4 Idle physical links.....	307
7.5 CRC.....	307
7.5.1 CRC overview	307
7.5.2 CRC generation	308
7.5.3 CRC checking	310
7.6 Scrambling.....	311
7.7 Bit order of CRC and scrambler	313
7.8 Address frames	316
7.8.1 Address frames overview.....	316
7.8.2 IDENTIFY address frame.....	318
7.8.3 OPEN address frame.....	321
7.9 Identification and hard reset sequence.....	324
7.9.1 Identification and hard reset sequence overview	324
7.9.2 SAS initiator device rules	326
7.9.3 Expander device rules	327
7.9.4 SL_IR (link layer identification and hard reset) state machines	327
7.9.4.1 SL_IR state machines overview	327
7.9.4.2 SL_IR transmitter and receiver.....	329
7.9.4.3 SL_IR_TIR (transmit IDENTIFY or HARD_RESET) state machine	329
7.9.4.3.1 SL_IR_TIR state machine overview	329
7.9.4.3.2 SL_IR_TIR1:Idle state	330
7.9.4.3.2.1 State description	330
7.9.4.3.2.2 Transition SL_IR_TIR1:Idle to SL_IR_TIR2:Transmit_Identify	330
7.9.4.3.2.3 Transition SL_IR_TIR1:Idle to SL_IR_TIR3:Transmit_Hard_Reset.....	330
7.9.4.3.3 SL_IR_TIR2:Transmit_Identify state.....	330
7.9.4.3.3.1 State description	330

7.9.4.3.3.2 Transition SL_IR_TIR2:Transmit_Identify to SL_IR_TIR4:Completed	330
7.9.4.3.4 SL_IR_TIR3:Transmit_Hard_Reset state	330
7.9.4.3.4.1 State description	330
7.9.4.3.4.2 Transition SL_IR_TIR3:Transmit_Hard_Reset to SL_IR_TIR4:Completed	330
7.9.4.3.5 SL_IR_TIR4:Completed state	330
7.9.4.4 SL_IR_RIF (receive IDENTIFY address frame) state machine	331
7.9.4.4.1 SL_IR_RIF state machine overview	331
7.9.4.4.2 SL_IR_RIF1:Idle state	331
7.9.4.4.2.1 State description	331
7.9.4.4.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame	331
7.9.4.4.3 SL_IR_RIF2:Receive_Identify_Frame state	331
7.9.4.4.3.1 State description	331
7.9.4.4.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed	332
7.9.4.4.4 SL_IR_RIF3:Completed state	332
7.9.4.5 SL_IR_IRC (identification and hard reset control) state machine	332
7.9.4.5.1 SL_IR_IRC state machine overview	332
7.9.4.5.2 SL_IR_IRC1:Idle state	332
7.9.4.5.2.1 State description	332
7.9.4.5.2.2 Transition SL_IR_IRC1:Idle to SL_IR_IRC2:Wait	332
7.9.4.5.3 SL_IR_IRC2:Wait state	332
7.9.4.5.3.1 State description	332
7.9.4.5.3.2 Transition SL_IR_IRC2:Wait to SL_IR_IRC3:Completed	333
7.9.4.5.4 SL_IR_IRC3:Completed state	333
7.10 Power management	333
7.11 SAS domain changes (Broadcast (Change) usage)	333
7.12 Connections	334
7.12.1 Connections overview	334
7.12.2 Opening a connection	335
7.12.2.1 Connection request	335
7.12.2.2 Results of a connection request	337
7.12.3 Arbitration fairness	337
7.12.4 Arbitration inside an expander device	338
7.12.4.1 Expander phy arbitration requirements	338
7.12.4.2 ECM arbitration requirements	339
7.12.4.2.1 ECM arbitration requirements overview	339
7.12.4.2.2 Arbitrating confirmations	339
7.12.4.2.3 Arb Won confirmation	340
7.12.4.2.4 Arb Lost confirmation	340
7.12.4.2.5 Arb Reject confirmation	341
7.12.4.3 Arbitration status	341
7.12.4.4 Partial Pathway Timeout timer	342
7.12.4.5 Pathway recovery	342
7.12.5 BREAK handling	343
7.12.6 Aborting a connection request	343
7.12.7 Closing a connection	345
7.12.8 Breaking a connection	346
7.13 Rate matching	347
7.14 SL (link layer for SAS logical phys) state machines	349
7.14.1 SL state machines overview	349
7.14.2 SL transmitter and receiver	351
7.14.3 SL_RA (receive OPEN address frame) state machine	352
7.14.4 SL_CC (connection control) state machine	353
7.14.4.1 SL_CC state machine overview	353
7.14.4.2 SL_CC0:Idle state	354
7.14.4.2.1 State description	354
7.14.4.2.2 Transition SL_CC0:Idle to SL_CC1:ArbSel	355
7.14.4.2.3 Transition SL_CC0:Idle to SL_CC2:Selected	355

7.14.4.3 SL_CC1:ArbSel state	355
7.14.4.3.1 State description	355
7.14.4.3.2 Transition SL_CC1:ArbSel to SL_CC0:Idle	356
7.14.4.3.3 Transition SL_CC1:ArbSel to SL_CC2:Selected	356
7.14.4.3.4 Transition SL_CC1:ArbSel to SL_CC3:Connected	357
7.14.4.3.5 Transition SL_CC1:ArbSel to SL_CC5:BreakWait	357
7.14.4.3.6 Transition SL_CC1:ArbSel to SL_CC6:Break	357
7.14.4.4 SL_CC2:Selected state	357
7.14.4.4.1 State description	357
7.14.4.4.2 Transition SL_CC2:Selected to SL_CC0:Idle	358
7.14.4.4.3 Transition SL_CC2:Selected to SL_CC3:Connected	358
7.14.4.4.4 Transition SL_CC2:Selected to SL_CC5:BreakWait	359
7.14.4.4.5 Transition SL_CC2:Selected to SL_CC6:Break	359
7.14.4.5 SL_CC3:Connected state	359
7.14.4.5.1 State description	359
7.14.4.5.2 Transition SL_CC3:Connected to SL_CC4:DisconnectWait	359
7.14.4.5.3 Transition SL_CC3:Connected to SL_CC5:BreakWait	359
7.14.4.5.4 Transition SL_CC3:Connected to SL_CC6:Break	359
7.14.4.5.5 Transition SL_CC3:Connected to SL_CC7:CloseSTP	359
7.14.4.6 SL_CC4:DisconnectWait state	360
7.14.4.6.1 State description	360
7.14.4.6.2 Transition SL_CC4:DisconnectWait to SL_CC0:Idle	360
7.14.4.6.3 Transition SL_CC4:DisconnectWait to SL_CC5:BreakWait	360
7.14.4.6.4 Transition SL_CC4:DisconnectWait to SL_CC6:Break	360
7.14.4.7 SL_CC5:BreakWait state	360
7.14.4.7.1 State description	360
7.14.4.7.2 Transition SL_CC5:BreakWait to SL_CC0:Idle	361
7.14.4.8 SL_CC6:Break state	361
7.14.4.8.1 State description	361
7.14.4.8.2 Transition SL_CC6:Break to SL_CC0:Idle	361
7.14.4.9 SL_CC7:CloseSTP state	361
7.14.4.9.1 State description	361
7.14.4.9.2 Transition SL_CC7:CloseSTP to SL_CC0:Idle	361
7.15 XL (link layer for expander logical phys) state machine	362
7.15.1 XL state machine overview	362
7.15.2 XL transmitter and receiver	366
7.15.3 XL0:Idle state	367
7.15.3.1 State description	367
7.15.3.2 Transition XL0:Idle to XL1:Request_Path	367
7.15.3.3 Transition XL0:Idle to XL5:Forward_Open	368
7.15.4 XL1:Request_Path state	368
7.15.4.1 State description	368
7.15.4.2 Transition XL1:Request_Path to XL0:Idle	369
7.15.4.3 Transition XL1:Request_Path to XL2:Request_Open	369
7.15.4.4 Transition XL1:Request_Path to XL4:Open_Reject	369
7.15.4.5 Transition XL1:Request_Path to XL5:Forward_Open	369
7.15.4.6 Transition XL1:Request_Path to XL9:Break	369
7.15.5 XL2:Request_Open state	370
7.15.5.1 State description	370
7.15.5.2 Transition XL2:Request_Open to XL3:Open_Confirm_Wait	370
7.15.6 XL3:Open_Confirm_Wait state	370
7.15.6.1 State description	370
7.15.6.2 Transition XL3:Open_Confirm_Wait to XL0:Idle	371
7.15.6.3 Transition XL3:Open_Confirm_Wait to XL1:Request_Path	371
7.15.6.4 Transition XL3:Open_Confirm_Wait to XL5:Forward_Open	371
7.15.6.5 Transition XL3:Open_Confirm_Wait to XL7:Connected	371
7.15.6.6 Transition XL3:Open_Confirm_Wait to XL9:Break	371

7.15.6.7 Transition XL3:Open_Confirm_Wait to XL10:Break_Wait.....	371
7.15.7 XL4:Open_Reject state.....	371
7.15.7.1 State description.....	371
7.15.7.2 Transition XL4:Open_Reject to XL0:Idle	372
7.15.7.3 Transition XL4:Open_Reject to XL5:Forward_Open	372
7.15.8 XL5:Forward_Open state.....	372
7.15.8.1 State description.....	372
7.15.8.2 Transition XL5:Forward_Open to XL6:Open_Response_Wait.....	372
7.15.9 XL6:Open_Response_Wait state.....	372
7.15.9.1 State description.....	372
7.15.9.2 Transition XL6:Open_Response_Wait to XL0:Idle	373
7.15.9.3 Transition XL6:Open_Response_Wait to XL1:Request_Path.....	373
7.15.9.4 Transition XL6:Open_Response_Wait to XL2:Request_Open	373
7.15.9.5 Transition XL6:Open_Response_Wait to XL7:Connected	374
7.15.9.6 Transition XL6:Open_Response_Wait to XL9:Break	374
7.15.9.7 Transition XL6:Open_Response_Wait to XL10:Break_Wait.....	374
7.15.10 XL7:Connected state	374
7.15.10.1 State description.....	374
7.15.10.2 Transition XL7:Connected to XL8:Close_Wait.....	374
7.15.10.3 Transition XL7:Connected to XL9:Break.....	375
7.15.10.4 Transition XL7:Connected to XL10:Break_Wait.....	375
7.15.11 XL8:Close_Wait state	375
7.15.11.1 State description.....	375
7.15.11.2 Transition XL8:Close_Wait to XL0:Idle.....	375
7.15.11.3 Transition XL8:Close_Wait to XL9:Break	375
7.15.11.4 Transition XL8:Close_Wait to XL10:Break_Wait.....	376
7.15.12 XL9:Break state	376
7.15.12.1 State description.....	376
7.15.12.2 Transition XL9:Break to XL0:Idle.....	376
7.15.13 XL10:Break_Wait state	376
7.15.13.1 State description.....	376
7.15.13.2 Transition XL10:Break_Wait to XL0:Idle	376
7.16 SSP link layer	376
7.16.1 Opening an SSP connection.....	376
7.16.2 Full duplex.....	377
7.16.3 SSP frame transmission and reception.....	377
7.16.4 SSP flow control.....	377
7.16.5 Interlocked frames	378
7.16.6 Breaking an SSP connection	380
7.16.7 Closing an SSP connection	380
7.16.8 SSP (link layer for SSP phys) state machines	381
7.16.8.1 SSP state machines overview.....	381
7.16.8.2 SSP transmitter and receiver	384
7.16.8.3 SSP_TIM (transmit interlocked frame monitor) state machine	385
7.16.8.4 SSP_TCM (transmit frame credit monitor) state machine	386
7.16.8.5 SSP_D (DONE control) state machine.....	386
7.16.8.6 SSP_TF (transmit frame control) state machine	388
7.16.8.6.1 SSP_TF state machine overview.....	388
7.16.8.6.2 SSP_TF1:Connected_Idle state	388
7.16.8.6.2.1 State description	388
7.16.8.6.2.2 Transition SSP_TF1:Connected_Idle to SSP_TF2:Tx_Wait.....	388
7.16.8.6.2.3 Transition SSP_TF1:Connected_Idle to SSP_TF4:Transmit_DONE.....	388
7.16.8.6.3 SSP_TF2:Tx_Wait state	388
7.16.8.6.3.1 State description	388
7.16.8.6.3.2 Transition SSP_TF2:Tx_Wait to SSP_TF3:Transmit_Frame.....	388
7.16.8.6.3.3 Transition SSP_TF2:Tx_Wait to SSP_TF4:Transmit_DONE.....	389
7.16.8.6.4 SSP_TF3:Transmit_Frame state	389

7.16.8.6.4.1 State description	389
7.16.8.6.4.2 Transition SSP_TF3:Transmit_Frame to SSP_TF1:Connected_Idle.....	389
7.16.8.6.5 SSP_TF4:Transmit_DONE state	389
7.16.8.7 SSP_RF (receive frame control) state machine	390
7.16.8.8 SSP_RCM (receive frame credit monitor) state machine.....	390
7.16.8.9 SSP_RIM (receive interlocked frame monitor) state machine.....	391
7.16.8.10 SSP_TC (transmit credit control) state machine	391
7.16.8.11 SSP_TAN (transmit ACK/NAK control) state machine.....	392
7.17 STP link layer	392
7.17.1 STP frame transmission and reception	392
7.17.2 STP flow control.....	393
7.17.3 Continued primitive sequence.....	396
7.17.4 Affiliations.....	397
7.17.5 Opening an STP connection	399
7.17.6 Closing an STP connection.....	401
7.17.7 STP connection management examples	402
7.17.8 STP (link layer for STP phys) state machines	405
7.17.9 SMP target port support.....	405
7.18 SMP link layer.....	405
7.18.1 SMP frame transmission and reception	405
7.18.2 SMP flow control	405
7.18.3 Opening an SMP connection	405
7.18.4 Closing an SMP connection.....	405
7.18.5 SMP (link layer for SMP phys) state machines.....	406
7.18.5.1 SMP state machines overview	406
7.18.5.2 SMP transmitter and receiver.....	406
7.18.5.3 SMP_IP (link layer for SMP initiator phys) state machine	406
7.18.5.3.1 SMP_IP state machine overview	406
7.18.5.3.2 SMP_IP1:Idle state	407
7.18.5.3.2.1 State description	407
7.18.5.3.2.2 Transition SMP_IP1:Idle to SMP_IP2:Transmit_Frame	407
7.18.5.3.3 SMP_IP2:Transmit_Frame state	408
7.18.5.3.3.1 State description	408
7.18.5.3.3.2 Transition SMP_IP2:Transmit_Frame to SMP_IP3:Receive_Frame	408
7.18.5.3.4 SMP_IP3:Receive_Frame state	408
7.18.5.4 SMP_TP (link layer for SMP target phys) state machine	409
7.18.5.4.1 SMP_TP state machine overview.....	409
7.18.5.4.2 SMP_TP1:Receive_Frame state	409
7.18.5.4.2.1 State description	409
7.18.5.4.2.2 Transition SMP_TP1:Receive_Frame to SMP_TP2:Transmit_Frame.....	410
7.18.5.4.3 SMP_TP2:Transmit_Frame state	410
8 Port layer.....	411
8.1 Port layer overview	411
8.2 PL (port layer) state machines.....	411
8.2.1 PL state machines overview	411
8.2.2 PL_OC (port layer overall control) state machine	413
8.2.2.1 PL_OC state machine overview.....	413
8.2.2.2 PL_OC1:Idle state	415
8.2.2.2.1 PL_OC1:Idle state description	415
8.2.2.2.2 Transition PL_OC1:Idle to PL_OC2:Overall_Control.....	416
8.2.2.3 PL_OC2:Overall_Control state.....	416
8.2.2.3.1 PL_OC2:Overall_Control state overview	416
8.2.2.3.2 PL_OC2:Overall_Control state establishing connections	417
8.2.2.3.3 PL_OC2:Overall_Control state connection established.....	420
8.2.2.3.4 PL_OC2:Overall_Control state unable to establish a connection.....	420
8.2.2.3.5 PL_OC2:Overall_Control state connection management.....	421

8.2.2.3.6 PL_OC2:Overall_Control state frame transmission	422
8.2.2.3.7 PL_OC2:Overall_Control state frame transmission cancellations	423
8.2.2.3.8 Transition PL_OC2:Overall_Control to PL_OC1:Idle	424
8.2.3 PL_PM (port layer phy manager) state machine	424
8.2.3.1 PL_PM state machine overview	424
8.2.3.2 PL_PM1:Idle state	426
8.2.3.2.1 PL_PM1:Idle state description	426
8.2.3.2.2 Transition PL_PM1:Idle to PL_PM2:Req_Wait	427
8.2.3.2.3 Transition PL_PM1:Idle to PL_PM3:Connected	427
8.2.3.3 PL_PM2:Req_Wait state	427
8.2.3.3.1 PL_PM2:Req_Wait state overview	427
8.2.3.3.2 PL_PM2:Req_Wait establishing a connection	427
8.2.3.3.3 PL_PM2:Req_Wait connection established	427
8.2.3.3.4 PL_PM2:Req_Wait unable to establish a connection	428
8.2.3.3.5 PL_PM2:Req_Wait connection management	429
8.2.3.3.6 Transition PL_PM2:Req_Wait to PL_PM1:Idle	429
8.2.3.3.7 Transition PL_PM2:Req_Wait to PL_PM3:Connected	429
8.2.3.3.8 Transition PL_PM2:Req_Wait to PL_PM4:Wait_For_Close	429
8.2.3.4 PL_PM3:Connected state	429
8.2.3.4.1 PL_PM3:Connected state description	429
8.2.3.4.2 Transition PL_PM3:Connected to PL_PM1:Idle	432
8.2.3.5 PL_PM4:Wait_For_Close state	432
8.2.3.5.1 PL_PM4:Wait_For_Close state description	432
8.2.3.5.2 Transition PL_PM4:Wait_For_Close to PL_PM1:Idle	433
9 Transport layer	434
9.1 Transport layer overview	434
9.2 SSP transport layer	435
9.2.1 SSP frame format	435
9.2.2 Information units	438
9.2.2.1 COMMAND frame - Command information unit	438
9.2.2.2 TASK frame - Task Management Function information unit	440
9.2.2.3 XFER_RDY frame - Transfer Ready information unit	442
9.2.2.4 DATA frame - Data information unit	443
9.2.2.5 RESPONSE frame - Response information unit	444
9.2.2.5.1 RESPONSE frame - Response information unit overview	444
9.2.2.5.2 Response information unit - NO_DATA format	445
9.2.2.5.3 Response information unit - RESPONSE_DATA format	445
9.2.2.5.4 Response information unit - SENSE_DATA format	446
9.2.3 Sequences of SSP frames	447
9.2.3.1 Sequences of SSP frames overview	447
9.2.3.2 Task management function sequence of SSP frames	447
9.2.3.3 Non-data command sequence of SSP frames	447
9.2.3.4 Write command sequence of SSP frames	448
9.2.3.5 Read command sequence of SSP frames	448
9.2.3.6 Bidirectional command sequence of SSP frames	449
9.2.4 SSP transport layer handling of link layer errors	449
9.2.4.1 SSP transport layer handling of link layer errors overview	449
9.2.4.2 COMMAND frame - handling of link layer errors	450
9.2.4.3 TASK frame - handling of link layer errors	450
9.2.4.4 XFER_RDY frame - handling of link layer errors	451
9.2.4.4.1 XFER_RDY frame overview	451
9.2.4.4.2 XFER_RDY frame with transport layer retries enabled	451
9.2.4.4.3 XFER_RDY frame with transport layer retries disabled	451
9.2.4.5 DATA frame - handling of link layer errors	452
9.2.4.5.1 DATA frame overview	452
9.2.4.5.2 DATA frame with transport layer retries enabled	452

9.2.4.5.3 DATA frame with transport layer retries disabled	452
9.2.4.6 RESPONSE frame - handling of link layer errors.....	453
9.2.5 SSP transport layer error handling summary	453
9.2.5.1 SSP transport layer error handling summary introduction.....	453
9.2.5.2 SSP initiator port transport layer error handling summary	453
9.2.5.3 SSP target port transport layer error handling summary.....	454
9.2.6 ST (transport layer for SSP ports) state machines	455
9.2.6.1 ST state machines overview	455
9.2.6.2 ST_I (transport layer for SSP initiator ports) state machines	456
9.2.6.2.1 ST_I state machines overview.....	456
9.2.6.2.2 ST_IFR (initiator frame router) state machine	458
9.2.6.2.2.1 ST_IFR state machine overview	458
9.2.6.2.2.2 Processing transport protocol service requests	458
9.2.6.2.2.3 Processing Frame Received confirmations.....	459
9.2.6.2.2.4 Processing Transmission Complete and Reception Complete messages.....	460
9.2.6.2.2.5 Processing miscellaneous requests.....	461
9.2.6.2.3 ST_ITS (initiator transport server) state machine	462
9.2.6.2.3.1 ST_ITS state machine overview	462
9.2.6.2.3.2 ST_ITS1:Initiator_Start state	463
9.2.6.2.3.2.1 State description.....	463
9.2.6.2.3.2.2 Transition ST_ITS1:Initiator_Start to ST_ITS3:Prepare_Command.....	463
9.2.6.2.3.2.3 Transition ST_ITS1:Initiator_Start to ST_ITS4:Prepare_Task	463
9.2.6.2.3.3 ST_ITS2:Initiator_Send_Frame state.....	463
9.2.6.2.3.3.1 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS1:Initiator_Start	467
9.2.6.2.3.3.2 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS5:Prepare_Data_Out.....	467
9.2.6.2.3.3.3 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS6:Process_Data_In	467
9.2.6.2.3.4 ST_ITS3:Prepare_Command state	468
9.2.6.2.3.4.1 State description.....	468
9.2.6.2.3.4.2 Transition ST_ITS3:Prepare_Command to ST_ITS2:Initiator_Send_Frame	468
9.2.6.2.3.5 ST_ITS4:Prepare_Task state.....	468
9.2.6.2.3.5.1 State description.....	468
9.2.6.2.3.5.2 Transition ST_ITS4:Prepare_Task to ST_ITS2:Initiator_Send_Frame	469
9.2.6.2.3.6 ST_ITS5:Prepare_Data_Out state	469
9.2.6.2.3.6.1 State description.....	469
9.2.6.2.3.6.2 Transition ST_ITS5:Prepare_Data_Out to ST_ITS2:Initiator_Send_Frame	470
9.2.6.2.3.7 ST_ITS6:Receive_Data_In state.....	470
9.2.6.2.3.7.1 State description.....	470
9.2.6.2.3.7.2 Transition ST_ITS6:Receive_Data_In to ST_ITS1:Initiator_Start	471
9.2.6.2.3.7.3 Transition ST_ITS6:Receive_Data_In to ST_ITS2:Initiator_Send_Frame	471
9.2.6.3 ST_T (transport layer for SSP target ports) state machines	471
9.2.6.3.1 ST_T state machines overview.....	471
9.2.6.3.2 ST_TFR (target frame router) state machine.....	474
9.2.6.3.2.1 ST_TFR state machine overview.....	474
9.2.6.3.2.2 Processing Frame Received confirmations.....	474
9.2.6.3.2.3 Processing transport protocol service requests and responses.....	476
9.2.6.3.2.4 Processing miscellaneous requests and confirmations	479
9.2.6.3.3 ST_TTS (target transport server) state machine	479
9.2.6.3.3.1 ST_TTS state machine overview	479
9.2.6.3.3.2 ST_TTS1:Target_Start state	480
9.2.6.3.3.2.1 State description.....	480
9.2.6.3.3.2.2 Transition ST_TTS1:Target_Start to ST_TTS3:Prepare_Data_In.....	481
9.2.6.3.3.2.3 Transition ST_TTS1:Target_Start to ST_TTS4:Prepare_Xfer_Rdy	481
9.2.6.3.3.2.4 Transition ST_TTS1:Target_Start to ST_TTS5:Receive_Data_Out.....	481
9.2.6.3.3.2.5 Transition ST_TTS1:Target_Start to ST_TTS7:Prepare_Response	481
9.2.6.3.3.3 ST_TTS2:Target_Send_Frame state.....	481
9.2.6.3.3.3.1 State description.....	481
9.2.6.3.3.3.2 Transition ST_TTS2:Target_Send_Frame to ST_TTS1:Target_Start.....	485

9.2.6.3.3.3 Transition ST_TTS2:Target_Send_Frame to ST_TTS3:Prepare_Data_In	485
9.2.6.3.3.4 Transition ST_TTS2:Target_Send_Frame to ST_TTS5:Receive_Data_Out	486
9.2.6.3.3.4 ST_TTS3:Prepare_Data_In state.....	486
9.2.6.3.3.4.1 State description.....	486
9.2.6.3.3.4.2 Transition ST_TTS3:Prepare_Data_In to ST_TTS2:Target_Send_Frame	487
9.2.6.3.3.5 ST_TTS4:Prepare_Xfer_Rdy state	487
9.2.6.3.3.5.1 State description.....	487
9.2.6.3.3.5.2 Transition ST_TTS4:Prepare_Xfer_Rdy to ST_TTS2:Target_Send_Frame	488
9.2.6.3.3.6 ST_TTS5:Receive_Data_Out state.....	488
9.2.6.3.3.6.1 State description.....	488
9.2.6.3.3.6.2 Transition ST_TTS5:Receive_Data_Out to ST_TTS1:Target_Start.....	489
9.2.6.3.3.6.3 Transition ST_TTS5:Receive_Data_Out to ST_TTS4:Prepare_Xfer_Rdy.....	489
9.2.6.3.3.7 ST_TTS6:Prepare_Response state	490
9.2.6.3.3.7.1 State description.....	490
9.2.6.3.3.7.2 Transition ST_TTS6:Prepare_Response to ST_TTS2:Target_Send_Frame	491
9.3 STP transport layer.....	491
9.3.1 Initial FIS.....	491
9.3.2 BIST Activate FIS.....	491
9.3.3 TT (transport layer for STP ports) state machines	491
9.4 SMP transport layer.....	492
9.4.1 SMP transport layer overview	492
9.4.2 SMP_REQUEST frame.....	493
9.4.3 SMP_RESPONSE frame	493
9.4.4 Sequence of SMP frames.....	494
9.4.5 MT (transport layer for SMP ports) state machines	494
9.4.5.1 SMP transport layer state machines overview	494
9.4.5.2 MT_IP (transport layer for SMP initiator ports) state machine	494
9.4.5.2.1 MT_IP state machine overview.....	494
9.4.5.2.2 MT_IP1:Idle state.....	495
9.4.5.2.2.1 State description	495
9.4.5.2.2.2 Transition MT_IP1:Idle to MT_IP2:Send	495
9.4.5.2.3 MT_IP2:Send state	496
9.4.5.2.3.1 State description	496
9.4.5.2.3.2 Transition MT_IP2:Send to MT_IP1:Idle	496
9.4.5.2.3.3 Transition MT_IP2:Send to MT_IP3:Receive	496
9.4.5.2.4 MT_IP3:Receive state	496
9.4.5.2.4.1 State description	496
9.4.5.2.4.2 Transition MT_IP3:Receive to MT_IP1:Idle	496
9.4.5.3 MT_TP (transport layer for SMP target ports) state machine.....	496
9.4.5.3.1 MT_TP state machine overview	496
9.4.5.3.2 MT_TP1:Idle state	497
9.4.5.3.2.1 State description	497
9.4.5.3.2.2 Transition MT_TP1:Idle to MT_TP2:Respond.....	497
9.4.5.3.3 MT_TP2:Respond state.....	498
9.4.5.3.3.1 State description	498
9.4.5.3.3.2 Transition MT_TP2:Respond to MT_TP1:Idle.....	498
10 Application layer.....	499
10.1 Application layer overview	499
10.2 SCSI application layer	499
10.2.1 SCSI transport protocol services	499
10.2.1.1 SCSI transport protocol services overview.....	499
10.2.1.2 Send SCSI Command transport protocol service.....	501
10.2.1.3 SCSI Command Received transport protocol service	501
10.2.1.4 Send Command Complete transport protocol service.....	502
10.2.1.5 Command Complete Received transport protocol service	503
10.2.1.6 Send Data-In transport protocol service.....	504

10.2.1.7 Data-In Delivered transport protocol service	505
10.2.1.8 Receive Data-Out transport protocol service	505
10.2.1.9 Data-Out Received transport protocol service	506
10.2.1.10 Terminate Data Transfer transport protocol service	506
10.2.1.11 Data Transfer Terminated transport protocol service	507
10.2.1.12 Send Task Management Request transport protocol service	507
10.2.1.13 Task Management Request Received transport protocol service	507
10.2.1.14 Task Management Function Executed transport protocol service	508
10.2.1.15 Received Task Management Function Executed transport protocol service	509
10.2.2 Application client error handling	510
10.2.3 Device server error handling	511
10.2.4 Task router and task manager error handling	511
10.2.5 SCSI transport protocol event notifications	512
10.2.6 SCSI commands	512
10.2.6.1 INQUIRY command	512
10.2.6.2 MODE SELECT and MODE SENSE commands	512
10.2.6.3 LOG SELECT and LOG SENSE commands	512
10.2.6.4 SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands	512
10.2.6.5 START STOP UNIT command	512
10.2.7 SCSI mode parameters	513
10.2.7.1 SCSI mode parameters overview	513
10.2.7.2 Disconnect-Reconnect mode page	513
10.2.7.2.1 Disconnect-Reconnect mode page overview	513
10.2.7.2.2 BUS INACTIVITY TIME LIMIT field	514
10.2.7.2.3 MAXIMUM CONNECT TIME LIMIT field	515
10.2.7.2.4 MAXIMUM BURST SIZE field	515
10.2.7.2.5 FIRST BURST SIZE field	515
10.2.7.3 Protocol-Specific Logical Unit mode page	516
10.2.7.4 Protocol-Specific Port mode page	516
10.2.7.5 Phy Control And Discover mode page	518
10.2.7.6 Shared Port Control mode page	521
10.2.7.6 SAS-2 Phy mode page	521
10.2.8 SCSI log parameters	524
10.2.8.1 Protocol-Specific Port log page	524
10.2.9 SCSI diagnostic parameters	528
10.2.9.1 SCSI diagnostic parameters overview	528
10.2.9.2 Protocol-Specific diagnostic page	529
10.2.9.3 Enclosure Control diagnostic page	533
10.2.9.4 Enclosure Status diagnostic page	533
10.2.9.5 Additional Element Status diagnostic page	534
10.2.10 SCSI power conditions	534
10.2.10.1 SCSI power conditions overview	534
10.2.10.2 SA_PC (SCSI application layer power condition) state machine	534
10.2.10.2.1 SA_PC state machine overview	534
10.2.10.2.2 SA_PC_0:Powered_On state	535
10.2.10.2.2.1 State description	535
10.2.10.2.2.2 Transition SA_PC_0:Powered_On to SA_PC_4:Stopped	535
10.2.10.2.2.3 Transition SA_PC_0:Powered_On to SA_PC_5:Active_Wait	536
10.2.10.2.3 SA_PC_1:Active state	536
10.2.10.2.3.1 State description	536
10.2.10.2.3.2 Transition SA_PC_1:Active to SA_PC_2:Idle	536
10.2.10.2.3.3 Transition SA_PC_1:Active to SA_PC_3:Standby	536
10.2.10.2.3.4 Transition SA_PC_1:Active to SA_PC_4:Stopped	536
10.2.10.2.4 SA_PC_2:Idle state	536
10.2.10.2.4.1 State description	536
10.2.10.2.4.2 Transition SA_PC_2:Idle to SA_PC_1:Active	536
10.2.10.2.4.3 Transition SA_PC_2:Idle to SA_PC_3:Standby	536

10.2.10.2.4.4 Transition SA_PC_2:Idle to SA_PC_4:Stopped.....	537
10.2.10.2.5 SA_PC_3:Standby state	537
10.2.10.2.5.1 State description	537
10.2.10.2.5.2 Transition SA_PC_3:Standby to SA_PC_4:Stopped	537
10.2.10.2.5.3 Transition SA_PC_3:Standby to SA_PC_5:Active_Wait.....	537
10.2.10.2.5.4 Transition SA_PC_3:Standby to SA_PC_6:Idle_Wait.....	537
10.2.10.2.6 SA_PC_4:Stopped state.....	537
10.2.10.2.6.1 State description	537
10.2.10.2.6.2 Transition SA_PC_4:Stopped to SA_PC_3:Standby	537
10.2.10.2.6.3 Transition SA_PC_4:Stopped to SA_PC_5:Active_Wait	537
10.2.10.2.6.4 Transition SA_PC_4:Stopped to SA_PC_6:Idle_Wait	538
10.2.10.2.7 SA_PC_5:Active_Wait state	538
10.2.10.2.7.1 State description	538
10.2.10.2.7.2 Transition SA_PC_5:Active_Wait to SA_PC_1:Active	538
10.2.10.2.7.3 Transition SA_PC_5:Active_Wait to SA_PC_3:Standby.....	538
10.2.10.2.7.4 Transition SA_PC_5:Active_Wait to SA_PC_4:Stopped	538
10.2.10.2.7.5 Transition SA_PC_5:Active_Wait to SA_PC_6:Idle_Wait.....	538
10.2.10.2.8 SA_PC_6:Idle_Wait state	539
10.2.10.2.8.1 State description	539
10.2.10.2.8.2 Transition SA_PC_6:Idle_Wait to SA_PC_2:Idle	539
10.2.10.2.8.3 Transition SA_PC_6:Idle_Wait to SA_PC_3:Standby.....	539
10.2.10.2.8.4 Transition SA_PC_6:Idle_Wait to SA_PC_4:Stopped	539
10.2.10.2.8.5 Transition SA_PC_6:Idle_Wait to SA_PC_5:Active_Wait.....	539
10.2.11 SCSI vital product data (VPD)	540
10.2.11.1 SCSI vital product data (VPD) overview.....	540
10.2.11.2 Device Identification VPD page.....	540
10.2.11.3 Protocol-Specific Logical Unit Information VPD page	541
10.3 ATA application layer.....	542
10.4 Management application layer.....	542
10.4.1 READY LED signal behavior	542
10.4.2 Management protocol services	544
10.4.3 SMP functions.....	544
10.4.3.1 SMP function request frame format.....	544
10.4.3.2 SMP function response frame format.....	546
10.4.3.3 REPORT GENERAL function.....	554
10.4.3.4 REPORT MANUFACTURER INFORMATION function	560
10.4.3.5 REPORT SELF-CONFIGURATION STATUS function	564
10.4.3.5.1 REPORT SELF-CONFIGURATION STATUS function overview	564
10.4.3.5.2 REPORT SELF-CONFIGURATION STATUS request	564
10.4.3.5.3 REPORT SELF-CONFIGURATION STATUS response	565
10.4.3.5.4 Self-configuration status descriptor	567
10.4.3.6 REPORT ZONE PERMISSION TABLE function.....	569
10.4.3.6.1 REPORT ZONE PERMISSION TABLE function overview	569
10.4.3.6.2 REPORT ZONE PERMISSION TABLE request.....	569
10.4.3.6.3 REPORT ZONE PERMISSION TABLE response	570
10.4.3.6.4 Zone permission descriptor	571
10.4.3.7 REPORT ZONE MANAGER PASSWORD function.....	572
10.4.3.8 REPORT BROADCAST function	573
10.4.3.8.1 REPORT BROADCAST function overview.....	573
10.4.3.8.2 REPORT BROADCAST request	574
10.4.3.8.3 REPORT BROADCAST response.....	575
10.4.3.8.4 REPORT BROADCAST response broadcast descriptor	576
10.4.3.9 DISCOVER function	577
10.4.3.10 REPORT PHY ERROR LOG function.....	591
10.4.3.11 REPORT PHY SATA function	594
10.4.3.12 REPORT ROUTE INFORMATION function	597
10.4.3.13 REPORT PHY EVENT function	600

10.4.3.13.1 REPORT PHY EVENT function overview.....	600
10.4.3.13.2 REPORT PHY EVENT request	600
10.4.3.13.3 REPORT PHY EVENT response.....	601
10.4.3.13.4 REPORT PHY EVENT response phy event descriptor	602
10.4.3.14 DISCOVER LIST function	602
10.4.3.14.1 DISCOVER LIST function overview.....	602
10.4.3.14.2 DISCOVER LIST request	603
10.4.3.14.3 DISCOVER LIST response.....	604
10.4.3.14.4 DISCOVER LIST response SHORT FORMAT descriptor	606
10.4.3.15 REPORT PHY EVENT LIST function.....	607
10.4.3.15.1 REPORT PHY EVENT LIST function overview	607
10.4.3.15.2 REPORT PHY EVENT LIST request.....	607
10.4.3.15.3 REPORT PHY EVENT LIST response	609
10.4.3.15.4 REPORT PHY EVENT LIST response phy event list descriptor	610
10.4.3.16 REPORT EXPANDER ROUTE TABLE LIST function	611
10.4.3.16.1 REPORT EXPANDER ROUTE TABLE LIST function overview.....	611
10.4.3.16.2 REPORT EXPANDER ROUTE TABLE LIST request	611
10.4.3.16.3 REPORT EXPANDER ROUTE TABLE LIST response.....	612
10.4.3.16.4 REPORT EXPANDER ROUTE TABLE descriptor	613
10.4.3.17 CONFIGURE GENERAL function	614
10.4.3.18 ENABLE DISABLE ZONING function	617
10.4.3.19 ZONED BROADCAST function	619
10.4.3.20 ZONE LOCK function	622
10.4.3.21 ZONE ACTIVATE function	624
10.4.3.22 ZONE UNLOCK function.....	626
10.4.3.23 CONFIGURE ZONE MANAGER PASSWORD function	627
10.4.3.24 CONFIGURE ZONE PHY INFORMATION function.....	629
10.4.3.24.1 CONFIGURE ZONE PHY INFORMATION function overview	629
10.4.3.24.2 CONFIGURE ZONE PHY INFORMATION request.....	630
10.4.3.24.3 Zone phy configuration descriptor	631
10.4.3.24.4 CONFIGURE ZONE PHY INFORMATION response	632
10.4.3.25 CONFIGURE ZONE PERMISSION TABLE function	632
10.4.3.25.1 CONFIGURE ZONE PERMISSION TABLE function overview	632
10.4.3.25.2 CONFIGURE ZONE PERMISSION TABLE request	633
10.4.3.25.3 Zone permission configuration descriptor.....	634
10.4.3.25.4 CONFIGURE ZONE PERMISSION TABLE response	636
10.4.3.26 CONFIGURE ROUTE INFORMATION function	636
10.4.3.27 PHY CONTROL function.....	638
10.4.3.28 PHY TEST FUNCTION function.....	644
10.4.3.29 CONFIGURE PHY EVENT function	646
10.4.3.29.1 CONFIGURE PHY EVENT function overview	646
10.4.3.29.2 CONFIGURE PHY EVENT request.....	647
10.4.3.29.3 CONFIGURE PHY EVENT request phy event configuration descriptor	648
10.4.3.29.4 CONFIGURE PHY EVENT response	648
Annex A Jitter tolerance patterns	650
A.1 Jitter tolerance pattern (JTPAT)	650
A.2 Compliant jitter tolerance pattern (CJTPAT)	652
A.3 Considerations for a phy transmitting JTPAT and CJTPAT	658
A.4 Considerations for a phy receiving JTPAT and CJTPAT	658
Annex B Signal performance measurements.....	660
B.1 Signal performance measurements overview	660
B.2 Simple physical link.....	660
B.2.1 Simple physical link overview	660
B.2.2 Assumptions for the structure of the transmitter device and the receiver device	661
B.2.3 Definition of receiver sensitivity and receiver device sensitivity	662

B.3 Measurement architecture requirements	663
B.3.1 General.....	663
B.3.2 Relationship between signal compliance measurements at interoperability points and operation in systems.....	663
B.4 De-embedding connectors in test fixtures.....	664
B.5 Measurement conditions for signal output at the transmitter device	664
B.6 Measurement conditions for signal tolerance at the transmitter device	665
B.7 Measurement conditions for signal output at the receiver device	666
B.8 Measurement conditions for signal tolerance at the receiver device	667
B.9 S-parameter measurements	668
B.9.1 S-parameter overview	668
B.9.2 S-parameter naming conventions.....	668
B.9.3 Use of single-ended instrumentation in differential applications.....	669
B.9.4 Measurement configurations for physical link elements	670
B.9.4.1 Measurement configuration overview	670
B.9.4.2 Transmitter device return loss.....	671
B.9.4.3 Receiver device return loss.....	672
B.9.4.4 Return loss (S_{11}) at IT or CT.....	672
B.9.4.5 Upstream return loss (S_{22}) at IR or CR.....	673
B.9.5 Summary for S-parameter measurements	674
Annex C SAS to SAS phy reset sequence examples	675
Annex D CRC.....	680
D.1 CRC generator and checker implementation examples	680
D.2 CRC implementation in C	680
D.3 CRC implementation with XORs	681
D.4 CRC examples.....	683
Annex E SAS address hashing	684
E.1 SAS address hashing overview	684
E.2 Hash collision probability.....	684
E.3 Hash generation.....	685
E.4 Hash implementation in C	685
E.5 Hash implementation with XORs	686
E.6 Hash examples	687
Annex F Scrambling	690
F.1 Scrambler implementation example	690
F.2 Scrambler implementation in C	690
F.3 Scrambler implementation with XORs.....	691
F.4 Scrambler examples.....	692
Annex G ATA architectural notes.....	693
G.1 STP differences from Serial ATA (SATA)	693
G.2 STP differences from Serial ATA II.....	693
G.3 Affiliation policies	693
G.3.1 Affiliation policies overview	693
G.3.2 Affiliation policy for static STP initiator port to STP target port mapping.....	694
G.3.3 Affiliation policy with SATA queued commands and multiple STP initiator ports	694
G.3.4 Applicability of affiliation for STP target ports.....	694
G.4 SATA port selector considerations	694
G.5 SATA device not transmitting initial Register Device-to-Host FIS	694
Annex H Minimum deletable primitive insertion rate summary	696
Annex I Zone permission configuration descriptor examples.....	697

Annex J Expander device handling of connections.....	700
J.1 Expander device handling of connections overview	700
J.2 Connection request - OPEN_ACCEPT	702
J.3 Connection request - OPEN_REJECT by end device	703
J.4 Connection request - OPEN_REJECT by expander device	704
J.5 Connection request - arbitration lost.....	705
J.6 Connection request - backoff and retry.....	706
J.7 Connection request - backoff and reverse path	707
J.8 Connection close - single step	708
J.9 Connection close - simultaneous	709
J.10 BREAK handling during path arbitration when the BREAK_REPLY method is disabled	710
J.11 BREAK handling during connection when the BREAK_REPLY method is disabled	711
J.12 BREAK handling during path arbitration when the BREAK_REPLY method is enabled	712
J.13 BREAK handling during connection when BREAK_REPLY method is enabled.....	713
J.14 STP connection - originated by STP initiator port.....	714
J.15 STP connection - originated by STP target port in an STP/SATA bridge	715
J.16 STP connection close - originated by STP initiator port	716
J.17 STP connection close - originated by STP target port in an STP/SATA bridge.....	717
J.18 Connection request - XL1:Request_Path to XL5:Forward_Open transition	718
J.19 Pathway blocked and pathway recovery example	719
 Annex K Primitive encoding	 721
 Annex L Discover process example implementation	 724
L.1 Discover process example implementation overview	724
L.2 Header file	724
L.3 Source file.....	741
 Annex M SAS icons	 763

Tables

	Page
1 Standards bodies	3
2 ISO and American numbering conventions	27
3 Multiplicity notation in class diagrams	28
4 Data dword containing a value	35
5 Data dword containing four one-byte fields	35
6 Logical links	42
7 Broadcast types	56
8 Names and identifiers	58
9 SAM-4 attribute mapping	58
10 NAA IEEE Registered format	58
11 Hashed SAS address code parameter	59
12 Device name created from the IDENTIFY (PACKET) DEVICE world wide name	60
13 Expander logical phy to ECM requests	80
14 Expander logical phy to ECM responses	80
15 ECM to expander logical phy confirmations	80
16 Expander logical phy to ECR to expander logical phy requests and indications	81
17 Expander logical phy to ECR to expander logical phy responses and confirmations	82
18 Expander logical phy to BPP requests	83
19 BPP to expander logical phy indications	83
20 Routing attributes and routing methods	84
21 Expander route table types	85
22 Expander route table levels for externally configurable expander device R phy A	97
23 Expander route table levels for externally configurable expander device N	97
24 Expander route entries for externally configurable expander device E0 phy 1	98
25 Expander route entries for externally configurable expander device F phy 0	99
26 Zone manager password	102
27 Zoning expander device zoning configuration after power on	103
28 Zone phy information	106
29 Zone phy information usage	106
30 Zone groups	108
31 Zone permission table	109
32 Zone permission table granting minimal permissions	109
33 Source zone group determination	111
34 Destination zone group determination	111
35 Zone phy information fields after a link reset sequence	113
36 Events that cause the ZONE GROUP field to be reset when the ZONE GROUP PERSISTENT bit set to zero ..	114
37 PHY EVENT SOURCE field	119
38 Connectors	128
39 SAS Drive connector pin assignments	132
40 Controller SAS 4i connector pin assignments and physical link usage	134
41 Backplane SAS 4i connector pin assignments and physical link usage	135
42 Controller Mini SAS 4i connector pin assignments and physical link usage	137
43 Backplane Mini SAS 4i connector pin assignments and physical link usage	138
44 SAS 4x cable plug connector icons	139
45 SAS 4x receptacle connector icons	140
46 SAS 4x connector pin assignments and physical link usage	141
47 Mini SAS 4x cable plug connector icons and key slot positions	142
48 Mini SAS 4x receptacle connector icons and key positions	144
49 Mini SAS 4x connector pin assignments and physical link usage	146
50 Requirements for internal cable assemblies using SAS Drive connectors and backplanes	163
51 Requirements for SAS internal cable assemblies using SAS 4i or Mini SAS 4i	164
52 Additional requirements for cable assemblies using SAS 4i	165
53 Additional requirements for cable assemblies using Mini SAS 4i	165
54 Requirements for external cable assemblies	166
55 Additional requirements for external cable assemblies using SAS 4x	167

56 Additional requirements for external cable assemblies using Mini SAS 4x	167
57 Compliance points	168
58 General electrical characteristics	183
59 General transmitter device electrical characteristics	184
60 Receiver device general electrical characteristics	185
61 Transmitter device signal output characteristics as measured with the zero-length test load at transmitter device compliance points IT and CT	190
62 Transmitter device signal output characteristics as measured with each test load at transmitter device compliance points IT and CT	191
63 Transmitter device maximum jitter as measured with each test load at transmitter device compliance points IT and CT	192
64 Delivered signal characteristics as measured with the zero length test load at receiver device compliance points IR and CR	194
65 Maximum delivered jitter at receiver device compliance points IR and CR	196
66 Receiver device jitter tolerance at receiver device compliance points IR and CR	197
67 SSC modulation types	198
68 SAS phy transmitter SSC modulation	199
69 Expander phy transmitter SSC modulation types	199
70 Receiver SSC modulation tolerance	200
71 Expander device center-spreading tolerance buffer	200
72 Output characteristics of the READY LED signal	202
73 Bit designations	203
74 Conversion from byte notation to character name example	204
75 Data characters	206
76 Control characters	211
77 Control character usage	211
78 Delayed code violation example	212
79 OOB signal timing specifications	215
80 OOB signal transmitter device requirements	216
81 OOB signal receiver device burst time detection requirements	218
82 OOB signal receiver device idle time detection requirements	218
83 OOB signal receiver device negation time detection requirements	218
84 SATA port selection signal transmitter device requirements	220
85 Phy reset sequence timing specifications	221
86 SATA speed negotiation sequence timing specifications	222
87 SAS speed negotiation sequence timing specifications	227
88 SNW rates used in SNW-1, SNW-2, and Final-SNW	229
89 SNW-3 phy capabilities bit	230
90 SNW-3 phy capabilities	231
91 Requested logical link rate	232
92 Multiplexing negotiation	232
93 Supported settings bit priorities	233
94 Example SNW-3 phy capabilities values	234
95 Training patterns	235
96 SP state machine timers	248
97 SP_DWS timers	271
98 Primitive format	279
99 Deletable primitives	280
100 Primitives not specific to type of connection	281
101 Primitives used only inside SSP and SMP connections	283
102 Primitives used only inside STP connections and on SATA physical links	284
103 Primitive encoding for deletable primitives	285
104 Primitive encoding for primitives not specific to type of connection	286
105 Primitive encoding for primitives used only inside SSP and SMP connections	288
106 Primitive encoding for primitives used only inside STP connections and on SATA physical links	289
107 Primitive sequences	290
108 AIP primitives	294

109 ALIGN primitives	294
110 BROADCAST primitives	295
111 CLOSE primitives	296
112 MUX primitives	297
113 NOTIFY primitives	297
114 Abandon-class OPEN_REJECT primitives	300
115 Retry-class OPEN_REJECT primitives	301
116 DONE primitives	303
117 NAK primitives	303
118 RRDY primitives	304
119 Physical link rate tolerance management deletable primitive insertion requirement	306
120 CRC notation and definitions	307
121 Scrambling for different data dword types	311
122 Address frame format	317
123 ADDRESS FRAME TYPE field	317
124 IDENTIFY address frame format	318
125 DEVICE TYPE field	318
126 REASON field	319
127 OPEN address frame format	321
128 PROTOCOL field	322
129 FEATURES field	322
130 CONNECTION RATE field	322
131 ARBITRATION WAIT TIME field	324
132 SL_IR_IRC timers	327
133 Connection Results of a connection request	337
134 Arbitration priority for OPEN address frames passing on a physical link	338
135 Arbitration priority for contending Request Path requests in the ECM when all requests have Retry Priority Status arguments of NORMAL	340
136 Arbitration priority for contending Request Path requests in the ECM among requests with Retry Priority Status arguments of IGNORE AWT	340
137 Pathway recovery priority	342
138 Results of aborting a connection request	343
139 Results of closing a connection	345
140 Results of breaking a connection	346
141 Rate matching deletable primitive insertion requirements	347
142 SL_CC timers	354
143 SL_CC state machine variables	354
144 OPEN_REJECT Received message to Open Failed confirmation mapping	356
145 XL timers	362
146 SSP frame interlock requirements	378
147 SSP link layer timers	382
148 STP link layer differences from SATA link layer during an STP connection	393
149 Affiliation policies	397
150 Affiliation context relative identifier example	399
151 PL_OC state machine timers	414
152 Confirmations from Unable To Connect or Retry Open messages	421
153 PL_PM state machine timers	424
154 Messages from Open Failed confirmations	428
155 SSP frame format	435
156 FRAME TYPE field	436
157 COMMAND frame - Command information unit	439
158 TASK ATTRIBUTE field	440
159 TASK frame - Task Management Function information unit	440
160 TASK MANAGEMENT FUNCTION field	441
161 XFER_RDY frame - Transfer Ready information unit	442
162 DATA frame - Data information unit	443
163 RESPONSE frame - Response information unit	444

164 DATAPRES field	445
165 RESPONSE DATA field	445
166 RESPONSE CODE field	446
167 Sequences of SSP frames	447
168 Confirmations sent to the SCSI application layer if a frame transmission or reception error occurs	461
169 ST_ITS state machine variables	462
170 ST_ITS state machine arguments	463
171 Messages sent to the ST_IFR state machine	465
172 Transmission Complete messages for XFER_RDY frame verification failures	466
173 Reception Complete messages for read DATA frame verification failures	470
174 ST_T state machine timers	472
175 Task Management Function Executed Service Response argument mapping to Request (Send Transport Response) service response argument	477
176 Confirmations sent to the SCSI application layer	478
177 ST_TTS state machine variables	480
178 ST_TTS state machine arguments	480
179 Messages sent to the ST_TFR state machine	484
180 Additional messages sent to the ST_TFR state machine	485
181 Reception Complete message for write DATA frame verification failures	488
182 Request argument to RESPONSE frame response data field mapping	490
183 SMP frame format	492
184 SMP FRAME TYPE field	492
185 SMP_REQUEST frame format	493
186 SMP_RESPONSE frame format	493
187 MT_TP time limits	497
188 SCSI architecture mapping	500
189 Send SCSI Command transport protocol service arguments	501
190 SCSI Command Received transport protocol service arguments	502
191 Send Command Complete transport protocol service arguments	503
192 Command Complete Received transport protocol service arguments	504
193 Send Data-In transport protocol service arguments	505
194 Data-In Delivered transport protocol service arguments	505
195 Receive Data-Out transport protocol service arguments	506
196 Data-Out Received transport protocol service arguments	506
197 Terminate Data Transfer transport protocol service arguments	506
198 Data Transfer Terminated transport protocol service arguments	507
199 Send Task Management Request transport protocol service arguments	507
200 Task Management Request Received transport protocol service arguments	508
201 Task Management Function Executed transport protocol service arguments	509
202 Received Task Management Function Executed transport protocol service arguments	510
203 Delivery Result to additional sense code mapping	511
204 SCSI transport protocol events	512
205 SSP target port mode pages	513
206 Disconnect-Reconnect mode page for SSP	514
207 Protocol-Specific Logical Unit mode page for SAS SSP	516
208 Protocol-Specific Port mode page for SAS SSP	517
209 I_T NEXUS LOSS TIME field	518
210 Phy Control And Discover mode page	519
211 SAS phy mode descriptor	520
212 Shared Port Control mode page	521
293 SAS-2 Phy mode page	522
294 SAS-2 phy mode descriptor	523
295 Protocol-Specific Port log page for SAS SSP	524
296 Protocol-Specific Port log parameter for SAS	525
297 Parameter control byte in the Protocol-Specific Port log parameter for SAS	526
298 SAS phy log descriptor	526
299 SSP target port diagnostic pages	528

300 Diagnostic pages affected by zoning	528
301 Protocol-Specific diagnostic page for SAS SSP	529
302 PHY TEST FUNCTION field	530
303 PHY TEST PATTERN field	531
304 PHY TEST PATTERN PHYSICAL LINK RATE field	531
305 PHY TEST PATTERN DWORDS CONTROL field	532
306 TWO_DWORDS phy test pattern examples	533
307 VPD pages with special requirements for SAS SSP	540
308 Device Identification VPD page designation descriptors for the SAS target port	540
309 Device Identification VPD page designation descriptors for the SAS target device	541
310 Logical Unit Information VPD page for SAS SSP	541
311 READY LED signal behavior	543
312 SMP request frame format	544
313 SMP functions (FUNCTION field)	544
314 SMP response frame format	546
315 FUNCTION RESULT field	547
316 Function result priority	550
317 REPORT GENERAL request	555
318 REPORT GENERAL response	555
319 NUMBER OF ZONE GROUPS field	559
320 REPORT MANUFACTURER INFORMATION request	561
321 REPORT MANUFACTURER INFORMATION response	562
322 REPORT SELF-CONFIGURATION STATUS request	564
323 REPORT SELF-CONFIGURATION STATUS response	565
324 Self-configuration status descriptor	567
325 STATUS TYPE field	567
326 REPORT ZONE PERMISSION TABLE request	569
327 REPORT TYPE field	569
328 REPORT ZONE PERMISSION TABLE response	570
329 Zone permission descriptor for a source zone group (i.e., s) with 128 zone groups	571
330 Zone permission descriptor for a source zone group (i.e., s) with 256 zone groups	571
331 Zone permission descriptor bit requirements	572
332 REPORT ZONE MANAGER PASSWORD request	572
333 REPORT ZONE MANAGER PASSWORD response	573
334 REPORT BROADCAST request	574
335 REPORT BROADCAST response	575
336 Broadcast descriptor	576
337 BROADCAST REASON field	577
338 DISCOVER request	578
339 DISCOVER response	579
340 ATTACHED DEVICE TYPE field	582
341 NEGOTIATED LOGICAL LINK RATE field	582
342 ATTACHED SATA PORT SELECTOR and ATTACHED SATA DEVICE bits	583
343 PROGRAMMED MINIMUM PHYSICAL LINK RATE and PROGRAMMED MAXIMUM PHYSICAL LINK rate fields	585
344 HARDWARE MINIMUM PHYSICAL LINK RATE and HARDWARE MAXIMUM PHYSICAL LINK RATE fields	586
345 ROUTING ATTRIBUTE field	587
346 ATTACHED DEVICE NAME field	588
347 SELF-CONFIGURATION STATUS field	589
348 SELF-CONFIGURATION LEVELS COMPLETED field	589
349 NEGOTIATED PHYSICAL LINK RATE field	590
350 REPORT PHY ERROR LOG request	592
351 REPORT PHY ERROR LOG response	593
352 REPORT PHY SATA request	594
353 REPORT PHY SATA response	595
354 REPORT ROUTE INFORMATION request	598
355 REPORT ROUTE INFORMATION response	599
356 REPORT PHY EVENT request	600

357 REPORT PHY EVENT response	601
358 Phy event descriptor	602
359 DISCOVER LIST request	603
360 PHY FILTER field	604
361 DESCRIPTOR TYPE field	604
362 DISCOVER LIST response	604
363 SHORT FORMAT descriptor	606
364 REPORT PHY EVENT LIST request	607
365 REPORT PHY EVENT LIST response	609
366 Phy event listdescriptor	610
367 REPORT EXPANDER ROUTE TABLE LIST request	611
368 REPORT EXPANDER ROUTE TABLE LIST response	612
369 REPORT EXPANDER ROUTE TABLE descriptor	613
370 CONFIGURE GENERAL request	615
371 STP SMP I_T NEXUS LOSS TIME field	616
372 CONFIGURE GENERAL response	617
373 ENABLE DISABLE ZONING request	618
374 SAVE field	618
375 ENABLE DISABLE ZONING field	619
376 ENABLE DISABLE ZONING response	619
377 ZONED BROADCAST request	620
378 BROADCAST TYPE field	621
379 ZONED BROADCAST response	621
380 ZONE LOCK request	623
381 ZONE LOCK response	624
382 ZONE ACTIVATE request	625
383 ZONE ACTIVATE response	625
384 ZONE UNLOCK request	626
385 ZONE UNLOCK response	627
386 CONFIGURE ZONE MANAGER PASSWORD request	628
387 SAVE field	628
388 CONFIGURE ZONE MANAGER PASSWORD response	629
389 CONFIGURE ZONE PHY INFORMATION request	630
390 SAVE field	631
391 Zone phy configuration descriptor	631
392 CONFIGURE ZONE PHY INFORMATION response	632
393 CONFIGURE ZONE PERMISSION TABLE request	633
394 NUMBER OF ZONE GROUPS field	634
395 SAVE field	634
396 Zone permission configuration descriptor for source zone group (i.e., s) for 128 zone groups	634
397 Zone permission configuration descriptor for source zone group (i.e., s) for 256 zone groups	635
398 Zone permission configuration descriptor bit requirements	635
399 CONFIGURE ZONE PERMISSION TABLE response	636
400 CONFIGURE ROUTE INFORMATION request	637
401 CONFIGURE ROUTE INFORMATION response	638
402 PHY CONTROL request	639
403 PHY OPERATION field	640
404 PROGRAMMED MINIMUM PHYSICAL LINK RATE and PROGRAMMED MAXIMUM PHYSICAL LINK RATE fields	643
405 PHY CONTROL response	643
406 PHY TEST FUNCTION request	644
407 PHY TEST FUNCTION field	645
408 PHY TEST PATTERN PHYSICAL LINK RATE field	646
409 PHY TEST FUNCTION response	646
410 CONFIGURE PHY EVENT request	647
411 Phy event configuration descriptor	648
412 CONFIGURE PHY EVENT response	648
A.1 JTPAT for RD+	650

A.2 JTPAT for RD-	651
A.3 JTPAT for RD+ and RD-	652
A.4 CJTPAT	654
A.5 CJTPAT with fixed content	658
D.1 CRC examples	683
E.1 Monte-Carlo simulation results	684
E.2 Hash results for simple SAS addresses	687
E.3 Hash results for realistic SAS addresses	687
E.4 Hash results for a walking ones pattern	688
E.5 Hash results for a walking zeros pattern	689
F.1 Scrambler examples	692
H.1 Minimum deletable primitive insertion rate examples	696
I.1 Zone permission table example initial value	697
I.2 CONFIGURE ZONE PERMISSION TABLE request example	698
I.3 Zone permission table after processing the first zone permission configuration descriptor	699
I.4 Zone permission table after processing the second zone permission configuration descriptor	699
J.1 Column descriptions for connection examples	701
K.1 Primitives with Hamming distance of 8	721
K.2 Primitives without Hamming distance of 8	723
L.1 C program files	724

Figures

	Page
1 SCSI document relationships	1
2 ATA document relationships	2
3 Classes in class diagrams	28
4 Association relationships in class diagrams	29
5 Aggregation relationships in class diagrams	30
6 Generalization relationships in class diagrams	31
7 Dependency relationships in class diagrams	31
8 Objects in object diagrams	32
9 State machine conventions	33
10 SAS Domain class diagram	37
11 Physical links and phys	38
12 Phy class diagram	39
13 SAS phy object diagram	40
14 Expander phy object diagram	41
15 Ports (narrow ports and wide ports)	43
16 Port class diagram	44
17 Port object diagram	45
18 SAS devices	46
19 Expander device	47
20 Expander Device class diagram	48
21 Domains	48
22 SAS domain bridging to ATA domains	49
23 SAS domains bridging to ATA domains with SATA port selectors	50
24 Devices spanning SAS domains	51
25 Single expander device topology example	51
26 Multiple expander device topologies	52
27 Potential pathways	53
28 Multiple connections on wide ports	55
29 State machines for SAS devices	61
30 State machines for expander devices	62
31 Transmit data path in a SAS phy	63
32 SSP link, port, SSP transport, and SCSI application layer state machines	64
33 SMP link, port, SMP transport, and management application layer state machines	65
34 STP link, port, STP transport, and ATA application layer state machines	66
35 Transmit data path and state machines in an expander phy	67
36 Receive data path in a SAS phy	68
37 Receive data path in an expander phy	69
38 State machines and SAS Device classes	70
39 State machine s and Expander Device classes	71
40 Reset terminology	72
41 Expander device model	75
42 Expander device interfaces	78
43 Expander device interface detail	79
44 Phy-based expander route table	86
45 Expander-based expander route table	87
46 Level-order traversal example	89
47 Examples of invalid topologies	92
48 Externally configurable expander device and table-to-table attachment	95
49 Expander route index levels example	96
50 Expander route index order example	98
51 One ZPSDS example	99
52 Zone manager location examples	100
53 Three ZPSDSes example	101
54 Extending a ZPSDS example	104
55 Overtaking a ZPSDS example	105

56 Zoning expander route table	110
57 Example of a ZPSDS	112
58 SATA connectors and cables	123
59 SAS Drive cable environments	124
60 SAS Drive backplane environment	124
61 SAS external cable environment	125
62 SAS internal symmetric cable environment - controller to backplane	125
63 SAS internal symmetric cable environment - controller to controller	126
64 SAS internal controller-based fanout cable environment	126
65 SAS internal backplane-based fanout cable environment	127
66 SAS Drive plug connector	129
67 Single-port SAS Drive cable receptacle connector	129
68 Dual-port SAS Drive cable receptacle connector	130
69 SAS Drive backplane receptacle connector	130
70 SAS 4i cable receptacle connector	133
71 SAS 4i plug connector	133
72 Mini SAS 4i cable plug connector	136
73 Mini SAS 4i receptacle connector	136
74 SAS 4x cable plug connector	139
75 SAS 4x receptacle connector	140
76 Mini SAS 4x cable plug connector	142
77 Mini SAS 4x cable plug connector that attaches to an enclosure out port	143
78 Mini SAS 4x cable plug connector that attaches to an enclosure in port	143
79 Mini SAS 4x receptacle connector	144
80 Mini SAS 4x receptacle connector - end device or enclosure universal port	145
81 Mini SAS 4x receptacle connector - enclosure out port	145
82 Mini SAS 4x receptacle connector - enclosure in port	145
83 Single-port SAS Drive cable assembly	147
84 Dual-port SAS Drive cable assembly	147
85 SAS internal symmetric cable assembly - SAS 4i	148
86 SAS internal symmetric cable assembly - Mini SAS 4i	149
87 SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i with vendor-specific sidebands	150
88 SAS internal symmetric cable assembly - SAS 4i controller to Mini SAS 4i backplane with SGPIO	152
89 SAS internal symmetric cable assembly - Mini SAS 4i controller to SAS 4i backplane with SGPIO	153
90 SAS internal controller-based fanout cable assembly - SAS 4i	155
91 SAS internal controller-based fanout cable assembly - Mini SAS 4i	156
92 SAS internal backplane-based fanout cable assembly - SAS 4i	157
93 SAS internal backplane-based fanout cable assembly - Mini SAS 4i	158
94 SAS external cable assembly - SAS 4x	159
95 SAS external cable assembly - Mini SAS 4x	160
96 SAS external cable assembly with Mini SAS 4x cable plug connectors	161
97 SAS external cable assembly - SAS 4x to Mini SAS 4x	162
98 SAS 4x and Mini SAS 4x cable assembly CT and CR compliance points	169
99 Backplane IT and IR compliance points	170
100 Backplane compliance points with SATA phy attached	171
101 SAS 4i and Mini SAS 4i cable assembly IT and IR compliance points	172
102 SAS 4i and Mini SAS 4i cable and backplane IT and IR compliance points	173
103 Internal cable and backplane IT and IR compliance points with SATA device attached	174
104 Internal cable IT and IR compliance points	175
105 Zero-length test load for transmitter device compliance point	176
106 Zero-length test load for receiver device compliance point	176
107 Zero-length test load S_{DD21}	177
108 TCTF test load	177
109 TCTF test load S_{DD21} and ISI loss requirements at IT at 3 Gbps	178
110 TCTF test load S_{DD21} and ISI loss requirements at CT at 3 Gbps	179
111 TCTF test load S_{DD21} and ISI loss requirements at IT at 1,5 Gbps	180
112 TCTF test load S_{DD21} and ISI loss requirements at CT at 1,5 Gbps	181

113 Low-loss TCTF test load	181
114 Low-loss TCTF test load S_{DD21} and ISI loss requirements at 3 Gbps	182
115 Transmitter device transient test circuit	186
116 Receiver device transient test circuit	186
117 Transmitter device eye mask	187
118 Receiver device eye mask	187
119 Deriving a receiver device jitter tolerance eye mask	188
120 Applied sinusoidal jitter	189
121 Center-spreading tolerance buffer	201
122 SAS bit transmission logic	213
123 SAS bit reception logic	214
124 OOB signal transmission	217
125 OOB signal detection	219
126 SATA port selection signal	220
127 SATA OOB sequence	221
128 SATA speed negotiation sequence	222
129 SAS to SATA OOB sequence	223
130 SAS to SAS OOB sequence	225
131 SNW-1, SNW-2, and Final-SNW	228
132 SNW-3	230
133 Train-SNW	235
134 SAS speed negotiation sequence SNW flowchart	237
135 SAS speed negotiation sequence (phy A: SNW-1 through SNW-3, phy B: SNW-2 only)	239
136 SAS speed negotiation sequence (both phys SNW-1 through SNW-3)	240
137 SAS speed negotiation sequence - phy reset problem in Final-SNW	241
138 SAS speed negotiation sequence - phy reset problem in SNW-3	242
139 SAS speed negotiation sequence - phy reset problem in Train-SNW	243
140 SAS speed negotiation sequence - multiple Train-SNWs	244
141 Hot-plug and the phy reset sequence	246
142 SP (phy layer) state machine - OOB sequence states	250
143 SP (phy layer) state machine - SAS speed negotiation states	256
144 SP (phy layer) state machine - SAS speed negotiation states for SNW-3 and Train-SNW	257
145 SP (phy layer) state machine - SATA host emulation states	265
146 SP (phy layer) state machine – SATA port selector state	269
147 SP (phy layer) state machine - SATA spinup hold state	270
148 SP_DWS (phy layer dword synchronization) state machine	272
149 Multiplexing disabled	277
150 Multiplexing enabled	277
151 Transmitting a repeated primitive sequence	290
152 Receiving a repeated primitive sequence	291
153 Extended primitive sequences	291
154 Triple primitive sequences	292
155 Redundant primitive sequences	293
156 Elasticity buffers	305
157 Address frame, SSP frame, and SMP frame CRC bit ordering	309
158 STP frame CRC bit ordering	310
159 Transmit path bit ordering	313
160 Receive path bit ordering	314
161 STP transmit path bit ordering	315
162 STP receive path bit ordering	316
163 Address frame transmission	316
164 Identification sequence	325
165 Hard reset sequence	326
166 SL_IR (link layer identification and hard reset) state machines	328
167 Example simultaneous connection recommendations for wide ports	336
168 Aborting a connection request with BREAK	344
169 Connection request timeout example	345

170 Closing a connection example	346
171 Rate matching example	348
172 SL (link layer for SAS logical phys) state machines (part 1)	350
173 SL (link layer for SAS logical phys) state machines (part 2)	351
174 XL (link layer for expander logical phys) state machine (part 1)	363
175 XL (link layer for expander logical phys) state machine (part 2)	364
176 XL (link layer for expander logical phys) state machine (part 3)	365
177 SSP frame transmission	377
178 Interlocked frames	379
179 Non-interlocked frames with the same tag	379
180 Non-interlocked frames with different tags	380
181 Closing an SSP connection example	381
182 SSP (link layer for SSP phys) state machines (part 1 - frame transmission)	383
183 SSP (link layer for SSP phys) state machines (part 2 - frame reception)	384
184 STP frame transmission	392
185 STP flow control	395
186 Transmitting a continued primitive sequence	396
187 Receiving a continued primitive sequence	397
188 Example simultaneous connection recommendations for an expander device	401
189 STP initiator port opening an STP connection	403
190 STP target port opening an STP connection	404
191 SMP frame transmission	405
192 SMP_IP (link layer for SMP initiator phys) state machine	407
193 SMP_TP (link layer for SMP target phys) state machine	409
194 Port layer examples	412
195 PL_OC (port layer overall control) state machine	415
196 PL_PM (port layer phy manager) state machine (part 1)	425
197 PL_PM (port layer phy manager) state machine (part 2)	426
198 Task management function sequence of SSP frames	447
199 Non-data command sequence of SSP frames	447
200 Write command sequence of SSP frames	448
201 Read command sequence of SSP frames	448
202 Bidirectional command sequence of SSP frames	449
203 ST_I (transport layer for SSP initiator ports) state machines	457
204 ST_T (transport layer for SSP target ports) state machines	473
205 Sequence of SMP frames	494
206 MT_IP (transport layer for SMP initiator ports) state machine	495
207 MT_TP (transport layer for SMP target ports) state machine	497
208 SA_PC (SCSI application layer power condition) state machine for SAS	535
A.1 CJTPAT pre-scrambling	653
B.1 A simple physical link	660
B.2 Transmitter device details	661
B.3 Receiver device details	662
B.4 De-embedding of connectors in test fixtures	664
B.5 Measurement conditions for signal output at the transmitter device	665
B.6 Transmitter device signal output measurement test fixture details	665
B.7 Measurement conditions for signal tolerance at the transmitter device	666
B.8 Calibration of test fixture for signal tolerance at the transmitter device	666
B.9 Measurement conditions for signal output at the receiver device	666
B.10 Measurement conditions for signal tolerance at the receiver device	667
B.11 Calibration of test fixture for signal tolerance at the receiver device	667
B.12 S-parameter port naming conventions	669
B.13 Four single-ended port or two differential port element	670
B.14 S-parameters for single-ended and differential systems	670
B.15 Measurement conditions for upstream return loss at the transmitter device connector	671
B.16 Measurement conditions for downstream return loss at the receiver device connector	672
B.17 Measurement conditions for downstream return loss at IT or CT	673

B.18 Measurement conditions for upstream return loss at IR or CR	674
C.1 SAS speed negotiation sequence (phy A: SNW-1 only, phy B: SNW-1 only)	675
C.2 SAS speed negotiation sequence (phy A: SNW-1, SNW-2, phy B: SNW-1, SNW-2)	676
C.3 SAS speed negotiation sequence (phy A: SNW-1, SNW-2, SNW-3, phy B: SNW-1, SNW-2)	677
C.4 SAS speed negotiation sequence (phy A: SNW-2, SNW-3, phy B: SNW-1, SNW-2)	678
C.5 SAS speed negotiation sequence (phy A: SNW-1 only, phy B: SNW-2 only)	679
D.1 CRC generator example	680
D.2 CRC checker example	680
E.1 BCH(69, 39, 9) code generator	685
F.1 Scrambler	690
J.1 Example topology	700
J.2 Connection request - OPEN_ACCEPT	702
J.3 Connection request - OPEN_REJECT by end device	703
J.4 Connection request - OPEN_REJECT by expander device	704
J.5 Connection request - arbitration lost	705
J.6 Connection request - backoff and retry	706
J.7 Connection request - backoff and reverse path	707
J.8 Connection close - single step	708
J.9 Connection close - simultaneous	709
J.10 BREAK handling during path arbitration when the BREAK_REPLY method is disabled	710
J.11 BREAK handling during a connection when the BREAK_REPLY method is disabled	711
J.12 BREAK handling during path arbitration when the BREAK_REPLY method is enabled	712
J.13 BREAK handling during a connection when the BREAK_REPLY method is enabled	713
J.14 STP connection - originated by STP initiator port	714
J.15 STP connection - originated by STP target port in an STP/SATA bridge	715
J.16 STP connection close - originated by STP initiator port	716
J.17 STP connection close - originated by STP target port in an STP/SATA bridge	717
J.18 XL1:Request_Path to XL5:Forward_Open transition	718
J.19 Partial pathway recovery	719
M.1 SAS primary icon	763
M.2 SAS alternate icon	763
M.3 SAS alternate icon with SAS letters	764

Revision Information

R.1 Revision sas2r00 (1 October 2005)

First release of SAS-2. The project proposal was 05-060r0.

Incorporated these changes per the January 2005 CAP WG (05-035r0) and T10 plenary (05-036r2):

- a) 04-372r2 SAM-4 SPC-4 SAS-2 I_T NEXUS RESET task management function (Rob Elliott, HP)
- b) 04-297r2 SAM-4 SAS-2 FCP-3 Retry delay (George Penokie, IBM)

No changes were approved in the March 2005 T10 plenary (05-097r0) or the May 2005 T10 plenary (05-165r1).

Incorporated these changes per the July 2005 SAS protocol WG (05-272r0) and T10 plenary (05-275r0):

- a) 05-167r5 SAS-2 New NOTIFY to indicate imminent power failure (George Penokie, IBM)

Incorporated these changes per the September 2005 SAS protocol WG (05-335r0) and T10 plenary (05-338r0):

- a) 05-299r1 Correct log page format tables in SPC-4, SBC-3, and SAS-2 (Ralph Weber, ENDL)
- b) 05-305r0 Maximum SMP response time (Rob Elliott, HP)
- c) 05-307r0 SAS-2 Modifications to NOTIFY to indicate imminent power failure (Mark Evans, Maxtor), dropping the proposed note that just restates the rule being added.

R.2 Revision sas2r01 (13 November 2005)

Incorporated these changes per the November 2005 SAS protocol WG (05-417r0) and T10 plenary (05-420r0):

- a) 04-222r6 SAS-2 More phy test patterns (Rob Elliott, HP). Added the PHY TEST PATTERN DWORDS CONTROL and PHY TEST PATTERN DWORDS fields to the SMP PHY TEST FUNCTION function.
- b) 05-306r2 SAS-2 STP connection time limits and STP/SMP I_T nexus loss (Rob Elliott, HP). Corrected all the length field values.
- c) 05-412r0 SAS-2 revision 0 minor corrections (Rob Elliott, HP)
- d) 05-370r2 SAS-2 NOTIFY (POWER FAILURE EXPECTED) fixes (George Penokie, IBM). Included a PROTOCOL IDENTIFIER field in the new subpage to follow the convention of the Protocol-Specific mode pages
- e) Changed the ST_ITS3 and ST_ITS4 states to set the TARGET PORT TRANSFER TAG field to FFFFh rather than zero (for COMMAND and TASK frames) to agree with the definition of the field in 9.2.1 (found by Ron Roberts, Broadcom)

R.3 Revision sas2r02 (18 January 2006)

Incorporated these changes per the January 2006 SAS protocol WG (06-038r0) and T10 plenary (06-041r0):

- a) 06-055r0 (aka 04-172r4) SAS-2 More counters (Rob Elliott, HP)
- b) 05-309r3 SAS-2 Add device name to IDENTIFY address frame (Rob Elliott, HP)
- c) 06-028r1 SAS-2 Redundant primitive sequence handling (Steve Gorshe, PMC-Sierra)
- d) 06-054r1 SAS-2 Expander issue resolutions (Rob Elliott, HP) covering issues raised by 05-373r2 SAS-2 Expander issues (Craig Stoops, Expert I/O)

R.4 Revision sas2r03 (20 March 2006)

Incorporated these changes per the March 2006 SAS protocol WG (06-124r0) and T10 plenary (06-127r0):

- a) 06-122r2 SAS-2 zoning - phy features (Ralph Weber, ENDL). Conflicts with 06-082r2 about whether the SMP function contains source or destination zone group numbers, especially related to the zone address resolved method, are described with editor's notes. How source zone group 2 is handled was not mentioned; made it based on the zone permission table like all other source zone groups and included an editor's note that more rewrite may be forthcoming. Vague rule about not sending "zoning information" out certain phys tagged with an editor's note. Included editor's notes by each reference to an SMP zone configuration function that is not yet defined.
- b) 06-130r1 SAS-2 sticky zone groups (Kevin Marks, Dell)

- c) 06-082r2 SAS-2 zoning-related SMP functions (Ralph Weber, ENDL and Steve Johnson, LSI Logic) - adds the ORIGINATE BROADCAST PRIMITIVES function. Changed name of the function to ZONED BROADCAST.
- d) 06-083r1 SAS-2 Handle STP CLOSE and SATA_X_RDY crossing on the wire (Bob Sheffield, Intel)
- e) 06-099r0 SAS-2 Transitions from SL_CC1:ArbSel (Brian Day, LSI Logic). Used "SOAF/Data Dwords/EOAF Transmitted message" instead of "transmitted the OPEN address frame", and changed two more instances of "transmitted the OPEN address frame" in Transition SL_CC1 to SL_CC2 and Transition SL_CC1 to SL_CC3.
- f) 06-101r0 SAS-2 Minor corrections to STP flow control formula (Brian Day, LSI Logic)
- g) 06-134r0 SAS-2 Renumber phy event information codes (Rob Elliott, HP). Also renumbered "Received SMP frame error count" from 62h to 63h to leave room for a possible "Transmitted SMP frame error count" value at 62h.
- h) In the edge expander device rules, changed "no phy using the subtractive routing method is available" to "...exists" for generation of Arb Reject (No Destination).

R.5 Revision sas2r03a (22 April 2006)

Incorporated these changes per the 23 March 2006 SAS physical WG teleconference (06-171r0) and the 20 April 2006 SAS zoning WG (06-205r0) (subsequently approved by the May T10 plenary (06-233r0)):

- a) 06-181r2 SAS-2 Zone group values after a link reset sequence (Kevin Marks, Dell)
- b) 06-165r0 SAS-2 Change SMP ZONE VIOLATION code values (Rob Elliott, HP)
- c) 06-167r1 SAS-2 Filtering OPEN content based on ZONE PARTICIPATING bit (Rob Elliott, HP)
- d) 06-168r1 SAS-2 OPEN address frame SOURCE ZONE GROUP field definition (Rob Elliott, HP)
- e) 06-166r4 SAS-2 Restrict access to SMP ZONED BROADCAST function (Rob Elliott, HP)
- f) 06-176r2 SAS-2 Add phy zone information to SMP DISCOVER (Tim Symons, PMC-Sierra and Rob Elliott, HP)
- g) 06-169r1 SAS-2 Correct receiver device jitter table footnotes (Rob Elliott, HP)
- h) added SAM-4 and SPC-4 to references list; changed SAM-3 reference to SAM-4 for the RETRY DELAY TIMER field in the SSP RESPONSE frame (did not change any others yet)
- i) added dword counts to each JTPAT table, and the INFORMATION UNIT field name to the left column of the CJTPAT table

R.6 Revision sas2r04 (16 May 2006)

Incorporated these changes per the May 2006 SAS protocol WG (06-230r1) and T10 plenary (06-233r0):

- a) 06-161r0 SAS-2 Alternate icon (Rob Elliott, HP)
- b) 06-119r2 SAS-2 BREAK_REPLY (Tim Hoglund, LSI Logic)
- c) 06-164r2 SAS-2 Require expanders transmit three AIPs (Rob Elliott, HP)
- d) 06-199r1 SAS-2 No BROADCASTs before link reset sequence completes (Luben Tuikov, Vitesse and Rob Elliott, HP)
- e) 06-044r3 SAS-2 BROADCAST (ASYNCHONOUS EVENT) (Steve Fairchild, HP)
- f) 06-177r3 SAS-2 ZONED BROADCAST clarification (Ed D'Avignon, Vitesse and Rob Elliott, HP)
- g) 06-097r4 SAS-2 Discovery - Configuring bit (Tim Symons, PMC-Sierra)
- h) 06-212r1 SAS-2 Zone group of an expander's SMP (Ed D'Avignon, Vitesse). Placed the rule in the zone phy information section defining the zone group field rather than in the zone permission table section.
- i) Changed RX1/RX2/RX3/RX4 to Rx 0/Rx 1/Rx 2/Rx 3 in the table of Additional requirements for SAS 4x cable assemblies
- j) Changed RXn to Rx n and TXn to Tx n in the table of Additional requirements for SAS 4i cable assemblies
- k) Normalized the table formatting for descriptor lists
- l) Changed COMPONENT REVISION ID to COMPONENT REVISION LEVEL in the REPORT MANUFACTURER INFORMATION response table to match the usage below the table
- m) Changed "not enabled" to "disabled" globally (just two times, all related to first burst)
- n) Changed "forwarded OPEN_REJECTs" to "OPEN_REJECTs in response to forwarded OPEN address frames" in the phy event information sources 22h and 24h
- o) Added "An SSP initiator port does not reuse a tag until it receives indication from the SSP target port that the tag is no longer in use (see 9.2.4, 9.2.5, and 10.2.2)." in 9.2.1 definition of the TAG field.

R.7 Revision sas2r04a (25 June 2006)

Incorporated these changes per the 20 June 2006 SAS zoning WG (06-290r0) (subsequently approved by the July T10 plenary (06-319r0)):

- a) 06-197r3 SAS-2 Add expander change count to most SMP functions (Rob Elliott, HP)
- b) 06-208r2 SAS-2 Restrict access to SMP write functions (Rob Elliott, HP)
- c) 06-213r2 SAS-2 REPORT GENERAL additions for zoning and self configuration (Steve Johnson, LSI Logic)
- d) 06-037r7 SAS-2 DISCOVER LIST function (Steve Johnson, LSI Logic). Throughout the standard, wherever DISCOVER was referenced, added mention of DISCOVER LIST if it is equally appropriate.
- e) 06-078r3 SAS-2 REPORT ROUTE TABLE function (Steve Johnson, LSI Logic). Updated the model to support this function. Defined two types of expander route tables - phy-based (accessed with the old functions, indexed by numbers dictated by the discover process) and expander-based (accessed with this function, indexed by "routed SAS address"). The latter is called a zoning expander route table in a zoning expander device.
- f) Changed "zoning expander device" to mean zoning is supported, not necessarily enabled. Sprinkled "if zoning is enabled" throughout.
- g) In 4.8.1, tweaked introduction from "Every phy in a ZPSDS belongs to a zone group" to "Every phy in the SAS domain may belong to a zone group." Mentioned that phys within the ZPSDS are in zone group 1, phys attached to the boundary pick up their zone groups from the zone phy information of the expander device to which they are attached, and phys beyond the boundary are in zone groups if the address resolved method is being used.

R.8 Revision sas2r05 (21 July 2006)

Incorporated these changes per the July 2006 SAS protocol WG (06-316r0), SAS physical WG (06-348r0), and T10 plenary (06-319r0):

- a) 05-322r4 SAS-2 Wide SSP target port simultaneous connection rules (Rob Elliott, HP)
- b) 06-187r2 SAS-2 Self-configuring expander status (Rob Elliott, HP)
- c) 06-273r1 SAS-2 Bus Inactivity Timer is Broken (Steve Finch, STMicroelectronics). Also changed all the timer-related unordered lists into ordered lists.
- d) 06-302r1 SAS-2 Multiple broadcasts on reset (George Penokie, IBM)
- e) 06-189r3 SAS-2 Allow table-to-table expander attachment (Rob Elliott, HP)
- f) 06-304r2 SAS-2 SMP function result priority (Zenta Darnell, Vitesse). Deleted PHY VACANT from the list of DISCOVER LIST responses; it can appear in the full DISCOVER responses but not in the outer DISCOVER LIST response itself. Added SMP ZONE VIOLATION to CONFIGURE GENERAL, PHY CONTROL, PHY TEST FUNCTION, and CONFIGURE PHY EVENT INFORMATION, with a priority just below PHY VACANT.
- g) 06-210r4 SAS-2 Reporting ZONE PARTICIPATING CAPABLE in the IDENTIFY address frame (Kevin Marks, Dell). Renamed ZONE PARTICIPATING in bit names to INSIDE ZPSDS to help the DISCOVER response table fit on the page (still requires a drop in font size).
- h) 06-332r1 SAS-2 PL_PM3 fixes (George Penokie, IBM)
- i) 06-259r1 SAM-4 et. al. making linked commands obsolete (Mark Evans, Maxtor)
- j) 06-323r1 SAM-4 et. al. Multiple service delivery subsystem editorial tweaks (Rob Elliott, HP). Updated all references to SAM-3 to SAM-4, SPC-3 to SPC-4, SBC-2 to SBC-3, and ATA/ATAPI-7 V1 to ATA8-AAM and ATA8-ACS. In the Protocol-Specific Port log parameter, merged the LBIN and LP bits into a FORMAT AND LINKING field, added a DS bit and an SPF bit in byte 0, and marked the DS bit in the parameter control byte obsolete to agree with SPC-4.
- k) corrected dword numbers in JTPAT tables
- l) fixed ms to μ s in the STP BUS INACTIVITY TIME LIMIT field and STP MAXIMUM CONNECT TIME LIMIT field descriptions (Symbol font was not preserved when 05-306r2 was incorporated)
- m) use "management device server" rather than "SMP target port" where appropriate in chapter 10
- n) added OPEN_REJECT (ZONE VIOLATION) handling throughout the state machines, generally wherever OPEN_REJECT (PROTOCOL NOT SUPPORTED) is handled. Add Arb Reject (Zone Violation) as lower priority than Arb Reject (Bad Connection Rate).
- o) added OPEN_REJECT (RESERVED ...) handling throughout the state machines wherever their "process as" peers are handled

- p) 05-251r0 requested deletion in SAS-1.1 of the last three words of “shall not complete the [START STOP UNIT] command with GOOD status” in the Power Condition state machine transition state descriptions, but that change was missed. 06-299r1 requests the same change again.
- q) Added “The ADDITIONAL RESPONSE BYTES may be present but shall be ignored.” to SMP function results of INVALID EXPANDER CHANGE COUNT, PHY EVENT INFORMATION SOURCE NOT SUPPORTED, and SMP ZONE VIOLATION. Although the functions that return those don’t yet return ADDITIONAL RESPONSE BYTES, functions could do so in the future and that general rule is intended to apply.
- r) Added a figure in front of the CJTPAT table showing how pre-scrambling the desired pattern results in the desired pattern on the wire.

R.9 sas2r05a (21 July 2006)

Incorporated the following proposals per the July SAS zoning WG (06-349r0)(r3 of each was subsequently approved by the September 2006 T10 plenary (06-417r0):

- a) 06-279r2 SAS-2 Allow more than one ZPSDS in a SAS domain (Rob Elliott, HP)
- b) 06-281r2 SAS-2 Enable and disable zoning (Rob Elliott, HP)

R.10 sas2r05b (1 September 2006)

Incorporated the following proposals per the August SAS zoning WG (06-376r1)(subsequently approved by the September 2006 T10 plenary (06-417r0)):

- a) 06-201r6 SAS-2 SMP CONFIGURE ZONE PHY INFORMATION function (Tim Symons, PMC-Sierra)
- b) 06-203r6 SAS-2 SMP REPORT ZONE PERMISSION TABLE function (Tim Symons, PMC-Sierra)
- c) 06-279r3 SAS-2 Allow more than one ZPSDS in a SAS domain (Rob Elliott, HP)
- d) 06-281r3 SAS-2 Enable and disable zoning (Rob Elliott, HP)
- e) 06-286r5 SAS-2 SMP ZONE LOCK function (Tim Symons, PMC-Sierra)
- f) 06-288r6 SAS-2 SMP ZONE ACTIVATE function (Tim Symons, PMC-Sierra)
- g) 06-289r6 SAS-2 SMP ZONE UNLOCK function (Tim Symons, PMC-Sierra)
- h) 06-377r1 SAS-2 Broadcast (Zone Activate) (Tim Symons, PMC-Sierra)
- i) Corrected “To” columns for BROADCAST primitives
- j) Renamed Arb Reject (Bad Connection Rate) to Arb Reject (Connection Rate Not Supported)
- k) Added function response priorities for ENABLE DISABLE ZONING
- l) In DISCOVER LIST SHORT FORMAT descriptor, defined the PHY IDENTIFIER field as always valid (from Douglas Gilbert). In the SHORT FORMAT descriptor, defined what to do with the remaining fields if the FUNCTION RESULT field is PHY VACANT or PHY DOES NOT EXIST.
- m) Added definitions on UNKNOWN DESCRIPTOR TYPE (renamed from UNKNOWN DESCRIPTOR LIST) and UNKNOWN PHY FILTER for the DISCOVER LIST function
- n) Renamed PHY EVENT INFORMATION NOT SUPPORTED to UNKNOWN PHY EVENT INFORMATION to match other “not supported” function result names
- o) Changed IDENTIFY address frame FEATURES field from “shall be set to zero” to a table showing 0h as the only defined value and all others as reserved. This should help people realize that non-zero values must be checked and treated as an error.
- p) Added footnotes to tables containing values that are a multiple of OOB indicating the nominal times they represent.

R.11 Revision sas2r06 (22 September 2006)

Incorporated these changes per the September 2006 SAS protocol WG (06-414r0), SAS physical WG (06-432r0), and T10 plenary (06-417r0):

- a) 06-202r9 SAS-2 SMP CONFIGURE ZONE PERMISSION function (Tim Symons, PMC-Sierra)
- b) 06-263r6 SAS-2 Spread spectrum clocking (Rob Elliott, HP)
- c) 06-299r1 SAS-2 Clarifications of the SCSI power conditions in SAS (Chris Owens and Kevin Marks, Dell)
- d) 06-326r2 SAS-2 SMP zone lock timer (Tim Symons, PMC-Sierra)
- e) 06-358r5 SAS-2 Zone configuration model (Tim Symons, PMC-Sierra)
- f) 06-384r1 SAS-2 OPEN_REJECT RETRY during zoning changes (Rob Elliott, HP)
- g) 06-402r0 SAS-2 Changes to ATTACHED SATA DEVICE bit (Brian Day, LSI Logic)
- h) 06-405r0 SAS-2 Transport layer retries fix (George Penokie, IBM)

- i) 06-408r1 SAS-2 Race condition for transition out of SP12:SAS_Fail state (Bill Martin, Sierra Logic)
- j) 06-409r1 SAS-2 IDENTIFY address frame REASON field (Rob Elliott, HP)
- k) 06-418r1 SAS-2 ENABLE DISABLE ZONING revision (Tim Symons, PMC-Sierra)

R.12 Revision sas2r07 (15 November 2006)

Incorporated these changes per the November 2006 SAS protocol WG (06-485r0), SAS physical WG (06-501r0), and T10 plenary (06-488r0):

- a) 05-381r7 SAS-2 Multiplexing (Rob Elliott, HP). Moved the model section from chapter 7 (link layer) to chapter 6, since it's negotiated in the phy layer now. Split the model section into 2 parts, the multiplexing sequence described in the phy reset sequence section and multiplexing overall described at the 6.xx level. Assign MUX (LOGICAL LINK 0) to a different primitive encoding than proposed, since BREAK_REPLY has already been assigned that encoding. Made most of the states in SP_DWS sensitive to Incorrect Mux.
- b) 06-473r0 SAS-2 REPORT EXPANDER ROUTE TABLE descriptor layout change (Rob Elliott, HP)
- c) 06-474r1 SAS-2 Broadcast (Zone Activate) only by ZONED BROADCAST (Rob Elliott, HP)
- d) 06-463r3 SAS-2 OOB transmission requirements (Alvin Cox, Seagate)
- e) 06-464r0 SAS-2 COMWAKE detection requirements (Alvin Cox, Seagate)
- f) Fixed location of "within 1 ms" phrase in NOTIFY (POWER LOSS EXPECTED) section per 05-307r0.
- g) Editorial changes from 06-489r0 SAS-2 Target transport layer read DATA flowchart (George Penokie, IBM)

R.13 Revision sas2r08 (26 January 2007)

Incorporated these changes per the January 2007 SAS protocol WG (07-031r0), SAS physical WG (07-043r0), and T10 plenary (07-034r0):

- a) 06-515r0 SAS-2 Modifications to SAS Speed Negotiation (Steve Finch, ST Microelectronics and Amr Wassal, PMC-Sierra) [which was a continuation of 06-324 and 06-295]
- b) 06-188r3 SAS-2 Support multiple STP affiliations (Rob Elliott, HP)
- c) 06-275r0 SAS-2 ALIGN insertion rate during STP connections (Rob Elliott, HP)
- d) 06-322r4 SAS-2 Response to abandon-class OPEN_REJECT (Rob Elliott, HP)
- e) 06-451r6 SAS-2 SAM-4 Miscellaneous state machine fixes (George Penokie, IBM)
- f) 06-466r1 SAS-2 OPEN_REJECT RETRY during self-configuration changes (Rob Elliott, HP)
- g) 06-470r4 SAS-2 Transport layer read data flowchart (George Penokie, IBM)
- h) 06-471r1 SAS-2 Change to phy reset sequence 10ms rule (Brian Day, LSI Logic)
- i) Changed ISO/IEC number for ATA/ATAPI-7 from 14776 to 27439

R.14 Revision sas2r09 (22 March 2007)

Incorporated these changes per the March 2007 SAS protocol WG (07-114r0), CAP WG (07-116r0), SAS physical WG (07-140r0), and T10 plenary (07-117r0):

- a) 06-467r3 SAS-2 Initiator handling of retransmit read DATA frames (Bob Sheffield, Intel)
- b) 06-478r2 SAS-2 Changes to NEGOTIATED PHYSICAL LINK RATE (Brian Day, LSI Logic). Assigned NEGOTIATED PHYSICAL LINK RATE field value of 6h for UNSUPPORTED_PHY_ATTACHED.
- c) 07-008r8 SAS-2 Expander notification of temporary shutdown (George Penokie, IBM) with these exceptions:
 - A) Changed "reduced function" to "reduced functionality"
 - B) Changed the "time to reduced function timer" to "reduced functionality delay" timer
 - C) In the OPEN_REJECT (RETRY) definition, added "c) an expander device in the pathway has reduced functionality (see 4.6.8)"
 - D) In the ECM Arb Reject confirmation rules, added "Arb Reject (Retry) if the expander device is unable to process the connection request because it has reduced functionality (see 4.6.8)"
- d) 07-027r2 SAS-2 Enabling and disabling Transport Layer Retries (Chris Martin and Rob Elliott, HP)
- e) 07-028r1 SAS-2 Remove address resolved zoning (Tim Symons, PMC-Sierra)
- f) 07-058r3 SAS-2 OOB and SSC (Steve Finch, STMicroelectronics)
- g) 07-059r1 SAS-2 Update SATA references to Serial ATA 2.6 (Rob Elliott, HP)
- h) 07-066r1 SAM-4 SAS-2 QUERY TASK SET task management function (Rob Elliott, HP)
- i) 07-076r1 SAM-4 SAS-2 QUERY UNIT ATTENTION task management function (Rob Elliott, HP)
- j) 07-082r1 SAS-2 Fix target device name PIV bit (Rob Elliott, HP)

- k) 07-083r0 SAS-2 Mini SAS 4i to SAS 4i cable assemblies with SGPIO (Barry Olawsky and Rob Elliott, HP)
- l) 07-084r1 SAS-2 Add ATTACHED REASON field (Rob Elliott, HP)
- m) 07-085r0 SAS-2 Change mode page subpage names (Rob Elliott, HP)
- n) 07-087r1 SAS-2 SES-2 Enclosure Connector Information (Brad Besmer, LSI Logic)
- o) 07-108r1 SAS-2 Transport level read fixes (George Penokie, IBM) with these exceptions:
 - A) In ST_TTS3, applied the “Read Data Frames Transmitted/ACKed to zero” to both the “with a Retry argument” lists, not the “without a Retry argument” list.
 - B) In ST_TTS2, added “minus the Data-In Application Client Buffer Offset argument” in two more places where the Read Data Offset variable was being compared to the Data-In Request Byte Count argument.
 - C) In ST_TTS3, noted two places where “plus the Data-In Application Client Buffer Offset argument” still needs to be added.
- p) 07-121r1 SAS-2 Add REASON field to DISCOVER and DISCOVER LIST (Steve Johnson, LSI Logic)
- q) Eliminated use of the word “comprise”
- r) Clarified in the SNW-3 REQUESTED LOGICAL LINK RATE field that bit 4 is the MSB; bit 7 is the LSB. Corrected the parity bit value in the example with rate of 9h; added another example with rate of 8h which highlights the endianness.
- s) Corrected cross-references relating to RETRANSMIT bit set to one for XFER_RDY and RESPONSE frames in ST_TTS and for TASK frames in ST_ITS

R.15 Revision sas2r09a (23 April 2007)

Incorporated these changes per the April 2007 SAS protocol WG (07-183r0). The SAS physical WG (07-183r0) did not recommend anything. Proposals are not yet approved by a T10 plenary:

- a) 07-124r0 SAS-2 Remove Messages Between State Machines annex (Rob Elliott, HP)
- b) 07-128r0 SAS-2 Target handling of retransmitted write DATA frames (Bob Sheffield, Intel)
- c) 07-167r0 SAS-2 Add SNTT timer to SP27:SAS_Settings (Bob Sheffield, Intel)
- d) 07-154r1 SAS-2 ATTACHED SAS ADDRESS for virtual phys (Bob Sheffield, Intel)
- e) 07-176r1 SAS-2 SMP function result for incomplete descriptor lists (Rob Elliott, HP)
- f) 07-178r0 SAS-2 Connection request livelock avoidance (Rob Elliott, HP)
- g) 07-155r0 SAS-2 Add expander reduced functionality to REASON field (Rob Elliott, HP)

R.16 Revision sas2r10 (15 May 2007)

Incorporated these changes per the May 2007 SAS protocol WG (07-210r0), SAS physical WG (07-243r0), and T10 plenary (07-213r0):

- a) 06-373r3 SAS-2 Zone manager key (Rob Elliott, HP). Assigned CONFIGURE ZONE MANAGER PASSWORD to function code 89h.
- b) 06-476r2 SAS-2 WWN-based Attached Device Name for SATA (Rob Elliott, HP)
- c) 07-017r2 SAS-2 SAS-2 More zone groups (Steve Johnson, LSI Logic)
- d) 07-094r2 SAS-2 UML changes for logical phys (George Penokie, IBM)
- e) 07-166r2 SAS-2 Clarify scope of retransmitted XFER_RDY (Bob Sheffield, Intel)
- f) 07-177r2 SAS-2 Port layer wide port ordering (Rob Elliott, HP)
- g) 07-209r0 SAS-2 Mini SAS 4x cable plug pull tab color (Rob Elliott, HP and Jay Neer, Molex)
- h) 07-228r0 SAS-2 ResetStatus port selector fixes (Brian Day, LSI)
- i) 07-233r0 SAS-2 Function result for 256 zone support (Tim Symons, PMC-Sierra). Renamed the value ZONE GROUP OUT OF RANGE and let the result continue to be applied to the CONFIGURE ZONE PHY INFORMATION function if the ZONE GROUP field is too large.
- j) Changed some uses of “detected” to “received”
- k) Corrected DJ CR 3 Gbps footnote to point to the measurement bandwidth of 1 800 kHz to 1 500 MHz, which was correct in sas-r05 but broken in SAS-1.1 when the table cells were rearranged.
- l) In class and object diagrams, changed class and attribute names to mixed cases to parallel a change planned for SAM-4 revision 11.
- m) In class and object diagrams, shaded class name cells tan and object name cells blue; removed any shading from attribute and operation cells.
- n) Added some multiplexing examples to the “Multiple connections on wide ports” figure

- o) Deleted “and in the SMP DISCOVER LIST response” from numerous places. It was not mentioned everywhere that DISCOVER was mentioned, so rather than imply different behavior with inconsistency, just removed the phrase.
- p) Define that the link layer SSP transmitter and the SMP transmitter are responsibly for inserting the CRC at the end of the frame (not the port layer or transport layer).
- q) Added the SMP Port class to the “State machines and Expander Device classes” class diagram with Port Layer and Transport Layer classes underneath it, paralleling the Port class diagram

R.17 Revision sas2r11 (23 July 2007)

Incorporated these changes per the July 2007 SAS protocol WG (07-313r0), SAS physical WG (07-338r0), CAP WG (07-315r0), and T10 plenary (07-316r0):

- a) 07-039r4 SAS-2 Self-Configuration Status Updates (Tyson Hartshorn and Stephen Johnson, LSI)
- b) 07-075r5 SAS-2 SMP REPORT BROADCAST function (George Penokie, IBM). Assigned function number 06h, since 05h is already taken.
- c) 07-091r3 SAS-2 SMP DISCOVER support for SNW-3 phy capabilities (Rob Elliott, HP). Placed the NEGOTIATED SSC bit in the DISCOVER response in byte 95, since byte 94 bit 4 is now taken by the REASON field.
- d) 07-102r3 SAS-2 Changes to Report Phy Event logging and reporting (Tyson Hartshorn, LSI). Assigned DISCOVER LIST to 20h, REPORT PHY EVENT INFORMATION LIST to 21h, and REPORT SELF-CONFIGURATION STATUS LIST to 22h.
- e) 07-130r4 SAS-2 CRC fixes (Luben Tuikov, Vitesse). Addressed preliminary letter ballot comments.
- f) 07-153r1 SAS-2 SPC-4 Protocol-Specific VPD pages (Rob Elliott, HP)
- g) 07-214r1 SAS-2 Mode and log page support for SNW-3 phy capabilities (Rob Elliott, HP)
- h) 07-277r0 SAS-2 SMP PHY CONTROL and affiliations (Rob Elliott, HP). Assigned AFFILIATION VIOLATION to 1Ah rather than 1Bh, since 07-287 deletes the assignment of 1Ah to LOGICAL LINK RATE NOT SUPPORTED.
- i) 07-280r1 SAS-2 SMP DISCOVER virtual phy clarifications (Rob Elliott, HP)
- j) 07-284r0 SAS-2 OPEN_REJECT ZONE VIOLATION priority (Rob Elliott, HP)
- k) 07-285r0 SAS-2 STP bus inactivity time limit clarification (Rob Elliott, HP)
- l) 07-287r0 SAS-2 SMP function result tweaks (Rob Elliott, HP)
- m) 07-288r0 SAS-2 TARGET PORT TRANSFER TAG ST_T state machine fix (Brian Day, LSI)
- n) 07-293r0 SAS-2 Zoning interaction with enclosure services (Rob Elliott, HP)
- o) 07-297r1 SAS-2 CJTPAT usage (Rob Elliott, HP)
- p) 07-300r1 SAS-2 Configurable expander table-to-table error handling (Rob Elliott, HP)
- q) 07-306r1 SAS-2 Route table discovery process (Tim Symons, PMC-Sierra)
- r) 07-307r1 SAS-2 Zone group valid bit (Tim Symons, PMC-Sierra)
- s) 07-311r1 SAS-2 More Transport Layer Retries Fixes (George Penokie, IBM)
- t) 07-312r2 SAS-2 Zone route table entries for subtractive ports (Tim Symons, PMC-Sierra). Merged the sentiments of the last proposed paragraph into item b).
- u) Corrected maximum COMMAND frame size from 284 to 280 bytes (Brian Day, LSI)
- v) Added NOTE in discover process section that old expander devices don't implement DISCOVER LIST.
- w) In the SAS phy transmitter SSC modulation table footnote on “None or downspreading”, fixed “attach to SAS” to “support being attached to SATA devices”, added the ppm range for SAS downspreading as an i.e., changed the SATA ppm range to not include +/- 350 ppm, and made other changes to try to address concerns from Doug Loree, Toshiba
- x) Fixed “hexadecimal” column for OPEN_REJECT

R.18 Revision sas2r12 (28 September 2007)

Incorporated these changes per the September 2007 SAS protocol WG (07-394r0) and T10 plenary (07-396r0):

- a) 07-317r2 SAS-2 Target transport layer write flowcharts (George Penokie, IBM)
- b) 07-334r2 SAS-2 Add minimum number of dwords after IDENTIFY frame (Gerry Houlder, Seagate)
- c) 07-388r1 SAS-2 Only issue BROADCAST (Change) for initial spinup hold (Brian Day, LSI)
- d) 07-401r0 SAS-2 SMP DISCOVER LIST field alignment (Brad Besmer, LSI)
- e) 07-305r3 SAS-2 Zone phy information clarifications (Rob Elliott, HP)

- f) In REPORT ZONE PERMISSION TABLE, renamed SOURCE INDEX DOES NOT EXIST to SOURCE ZONE GROUP DOES NOT EXIST and assigned it function result code 28h (missed while incorporating 06-203r6 into sas2r05b)
- g) Corrected last scrambler output in CJTPAT table (the one XORed with the CRC).
- h) Corrected header and CRC for CJTPAT with fixed content (the header with mostly zeros is not the desired output on the physical link)
- i) Redrew the CJTPAT scrambling figure to show the frame header and CRC not being pre-scrambled. Changed the column order in the tables to have pre-scrambled values on the left and physical link values on the right.
- j) In REPORT GENERAL response, changed the minimum increment rule for the EXPANDER CHANGE COUNT field from REPORT GENERAL responses to any SMP response frame containing an EXPANDER CHANGE COUNT field
- k) In REPORT GENERAL response, changed the note about wrapping to discuss 65 535 rather than 65 536 increments, since the field now wraps to 0001h rather than 0000h.
- l) In Arbitration status section, changed “transmitting” to “originating” in the rule about transmitting at least one AIP every 128 dwords. This does not apply to dwords that are being forwarded (e.g. after originating AIP (WAITING ON DEVICE)).
- m) In the phy event information source description of 05h elasticity buffer overflows, added “outside of phy reset sequences” (as is done in 01h and 02h). Elasticity buffer overflows might occur during Train-SNW (since it doesn’t send ALIGNs) but are irrelevant.
- n) In the list of reasons for a Service Response of SERVICE DELIVERY OR TARGET FAILURE for the Command Complete Received transport protocol service arguments, added “ or the ST_IFR state machine detects an error as described in 9.2.6.2.2.3 and 9.2.6.2.2.4” to cover additional cases where the initiator port reports the error on its own.
- o) Renamed “phy event information” to “phy event” throughout (inconsistent in sas2r11; the shorter name seems adequate). This affects function names REPORT PHY EVENT INFORMATION, REPORT PHY EVENT INFORMATION LIST, and CONFIGURE PHY EVENT INFORMATION and one function result name.

R.19 Revision sas2r13 (16 November 2007)

Incorporated these changes per the November 2007 SAS protocol WG (07-473r0), SAS physical WG (07-502r0), and T10 plenary (07-476r0):

- a) 07-304r3 SAS-2 Zero-length test load (Barry Olawsky, HP). Since this required drawing a new S_{DD21} graph, also redrew the TCTF test load graphs based on Excel plots of the real equations. Separated IT and CT graphs into separate figures since they have different curvatures.
- b) 07-090r3 SAS-2 Transmit IDENTIFY three times (Rob Elliott, HP)
- c) 07-392r2 SAS-2 Remove AWT reset on receipt of OPEN_REJECT (RETRY) (George Penokie, IBM)
- d) 07-482r1 SAS-2 PHY CHANGE COUNT increment due to zoning changes (Brad Besmer, LSI)
- e) 07-424r1 SAS-2 Initiator Side Transport layer write flowcharts (George Penokie, IBM)
- f) 07-103r3 SAS-2 Changes to Report Phy Broadcast Counts logging and retrieval method (Tyson Hartshorn, LSI)
- g) 07-480r0 SAS-2 More phy test patterns - TRAIN, TRAIN_DONE, and idle dwords (Rob Elliott, HP)
- h) Added more cross-references in the definitions section when using other defined terms
- i) Added subsections to the expander route table section to isolate phy-based and expander-based sentences
- j) Moved the paragraphs in 7.9.2 SAS initiator device rules (for handling the identification sequence) about Broadcast (Expander) handling into 4.6.8 Expander reduced functionality

R.20 Pending changes

These physical layer proposals/changes are pending acceptance by the appropriate WG(s) and the T10 plenary:

- a) 07-339 SAS-2 6 Gbps PHY specification (Alvin Cox, Seagate)
- b) 07-486 SAS-2 Receiver device physical testing (Kevin Witt and Mahbubul Bari, Maxim)
- c) 07-443 SAS-2 Calibration of jitter measurement devices (Alvin Cox, Seagate and Chuck Hill, Alta Engineering)
- d) 07-449 SAS-2 Mini SAS 8i connectors and cable assemblies (Rob Elliott, HP)

- e) 07-471 SAS-2 Proposed cable tables (Greg McSorley, Amphenol)
- f) resolve various editor's notes

These non-physical layer proposals/changes are pending acceptance by the appropriate WG(s) and the T10 plenary:

- g) 06-459 SAS-2 Broadcast after count update (Luben Tuikov, Vitesse)
- h) 07-089 SAS-2 Redundant primitive sequence error handling (Rob Elliott, HP)
- i) 07-119 SAS-2 Far end retimed loopback phy test function (Rob Elliott, HP)
- j) 07-391 SAS-2 Limiting SAS target response to OPEN_REJECT (RETRY) (George Penokie, IBM)
- k) 07-397 SAS-2 Indeterminate response length to SMP REPORT GENERAL function (George Penokie, IBM)
- l) 07-403 SAS-2 Add SMP REPORT GENERAL version information (Brad Besmer, LSI)
- m) 07-479 SAS-2 Phy test pattern transceiver controls (Rob Elliott, HP)
- n) resolve various editor's notes

Foreword (This foreword is not part of this standard)

This standard defines the Serial Attached SCSI (SAS) interconnect and three transport protocols that use the SAS interconnect:

- a) Serial SCSI Protocol (SSP): a mapping of SCSI supporting multiple initiators and targets;
- b) Serial ATA Tunneled Protocol (STP): a mapping of Serial ATA expanded to support multiple initiators and targets; and
- c) Serial Management Protocol (SMP): a management protocol.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, International Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This standard was processed and approved for submittal to ANSI by the International Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time it approved this standard, INCITS had the following members:

INCITS Technical Committee T10 on Lower Level Interfaces, which developed and reviewed this standard, had the following members:

John B. Lohmeyer, Chair George O. Penokie, Vice-Chair Ralph O. Weber, Secretary

Introduction

This standard defines the Serial Attached SCSI (SAS) interconnect and three transport protocols that use the SAS interconnect:

- a) Serial SCSI Protocol (SSP): a mapping of SCSI supporting multiple initiators and targets;
- b) Serial ATA Tunneled Protocol (STP): a mapping of Serial ATA expanded to support multiple initiators and targets; and
- c) Serial Management Protocol (SMP): a management protocol.

The standard is organized as follows:

- Clause 1 (Scope) describes the relationship of this standard to the SCSI and ATA families of standards.
- Clause 2 (Normative references) provides references to other standards and documents.
- Clause 3 (Definitions, symbols, abbreviations, keywords, and conventions) defines terms and conventions used throughout this standard.
- Clause 4 (General) describes architecture, names and identifiers, state machines, resets, I_T nexus loss, and provides an expander device model.
- Clause 5 (Physical layer) describes the physical layer. It describes passive interconnect components (connectors, cables, and backplanes) and defines the transmitter and receiver electrical characteristics.
- Clause 6 (Phy layer) describes the phy layer. It describes 8b10b encoding, bit order, out of band (OOB) signals, phy reset sequences, phy layer state machines, and spin-up.
- Clause 7 (Link layer) describes the link layer. It describes primitives, clock skew management, idle physical links, CRC, scrambling, address frames, the identification sequence and its state machine, power management, SAS domain changes, connections, rate matching, and SSP, STP, and SMP connection rules and link layer state machines.
- Clause 8 (Port layer) describes the port layer, which sits between one or more link layers and one or more transport layers. It includes port layer state machines.
- Clause 9 (Transport layer) describes the transport layer. It includes SSP, STP, and SMP frame definitions and transport layer state machines.
- Clause 10 (Application layer) describes the application layer. It describes SCSI protocol services, mode parameters, log parameters, and power conditions, ATA specifics, and SMP functions.

Normative Annex A (Jitter tolerance patterns) describes the jitter tolerance patterns.

Normative Annex B (Signal performance measurements) describes signal measurement techniques.

Informative Annex C (SAS to SAS phy reset sequence examples) provides additional phy reset sequence examples.

Informative Annex D (CRC) provides information and example implementations of the CRC algorithm.

Informative Annex E (SAS address hashing) provides information and example implementations of the hashing algorithm.

Informative Annex F (Scrambling) provides information and example implementations of the scrambling algorithm.

Informative Annex G (ATA architectural notes) describes ATA architectural differences from Serial ATA and Serial ATA II.

Informative Annex H (Minimum deletable primitive insertion rate summary) describes the minimum ALIGN and/or NOTIFY insertion rates for clock skew management and rate matching.

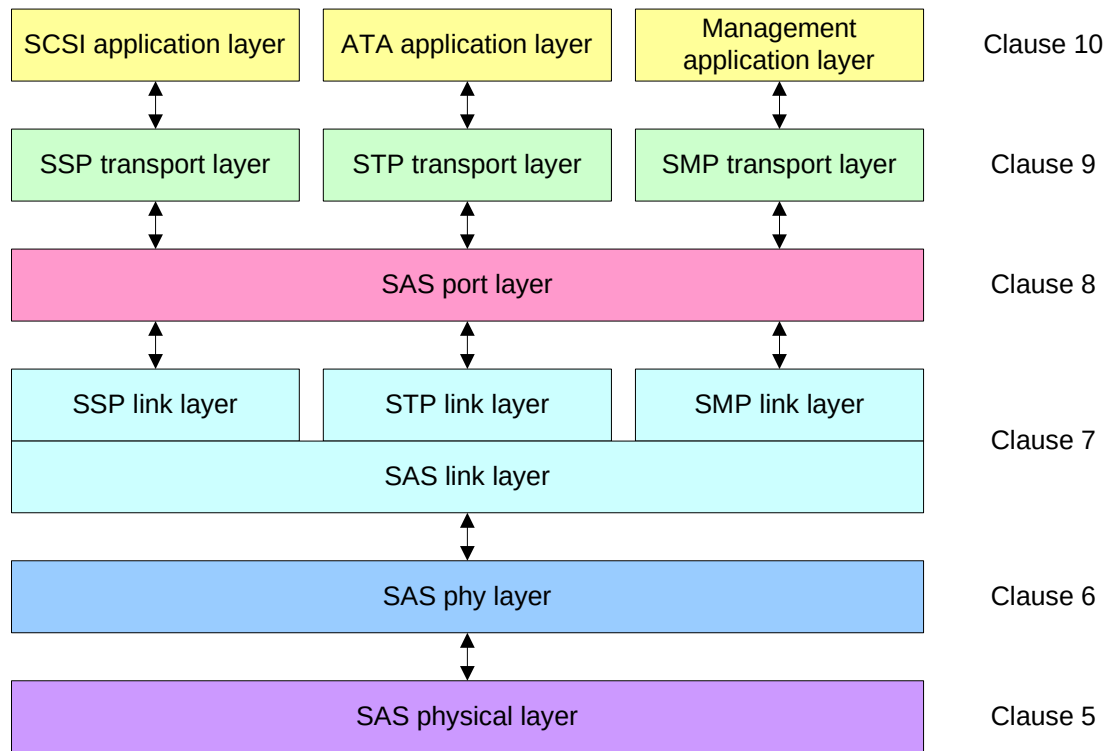
Informative Annex J (Expander device handling of connections) describes expander device behavior in a variety of connection examples.

Informative Annex K (Primitive encoding) lists the primitive encodings available for future versions of this standard.

Informative Annex L (Discover process example implementation) provides an example implementation of the discover process.

Informative Annex M (SAS icons) defines the SAS icons.

The following figure shows the organization of the layers of this standard.



Organization of this standard

American National Standard for Information Technology -

Serial Attached SCSI - 2 (SAS-2)

1 Scope

The SCSI family of standards provides for many different transport protocols that define the rules for exchanging information between different SCSI devices. This standard defines the rules for exchanging information between SCSI devices using a serial interconnect. Other SCSI transport protocol standards define the rules for exchanging information between SCSI devices using other interconnects.

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards.

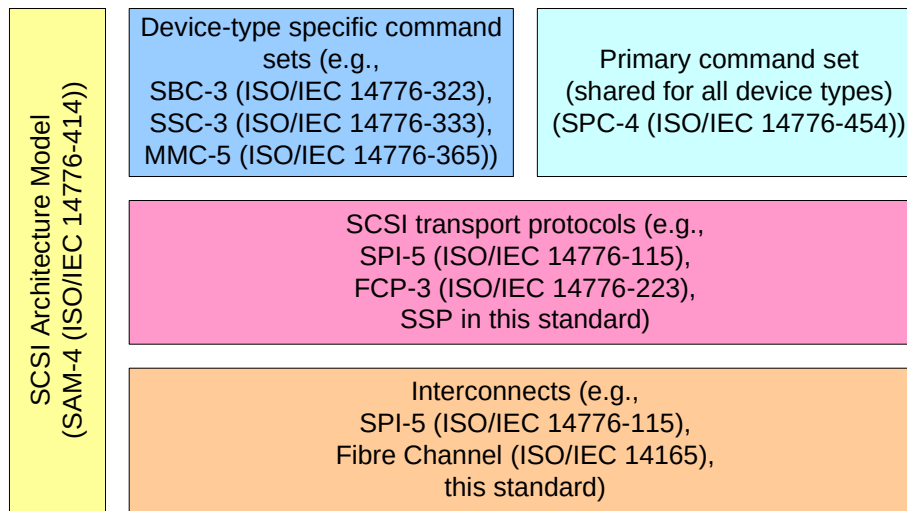


Figure 1 — SCSI document relationships

This standard also defines the rules for exchanging information between ATA hosts and ATA devices using the same serial interconnect. Other ATA transport protocol standards define the rules for exchanging information between ATA hosts and ATA devices using other interconnects.

Figure 2 shows the relationship of this standard to other standards and related projects in the ATA family of standards.

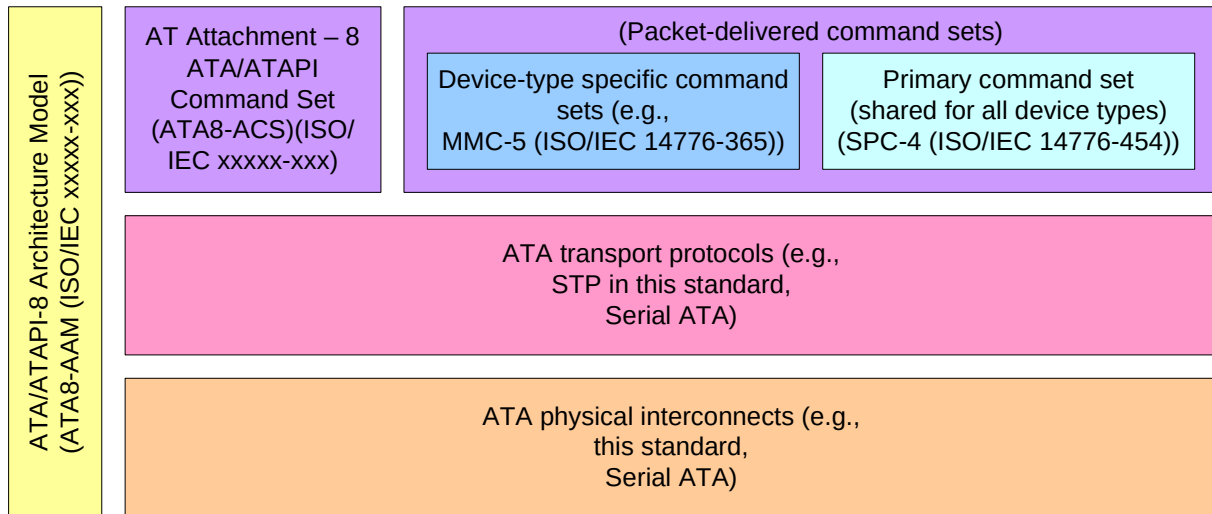


Figure 2 — ATA document relationships

Figure 1 and figure 2 show the general relationship of the documents to one another, and do not imply a relationship such as a hierarchy, protocol stack or system architecture.

These standards specify the interfaces, functions and operations necessary to ensure interoperability between conforming implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

2 Normative references

2.1 Normative references

Referenced standards and specifications contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI:

- a) approved ANSI standards;
- b) approved and draft international and regional standards (e.g., ISO, IEC); and
- c) approved and draft foreign standards (e.g., JIS and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at <http://www.ansi.org>.

Additional availability contact information is provided below as needed.

Table 1 shows standards bodies and their web sites.

Table 1 — Standards bodies

Abbreviation	Standards body	Web site
ANSI	American National Standards Institute	http://www.ansi.org
DIN	German Institute for Standardization	http://www.din.de
IEC	International Engineering Consortium	http://www.iec.ch
IEEE	Institute of Electrical and Electronics Engineers	http://www.ieee.org
INCITS	International Committee for Information Technology Standards	http://www.incits.org
ISO	International Standards Organization	http://www.iso.ch
ITI	Information Technology Industry Council	http://www.itic.org
JIS	Japanese Industrial Standards Committee	http://www.jisc.org
T10	INCITS T10 SCSI storage interfaces	http://www.t10.org
T11	INCITS T11 Fibre Channel interfaces	http://www.t11.org
T13	INCITS T13 ATA storage interface	http://www.t13.org

2.2 Approved references

At the time of publication, the following referenced standards or technical reports were approved:

ANSI INCITS TR-35-2004, *Methodologies for Jitter and Signal Quality Specification (MJSQ)*. When MJSQ is referenced from this standard, the FC Port terminology used within MJSQ should be substituted with SAS phy terminology.

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC xxxxx-xxx, *ATA Attachment-8 ATA/ATAPI Architecture Model (ATA8-AAM)* (T13/1700-D)

ISO/IEC xxxxx-xxx, *ATA Attachment-8 ATA/ATAPI Command Set (ATA8-ACS)* (T13/1699-D)

ISO/IEC 14776-414, *SCSI Architecture Model-4 (SAM-4)* (T10/1683-D)
 ISO/IEC 14776-454, *SCSI Primary Commands-4 (SPC-4)* (T10/1731-D)
 ISO/IEC 14776-323, *SCSI Block Commands-3 (SBC-3)* (T10/1799-D)
 ISO/IEC 14776-372, *SCSI Enclosure Services-2 (SES-2)* (T10/1559-D)

NOTE 1 - For more information on the current status of these documents, contact the INCITS Secretariat at 202-737-8888 (phone), 202-638-4922 (fax) or via Email at incits@itic.org. To obtain copies of these documents, contact Global Engineering at 15 Inverness Way, East Englewood, CO 80112-5704 at 303-792-2181 (phone), 800-854-7179 (phone), or 303-792-2192 (fax) or see <http://www.incits.org>.

2.4 Other references

For information on the current status of the listed documents, or regarding availability, contact the indicated organization.

Serial ATA 2.6 (SATA-2). 15 February 2007

NOTE 2 - For information on the current status of Serial ATA documents, see the Serial ATA International Organization at <http://www.sata-io.org>.

SFF-8086, *Compact Multilane Series: Common Elements*
 SFF-8087, *Compact Multilane Series: Unshielded*
 SFF-8088, *Compact Multilane Series: Shielded*
 SFF-8223, *2.5" Drive Form Factor with Serial Connector*
 SFF-8323, *3.5" Drive Form Factor with Serial Connector*
 SFF-8523, *5.25" Drive Form Factor with Serial Connector*
 SFF-8410, *HSS Copper Testing and Performance Requirements*
 SFF-8416, *Measurement and Performance Requirements for HPEI Bulk Cable*
 SFF-8460, *HSS Backplane Design Guidelines*
 SFF-8470, *Shielded High Speed Multilane Copper Connector*
 SFF-8482, *Unshielded Dual Port Serial Attachment Connector*
 SFF-8484, *Multi-Lane Unshielded Serial Attachment Connectors*
 SFF-8485, *Serial GPIO (SGPIO) Bus*

NOTE 3 - For more information on the current status of SFF document, contact the SFF Committee at 408-867-6630 (phone), or 408-867-2115 (fax). To obtain copies of these documents, contact the SFF Committee at 14426 Black Walnut Court, Saratoga, CA 95070 at 408-867-6630 (phone) or 408-741-1600 (fax) or see <http://www.sffcommittee.org>.

Editor's Note 1: Upgrade these to IEC references, if available. SFF revision numbers could be included, but the SFF Committee does not maintain old revisions. SFF specifications normally migrate from Development to Published status, but can change from Published back to Development if the project is reopened.

OMG Unified Modeling Language (UML) Specification. Version 1.5, March 2003.

NOTE 4 - For more information on the UML specification, contact the Object Modeling Group at <http://www.omg.org>.

PANTONE® Color Formula Guide

NOTE 5 - Pantone® and PANTONE MATCHING SYSTEM® are registered trademarks of Pantone, Inc. For more information on Pantone colors, contact Pantone, Inc. at <http://www.pantone.com>.

3 Definitions, symbols, abbreviations, keywords, and conventions

3.1 Definitions

3.1.1 8b10b coding: A coding scheme that represents an 8-bit byte (i.e., a control byte or data byte) as a 10-bit character (i.e., a control character or data character). See 6.2.

3.1.2 8b10b encoding: Encoding an 8-bit byte (i.e., a control byte or data byte) into a 10-bit character (i.e., a control character or data character). See 6.2.

3.1.3 10b8b decoding: Decoding a 10-bit character (i.e., a control character or data character) into an 8-bit byte (i.e., a control byte or data byte). See 6.2.

3.1.4 actual lock time (ALT): The time during Train-SNW at which a phy locks on received ALIGN (0)s and/or ALIGN (1)s and changes from transmitting ALIGN (0) to ALIGN (1) (see 6.7.4.2.3.4).

3.1.5 actual training time (ATT): The time during Train-SNW at which training (see 3.1.274) is complete (see 6.7.4.2.3.4).

3.1.6 active zone manager: The zone manager (see 3.1.296) that successfully locked a zoning expander device (see 3.1.300). See 4.9.6.

3.1.7 affiliation: The STP target port (see 3.1.263) state of limiting acceptance of connection requests to those from one or more STP initiator ports (see 3.1.258). See 7.17.4.

3.1.8 affiliation context: A set of ATA task file registers maintained by an STP target port for an STP initiator port holding an affiliation. See 7.17.4.

3.1.9 aggregation: When referring to classes (see 3.1.29), a form of association (see 3.1.11) that defines a whole-part relationship between the whole (i.e., aggregate) class (see 3.1.28) and its parts. See 3.5.

3.1.10 application client: An object that is the source of SCSI commands and task management function requests (see SAM-4), ATA commands (see ATA8-AAM), or management function requests. See 4.1.5.

3.1.11 association: When referring to classes (see 3.1.29), a relationship between two or more classes (see 3.1.28) that specifies connections among their objects (see 3.1.148) (i.e., a relationship that specifies that objects of one class are connected to objects of another class). See 3.5.

3.1.12 attached SAS address: The SAS address (see 3.1.192) of the attached phy (e.g., received by a logical phy in the incoming IDENTIFY address frame during the initialization sequence (see 4.1.2)), or the SAS address of the STP target port in an STP/SATA bridge (see 4.6.2).

3.1.13 attribute: When referring to classes (see 3.1.29) or objects (see 3.1.149), a named property of a class (see 3.1.28) that describes the range of values that its objects (see 3.1.148) may hold. See 3.5.

3.1.14 AT Attachment (ATA): A standard for the internal attachment of storage devices to hosts. See ATA8-AAM.

3.1.15 ATA device: A storage peripheral (analogous to a SCSI target device). See ATA8-AAM.

3.1.16 ATA domain: An I/O system consisting of an ATA host and one or more ATA devices that communicate with one another by means of a service delivery subsystem (see 3.1.229). See ATA8-AAM.

3.1.17 ATA host: A host device that originates requests to be processed by an ATA device (analogous to a SCSI initiator device). See ATA8-AAM.

3.1.18 baud: A unit of signaling speed, expressed as the maximum number of times per second the signal (see 3.1.230) may change the state of the physical link (see 3.1.161). The state change produces a transition (i.e., signal edge). Units of baud are symbols/second and in this standard, a symbol represents a single bit or single transition if the maximum transition rate (i.e., a 0101b pattern) is occurring.

3.1.19 big-endian: A format for storage or transmission of binary data in which the most significant byte appears first. In a multi-byte value, the byte containing the most significant bit is stored in the lowest memory address and transmitted first, and the byte containing the least significant bit is stored in the highest memory address and transmitted last (e.g., for the value 0080h, the byte containing 00h is stored in the lowest memory address, and the byte containing 80h is stored in the highest memory address).

3.1.20 bit cell time (BCT): The time for transmitting a phy capabilities bit during SNW-3 (see 6.7.4.2.3.3).

3.1.21 bit error ratio (BER): The number of logical bits output from a receiver circuit that differ from the correct transmitted logical bits, divided by the number of transmitted logical bits. The BER is usually expressed as a coefficient and a power of 10 (e.g., 2 erroneous bits out of 100 000 bits transmitted is expressed as 2 out of 10^5 or 2×10^{-5}). See MJSQ.

3.1.22 Broadcast: Information about an event in the SAS domain, communicated between phys with the BROADCAST primitive sequence (see 7.2.5.5) and/or the SMP ZONED BROADCAST function (see 10.4.3.19). See 4.1.13.

3.1.23 Broadcast propagation processor (BPP): An object within an expander function (see 3.1.74) that manages Broadcasts (see 3.1.22). See 4.6.5.

3.1.24 burst time: The part of an OOB signal (see 3.1.153) where the OOB burst (see 3.1.150) is transmitted. See 6.6.

3.1.25 byte: A sequence of eight contiguous bits considered as a unit. A byte is encoded as a character (see 3.1.27) using 8b10b coding (see 6.2).

3.1.26 cable assembly: Bulk cable plus a separable connector at each end plus any retention, backshell, or shielding features. See 5.2.4.

3.1.27 character: A sequence of ten contiguous bits considered as a unit. A byte (see 3.1.25) is encoded as a character using 8b10b coding (see 6.2).

3.1.28 class: A description of a set of objects (see 3.1.148) that share the same attributes (see 3.1.13), operations (see 3.1.154), relationships, and semantics. Classes may have attributes and may support operations.

3.1.29 class diagram: A diagram that shows a collection of classes (see 3.1.28) and their contents and relationships. See 3.5.

3.1.30 clock data recovery (CDR): The function provided by the receiver circuit responsible for producing a regular clock signal (i.e., the recovered clock) from the received signal and for aligning the recovered clock to the symbols (i.e., bits) being transmitted with the signal. The CDR uses the recovered clock to recover the bits. See MJSQ.

3.1.31 command descriptor block (CDB): A structure used to communicate a command from a SCSI application client to a SCSI device server. See SAM-4.

3.1.32 compliance point: An interoperability point where interoperability specifications are met. See 5.3.1.

3.1.33 compliant jitter tolerance pattern (CJT PAT): A test pattern for jitter testing. See 5.3.5.4 and A.2.

3.1.34 configuration subprocess: A subprocess invoked from the discover process (see 3.1.58) to configure an externally configurable expander device (see 3.1.82). See 4.7.

3.1.35 confirmation: Information passed from a lower layer state machine to a higher layer state machine, usually responding to a request (see 3.1.181) from that higher layer state machine, and sometimes relaying a response (see 3.1.183) from a peer higher layer state machine. See 3.6.

3.1.36 connection: A temporary association between a SAS initiator port (see 3.1.197) and a SAS target port (see 3.1.203) using a pathway (see 3.1.156). See 7.12.

3.1.37 connection rate: The effective rate of dwords through the pathway (see 3.1.156) between a SAS initiator phy and a SAS target phy, established through the connection request.

3.1.38 connection request: A request to establish a connection originated by a SAS initiator port (see 3.1.197) or a SAS target port using an OPEN address frame(see 7.8.3). See 7.12.2.1.

3.1.39 connector: Electro-mechanical components consisting of a receptacle and a plug that provide a separable interface between two transmission segments. See 5.2.3.

3.1.40 constraint: When referring to classes (see 3.1.29) and objects (see 3.1.149), a mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations (see 3.1.11)). See 3.5.

3.1.41 control byte: A byte containing control information defined in table 76 (see 6.2).

3.1.42 control character (Kxx.y): A character containing control information defined in table 76 (see 6.2).

3.1.43 cumulative distribution function (CDF): The probability that jitter is less than a given value. See MJSQ.

3.1.44 cyclic redundancy check (CRC): An error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum. See 7.5.

3.1.45 D.C. idle: A differential signal level that is nominally 0 V(P-P), used during the idle time (see 3.1.104) and negation time (see 3.1.144) of an OOB signal (see 3.1.153). See 5.3.6.

3.1.46 data byte: A byte containing data information defined in table 75 (see 6.2).

3.1.47 data character (Dxx.y): A character containing data information defined in table 75 (see 6.2).

3.1.48 data dword: A dword containing four data bytes, or four data characters with correct disparity.

3.1.49 deadlock: A condition in which two or more processes (e.g., connection requests) are waiting on the others to complete, resulting in none of the processes completing.

3.1.50 decibel (dB): One-tenth of the common logarithm (i.e., $\log_{10}$) of the relative power ratio (i.e., $\text{dB} = 10 \log_{10} (P_1 / P_2)$). Equivalent to one-twentieth of the common logarithm of the relative voltage ratio (i.e., $\text{dB} = 20 \log_{10} (V_1 / V_2)$).

3.1.51 deletable primitive: An ALIGN (see 7.2.5.2), NOTIFY (see 7.2.5.11), or MUX (see 7.2.5.10), which may be deleted by a receiver instead of being placed into its elasticity buffer. See 7.2.

3.1.52 dependency: When referring to classes (see 3.1.29), a relationship between two classes (see 3.1.28) where a change to one class (i.e., the independent class) may cause a change in the other class (i.e., the dependent class). See 3.5.

3.1.53 device name: A worldwide unique name for a device within a transport protocol. See 4.2.4.

3.1.54 device server: An object that processes SCSI tasks (see SAM-4), ATA commands (see ATA8-AAM), or management functions. See 4.1.5.

3.1.55 direct current (D.C.): The non-A.C. component of a signal. In this standard, all frequency components below 100 kHz.

3.1.56 direct routing attribute: The attribute of an expander phy that indicates it may be used by the ECM (see 3.1.71) to route a connection request to an end device. See 4.6.7.1.

3.1.57 direct routing method: The method the ECM (see 3.1.71) uses to route connection requests to an attached end device or an attached expander device (see 3.1.73). See 4.6.7.1.

3.1.58 discover process: The process performed by a management application client to discover all the SAS devices (see 3.1.193) and expander devices (see 3.1.73) in the SAS domain (see 3.1.194) that invokes the configuration subprocess (see 3.1.34) as needed. See 4.7.

3.1.59 disparity: The difference between the number of ones and zeros in a character (see 6.2).

3.1.60 dispersion: Signal pulse broadening and distortion from all causes.

3.1.61 domain: A SAS domain (see 3.1.194), a SCSI domain (see 3.1.215), or an ATA domain (see 3.1.16).

3.1.62 dword: A sequence of four contiguous bytes or four contiguous characters considered as a unit. The meaning depends on the context (e.g., when discussing the bits being transmitted over a physical link, dword represents four characters (i.e., 40 bits). When discussing the contents of a frame after 10b8b decoding (see 3.1.3), dword represents four bytes (i.e., 32 bits)).

3.1.63 dword synchronization: Detection of an incoming stream of dwords from a physical link by a phy. See 6.9.

3.1.64 electromagnetic interference (EMI): Any electromagnetic disturbance that interrupts, obstructs, or otherwise degrades or limits the effective performance of electronics/electrical equipment.

3.1.65 enclosure: The box, rack, or set of boxes providing the powering, cooling, mechanical protection, EMI protection, and external electronic interfaces for one or more end device(s) (see 3.1.69) and/or expander device(s) (see 3.1.73). The enclosure provides the outermost electromagnetic boundary and acts as an EMI barrier. An enclosure is not a class (see 3.1.28) in this standard.

3.1.66 enclosure in port: A set of expander phys with subtractive routing attributes using the same external connector (see 5.2.3.3). See 4.6.2.

3.1.67 enclosure out port: A set of expander phys with table routing attributes in an expander device that does not support table-to-table attachment using the same external connector (see 5.2.3.3). See 4.6.2.

3.1.68 enclosure universal port: A set of expander phys with table routing attributes in an expander device that supports table-to-table attachment using the same external connector (see 5.2.3.3). See 4.6.2.

3.1.69 end device: A SAS device (see 3.1.193) or SATA device (see 3.1.206) that is not contained within an expander device (see 3.1.73).

3.1.70 event notification: Information passed from the transport layer to the application layer notifying the application layer that a SCSI event has occurred. See SAM-4.

3.1.71 expander connection manager (ECM): An object within an expander function (see 3.1.74) that manages routing. See 4.6.3.

3.1.72 expander connection router (ECR): The portion of an expander function (see 3.1.74) that routes messages between expander phys. See 4.6.4.

3.1.73 expander device: A device that is part of a service delivery subsystem (see 3.1.229), facilitates communication between SAS devices (see 3.1.193) and SATA devices (see 3.1.206), and is either an externally configurable expander device (see 3.1.82) or a self-configuring expander device (see 3.1.223). See 4.1.7.

3.1.74 expander function: An object within an expander device (see 3.1.73) that contains an expander connection manager (see 3.1.71), expander connection router (see 3.1.72), and Broadcast propagation processor (see 3.1.23). See 4.6.1.

3.1.75 expander logical phy: An expander phy (see 3.1.76) or a multiplexed portion of an expander phy. See 4.1.2.

3.1.76 expander phy: A phy in an expander device (see 3.1.73) that interfaces to a service delivery subsystem (see 3.1.229).

3.1.77 expander port: An expander device object that interfaces to a service delivery subsystem (see 3.1.229) and to SAS ports in other devices. See 4.6.2.

3.1.78 expander route entry: A SAS address and an enable/disable bit in an expander route table (see 3.1.80).

3.1.79 expander route index: A value used in combination with a phy identifier to select an expander route entry in an expander route table (see 3.1.80) in an externally configurable expander device (see 3.1.82). See 4.6.7.3.

3.1.80 expander route table: A table of expander route entries (see 3.1.78) within an expander device (see 3.1.73). The table is used by the expander function (see 3.1.74) to resolve connection requests. See 4.6.7.3.

3.1.81 external connector: A bulkhead connector (see 3.1.39), whose purpose is to carry signals into and out of an enclosure (see 3.1.65), that exits the enclosure with only minor compromise to the shield effectiveness of the enclosure. See 5.2.3.3.1.1 and 5.2.3.3.1.2.

3.1.82 externally configurable expander device: A non-self-configuring expander device (see 3.1.73) that contains an expander route table (see 3.1.80) that is configurable with the SMP CONFIGURE ROUTE INFORMATION function (see 10.4.3.26). See 4.1.7.

3.1.83 eye contour: The locus of points in a signal level versus time eye diagram where the CDF of 10^{-12} in the actual signal population exists. Comparison of the measured eye contour to the jitter eye mask determines whether a jitter eye mask violation has occurred. See 5.3.5 and MJSQ.

3.1.84 fall time: The time interval for the falling signal edge to transit between specified percentages of the signal amplitude. In this standard, the measurement points are the 80 % and 20 % voltage levels. Also see rise time (see 3.1.187).

3.1.85 fanout cable assembly: A cable assembly that attaches a wide connector on one end to multiple narrow connectors on the other end. See 5.2.4.1.3.

3.1.86 far-end crosstalk: Crosstalk that is propagated in a disturbed channel in the same direction as the propagation of a signal in the disturbing channel. The terminals of the disturbed channel, at which the far-end crosstalk is present, and the energized terminals of the disturbing channel are usually remote from each other.

3.1.87 field: A group of one or more contiguous bits.

3.1.88 frame: A sequence of data dwords between a start of frame primitive (i.e., SOF, SOAF, or SATA_SOF) and an end of frame primitive (i.e., EOF, EOAF, or SATA_EOF).

3.1.89 frame information structure (FIS): The SATA frame format. See SATA-2.

3.1.90 generalization: When referring to classes (see 3.1.29), a relationship among classes (see 3.1.28) where one class (i.e., the superclass) shares the attributes (see 3.1.13) and operations (see 3.1.154) of one or more other classes (i.e., the subclasses). See 3.5.

3.1.91 golden phase lock loop (golden PLL): A function that conforms to the jitter timing reference frequency response requirements in MJSQ that extracts the jitter timing reference from the data stream under test to be used as the timing reference for the instrument used for measuring the jitter in the signal under test. See MJSQ.

3.1.92 hard reset: A SAS device or expander device action in response to a reset event in which the device performs the operations described in 4.4.

3.1.93 hard reset sequence: A sequence that causes a hard reset (see 4.4).

3.1.94 hardware maximum physical link rate: The maximum physical link rate capability of a phy.

3.1.95 hardware minimum physical link rate: The minimum physical link rate capability of a phy.

3.1.96 hash function: A mathematical function that maps values from a larger set of values into a smaller set of values, reducing a long value into a shorter hashed value.

3.1.97 I_T nexus: When referring to SAS ports (see 3.1.200), a nexus between a SAS initiator port and a SAS target port. When referring to SCSI ports (see 3.1.218), a nexus between a SCSI initiator port and a SCSI target port. See SAM-4.

3.1.98 I_T nexus loss: When referring to SAS ports (see 3.1.200), a condition where a SAS port determines that another SAS port is no longer available. See 4.5. When referring to SCSI ports (see 3.1.218)(e.g., SSP ports), a condition resulting from the events defined by SAM-4 in which the SCSI device performs the I_T nexus loss operations described in SAM-4, SPC-4, and the appropriate command standards. See SAM-4.

3.1.99 I_T_L nexus: A nexus that exists between a SCSI initiator port, a SCSI target port, and a logical unit. This relationship extends the prior I_T nexus. See SAM-4.

3.1.100 I_T_L_Q nexus: A nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a task. This relationship extends the prior I_T nexus or I_T_L nexus. See SAM-4.

3.1.101 IDENTIFY (PACKET) DEVICE data: IDENTIFY DEVICE data from an ATA device, or IDENTIFY PACKET DEVICE data from an ATAPI device. See ATA8-ACS.

3.1.102 identification sequence: A sequence where phys exchange IDENTIFY address frames. See 7.9.

3.1.103 idle dword: A vendor-specific data dword that is scrambled and is transmitted outside a frame. See 7.4.

3.1.104 idle time: The part of an OOB signal (see 3.1.153) where D.C. idle (see 5.3.6) is being transmitted interleaved with OOB burst (see 3.1.150). See 6.6.

3.1.105 indication: Information passed from a lower layer state machine to a higher layer state machine, usually relaying a request (see 3.1.181) from a peer higher layer state machine. See 3.6.

3.1.106 information unit (IU): The portion of an SSP frame that carries command, task management function, data, response, or transfer ready information. See 9.2.2.

3.1.107 insertion loss (S_{21}): The ratio, usually expressed in dB, of delivered power to incident power. See B.9.

Editor's Note 2: The insertion loss and return loss definitions are in terms of power. However, the voltage, not power, value is generally used in discussions of signal levels. Insertion loss = $10 \log (P_{in}/P_{out})$ and = $20 \log (V_{in}/V_{out})$. 07-339 will include a definition of decibel (dB) that describes this.

3.1.108 insertion loss, differential (S_{DD21}): Differential insertion loss measured at port 1. See B.9.

3.1.109 insertion loss, upstream differential (S_{DD12}): Differential insertion loss measured at port 2. See B.9.

3.1.110 intersymbol interference (ISI): Reduction in the distinction of a pulse caused by overlapping energy from neighboring pulses. Neighboring pulses are pulses that are close enough to have significant energy overlapping and does not imply or exclude adjacent pulses (i.e., many bit times may separate the pulses, especially in the case of reflections). ISI may result in DDJ and vertical eye closure. Several mechanisms produce ISI (e.g., dispersion, reflections, and circuits that lead to baseline wander). See MJSQ.

3.1.111 invalid character: A character that is not a control character (see 3.1.42) or a data character (see 3.1.47).

3.1.112 invalid dword: A dword that is not a data dword (see 3.1.48) or a primitive (see 3.1.169) (i.e., in the character context, a dword that contains an invalid character, a control character in other than the first character position, a control character other than K28.3 or K28.5 in the first character position, or one or more characters with a running disparity error).

Editor's Note 3: See editor's note about "primitive"

3.1.113 jitter: The collection of instantaneous deviations of signal edge times at a defined signal level of the signal from the reference times (e.g., as defined by the jitter timing reference) for those events. See MJSQ.

3.1.114 jitter, data dependent (DDJ): Jitter that is added when the transmission pattern is changed from a clock-like to a non-clock-like pattern. See MJSQ.

3.1.115 jitter, deterministic (DJ): Jitter with non-Gaussian distribution. See MJSQ.

3.1.116 jitter, random, (RJ): Jitter that is characterized by a Gaussian distribution and is unbounded. See MJSQ.

3.1.117 jitter, sinusoidal (SJ): Single frequency jitter applied during signal tolerance testing. See MJSQ.

3.1.118 jitter, total (TJ): Jitter from all sources. See MJSQ.

3.1.119 jitter timing reference: The signal used as the basis for calculating the jitter in the signal under test. See MJSQ.

3.1.120 jitter tolerance: The ability of the receiver device to recover transmitted bits in an incoming data stream in the presence of specified jitter in the signal applied to the receiver device compliance point. See MJSQ.

3.1.121 jitter tolerance pattern (JTPAT): A test pattern for jitter testing. See 5.3.5.4 and A.1.

3.1.122 least significant bit (LSB): In a binary code, the bit or bit position with the smallest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 0001b, the bit that is set to one).

3.1.123 link reset: Performing the link reset sequence (see 3.1.124).

3.1.124 link reset sequence: For SATA, a phy reset sequence (see 3.1.160). For SAS, a phy reset sequence followed by an identification sequence (see 3.1.102), or a phy reset sequence followed by a hard reset sequence (see 3.1.93), another phy reset sequence, and an identification sequence. See 4.4.

3.1.125 little-endian: A format for storage or transmission of binary data in which the least significant byte appears first. In a multi-byte value, the byte containing the least significant bit is stored in the lowest memory address and transmitted first, and the byte containing the most significant bit is stored in the highest memory address and transmitted last (e.g., for the value 0080h, the byte containing 80h is stored in the lowest memory address, and the byte containing 00h is stored in the highest memory address).

3.1.126 livelock: A condition where two or more processes (e.g., connection requests) continually change their state in response to changes in other processes, resulting in none of the processes completing.

3.1.127 locked zoning expander device: A zoning expander device (see 3.1.300) that has been locked by a zone manager (see 3.1.296). See 4.9.6.2.

3.1.128 logical link: A physical link (see 3.1.161) or a multiplexed portion of a physical link. See 4.1.3.

3.1.129 logical link rate: A link rate between two logical phys established as a result of speed negotiation and multiplexing negotiation between the physical phys containing those logical phys.

3.1.130 logical phy: A phy (see 3.1.158) or a multiplexed portion of a phy. See 4.1.2.

3.1.131 logical unit: An externally addressable entity within a SCSI target device that implements a SCSI device model and contains a device server. See SAM-4.

3.1.132 logical unit number (LUN): An identifier for a logical unit (see 3.1.131). See SAM-4.

3.1.133 maximum training time (MTT): The maximum time during Train-SNW for a phy receiver to complete training (see 3.1.274) (see 6.7.4.2.3.4).

3.1.134 maximum Train-SNW window time (MTWT): The maximum duration of Train-SNW (see 6.7.4.2.3.4).

3.1.135 media: Plural of medium (see 3.1.136).

3.1.136 medium: The material on which data is stored (e.g., a magnetic disk).

3.1.137 message: Information sent between state machines. See 3.6.

3.1.138 most significant bit (MSB): In a binary code, the bit or bit position with the largest numerical weighting in a group of bits that, when taken as a whole, represent a numerical value (e.g., in the number 1000b, the bit that is set to one).

3.1.139 multiplexing: Dividing a physical link into two logical links, and dividing a phy into two logical phys, by interleaving dwords. See 6.10.

3.1.140 multiplicity: When referring to classes (see 3.1.29), an indication of the range of allowable instances of an object (see 3.1.148) or an attribute (see 3.1.13). See 3.5.

3.1.141 narrow link: A physical link that attaches a narrow port to another narrow port. See 4.1.4.

3.1.142 narrow port: A port that contains exactly one phy. See 4.1.4.

3.1.143 near-end crosstalk (NEXT): Crosstalk that is propagated in a disturbed channel in the opposite direction as the propagation of a signal in the disturbing channel. The terminals of the disturbed channel, at which the near-end crosstalk is present, and the energized terminals of the disturbing channel are usually near each other.

3.1.144 negation time: The part of an OOB signal (see 3.1.153) during which D.C. idle (see 3.1.45) is transmitted after the last OOB burst (see 3.1.150). See 6.6.

3.1.145 negotiated logical link rate: The current operational logical link rate (see 3.1.129).

3.1.146 negotiated physical link rate: The current operational physical link rate (see 3.1.162).

3.1.147 nexus: When referring to SAS ports (see 3.1.200), a relationship between a SAS initiator port and a SAS target port. When referring to SCSI ports (see 3.1.218), a relationship between a SCSI initiator port and a SCSI target port that may extend to a logical unit and a task. See SAM-4.

3.1.148 object: An entity with a well-defined boundary and identity that encapsulates state and behavior. All objects are instances of classes (see 3.1.28)(i.e., a concrete manifestation of a class is an object).

3.1.149 object diagram: A diagram that encompasses objects and their relationships at a point in time. See 3.5.

3.1.150 OOB burst: The transmission of signal transitions for a burst time (see 3.1.24). See 6.6.1.

3.1.151 OOB interval: The time basis for burst times (see 3.1.24), idle times (see 3.1.104), negation times (see 3.1.144), and signal times (see 3.1.233) used to create OOB signals (see 3.1.153). See 6.6.1.

3.1.152 OOB sequence: A sequence where two phys exchange OOB signals (see 3.1.153). See 6.7.2.1 and 6.7.4.1.

3.1.153 OOB signal: A pattern of idle time (see 3.1.104), burst time (see 3.1.24), and negation time (see 3.1.144) used during the link reset sequence. See 6.6.

3.1.154 operation: When referring to classes (see 3.1.29), a service that may be requested from any object (see 3.1.148) of the class (see 3.1.28) in order to effect behavior. Operations describe what a class is allowed to do and may be a request or a query. A request may change the state of the object but a query should not. See 3.5.

3.1.155 partial pathway: The set of logical links participating in a connection request that have not yet conveyed a connection response. See 4.1.11.

3.1.156 pathway: A set of logical links between a SAS initiator phy and a SAS target phy being used by a connection (see 3.1.36). See 4.1.11.

3.1.157 pathway blocked count: The number of times the port has retried this connection request due to receiving OPEN_REJECT (PATHWAY BLOCKED), OPEN_REJECT (RESERVED STOP 0), or OPEN_REJECT (RESERVED STOP 1).

3.1.158 phy: A object in a device that is used to interface to other devices (e.g., an expander phy (see 3.1.76) or a SAS phy (see 3.1.198)). See 4.1.2.

3.1.159 phy identifier: The value by which a phy is identified within a device. See 4.2.8.

3.1.160 phy reset sequence: An OOB sequence (see 3.1.152) followed by a speed negotiation sequence (see 3.1.244). See 4.4.

3.1.161 physical link: Two differential signal pairs, one pair in each direction, that connect two physical phys (see 3.1.163). See 4.1.2.

3.1.162 physical link rate: A link rate between two physical phys established as a result of speed negotiation between those phys.

3.1.163 physical phy: A phy (see 3.1.158) that contains a transceiver (see 3.1.277) and electrically interfaces to a physical link to communicate with another physical phy. See 4.1.2.

3.1.164 port: A SAS port (see 3.1.200) or an expander port (see 3.1.77). Each port contains one or more phys (see 3.1.158). See 4.1.4.

3.1.165 port identifier: The value by which a port is identified within a domain. See 4.2.7.

3.1.166 port name: A worldwide unique name for a port within a transport protocol. See 4.2.6.

3.1.167 potential pathway: A set of logical links between a SAS initiator phy and a SAS target phy. See 4.1.11.

3.1.168 power on: Power being applied.

3.1.169 primitive: A dword containing a 7Ch or BCh control byte followed by three data bytes, or a K28.3 or K28.5 control character with correct disparity followed by three data characters with correct disparity. See 7.2.

Editor's Note 4: An end device that does not support being attached to SATA (e.g., a SAS disk drive) should be allowed to consider K28.3 based dwords as not being primitives, and thus treat them as illegal dwords. (related to editor's note on "invalid dword")

3.1.170 primitive sequence: A set of primitives treated as a single entity. See 7.2.4.

3.1.171 probe point: Physical position in a test load where signal characteristics for compliance points are measured See 5.3.2.

3.1.172 programmed maximum physical link rate: The maximum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function (see 10.4.3.27) or the Phy Control and Discover mode page (see 10.2.7.5)).

3.1.173 programmed minimum physical link rate: The minimum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function (see 10.4.3.27) or the Phy Control and Discover mode page (see 10.2.7.5)).

3.1.174 protocol: Either SSP (see 3.1.228), SMP (see 3.1.227), or STP (see 3.1.225).

3.1.175 rate: Data transfer rate of a physical or logical link (e.g., 1,5 Gbps, 3 Gbps, or 6 Gbps).

3.1.176 rate change delay time (RCDT): The time between rates during the speed negotiation sequence (see 6.7.4.2).

3.1.177 read data: Data transferred to the SCSI application client's data-in buffer from the SCSI device server, as requested by the Send Data-In transport protocol service (see 10.2.1.6).

3.1.178 receiver circuit: An electronic circuit that converts an analog serial input signal to a logic signal.

3.1.179 receiver device (Rx): The device downstream from an IR or CR compliance point containing a portion of the physical link and a receiver circuit (see 3.1.178).

3.1.180 reflection coefficient (ρ): The ratio of reflected voltage to incident voltage.

3.1.181 request: Information passed from a higher layer state machine to a lower layer state machine, usually to initiate some action. See 3.6.

3.1.182 reset event: An event that triggers a hard reset (see 4.4.2) in a SAS device.

3.1.183 response: Information passed from a higher layer state machine to a lower layer state machine, usually in response to an indication (see 3.1.105). See 3.6.

3.1.184 return loss (S_{11}): The ratio, usually expressed in dB, of incident power to reflected power. See B.9.

3.1.185 return loss, differential (S_{DD21}): Differential return loss measured at port 1. See B.9.

3.1.186 return loss, upstream differential (S_{DD12}): Differential return loss measured at port 2. See B.9.

3.1.187 rise time: The time interval for the rising signal edge to transit between specified percentages of the signal amplitude. In this standard, the measurement points are the 20 % and 80 % voltage levels. Also see fall time (see 3.1.84).

3.1.188 role: When referring to classes (see 3.1.29) and objects (see 3.1.149), a label at the end of an association (see 3.1.11) or aggregation (see 3.1.9) that defines a relationship to the class on the other side of the association or aggregation. See 3.5.

3.1.189 route table optimization: A configuration subprocess (see 3.1.34) algorithm that reduces the number of entries required in an expander route table (see 3.1.80) in an externally configurable expander device (see 3.1.82). See 4.8.3.

3.1.190 run length: Number of consecutive identical bits in the transmitted signal (e.g., the pattern 0011111010 includes the following run lengths: five 1s, one 0, one 1, and indeterminate run lengths of 0s at the start and end).

3.1.191 running disparity (RD): A binary parameter with a negative (-) or positive (+) value indicating the cumulative encoded signal imbalance between the one and zero signal state of all characters since dword synchronization has been achieved. See 6.2.

3.1.192 SAS address: A worldwide unique NAA IEEE Registered format identifier assigned to a SAS port (see 3.1.200) or expander device (see 3.1.73). See 4.2.2.

3.1.193 SAS device: A SAS initiator device, SAS target device, or SAS target/initiator device.

3.1.194 SAS domain: The I/O system defined by this standard that may serve as a SCSI domain. See 4.1.9.

3.1.195 SAS initiator device: A device containing SSP, STP, and/or SMP initiator ports in a SAS domain. See 4.1.6.

3.1.196 SAS initiator phy: A logical phy (see 3.1.130) in a SAS initiator device.

3.1.197 SAS initiator port: An SSP initiator port (see 3.1.250), STP initiator port (see 3.1.258), and/or SMP initiator port (see 3.1.236) in a SAS domain.

3.1.198 SAS phy: A phy in a SAS device that interfaces to a service delivery subsystem (see 3.1.229).

3.1.199 SAS logical phy: A SAS phy (see 3.1.198) or a multiplexed portion of a SAS phy. See 4.1.2.

3.1.200 SAS port: A SAS initiator port (see 3.1.197), SAS target port (see 3.1.203), or SAS target/initiator port (see 3.1.205).

3.1.201 SAS target device: A device containing SSP, STP, and/or SMP target ports in a SAS domain. See 4.1.6.

3.1.202 SAS target phy: A logical phy (see 3.1.130) in a SAS target device.

3.1.203 SAS target port: An SSP target port (see 3.1.254), STP target port (see 3.1.263), and/or SMP target port (see 3.1.240) in a SAS domain.

3.1.204 SAS target/initiator device: A device that has all the characteristics of a SAS target device and a SAS initiator device.

3.1.205 SAS target/initiator port: A port that has all the characteristics of a SAS target port and a SAS initiator port in a SAS domain.

3.1.206 SATA device: An ATA device that contains a SATA device port in an ATA domain (analogous to a SCSI target device).

3.1.207 SATA device port: An ATA device object that interfaces to a service delivery subsystem (see 3.1.229) with SATA. Analogous to a SCSI target port.

3.1.208 SATA host: An ATA host that contains a SATA host port in an ATA domain (analogous to a SCSI initiator device).

3.1.209 SATA host port: An ATA host object that interfaces to a service delivery subsystem (see 3.1.229) with SATA. Analogous to a SCSI initiator port.

3.1.210 SATA phy: A phy in a SATA device or SATA port selector that interfaces to a service delivery subsystem (see 3.1.229). Analogous to a SAS phy.

3.1.211 SATA port selector: A device that attaches to two SATA host ports (i.e., two ATA domains) and one SATA device port, and provides the means for one SATA host to access the device at any given time (see SATA-2).

3.1.212 saturating counter: A counter that remains at its maximum value after reaching its maximum value.

3.1.213 scrambling: Modifying data by XORing each bit with a pattern generated by a linear feedback shift register to minimize repetitive character patterns. See 7.6.

3.1.214 SCSI device: A device that contains one or more SCSI ports that are connected to a service delivery subsystem (see 3.1.229) and supports a SCSI application protocol. See SAM-4.

3.1.215 SCSI domain: An I/O system consisting of a set of SCSI devices that communicate with one another by means of a service delivery subsystem (see 3.1.229). See SAM-4.

3.1.216 SCSI initiator device: A SCSI device containing SCSI application clients and SCSI initiator ports that originates device service and task management requests to be processed by a SCSI target device and receives device service and task management responses from SCSI target devices. See SAM-4.

3.1.217 SCSI initiator port: A SCSI initiator device object that acts as the connection between SCSI application clients and a service delivery subsystem (see 3.1.229) through which indications and responses are routed. See SAM-4.

3.1.218 SCSI port: A SCSI initiator port, SCSI target port, or SCSI target/initiator port. See SAM-4.

3.1.219 SCSI target device: A SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices. See SAM-4.

3.1.220 SCSI target port: A SCSI target device object that contains a task router and acts as the connection between SCSI device servers and task managers and a service delivery subsystem (see 3.1.229) through which requests and confirmations are routed. See SAM-4.

3.1.221 SCSI target/initiator device: A device that has all the characteristics of a SCSI target device and a SCSI initiator device. See SAM-4.

3.1.222 SCSI target/initiator port: A SCSI target/initiator device object that has all the characteristics of a SCSI target port and a SCSI initiator port. See SAM-4.

3.1.223 self-configuring expander device: An expander device (see 3.1.73) containing an SMP initiator port and a management application client to perform the discover process (see 3.1.58) to configure its own expander route table (see 3.1.80). See 4.7.2.

3.1.224 Serial ATA (SATA): The protocol defined by SATA-2 (see 2.3).

3.1.225 Serial ATA Tunneled Protocol (STP): The protocol defined in this standard used by STP initiator ports (see 3.1.258) to communicate with STP target ports (see 3.1.263) in a SAS domain. See 7.17 and 9.3.

3.1.226 Serial Attached SCSI (SAS): The set of protocols and the interconnect defined by this standard.

3.1.227 Serial Management Protocol (SMP): The protocol defined in this standard used by SMP initiator ports (see 3.1.236) to communicate with SMP target ports (see 3.1.240) in a SAS domain. See 7.18 and 9.4.

3.1.228 Serial SCSI Protocol (SSP): The protocol defined in this standard used by SSP initiator ports (see 3.1.250) to communicate with SSP target ports (see 3.1.254) in a SAS domain. See 7.16 and 9.2.

3.1.229 service delivery subsystem: The part of a SCSI I/O system that transmits information between a SCSI initiator port and a SCSI target port, or the part of an ATA I/O system that transmits information between an ATA host and an ATA device, or the part of a SAS I/O system that transmits information between a SAS initiator port and a SAS target port.

3.1.230 signal: The entire voltage waveform during transmission.

3.1.231 signal amplitude: A property of the overall signal (see 3.1.230) that describes the peak or peak-to-peak values of the signal level (see 3.1.232).

3.1.232 signal level: The instantaneous intensity of a signal (see 3.1.230) measured in volts.

3.1.233 signal time: The time of an OOB signal (see 3.1.153), consisting of six burst times (see 3.1.24), six idle times (see 3.1.104), and one negation time (see 3.1.144). See 6.6.

3.1.234 signal tolerance: The ability of the receiver device to recover transmitted bits in an incoming data stream with maximum jitter and minimum amplitude. See MJSQ.

3.1.235 SMP initiator phy: A SAS initiator phy (see 3.1.196) in an SMP initiator port (see 3.1.236).

3.1.236 SMP initiator port: A SAS initiator device object in a SAS domain that interfaces to a service delivery subsystem with SMP.

3.1.237 SMP phy: A SAS logical phy (see 3.1.199) in an SMP port.

3.1.238 SMP port: An SMP initiator port (see 3.1.236), SMP target port (see 3.1.240), or SMP target/initiator port (see 3.1.241).

3.1.239 SMP target phy: A SAS target phy (see 3.1.202) in an SMP target port (see 3.1.240).

3.1.240 SMP target port: A SAS target device object in a SAS domain that interfaces to a service delivery subsystem (see 3.1.229) with SMP.

3.1.241 SMP target/initiator port: A port that has all the characteristics of an SMP initiator port and an SMP target port.

3.1.242 SMP zone configuration function: An SMP function that configures zoning expander shadow values (see 3.1.302) and is only accepted by a locked zoning expander device (see 3.1.127). See 4.9.6.3.

3.1.243 speed negotiation lock time (SNLT): The maximum time during an SNW (see 3.1.246) for a transmitter to reply with ALIGN (1) (see 6.7.4.2).

3.1.244 speed negotiation sequence: A sequence in which two phys negotiate the operational physical link rate. See 6.7.2.2 and 6.7.4.2.

3.1.245 speed negotiation transmit time (SNTT): The time during which ALIGN (0) or ALIGN (1) is transmitted during the speed negotiation sequence (see 6.7.4.2).

3.1.246 speed negotiation window (SNW): A portion of the SAS speed negotiation sequence (see 3.1.244). See 6.7.4.2.3.

3.1.247 speed negotiation window time (SNWT): The duration of an SNW (see 3.1.247). See 6.7.4.2.3.

3.1.248 spread spectrum clocking (SSC): The technique of modulating the operating frequency of a transmitted signal (i.e., the physical link rate) to reduce the measured peak amplitude of radiated emissions. See 5.3.8.

3.1.249 SSP initiator phy: A SAS initiator phy (see 3.1.196) in an SSP initiator port (see 3.1.250).

3.1.250 SSP initiator port: A SCSI initiator port in a SAS domain that implements SSP.

3.1.251 SSP phy: A SAS logical phy (see 3.1.199) in an SSP port.

3.1.252 SSP port: An SSP initiator port (see 3.1.250), SSP target port (see 3.1.254), or SSP target/initiator port (see 3.1.255).

3.1.253 SSP target phy: A SAS target phy (see 3.1.202) in an SSP target port (see 3.1.254).

3.1.254 SSP target port: A SCSI target port in a SAS domain that implements SSP.

3.1.255 SSP target/initiator port: A port that has all the characteristics of an SSP initiator port and an SSP target port.

3.1.256 state machine variable: A variable that exists within the context of a state machine that may log status determined in one state that is used in another state of the same state machine affecting subsequent state transitions or state machine outputs.

3.1.257 STP initiator phy: A SAS initiator phy (see 3.1.196) in an STP initiator port (see 3.1.258).

3.1.258 STP initiator port: A SAS initiator device object in a SAS domain that interfaces to a service delivery subsystem (see 3.1.229) with STP.

3.1.259 STP phy: A SAS logical phy (see 3.1.199) in an STP port.

3.1.260 STP port: An STP initiator port (see 3.1.258), STP target port (see 3.1.263), or STP target/initiator port (see 3.1.264).

3.1.261 STP primitive: A primitive used only inside STP connections and on SATA physical links. See 7.2.2.

3.1.262 STP target phy: A SAS target phy (see 3.1.202) in an STP target port (see 3.1.263).

3.1.263 STP target port: A SAS target device object in a SAS domain that interfaces to a service delivery subsystem (see 3.1.229) with STP.

3.1.264 STP target/initiator port: A port that has all the characteristics of an STP initiator port and an STP target port.

3.1.265 STP/SATA bridge: An expander device object containing an STP target port, a SATA host port, and the functions required to forward information between the STP target port and SATA host port to enable STP initiator ports in a SAS domain to communicate with SATA devices in an ATA domain.

3.1.266 subtractive routing attribute: The attribute of an expander phy that indicates it may be used by the ECM (see 3.1.71) to route connection requests to an attached expander device that were not resolved using the direct routing method or table routing method. See 4.6.7.1.

3.1.267 subtractive routing method: The method the ECM (see 3.1.71) uses to route connection requests not resolved using the direct routing method or table routing method to an attached expander device (see 3.1.73). See 4.6.7.1.

3.1.268 symbol: The smallest unit of data transmission on a physical link (i.e., a bit).

3.1.269 table routing attribute: The attribute of an expander phy that indicates it may be used by the ECM (see 3.1.71) to route connection requests using an expander route table (see 3.1.80). See 4.6.7.1.

3.1.270 table routing method: The method the ECM (see 3.1.71) uses to route connection requests to an attached expander device (see 3.1.73) using an expander route table (see 3.1.80). See 4.6.7.1.

3.1.271 task: An object within the logical unit representing the work associated with a command. See SAM-4.

3.1.272 task management function: A task manager service capable of being requested by a SCSI application client to affect the processing of one or more tasks. See SAM-4.

3.1.273 task manager: An object within a SCSI target device that controls the sequencing of SCSI commands and processes SCSI task management functions. See SAM-4.

3.1.274 training : Definition TBD.

Editor's Note 5: A definition of "training" will be added as part of the 6 Gbps phy specifications

3.1.275 training lock time (TLT): The maximum time during Train-SNW for a receiver to complete training (see 3.1.274) and transmit TRAIN_DONE patterns instead of TRAIN patterns (see 6.7.4.2.3.4).

3.1.276 Train-SNW window time (TWT): The actual duration of Train-SNW (see 6.7.4.2.3.4).

3.1.277 transceiver: A physical entity that contains both a transmitter device (see 3.1.280) and a receiver device (see 3.1.179).

3.1.278 transmitter circuit: An electronic circuit that converts a logic signal to an analog serial output signal.

3.1.279 transmitter compliance transfer function (TCTF): The mathematical statement of the transfer function through which the transmitter shall be capable of producing acceptable signals as defined by a receive mask. See 5.3.6.1.

3.1.280 transmitter device (Tx): The device upstream from an IT or CT compliance point containing a portion of the physical link and a transmitter circuit (see 3.1.278).

3.1.281 transport protocol service confirmation: Information passed from the transport layer to the application layer (i.e., from the SSP initiator port to the SCSI application client) that notifies the application layer that a SCSI transport protocol service has completed.

3.1.282 transport protocol service indication: Information passed from the transport layer to the application layer notifying the application layer (i.e., from the SSP target port to the SCSI device server) to begin a SCSI transport protocol service.

3.1.283 transport protocol service request: Information passed from the SCSI application layer to the SSP transport layer (i.e., from the SCSI application client to the SCSI initiator port) to begin a SCSI transport protocol service.

3.1.284 transport protocol service response: Information passed from the application layer to the transport layer (i.e., from the SCSI device server to the SSP target port) that completes the SCSI transport protocol service.

3.1.285 TxRx connection: The complete simplex signal path between the transmitter circuit (see 3.1.278) and receiver circuit (see 3.1.178). See 5.3.3.

3.1.286 TxRx connection segment: That portion of a TxRx connection (see 3.1.285) delimited by separable connectors or changes in the conductive material. See 5.3.3.

3.1.287 unit interval (UI): The normalized, dimensionless, nominal duration of a signal transmission bit (e.g., 666,6 ps at 1,5 Gbps, 333,3 ps at 3 Gbps, and 166,6 ps at 6 Gbps). Unit interval is a measure of time that has been normalized such that 1 UI is equal to 1/baud seconds.

3.1.288 valid character: A character that is a control character (see 3.1.42) or a data character (see 3.1.47).

3.1.289 valid dword: A dword that is a data dword (see 3.1.48) or a primitive (see 3.1.169).

3.1.290 virtual phy: A phy (see 3.1.158) that interfaces with a vendor-specific interface to another virtual phy inside the same device. See 4.1.2.

3.1.291 wide link: A group of physical links that attaches a wide port to another wide port. See 4.1.4.

3.1.292 wide port: A port that contains more than one phy. See 4.1.4.

3.1.293 wrapping counter: A counter that wraps back to zero after reaching its maximum value.

3.1.294 write data: Data transferred from the SCSI application client's data-out buffer to the SCSI device server, as requested by the Request Data-Out transport protocol service (see 10.2.1.8).

3.1.295 zone group: A set of phys in a ZPSDS (see 3.1.298) that all have the same access permission. See 4.9.

3.1.296 zone manager: The entity responsible for configuring a ZPSDS (see 3.1.298). See 4.9.1.

3.1.297 zone permission table: The table that defines access permission between the zone group of a source phy and the zone group of the destination phy. See 4.9.3.3.

3.1.298 zoned portion of a service delivery subsystem (ZPSDS): A group of zoning expander devices (see 3.1.300) that cooperate to control access between phys. The ZPSDS may include all or part of a service delivery subsystem (see 3.1.229). See 4.9.

3.1.299 zoning expander active values: The active zone permission table (see 3.1.297) and zone phy information in a zoning expander device (see 3.1.300). See 4.9.6.

3.1.300 zoning expander device: An expander device (see 3.1.73) that supports zoning. See 4.9.

3.1.301 zoning expander phy: An expander phy in a zoning expander device (see 3.1.300).

3.1.302 zoning expander shadow values: The shadow zone permission table (see 3.1.297) and zone phy information in a zoning expander device, which are changed by SMP zone configuration functions (see 3.1.242) but do not become active until the activate step (see 4.9.6.4) is performed. See 4.9.6.

3.2 Symbols and abbreviations

See 2.1 for abbreviations of standards bodies (e.g., ISO). Units and abbreviations used in this standard:

Abbreviation	Meaning
AA	ATA application layer (see 10.3)
A.C.	alternating current
ACK	acknowledge primitive (see 7.2.6.1)
AIP	arbitration in progress primitive (see 7.2.5.1)
ALT	actual lock time (see 3.1.4)
ATA	AT attachment (see 3.1.14)
ATAPI	AT attachment packet interface
ATA8-AAM	AT Attachment - 8 ATA/ATAPI Architecture Model standard (see 2.3)
ATA8-ACS	AT Attachment - 8 ATA/ATAPI Command Set standard (see 2.3)
ATT	actual training time (see 3.1.5)
AWG	American wire gauge
AWT	arbitration wait time
BCH	Bose, Chaudhuri and Hocquenghem code (see 4.2.3)
BCT	bit cell time (see 3.1.20)
BER	bit error ratio (see 3.1.21)
BIST	built in self test
BPP	broadcast propagation processor (see 3.1.23)
CDB	command descriptor block (see 3.1.31)
CDF	cumulative distribution function (see 3.1.43)
CDR	clock data recovery (see 3.1.30)
CJTPAT	compliant jitter tolerance pattern (see 3.1.33)
CRC	cyclic redundancy check (see 3.1.44)
dB	decibel (see 3.1.50)
D.C.	direct current (see 3.1.55)
Dxx.y	data character (see 3.1.47)
DDJ	data dependent jitter (see 3.1.114)
DJ	deterministic jitter (see 3.1.115)
e	2,718 28..., the base of the natural (i.e., hyperbolic) system of logarithms
ECM	expander connection manager (see 3.1.71)
ECR	expander connection router (see 3.1.72)
EMI	electromagnetic interference (see 3.1.64)

Abbreviation	Meaning
EOAF	end of address frame primitive (see 7.2.5.7)
EOF	end of frame primitive (see 7.2.6.4)
ESD	electrostatic discharge
Final-SNW	final speed negotiation window for 1,5 or 3 Gbps without training (see 6.7.4.2.3.2)
FIS	frame information structure (see 3.1.89)
G1	generation 1 physical link rate (1,5 Gbps)
G2	generation 2 physical link rate (3 Gbps)
G3	generation 3 physical link rate (6 Gbps)
G4	generation 4 physical link rate (defined in a future version of this standard)
Gbps	gigabits per second (10^9 bits per second)
Gen1i	SATA generation 1 physical link rate (1,5 Gbps)(see SATA-2)
Gen1x	SATA generation 1 physical link rate (1,5 Gbps), extended length (see SATA-2)
Gen2i	SATA generation 2 physical link rate (3 Gbps)(see SATA-2)
Gen2x	SATA generation 2 physical link rate (3 Gbps), extended length (see SATA-2)
GHz	gigahertz (10^9 cycles per second)
Hz	hertz (cycles per second)
ISI	intersymbol interference (see 3.1.110)
IU	information unit (see 3.1.106)
JTPAT	jitter tolerance pattern (see 3.1.121)
kHz	kilohertz (10^3 cycles per second)
Kxx.y	control character (see 3.1.42)
LED	light-emitting diode
LSB	least significant bit (see 3.1.122)
LUN	logical unit number (see 3.1.132)
μ A	microampere (10^{-6} amperes)
μ s	microsecond (10^{-6} seconds)
MA	management application layer (see 10.4)
Mbaud	megabaud (10^6 symbols per second)
MBps	megabytes per second (10^6 bytes per second)
MHz	megahertz (10^6 cycles per second)
MSB	most significant bit (see 3.1.138)
ms	millisecond (10^{-3} seconds)
MT	SMP transport layer state machines (see 9.4)
MTT	maximum training time (see 3.1.133)
MTWT	maximum Train-SNW window time (see 3.1.134)
mV	millivolt (10^{-3} volts)
N/A	not applicable
NAA	name address authority (see 4.2)
NAK	negative acknowledge primitive (see 7.2.6.5)
NEXT	near-end crosstalk (see 3.1.143)
nF	nanofarad (10^{-9} Farads)

Abbreviation	Meaning
ns	nanosecond (10^{-9} seconds)
NVRAM	non-volatile random-access memory
OOB	out-of-band
OOBI	out-of-band interval (see 3.1.151)
OUI	organizationally unique identifier (i.e., company identifier)
PCB	printed circuit board
PL	port layer state machines (see 8.2)
PLL	phase lock loop
P-P	peak-to-peak
ppm	parts per million (10^{-6})
ps	picosecond (10^{-12} seconds)
ρ	reflection coefficient (rho)(see 3.1.180)
RCDT	rate change delay time (see 3.1.176)
RD	running disparity (see 3.1.191)
RJ	random jitter (see 3.1.116)
RRDY	receiver ready primitive (see 7.2.6.6)
Rx	receiver device (see 3.1.179)
SA	SCSI application layer (see 10.2)
SAM-4	SCSI Architecture Model - 4 standard (see 2.3)
SAS	Serial Attached SCSI (see 3.1.226)
SATA	Serial ATA (see 3.1.224)
SATA-2	Serial ATA 2.6 specification (see 2.4)
SBC-3	SCSI Block Commands - 3 standard (see 2.3)
SCSI	Small Computer System Interface family of standards
S_{ij}	S-parameter for port j to port i (see B.9)
S_{CCij}	S-parameter for common-mode to common-mode port j to port i (see B.9)
S_{CDij}	S-parameter for differential to common-mode port j to port i (see B.9)
S_{DCij}	S-parameter for common-mode to differential port j to port i (see B.9)
S_{DDij}	S-parameter for differential to differential port j to port i (see B.9)
SJ	sinusoidal jitter (see 3.1.117)
SL	link layer for SAS phys state machines (see 7.14)
SL_IR	link layer identification and hard reset state machines (see 7.9.4)
SMP	Serial Management Protocol (see 3.1.227), or link layer for SMP phys state machines (see 7.18.5)
SNLT	speed negotiation lock time (see 3.1.243)
SNTT	speed negotiation transmit time (see 3.1.245)
SNW	speed negotiation window (see 3.1.246)
SNW-1	speed negotiation window for 1.5 Gbps without training (see 6.7.4.2.3.2)
SNW-2	speed negotiation window for 3 Gbps without training (see 6.7.4.2.3.2)
SNW-3	speed negotiation window negotiating physical link rates with training (see 6.7.4.2.3.3)
SNWT	speed negotiation window time (see 3.1.247)
SOAF	start of address frame primitive (see 7.2.5.14)

Abbreviation	Meaning
SOF	start of frame primitive (see 7.2.6.7)
SP	phy layer state machine (see 6.8)
SP_DWS	phy layer dword synchronization state machine (see 6.9)
SPC-4	SCSI Primary Commands - 4 standard (see 2.3)
SSP	Serial SCSI Protocol (see 3.1.228), or link layer for SSP phys state machines (see 7.16.8)
ST	SSP transport layer state machines (see 9.2)
STP	Serial ATA Tunneled Protocol (see 3.1.225), or link layer for STP phys state machines (see 7.17.8)
s	second (unit of time)
TCTF	transmitter compliance transfer function (see 3.1.279)
TDNA	time domain network analyzer (i.e., TDR/TDT plus analysis software that performs a VNA-style output)
TDR	time domain reflectometer
TDT	time domain transmission
TJ	total jitter (see 3.1.118)
TLT	training lock time (see 3.1.275)
Train-SNW	speed negotiation window with training (see 6.7.4.2.3.4)
TT	STP transport layer state machines (see 9.3)
TWT	Train-SNW window time (see 3.1.276)
Tx	transmitter device (see 3.1.280)
UI	unit interval (see 3.1.287)
V	volt
VNA	vector network analyzer
VPD	vital product data (see 10.2.11)
XL	link layer for expander phys state machine (see 7.15)
XOR	exclusive logical OR
ZP[s, d]	Zone permission bit for a source zone group (i.e., s) and a destination zone group (i.e., d) in the zone permission table (see 4.9.3.3)
ZPSDS	Zoned portion of a service delivery subsystem (see 3.1.298)
^	exclusive logical OR
x	multiplication
/	division

3.3 Keywords

3.3.1 invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

3.3.2 mandatory: A keyword indicating an item that is required to be implemented as defined in this standard.

3.3.3 may: A keyword that indicates flexibility of choice with no implied preference (equivalent to “may or may not”).

3.3.4 may not: Keywords that indicate flexibility of choice with no implied preference (equivalent to “may or may not”).

3.3.5 obsolete: A keyword indicating that an item was defined in prior standards but has been removed from this standard.

3.3.6 optional: A keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined in this standard is implemented, then it shall be implemented as defined in this standard.

3.3.7 reserved: A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.8 restricted: A keyword referring to bits, bytes, words, and fields that are set aside for use in other standards or for other data structures in this standard. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

3.3.9 shall: A keyword indicating a mandatory requirement (equivalent to “is required”). Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.10 should: A keyword indicating flexibility of choice with a strongly preferred alternative (equivalent to “is strongly recommended”).

3.3.11 vendor specific: Something (e.g., a bit, field, or code value) that is not defined by this standard and may be used differently in various implementations.

3.4 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

Names of signals, address frames, primitives and primitive sequences, SMP functions, state machines, SCSI and ATA commands, SCSI statuses, SCSI sense keys, and SCSI additional sense codes are in all uppercase (e.g., REQUEST SENSE command).

Names of messages, requests, confirmations, indications, responses, event notifications, timers, SCSI diagnostic pages, SCSI mode pages, and SCSI log pages are in mixed case (e.g., Disconnect-Reconnect mode page).

Names of fields are in small uppercase (e.g., DESTINATION SAS ADDRESS). Normal case is used when the contents of a field are being discussed. Fields containing only one bit are usually referred to as the NAME bit instead of the NAME field.

Normal case is used for words having the normal English meaning.

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included between characters in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included between characters in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

This standard uses the ISO convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a space, and a comma is used as the decimal point). Table 2 shows some examples of decimal numbers using the ISO and American numbering conventions.

Table 2 — ISO and American numbering conventions

ISO	American
0,6	0.6
3,141 592 65	3.14159265
1 000	1,000
1 323 462,95	1,323,462.95

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666,6̄ means 666,666 666... or 666 2/3, and 12.142 857̄ means 12.142 857 142 857... or 12 1/7).

Lists sequenced by letters (e.g., a) red, b) blue, c) green) show no ordering relationship between the listed items. Lists sequenced by numbers (e.g., 1) red, 2) blue, 3) green) show an ordering relationship between the listed items.

In the event of conflicting information the precedence for requirements defined in this standard is:

- 1) text;
- 2) tables; then
- 3) figures.

Notes do not constitute any requirements for implementers.

3.5 Class diagram and object diagram conventions

The notation used in class diagrams and object diagrams is based on the Unified Modeling Language (UML) specification.

Figure 3 shows the notation used for classes in class diagrams.

Notation for a class with no attributes or operations:

Class Name

Notation for a class with attributes and no operations:

Class Name
Attribute 1
Attribute 2

Notation for a class with operations and no attributes:

Class Name
Operation 1()
Operation 2()

Notation for a class with attributes and operations:

Class Name
Attribute 1
Attribute 2
Operation 1()
Operation 2()

Notation for a class with attributes showing multiplicity and operations:

Class Name
Attribute 1[1..*]
Attribute 2[1]
Operation 1()
Operation 2()

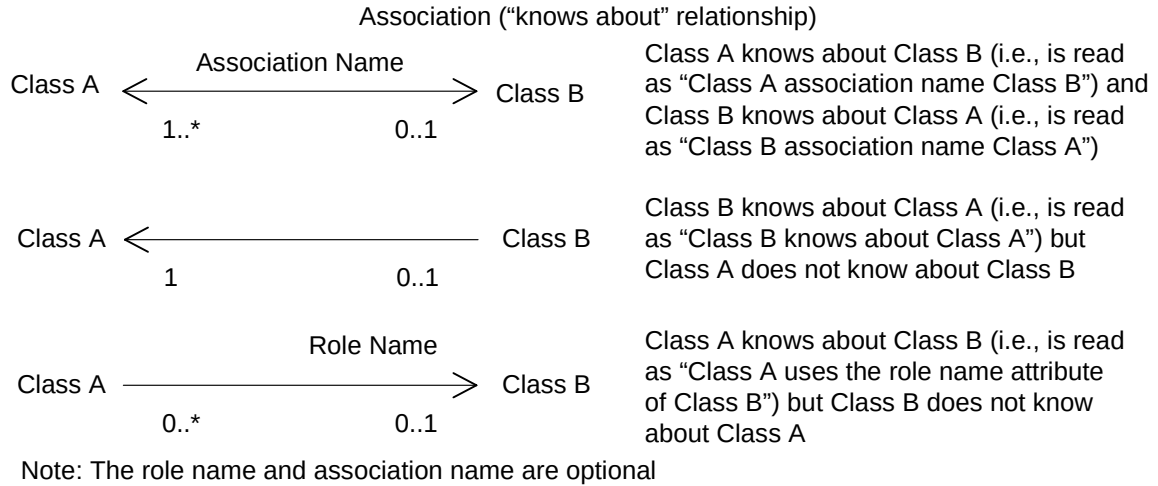
Figure 3 — Classes in class diagrams

Table 3 defines the notation used for multiplicity in class diagrams.

Table 3 — Multiplicity notation in class diagrams

Notation	Description
<none>	The number of instances of the object or attribute is not specified.
1	One instance of the object or attribute exists.
0..*	Zero or more instances of the object or attribute exist.
1..*	One or more instances of the object or attribute exist.
0..1	Zero or one instances of the object or attribute exist.
n..m	n to m instances of the object or attribute exist (e.g., 2..8).
x, n..m	Multiple disjoint instances of the object or attribute exist (e.g., 2, 8..15).

Figure 4 defines the notation used for association relationships between classes.



Examples of class diagrams using associations:

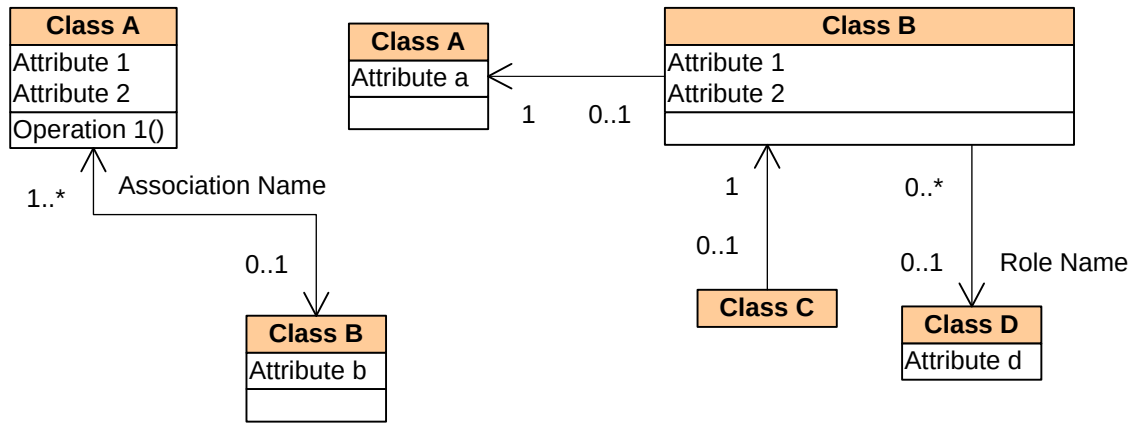


Figure 4 — Association relationships in class diagrams

Figure 5 defines the notation used for aggregation relationships between classes.

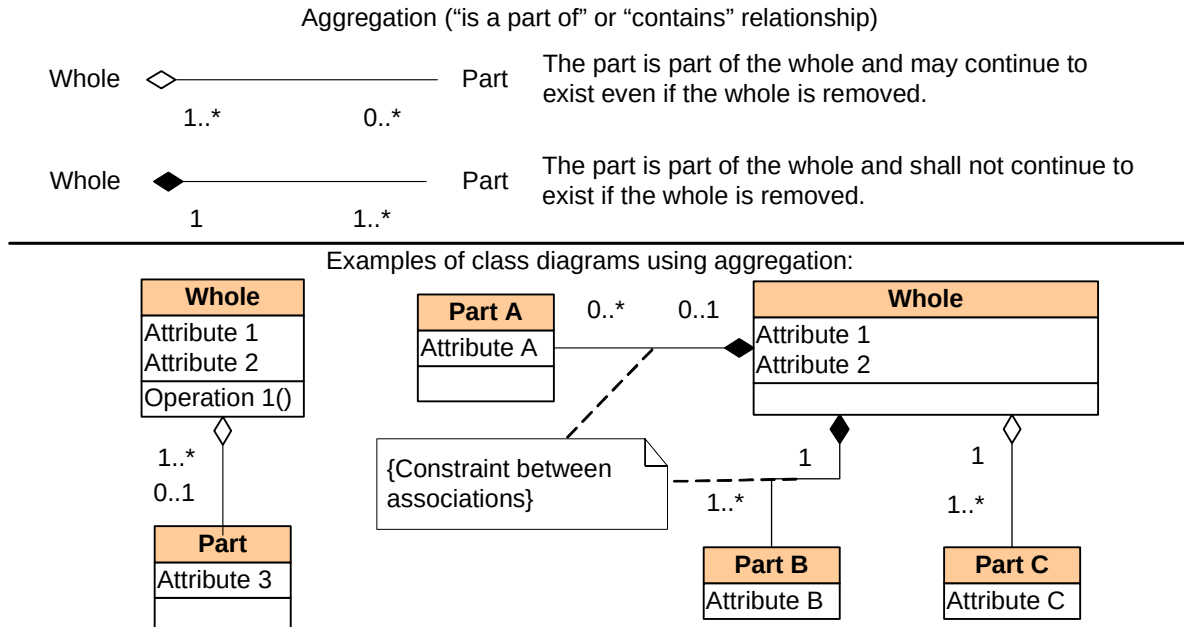


Figure 5 — Aggregation relationships in class diagrams

Editor's Note 6: "Aggregation sets" might be helpful for modeling SAS ports and phys, where we have struggled to define the protocol (SMP, STP, SSP) and role (initiator/target) combinations.

Figure 6 defines the notation used for generalization relationships between classes.

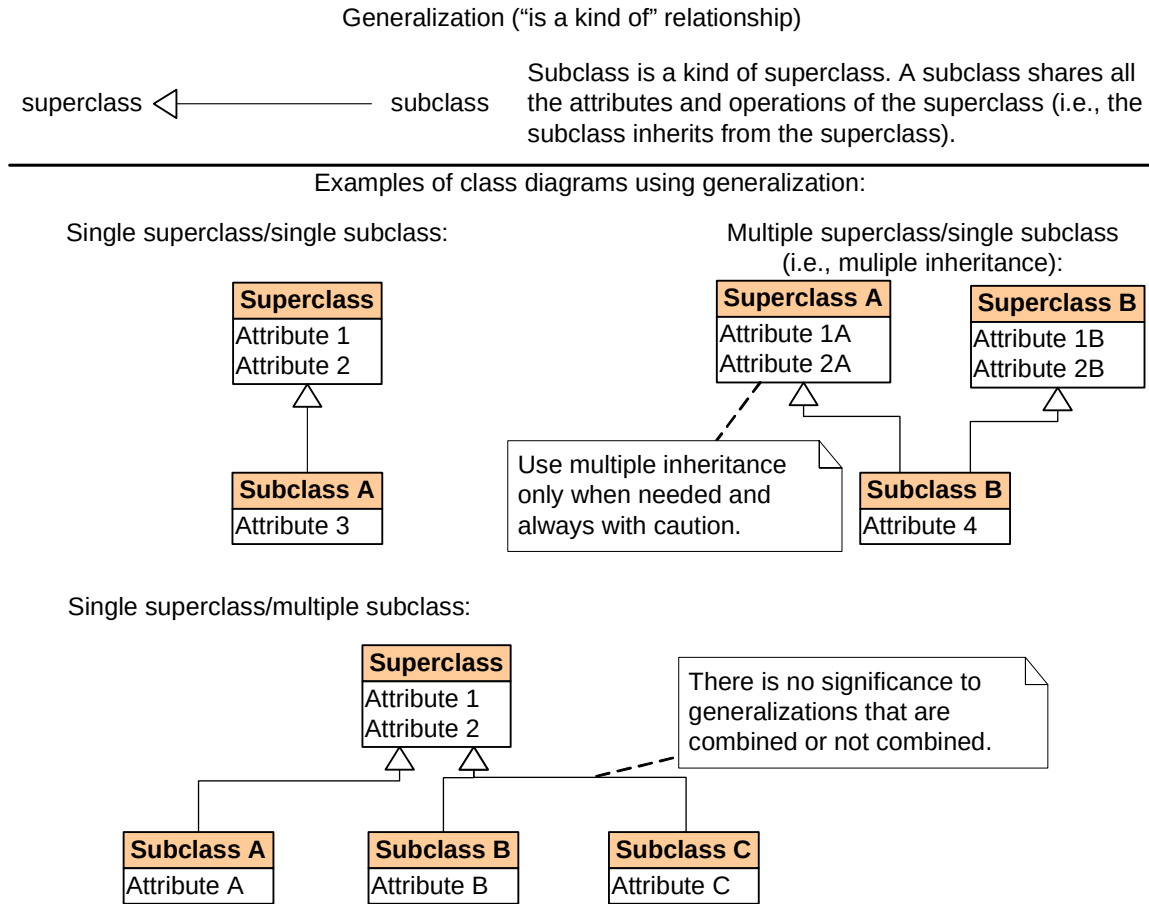


Figure 6 — Generalization relationships in class diagrams

Figure 7 defines the notation used for dependency relationships between classes.

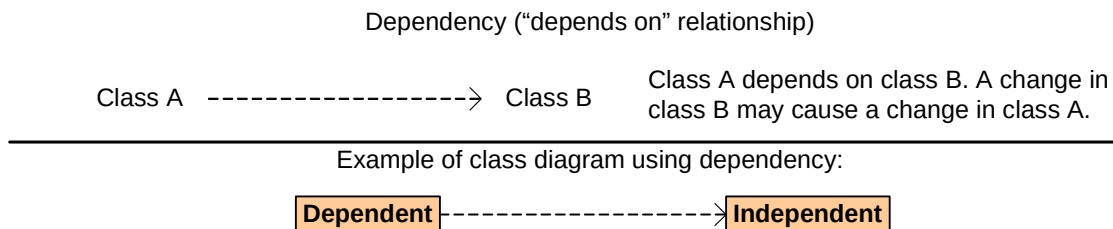


Figure 7 — Dependency relationships in class diagrams

Figure 8 defines the notation used for objects in object diagrams.

Notation for a named object with no attributes:

label : Class Name

Notation for a named object with attributes:

label : Class Name
Attribute 1 = x
Attribute 2 = y

Notation for an anonymous object with no attributes:

: Class Name

Notation for an anonymous object with attributes:

: Class Name
Attribute 1 = x
Attribute 2 = y

Figure 8 — Objects in object diagrams

A constraint is specified in a class diagram or an object diagram as text encapsulated with a { } notation within a box. See figure 5 for an example of a constraint.

A note is specified in a class diagram or an object diagram as text within a box. See figure 6 for an example of a note.

3.6 State machine conventions

3.6.1 State machine conventions overview

Figure 9 shows how state machines are described. See 4.3 for a summary of the state machines in this standard.

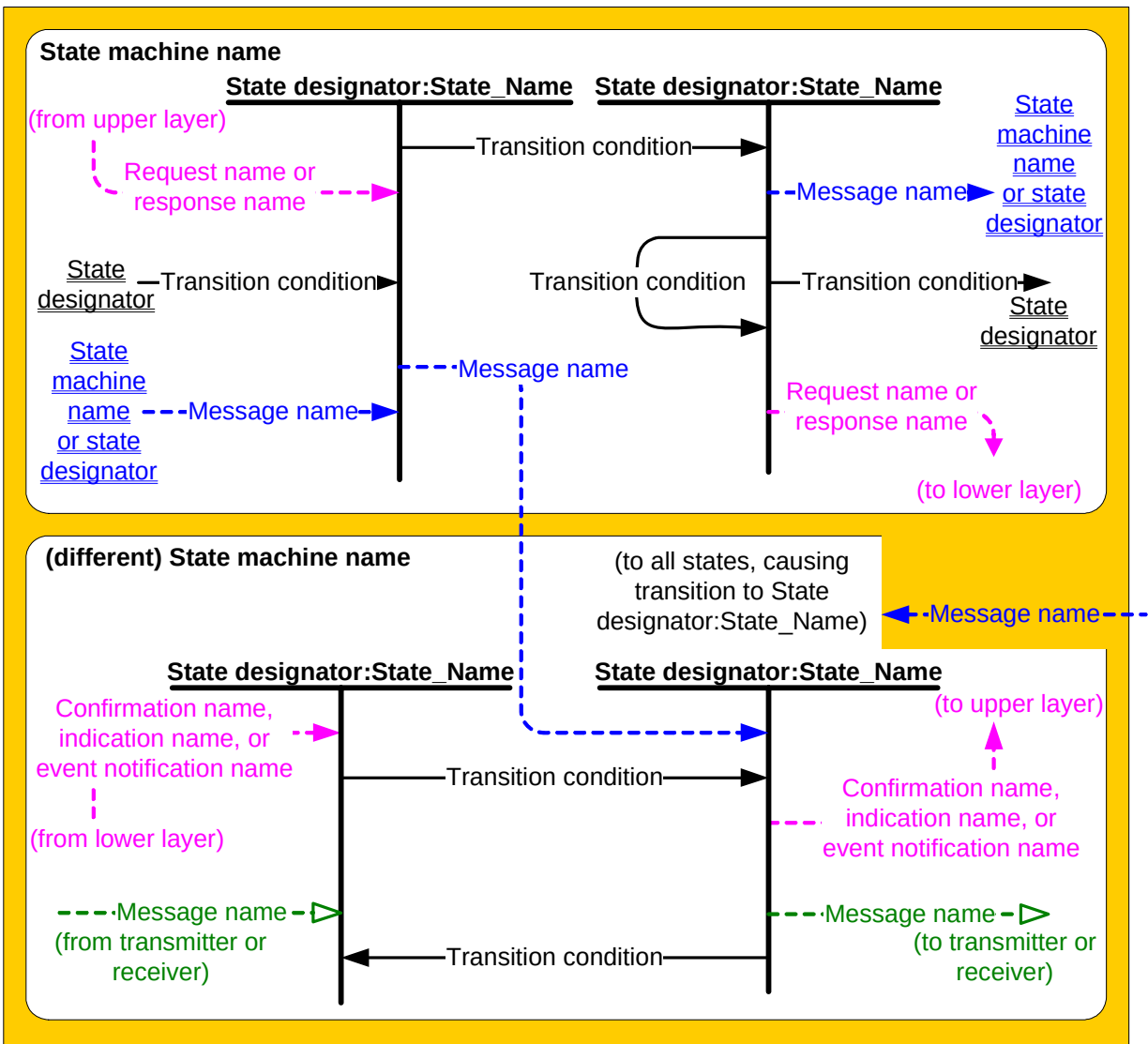


Figure 9 — State machine conventions

When multiple state machines are present in a figure, they are enclosed in boxes with rounded corners.

Each state machine is identified by a state machine name. In state machines with one state, the state machine is identified by a state designator. In state machines with multiple states, each state is identified by a state designator and a state name. The state designator (e.g., SL1) is unique among all state machines in this standard. The state name (e.g., Idle) is a brief description of the primary action taken during the state, and the same state name may be used by other state machines. Actions taken while in each state are described in the state description text.

3.6.2 Transitions

Transitions between states are shown with solid lines, with an arrow pointing to the destination state. A transition may be labeled with a transition condition label, a brief description of the event or condition that causes the transition to occur.

If the state transition leaves the page, the transition label goes to or from a state designator label with double underlines rather than to or from a state.

The conditions and actions are described fully in the transition description text. In case of a conflict between a figure and the text, the text shall take precedence.

Upon entry into a state, all actions to be processed in that state are processed. If a state is re-entered from itself, all actions to be processed in the state are processed again. A state may be entered and exited in zero time if the conditions for exiting the state are valid upon entry into the state. Transitions between states are instantaneous.

3.6.3 Messages, requests, indications, confirmations, responses, and event notifications

Messages passed between state machines are shown with dashed lines labeled with a message name. When messages are passed between state machines within the same layer of the protocol, they are identified by either:

- a) a dashed line to or from a state machine name label with double underlines and/or state name label with double underlines, if the destination is in a different figure from the source;
- b) a dashed line to or from a state in another state machine in the same figure; or
- c) a dashed line from a state machine name label with double underlines to a "(to all states)" label, if the destination is every state in the state machine.

The meaning of each message is described in the state description text.

Requests, indications, confirmations, responses, and event notifications are shown with curved dashed lines originating from or going toward the top or bottom of the figure. Each request, indication, confirmation, response, and event notification is labeled. The meaning of each request, indication, confirmation, response, and event notification is described in the state description text.

Messages with unfilled arrowheads are passed to or from the state machine's transmitter or receiver, not shown in the state machine figures, and are directly related to data being transmitted on or received from the physical link.

The state machine description text for each state wholly defines the messages sent while the state machine is in that state. If a state machine in one state repeatedly sending a message transitions to another state, it stops repeatedly sending that message unless stated otherwise in the new state.

3.6.4 State machine counters, timers, and variables

State machines may contain counters, timers, and variables that affect the operation of the state machine. The following properties apply to counters, timers, and variables:

- a) Their scope is the state machine itself;
- b) They are created and deleted with the state machines with which they are associated;
- c) Their initialization and modification is specified in the state descriptions and the transition descriptions; and
- d) Their current values may be used to determine the behavior of a state and select the transition out of a state.

State machine timers may continue to run while a state machine is in a given state, and a timer may cause a state transition upon reaching a defined threshold value (e.g., zero for a timer that counts down).

3.6.5 State machine arguments

State machines may contain an argument received in a message or confirmation as a state machine arguments. The following properties apply to state machine arguments:

- a) the state machine that sends the argument owns that argument's value;
- b) the state machine that receives the argument shall not modify those argument's values;
- c) the state machine that sends the argument may resend those arguments with different values;
- d) the scope of a state machine argument is the state machine itself; and
- e) state machine argument usage is described in the state descriptions and the transition descriptions.

3.7 Bit and byte ordering

In a field in a table consisting of more than one bit that contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left; bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of 8 or fewer bits.

In a field in a table consisting of more than one byte that contains a single value (e.g., a number), the byte containing the MSB is stored at the lowest address and the byte containing the LSB is stored at the highest address (i.e., big-endian byte ordering). The MSB and LSB are labeled.

NOTE 6 - SATA numbers bits within fields the same as this standard, but uses little-endian byte ordering.

In a field in a table consisting of more than one byte that contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB, but they are not labeled.

In a field containing a text string (e.g., ASCII or UTF-8), the MSB label is the MSB of the first character and the LSB label is the LSB of the last character.

Multiple byte fields are represented with only two rows, with the non-sequentially increasing byte number indicating the presence of additional bytes.

A data dword consists of 32 bits. Table 4 shows a data dword containing a single value, where the MSB is on the left in bit 31 and the LSB is on the right in bit 0.

Table 4 — Data dword containing a value

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0					
MSB																Value																LSB				

Table 5 shows a data dword containing four one-byte fields, where byte 0 (the first byte) is on the left and byte 3 (the fourth byte) is on the right. Each byte has an MSB on the left and an LSB on the right.

Table 5 — Data dword containing four one-byte fields

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																
MSB				Byte 0 (First byte)				LSB				MSB				Byte 1 (Second byte)				LSB				MSB				Byte 2 (Third byte)				LSB				MSB				Byte 3 (Fourth byte)				LSB			

3.8 Notation for procedures and functions

In this standard, the model for functional interfaces between objects is the callable procedure. Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] ...), OUT ([output-1] [,output-2] ...))

where:

Result	A single value representing the outcome of the procedure or function.
Procedure Name:	A descriptive name for the function to be performed.
IN (Input-1, Input-2, ...)	A comma-separated list of names identifying caller-supplied input data objects.
OUT (Output-1, Output-2, ...)	A comma-separated list of names identifying output data objects to be returned by the procedure.
[...]	Brackets enclose optional or conditional parameters and arguments.

This notation allows data objects to be specified as inputs and outputs.

4 General

4.1 Architecture

4.1.1 Architecture overview

A SAS domain (see 4.1.9) contains one or more SAS devices and a service delivery subsystem. A SAS domain may be a SCSI domain (see SAM-4).

A SAS device (see 4.1.6) contains one or more SAS ports (see 4.1.4). A SAS device may be a SCSI device (see SAM-4).

A SAS port (see 4.1.4) contains one or more phys (see 4.1.2). A SAS port may be a SCSI port (see SAM-4).

The service delivery subsystem (see 4.1.8) in a SAS domain may contain expander devices (see 4.1.7).

Expander devices contain expander ports (see 4.1.4) and one SMP port.

An expander port contains one or more phys (see 4.1.2).

An expander device shares its phys with the SAS device(s) contained within the expander device.

Figure 10 shows the class diagram for a SAS domain, showing the relationships between SAS Domain, SCSI Domain, Service Delivery Subsystem, Expander Device, Expander Port, SAS Device, SCSI Device, SAS Port, SCSI Port, and Phy classes. Not all attributes are shown.

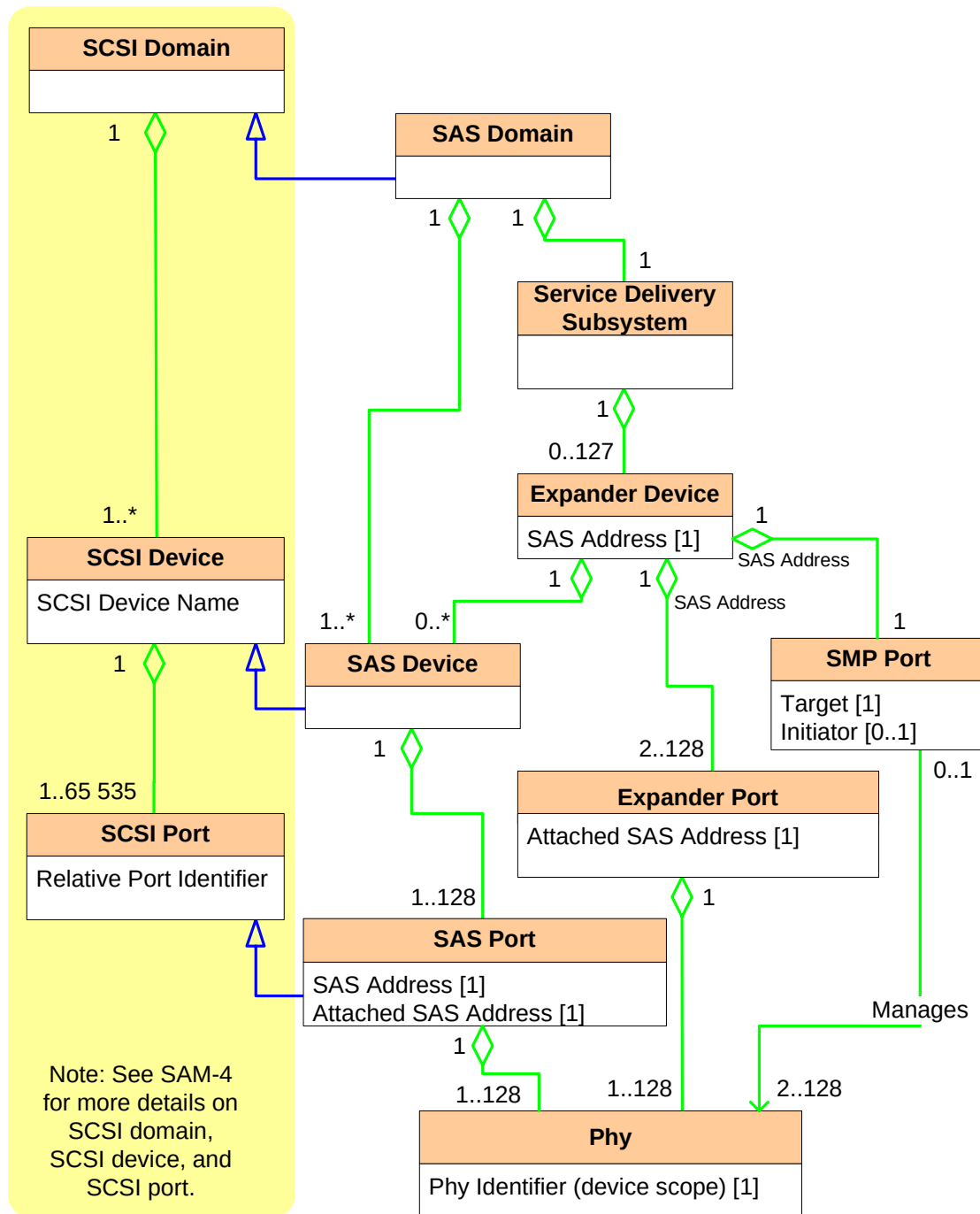


Figure 10 — SAS Domain class diagram

4.1.2 Physical links and phys

A physical link is a set of four wires used as two differential signal pairs. One differential signal transmits in one direction while the other differential signal transmits in the opposite direction. Data may be transmitted in both directions simultaneously.

A physical phy contains a transceiver which electrically interfaces to a physical link, which attaches to another physical phy. A virtual phy contains a vendor-specific interface to another virtual phy.

Phys are contained in ports (see 4.1.4). Phys interface to a service delivery subsystem (see 4.1.8).

Figure 11 shows two phys attached with a physical link.

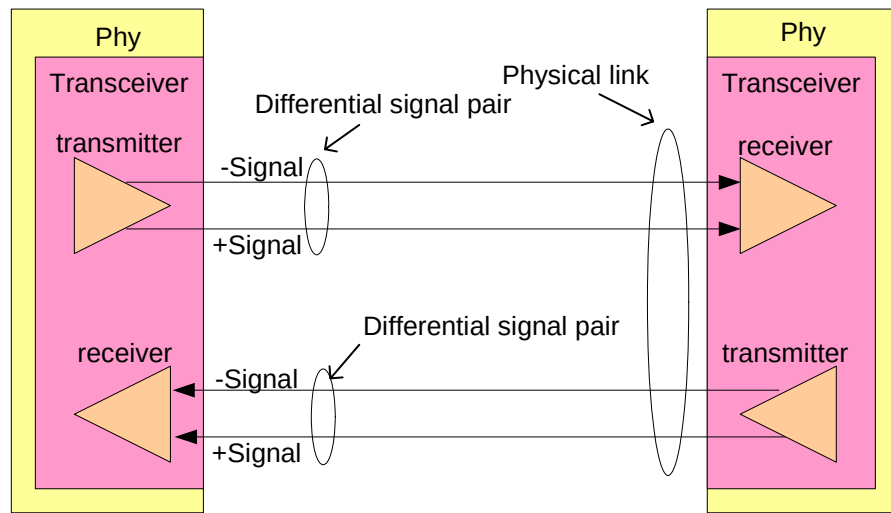


Figure 11 — Physical links and phys

An attached phy is the phy to which a phy is attached over a physical link.

A device (i.e., a SAS device (see 4.1.6) or expander device (see 4.1.7)) contains one or more phys.

Each phy has:

- a) a SAS address (see 4.2.2), inherited from the SAS port (see 4.1.4) or expander device;
- b) a phy identifier (see 4.2.8) which is unique within the device;
- c) optionally, support for being an SSP initiator phy;
- d) optionally, support for being an STP initiator phy;
- e) optionally, support for being an SMP initiator phy;
- f) optionally, support for being an SSP target phy;
- g) optionally, support for being an STP target phy; and
- h) optionally, support for being an SMP target phy.

A phy may be used as one or two logical phys based on multiplexing (see 6.10).

During the identification sequence (see 7.9), a logical phy:

- a) transmits an IDENTIFY address frame including the device type (i.e., end device or expander device) of the device containing the phy, the SAS address of the SAS port or expander device containing the logical phy, and other information; and
- b) receives an IDENTIFY address frame containing the same set of information from the attached logical phy, including the attached device type, the attached SAS address, the attached device name, and other attached information.

The transceiver follows the electrical specifications defined in 5.3. Phys transmit and receive bits at physical link rates defined in 5.3. The physical link rates supported by a phy are specified or indicated by the following fields in the SMP DISCOVER response (see 10.4.3.9), the SMP PHY CONTROL request (see 10.4.3.27), and the Phy Control and Discover mode page (see 10.2.7.5):

- c) the NEGOTIATED PHYSICAL LINK RATE field;
- d) the HARDWARE MINIMUM PHYSICAL LINK RATE field;
- e) the HARDWARE MAXIMUM PHYSICAL LINK RATE field;
- f) the PROGRAMMED MINIMUM PHYSICAL LINK RATE field; and
- g) the PROGRAMMED MAXIMUM PHYSICAL LINK RATE field.

The bits are part of dwords (see 6.2.1), each of which has been encoded using 8b10b coding into four 10-bit characters (see 6.2).

Figure 12 defines the Phy classes, showing the relationships between the following classes:

- a) Phy;
- b) Logical Phy;
- c) SAS phy;
- d) Logical SAS Phy;
- e) Expander Phy;
- f) Logical Expander Phy;
- g) SAS Initiator Phy;
- h) SAS Target Phy;
- i) SSP Phy;
- j) STP Phy; and
- k) SMP Phy.

SATA phys are also referenced in this standard but are defined by SATA (see SATA-2).

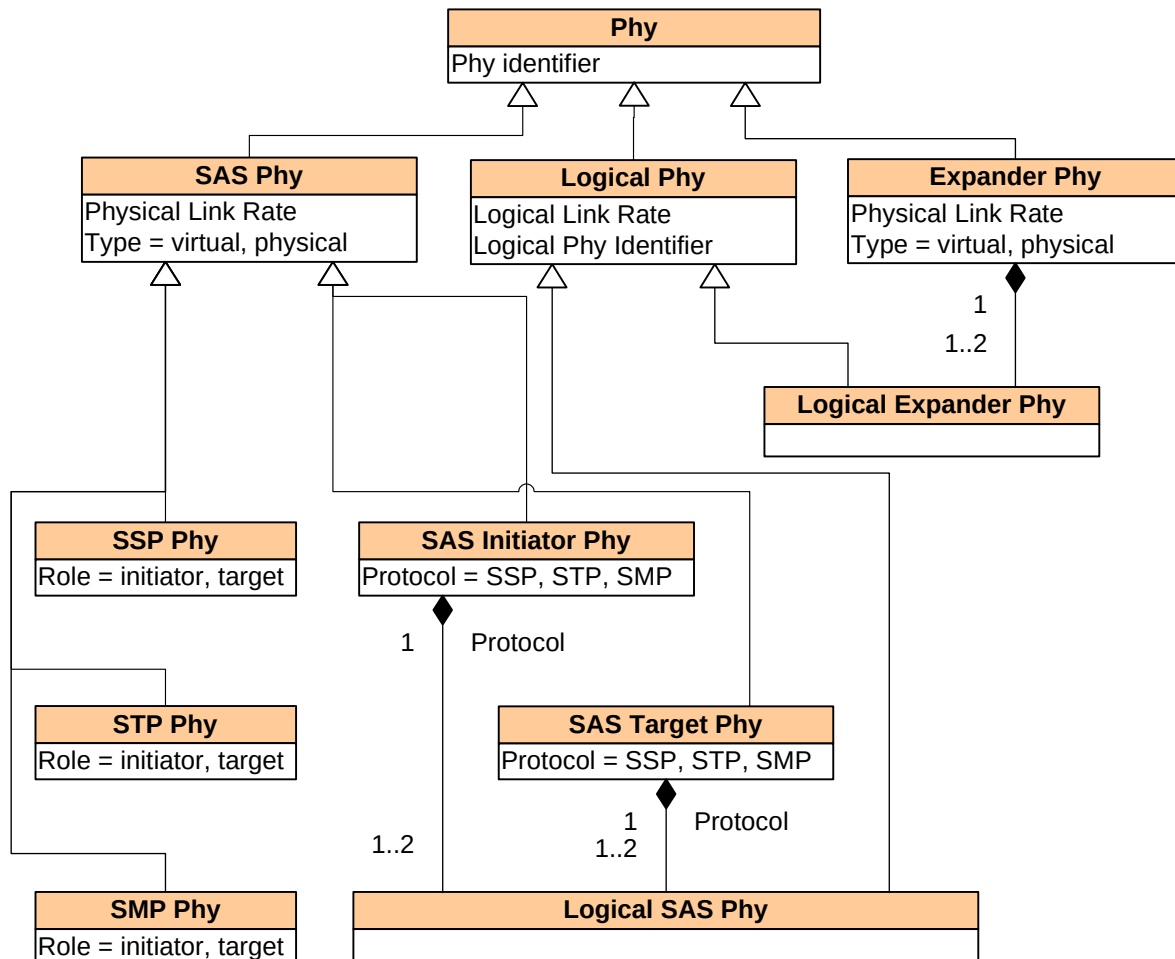


Figure 12 — Phy class diagram

Figure 13 shows example objects instantiated from the SAS Phy class, including:

- a) SSP initiator phy;
- b) SSP target phy;
- c) virtual SMP initiator phy;
- d) virtual SMP target phy;
- e) STP initiator phy;
- f) logical SAS phy.

A SAS phy is represented by one of these objects during each connection. A SAS phy may be represented by different phy objects in different connections.

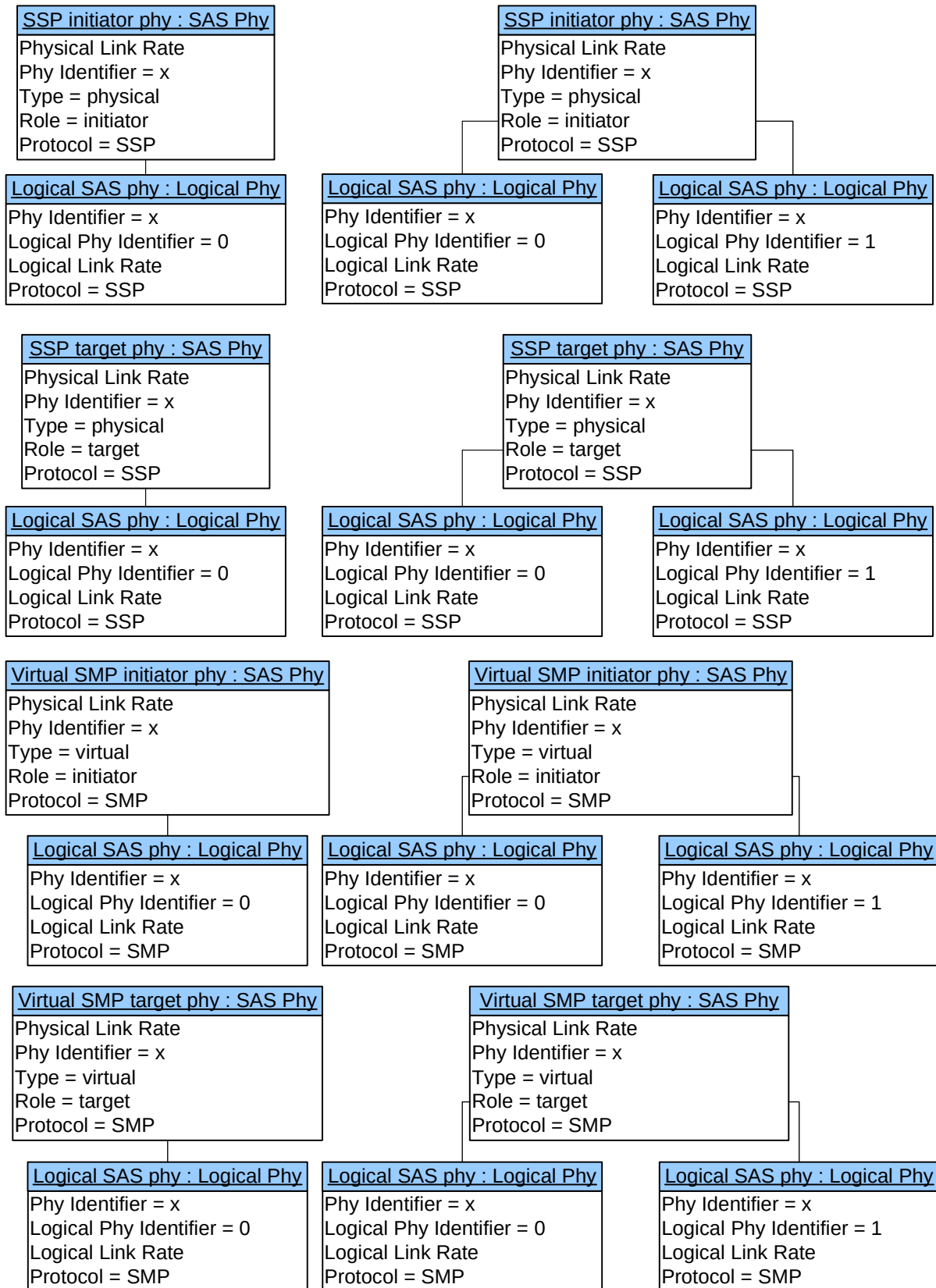


Figure 13 — SAS phy object diagram

Figure 3 shows the objects instantiated from the Expander Phy class, including:

- a) expander phy;
- b) virtual expander phy; and
- c) logical expander phy.

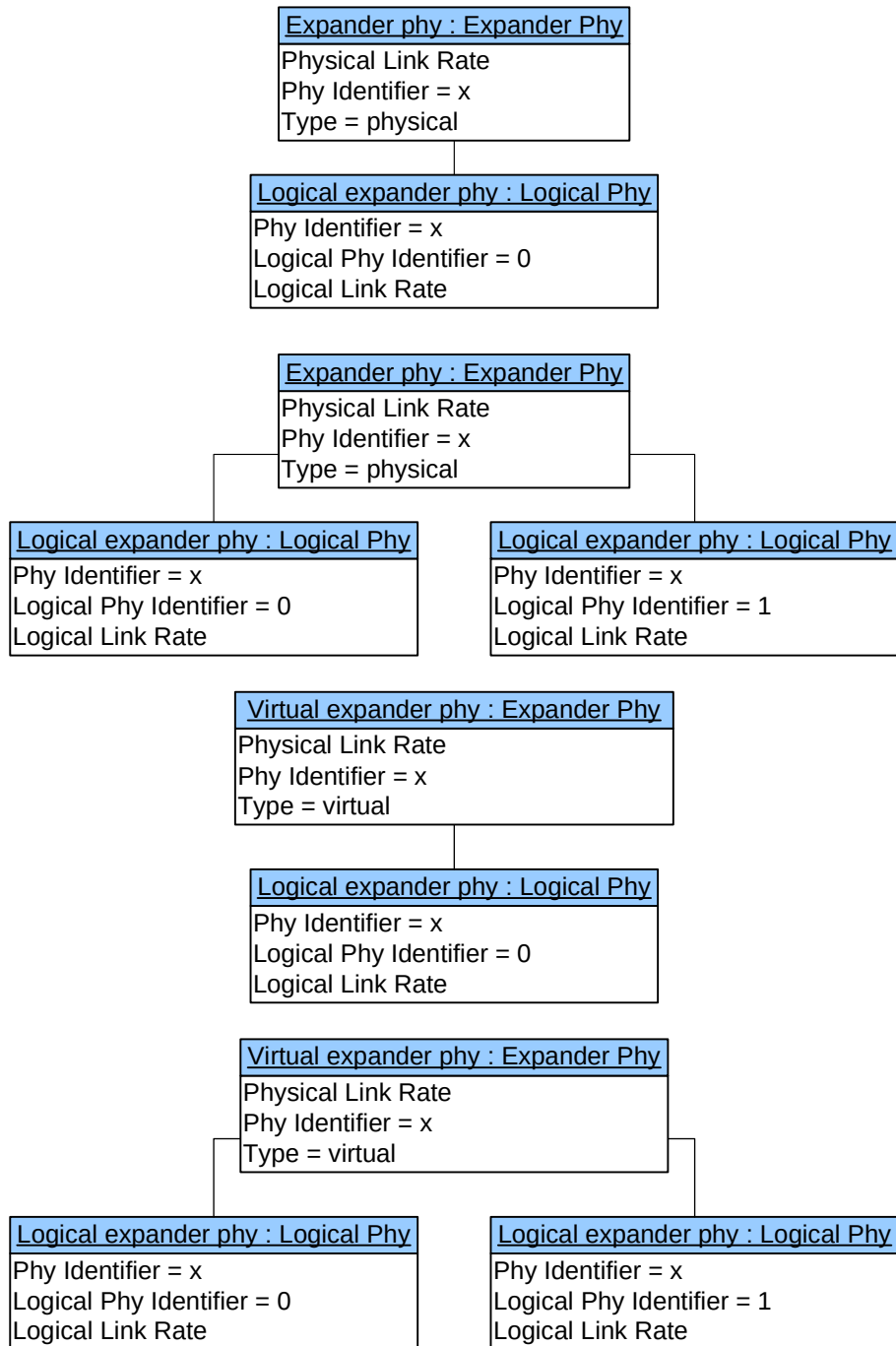


Figure 14 — Expander phy object diagram

4.1.3 Logical links

A physical link with a physical link rate greater than 1,5 Gbps may be multiplexed into two logical links as defined in table 6.

Table 6 — Logical links

Physical link rate	Logical link(s)
6 Gbps	One 6 Gbps logical link
	Two 3 Gbps logical links
3 Gbps	One 3 Gbps logical link
	Two 1,5 Gbps logical links
1,5 Gbps	One 1,5 Gbps logical link

Multiplexing is defined in 6.10.

4.1.4 Ports (narrow ports and wide ports)

A port contains one or more phys. Ports in a device are associated with physical phys based on the identification sequence (see 7.9). Ports are associated with virtual phys based on the design of the device.

A port is created from one or more physical phys if, during the identification sequence (see 7.9), they:

- a) transmitted the same SAS address (see 4.2) that the other physical phys in that port also transmitted in their outgoing IDENTIFY address frames (i.e., the SAS address is the same); and
- b) received the same SAS address that the other physical phys in that port also received in their incoming IDENTIFY address frames (i.e., the attached SAS address is the same).

A port is a wide port if there are more than one phy in the port. A port is a narrow port if there is only one phy in the port.

A wide link is the set of physical links that attach a wide port to another wide port. A narrow link is the physical link that attaches a narrow port to another narrow port.

Attaching phys within a wide port to other phys in the same wide port (i.e., the SAS address transmitted in the outgoing IDENTIFY address frame is the same as the SAS address received in the incoming IDENTIFY address frame) is outside the scope of this standard.

Phys that are able to become part of the same wide port shall set the DEVICE TYPE field, BREAK_REPLY CAPABLE bit, SSP INITIATOR PORT bit, STP INITIATOR PORT bit, SMP INITIATOR PORT bit, SSP TARGET PORT bit, STP TARGET PORT bit, SMP TARGET PORT bit, and SAS ADDRESS field in the IDENTIFY address frame (see 7.8.2) transmitted during the identification sequence to the same set of values on each phy in the wide port. Recipient wide ports are not required to check the consistency of these fields across their phys.

Figure 15 shows examples of narrow ports and wide ports, with a representation of the SAS address transmitted during the identification sequence. Although several phys on the left transmit SAS addresses of B, only phys attached to the same SAS addresses become part of the same ports. The set of phys with SAS address B attached to the set of phys with SAS address Y become one port, while the set of phys with SAS address B attached to the set of phys with SAS address Z become another port.

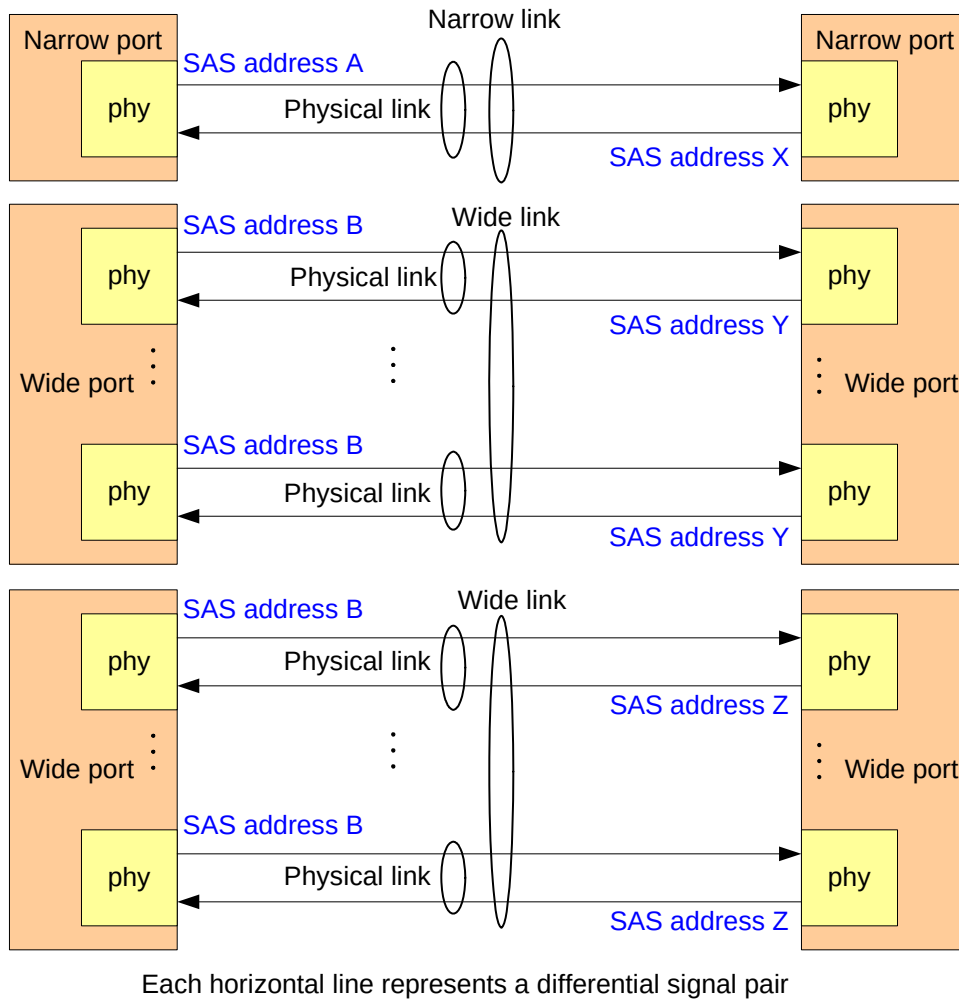


Figure 15 — Ports (narrow ports and wide ports)

In figures in this standard that show ports but not phys, the phy level of detail is not shown; however, each port always contains one or more phys.

Figure 16 defines the Port classes, showing the relationships between the following classes:

- Port;
- Expander Port;
- SAS Port;
- SAS Initiator Port;
- SAS Target Port;
- SSP Port;
- STP Port; and
- SMP Port.

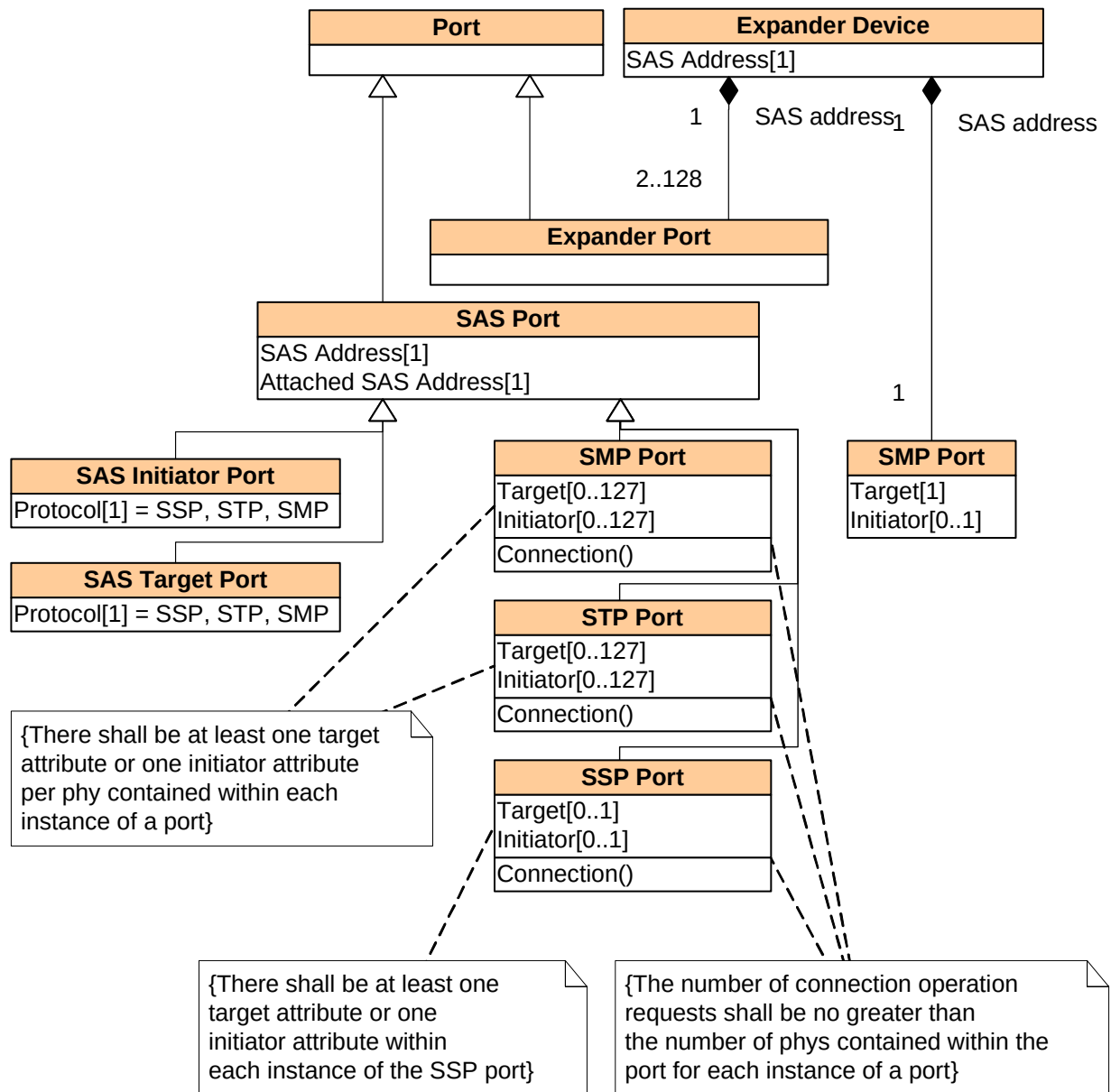


Figure 16 — Port class diagram

Figure 17 shows the objects instantiated from the Port classes:

- SAS Target Port class (i.e., SSP target port, STP target port, SMP target port);
- SAS Initiator Port class (i.e., SSP initiator port, STP initiator port, SMP initiator port);
- STP Port class (i.e., STP initiator port, STP target port, STP port);
- SMP Port class (i.e., SMP initiator port, SMP target port, SMP port);
- SSP Port class (i.e., SSP initiator port, SSP target port, SSP port);
- Expander Device SMP Port class (i.e., SMP target port, SMP port); and
- Expander Port class (i.e., expander port).

Port objects remain instantiated even when no connection is open on any of the phys within the port.

Valid objects for the SAS Initiator Port class	Valid objects for the SAS Target Port class	Valid objects for the Expander Device SMP Port class
<u>SSP initiator port : SAS Initiator Port</u> SAS Address Attached SAS Address Protocol = SSP	<u>SSP target port : SAS Target Port</u> SAS Address Attached SAS Address Protocol = SSP	<u>SMP target port : SMP port</u> Target
<u>STP initiator port : SAS Initiator Port</u> SAS Address Attached SAS Address Protocol = STP	<u>STP target port : SAS Target Port</u> SAS Address Attached SAS Address Protocol = STP	<u>SMP port : SMP Port</u> Target Initiator
<u>SMP initiator port : SAS Initiator Port</u> SAS Address Attached SAS Address Protocol = SMP	<u>SMP target port : SAS Target Port</u> SAS Address Attached SAS Address Protocol = SMP	Valid object for the Expander Port class <u>Expander port : Expander Port</u> SAS Address Attached SAS Address

Valid objects for the SSP Port class	Examples of valid objects for the STP Port class	Examples of valid objects for the SMP Port class
<u>SSP target port : SSP Port</u> SAS Address Attached SAS Address Target	<u>STP target port : STP Port</u> SAS Address Attached SAS Address Target01 Target02	<u>SMP target port : SMP Port</u> SAS Address Attached SAS Address Target
<u>SSP initiator port : SSP Port</u> SAS Address Attached SAS Address Initiator	<u>STP initiator port : STP Port</u> SAS Address Attached SAS Address Initiator	<u>SMP initiator port : SMP Port</u> SAS Address Attached SAS Address Initiator01 Initiator02
<u>SSP port : SSP Port</u> SAS address Attached SAS address Initiator Target	<u>STP port : STP Port</u> SAS Address Attached SAS Address Target01 Target02 Initiator	<u>SMP port : SMP port</u> SAS Address Attached SAS Address Target01 Target02 Initiator01 Initiator02

Figure 17 — Port object diagram

4.1.5 Application clients and device servers

This standard defines the following application clients:

- a SCSI application client (see SAM-4) is the source of SCSI commands and task management function requests. A SCSI application client uses an SSP initiator port to interface to a service delivery subsystem;

- b) an ATA application client (see ATA8-AAM) is the source of ATA commands and device management operation requests. An ATA application client uses an STP initiator port to interface to a service delivery subsystem; and
- c) a management application client is the source of management function requests. A management application client uses an SMP initiator port to interface to a service delivery subsystem.

This standard defines the following device servers:

- a) a SCSI device server (see SAM-4) processes SCSI commands. A SCSI device server uses an SSP target port to interface to a service delivery subsystem;
- b) an ATA device server (see ATA8-AAM) processes ATA commands. An device server uses an STP target port to interface to a service delivery subsystem; and
- c) a management device server processes management functions. A management device server uses an SMP target port to interface to a service delivery subsystem;

4.1.6 SAS devices

A SAS device contains one or more SAS ports, each containing one or more phys (i.e., a SAS port may be a narrow port or a wide port).

Figure 18 shows examples of SAS devices with different port and phy configurations.

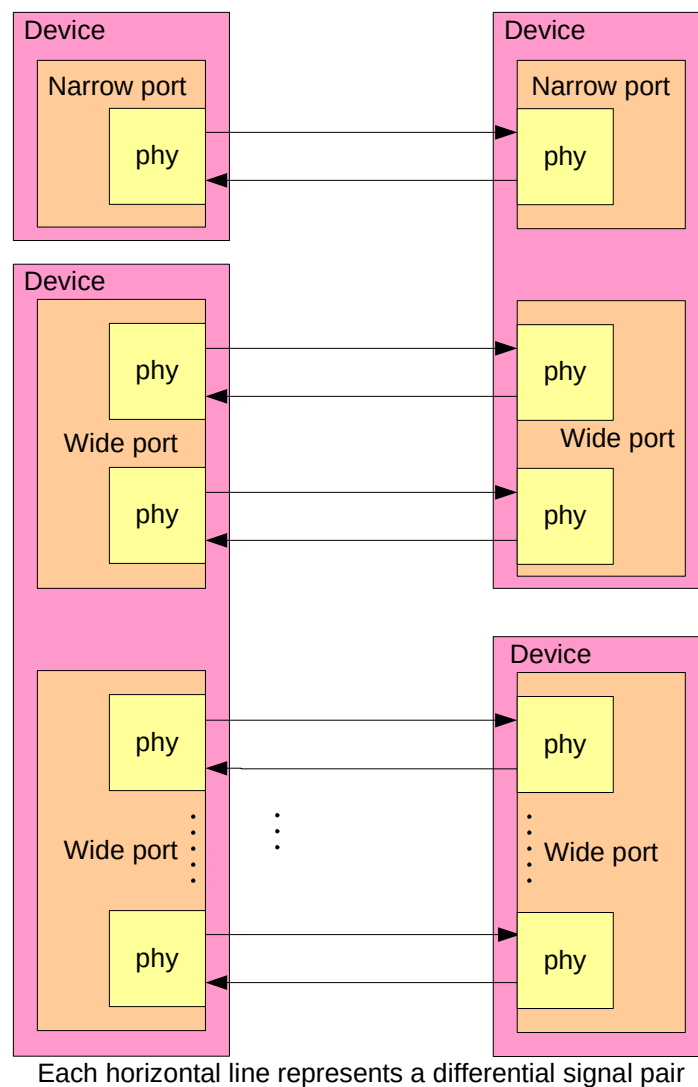


Figure 18 — SAS devices

An end device is a SAS device that is not contained in an expander device (see 4.1.7).

4.1.7 Expander devices

Expander devices are part of a service delivery subsystem and facilitate communication between multiple SAS devices. Expander devices contain two or more external expander ports. Each expander device contains one SMP target port and one management device server, contains one SMP initiator port and one management application client if it is self-configuring and may contain one SMP initiator port and one management application client if it is not self-configuring, and may contain SAS devices (e.g., an expander device may include an SSP target port for access to a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-4 and SES-2)).

Figure 19 shows an expander device.

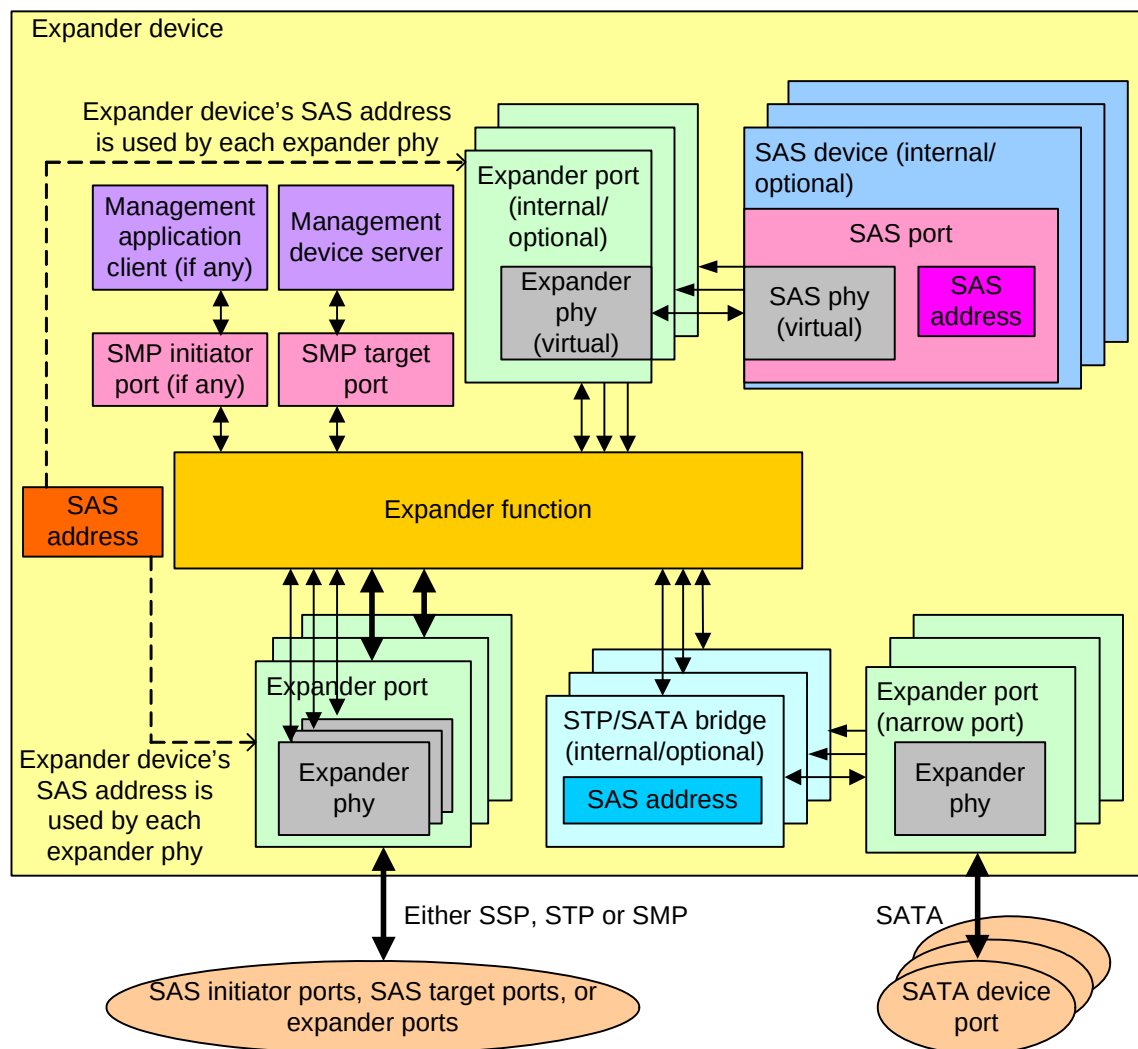


Figure 19 — Expander device

See 4.6 for a detailed model of an expander device.

Figure 20 defines the Expander Device class.

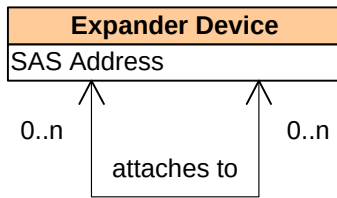


Figure 20 — Expander Device class diagram

Each expander phy has one of the following routing attributes (see 4.6.7.1):

- a) direct routing attribute;
- b) table routing attribute; or
- c) subtractive routing attribute.

Expander devices with expander phys with the table routing attribute contain an expander route table (see 4.6.7.3). An externally configurable expander device depends on a management application client within the SAS domain to use the discover process (see 4.7) and the configuration subprocess (see 4.8) to configure the expander route table. A self-configuring expander device contains a management application client and an SMP initiator port to perform the discover process (see 4.7) to configure its own expander route table.

4.1.8 Service delivery subsystem

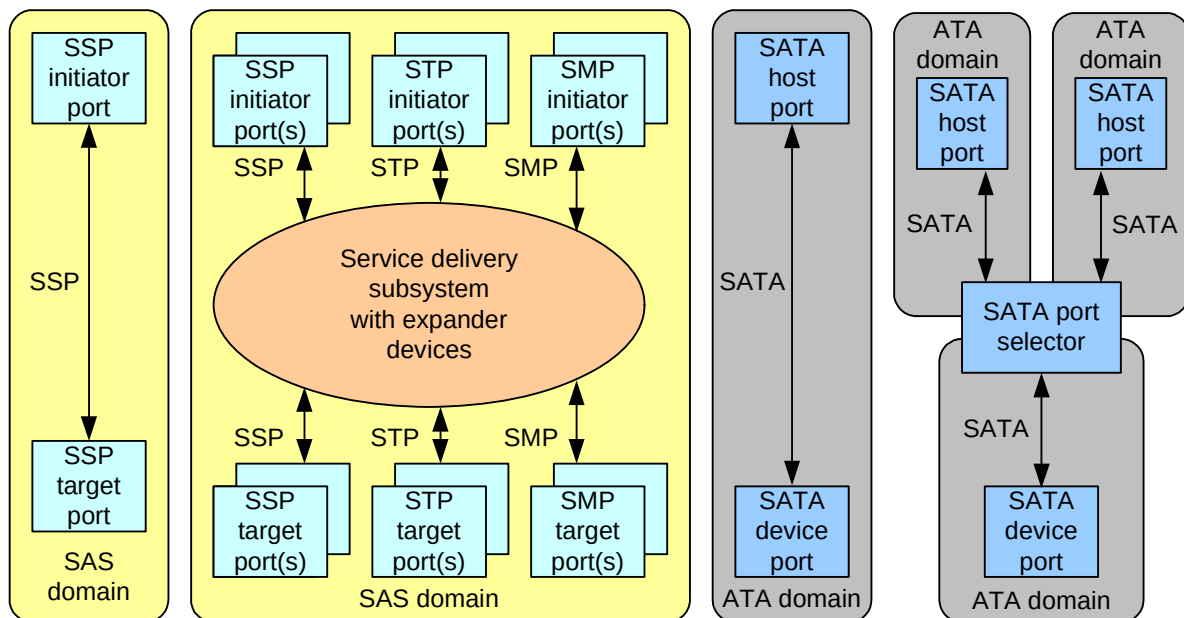
A service delivery subsystem is either:

- a) a set of physical links between a SAS initiator port and a SAS target port; or
- b) a set of physical links and expander devices, supporting more than two SAS ports.

See 4.1.10 for rules on constructing service delivery subsystems from multiple expander devices.

4.1.9 Domains

Figure 21 shows examples of SAS domains and ATA domains.



Note: When expander devices are present, SAS target ports may be located in SAS devices contained in expander devices.

Figure 21 — Domains

Figure 22 shows a SAS domain bridging to one or more ATA domains.

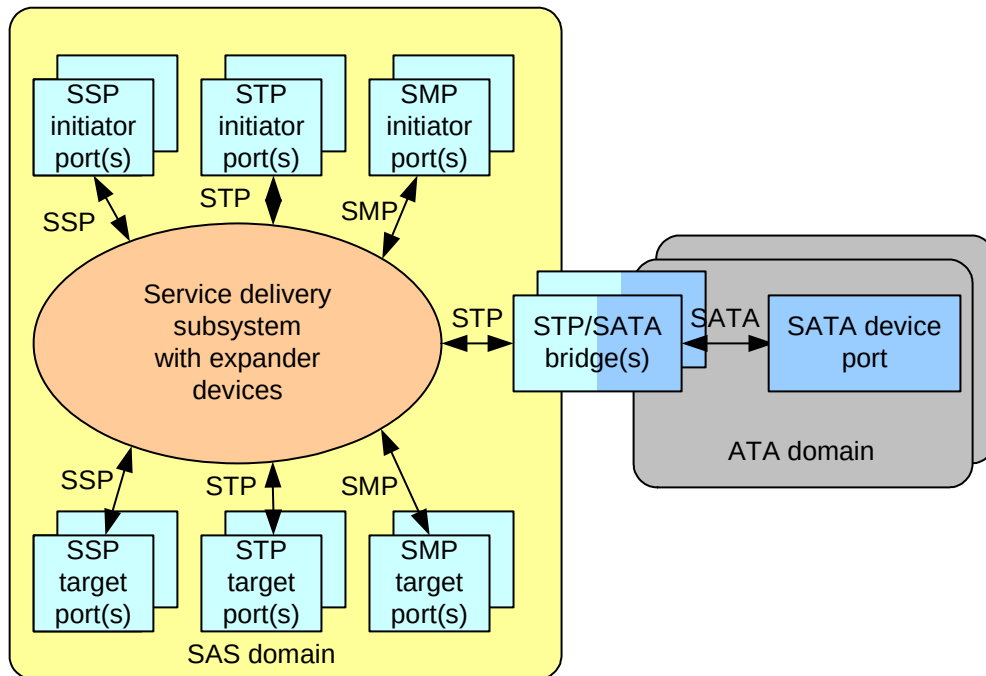


Figure 22 — SAS domain bridging to ATA domains

Figure 23 shows two SAS domains bridging to one or more ATA domains containing SATA devices with SATA port selectors.

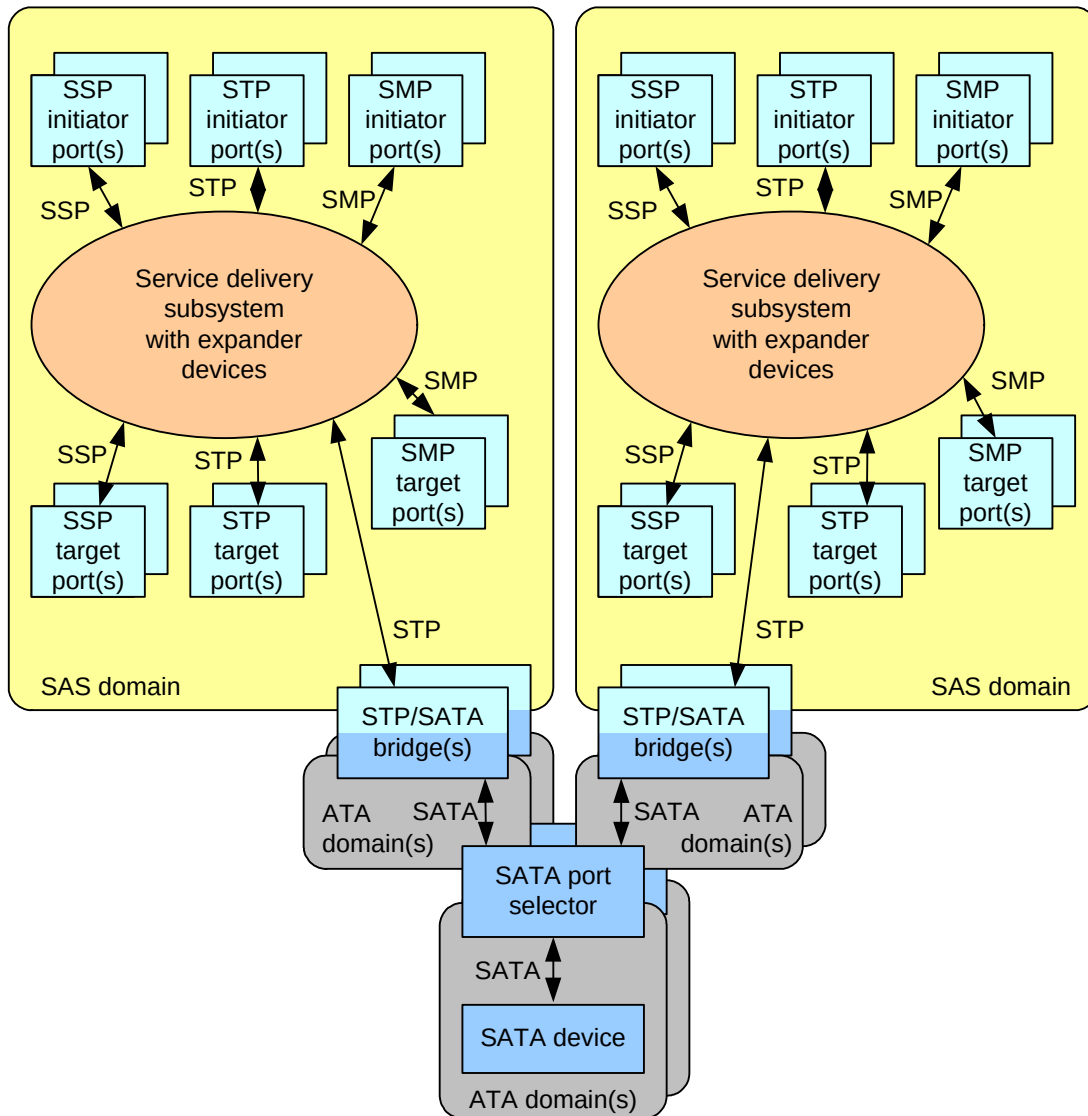


Figure 23 — SAS domains bridging to ATA domains with SATA port selectors

Figure 24 shows SAS initiator devices and SAS target devices with SAS ports in the same SAS domains and in different SAS domains. When a SAS device has ports in different SAS domains, the ports may have the same SAS address (see 4.2); when its ports are in the same SAS domain, they shall have different SAS addresses.

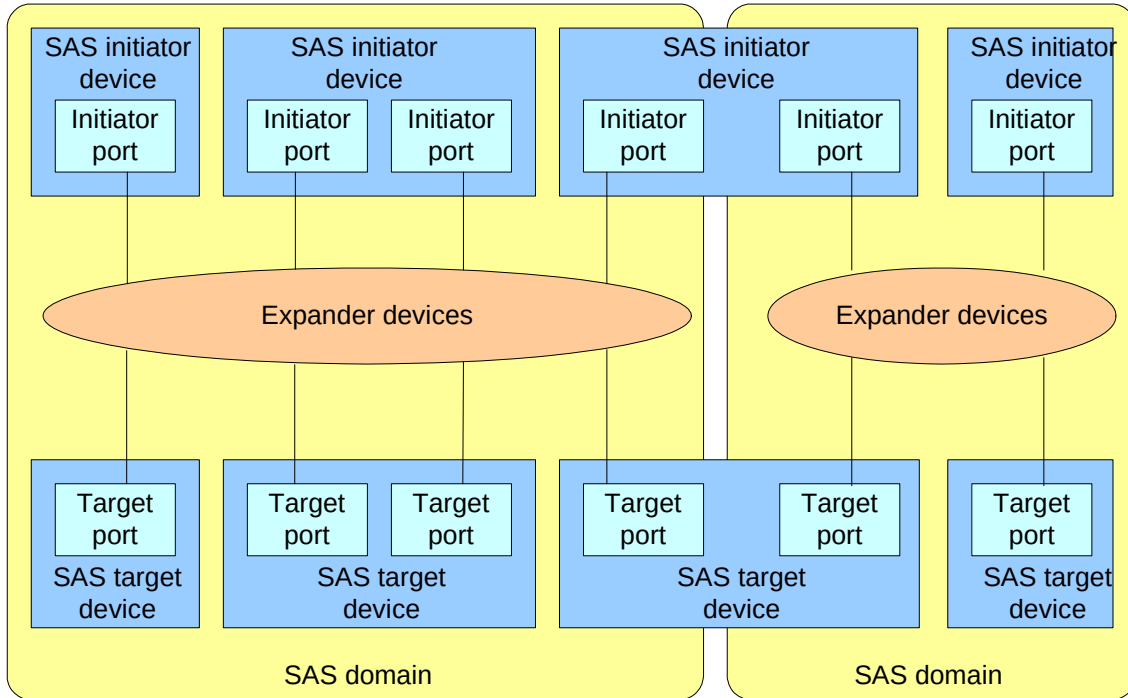


Figure 24 — Devices spanning SAS domains

4.1.10 Expander device topologies

4.1.10.1 Expander device topology overview

More than one expander device may be part of a service delivery subsystem.

To avoid an overflow of an expander route index during the configuration subprocess (see 4.8), a SAS domain containing an externally configurable expander device shall be constructed such that the number of expander route indexes available for each expander phy with the table routing attribute is greater than or equal to the number of SAS addresses addressable through that expander phy.

4.1.10.2 Expander device topologies

Figure 25 shows an example of an expander topology with one expander device.

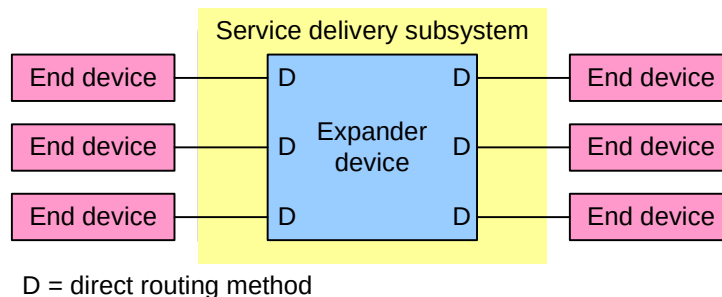


Figure 25 — Single expander device topology example

Figure 26 shows examples of expander topologies with multiple expander devices.

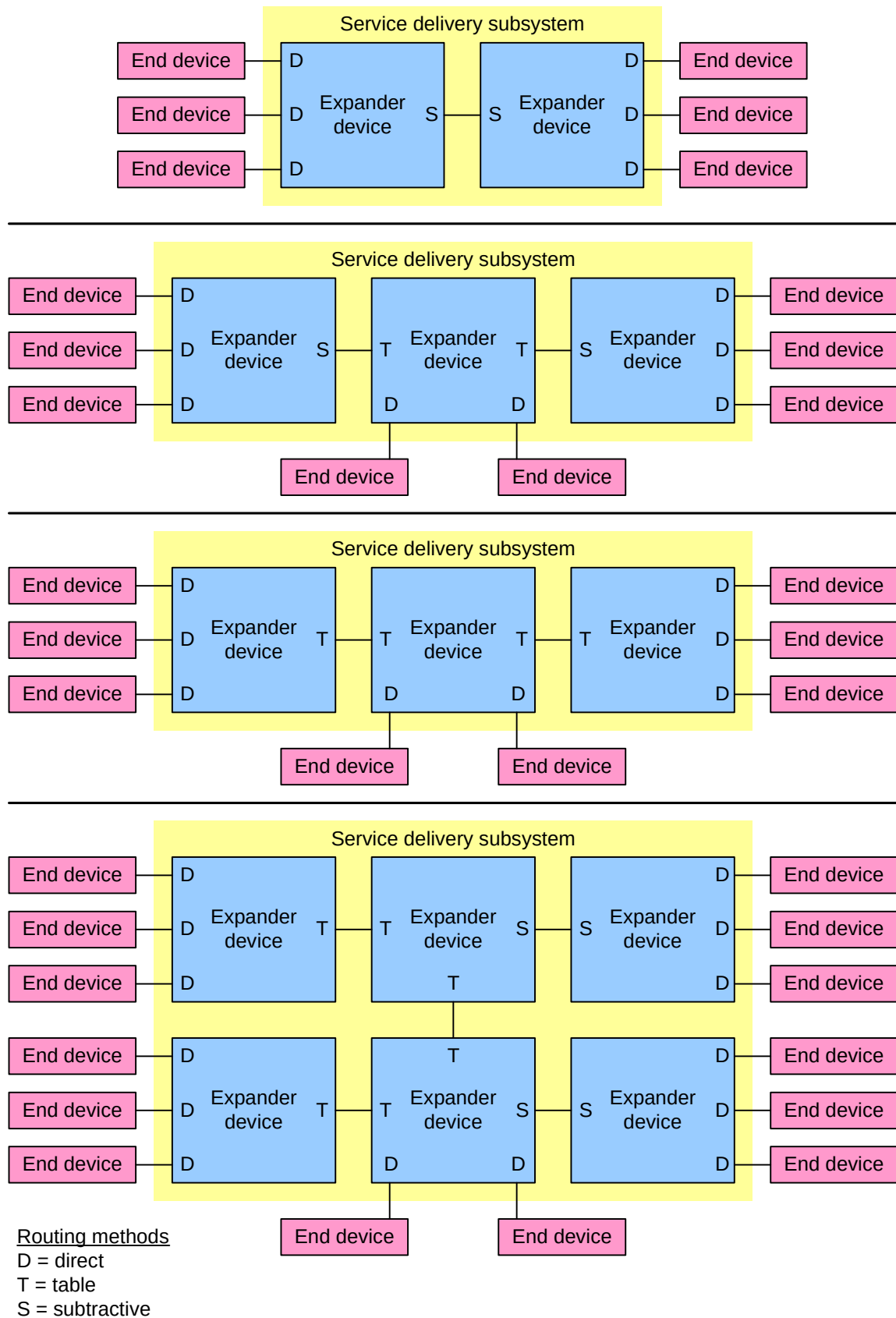


Figure 26 — Multiple expander device topologies

4.1.11 Pathways

A potential pathway is a set of logical links between a SAS initiator phy and a SAS target phy. When a SAS initiator phy is directly attached to a SAS target phy with a non-multiplexed physical link, there is one potential pathway. When the physical link is multiplexed or there are expander devices between a SAS initiator phy and a SAS target phy, it is possible that there is more than one potential pathway, each consisting of a set of logical links between the SAS initiator phy and the SAS target phy. The physical links may or may not be using the same physical link rate.

A pathway is a set of logical links between a SAS initiator phy and a SAS target phy being used by a connection (see 4.1.12).

Figure 27 shows examples of potential pathways.

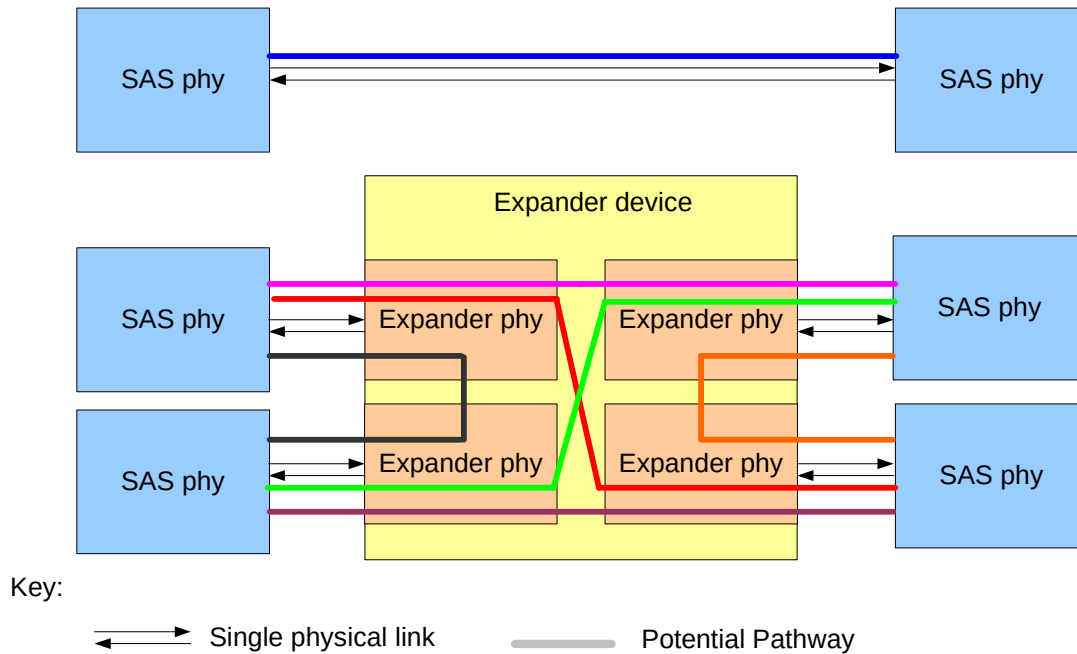


Figure 27 — Potential pathways

A partial pathway is the set of logical links participating in a connection request that have not yet conveyed a connection response (see 7.12).

A partial pathway is blocked when path resources it requires are held by another partial pathway (see 7.12).

4.1.12 Connections

A connection is a temporary association between a SAS initiator phy and a SAS target phy. During a connection all dwords from the SAS initiator phy are forwarded to the SAS target phy, and all dwords from the SAS target phy are forwarded to the SAS initiator phy.

A connection is pending when an OPEN address frame has been delivered along a completed pathway to the destination phy but the destination phy has not yet responded to the connection request. A connection is established when an OPEN_ACCEPT is received by the source phy.

A connection enables communication for one protocol: SSP, STP, or SMP. For SSP and STP, connections may be opened and closed multiple times during the processing of a command (see 7.12).

The connection rate is the effective rate of dwords through the pathway between a SAS initiator phy and a SAS target phy, established through the connection request. Every logical phy shall support a 1,5 Gbps connection rate regardless of its logical link rate.

No more than one connection is active on a logical link at a time. If the connection is an SSP or SMP connection and there are no dwords to transmit associated with that connection, idle dwords are transmitted.

If the connection is an STP connection and there are no dwords to transmit associated with that connection, then SATA_SYNCs, SATA_CONTs, or vendor-specific scrambled data dwords are transmitted as defined in SATA-2. If there is no connection on a logical link then idle dwords are transmitted.

The number of connections established by a SAS port shall not exceed the number of SAS logical phys within the SAS port (i.e., only one connection per SAS logical phy is allowed). There shall be a separate connection on each logical link.

If multiple potential pathways exist between the SAS initiator port(s) and the SAS target port(s), multiple connections may be established by a SAS port between the following:

- a) one SAS initiator port to multiple SAS target ports;
- b) one SAS target port to multiple SAS initiator ports; or
- c) one SAS initiator port to one SAS target port.

Once a connection is established, the pathway used for that connection shall not be changed (i.e., all the logical links that make up the pathway remain dedicated to the connection until it is closed).

Figure 28 shows examples of connections between wide and narrow ports. All the connections shown may occur simultaneously. Additionally:

- a) the connections labeled A and B are an example of one SAS initiator port with connections to multiple SAS target ports;
- b) the connections labeled A and C are an example of one SAS target port with connections to multiple SAS initiator ports;
- c) the connections labeled E and F are an example of multiple connections between one SAS initiator port and one SAS target port; and
- d) the connections labeled C, D, E, and F are an example of one SAS initiator port with connections to multiple SAS target ports with one of those SAS target ports having multiple connections with that SAS initiator port.

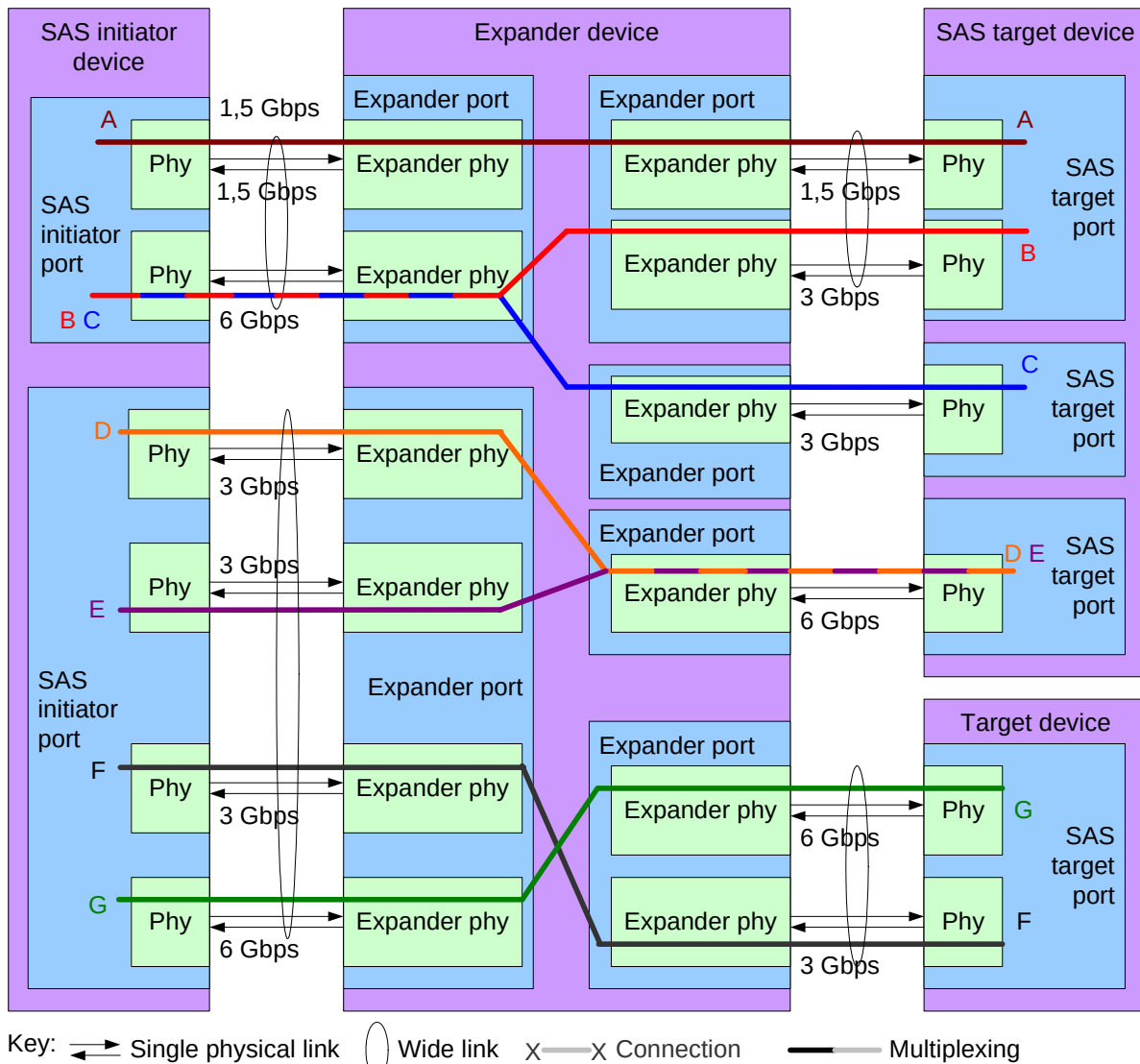


Figure 28 — Multiple connections on wide ports

4.1.13 Broadcasts

Broadcasts are used to notify all phys in the SAS domain about certain events. Broadcasts are transmitted using BROADCAST (see 7.2.5.5) and/or the SMP ZONED BROADCAST function (see 10.4.3.19).

Table 7 defines the Broadcast types.

Table 7 — Broadcast types (part 1 of 2)

Broadcast	Primitive ^a	Description
Broadcast (Change)	yes	Originated by an expander device to notify SAS initiator ports that a SAS domain change has occurred (see 7.11). May also be originated by SAS initiator ports. Ignored by SAS target ports.
Broadcast (Reserved Change 0)	yes	Reserved. SAS ports (i.e., SAS initiator ports and SAS target ports) shall process this Broadcast the same as Broadcast (Change).
Broadcast (Reserved Change 1)	yes	Reserved. SAS ports shall process this Broadcast the same as Broadcast (Change).
Broadcast (SES)	yes	Originated by a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-4 and SES-2) accessible through a SAS target port in the SAS domain to notify SAS initiator ports of an asynchronous event. SSP initiator ports should poll all the logical units in the SAS domain with peripheral device types set to 0Dh to determine the source. SAS target ports shall ignore this Broadcast.
Broadcast (Expander)	yes	Originated by an expander device to notify SAS initiator ports that an expander event has occurred, including: a) expander device is temporarily going to have reduced functionality for a period of time (see 4.6.8); b) a phy event peak value detector has reached its threshold value; or c) a phy event peak value detector has been cleared by an SMP CONFIGURE PHY EVENT function (see 10.4.3.29). Expander events do not include SAS domain changes, which are communicated with Broadcast (Change).
Broadcast (Asynchronous Event)	yes	Originated by an SSP target port when an event occurs that causes one or more unit attention conditions to be established for one or more logical units accessible through the SSP target port. An SSP target port shall only originate one Broadcast (Asynchronous Event) for each event that affects multiple logical units accessible through the SSP target port (e.g., only one Broadcast (Asynchronous Event) is originated when a hard reset occurs).
^a All Broadcasts are supported by the SMP ZONED BROADCAST function (see 10.4.3.19), which defines additional reserved Broadcast types. Broadcasts labeled “yes” are also transmitted via BROADCAST primitive sequences (see 7.2.5.5).		

Table 7 — Broadcast types (part 2 of 2)

Broadcast	Primitive ^a	Description
Broadcast (Reserved 3)	yes	Reserved. SAS ports shall ignore this Broadcast.
Broadcast (Reserved 4)	yes	Reserved. SAS ports shall ignore this Broadcast.
Broadcast (Zone Activate)	no	Initiates the zone activate step (see 4.9.6.4). Devices that are not locked zoning expander devices shall ignore this Broadcast.
^a All Broadcasts are supported by the SMP ZONED BROADCAST function (see 10.4.3.19), which defines additional reserved Broadcast types. Broadcasts labeled “yes” are also transmitted via BROADCAST primitive sequences (see 7.2.5.5).		

When an expander port receives a Broadcast, the BPP (see 4.6.5) shall forward the Broadcast on at least one phy in each other expander port if zoning is disabled, or forward the Broadcast as described in 4.9.5 if zoning is enabled.

An expander device is not required to queue multiple identical Broadcasts for the same expander port. If a second identical Broadcast is requested before the first Broadcast has been transmitted, the second Broadcast may be ignored.

A SAS device or an expander device is not required to maintain originated Broadcast count information in non-volatile storage or across reset events.

A SAS device or expander device may implement counters for Broadcasts it originates and report them in the REPORT BROADCAST response (see 10.4.3.8). If supported, the SAS device or expander device shall, for each combination of Broadcast type and Broadcast reason that the SAS device or expander device supports:

- a) if the Broadcast is related to a phy, then maintain a separate broadcast counter for each phy; and
- b) if the Broadcast is not related to a phy, then maintain one originated broadcast counter.

Broadcast (Change)s originated by an expander device are counted and reported in the REPORT GENERAL response (see 10.4.3.3) and other SMP response frames containing an EXPANDER CHANGE COUNT field.

An expander device may implement counters for Broadcasts received from attached end devices and report them in the REPORT BROADCAST response (see 10.4.3.8).

See 4.11 for details on phy events.

4.2 Names and identifiers

4.2.1 Names and identifiers overview

Device names are worldwide unique names for devices within a transport protocol. Port names are worldwide unique names for ports within a transport protocol. Port identifiers are the values by which ports are identified within a domain. Phy identifiers are the values by which phys are identified within a device.

Table 8 describes the definitions of names and identifiers for SAS.

Table 8 — Names and identifiers

Attribute	Format	SAS usage	Reference
Device name	NAA IEEE Registered format (see 4.2.2) for: a) SAS devices and expander devices; and b) SATA devices with worldwide names	Reported in: a) the IDENTIFY address frame (see 7.8.2) DEVICE NAME field; b) the Device Identification VPD page (see 10.2.11.2); and c) the DISCOVER response (see 10.4.3.9) ATTACHED DEVICE NAME field	4.2.4 and 4.2.5
Port name	Not defined		4.2.6
Port identifier	NAA IEEE Registered format (see 4.2.2)	Reported in the IDENTIFY address frame (see 7.8.2) for SAS ports.	4.2.7
Phy identifier	7-bit value	Phy identifier	4.2.8

Table 9 describes how various SAM-4 attributes are implemented in SSP.

Table 9 — SAM-4 attribute mapping

SAM-4 attribute	SSP implementation
Initiator port identifier	SAS address of an SSP initiator port
Initiator port name	Not defined
Target port identifier	SAS address of an SSP target port
Target port name	Not defined
SCSI device name	Device name of SAS device containing an SSP port

4.2.2 NAA IEEE Registered format identifier

Table 10 defines the NAA IEEE Registered format identifier used by device names and port identifiers. This format is the same as that defined in SPC-4.

Table 10 — NAA IEEE Registered format

Byte\Bit	7	6	5	4	3	2	1	0
0	NAA (5h)				(MSB)			
1	IEEE COMPANY ID							
2								
3								
4	VENDOR-SPECIFIC IDENTIFIER							
5								
6								
7								

The NAA field contains 5h.

The IEEE COMPANY ID field contains a 24-bit canonical form company identifier (i.e., organizationally unique identifier or OUI) assigned by the IEEE.

Bit 5 of byte 1, which serves as the UNIVERSALLY/LOCALLY ADMINISTERED ADDRESS bit, shall be set to zero.

Bit 4 of byte 1, which serves as the INDIVIDUAL/GROUP ADDRESS bit, shall be set to zero.

NOTE 7 - Information about IEEE company identifiers may be obtained from the IEEE Registration Authority web site at <http://standards.ieee.org/regauth/oui>.

The VENDOR-SPECIFIC IDENTIFIER field contains a 36-bit value that is assigned by the organization associated with the company identifier in the IEEE COMPANY ID field. The VENDOR-SPECIFIC IDENTIFIER field shall be assigned so the NAA IEEE Registered format identifier is worldwide unique.

An identifier value of 00000000_00000000h indicates an invalid identifier.

4.2.3 Hashed SAS addresses

SSP frames include hashed versions of SAS addresses of SAS ports (i.e., NAA IEEE Registered format identifiers for SAS ports) to provide an additional level of verification of proper frame routing.

The code used for the hashing algorithm is a cyclic binary Bose, Chaudhuri, and Hocquenghem (BCH) (63, 39, 9) code. Table 11 lists the parameters for the code.

Table 11 — Hashed SAS address code parameter

Parameter	Value
Number of bits per codeword	63
Number of data bits	39
Number of redundant bits	24
Minimum distance of the code	9

The generator polynomial for this code is:

$$G(x) = (x^6 + x + 1) (x^6 + x^4 + x^2 + x + 1) (x^6 + x^5 + x^2 + x + 1) (x^6 + x^3 + 1)$$

After multiplication of the factors, the generator polynomial is:

$$G(x) = x^{24} + x^{23} + x^{22} + x^{20} + x^{19} + x^{17} + x^{16} + x^{13} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^2 + x + 1$$

Annex E contains additional information on SAS address hashing.

4.2.4 Device names and expander device SAS addresses

Each expander device, SAS initiator device, SAS target device, and SAS target/initiator device shall include an NAA IEEE Registered format identifier (see 4.2.2) as its device name. The device name of an expander device is called its SAS address. An NAA IEEE Registered format identifier used as a device name shall not be used as any other name or identifier (e.g., a device name, port name, port identifier, or logical unit name (see SAM-4)).

SAS devices and expander devices report their device names in the IDENTIFY address frame (see 7.8.2).

NOTE 8 - When a set of expander phys transmit the same SAS address in the identification sequence but receive different SAS addresses, indicating they are attached to separate SAS ports or expander ports, they become part of separate expander ports in the same domain.

Logical units accessed through SSP target ports report SAS target device names through SCSI vital product data (see 10.2.11).

4.2.5 Device name for SATA devices with world wide names

Table 12 defines the NAA IEEE Registered format identifier used by device names for SATA devices that provide world wide names in their IDENTIFY (PACKET) DEVICE data.

Table 12 — Device name created from the IDENTIFY (PACKET) DEVICE world wide name

Subformat field name (see table 10)	Specific bits in	Contents ^a
NAA	Byte 0 bits 7:4	IDENTIFY (PACKET) DEVICE data word 108 bits 15:12 ^b
IEEE COMPANY_ID	Byte 0 bits 3:0	IDENTIFY (PACKET) DEVICE data word 108 bits 11:8
	Byte 1	IDENTIFY (PACKET) DEVICE data word 108 bits 7:0
	Byte 2	IDENTIFY (PACKET) DEVICE data word 109 bits 15:8
	Byte 3 bits 7:4	IDENTIFY (PACKET) DEVICE data word 109 bits 7:4
VENDOR SPECIFIC IDENTIFIER	Byte 3 bits 3:0	IDENTIFY (PACKET) DEVICE data word 109 bits 3:0
	Byte 4	IDENTIFY (PACKET) DEVICE data word 110 bits 15:8
	Byte 5	IDENTIFY (PACKET) DEVICE data word 110 bits 7:0
	Byte 6	IDENTIFY (PACKET) DEVICE data word 111 bits 15:8
	Byte 7	IDENTIFY (PACKET) DEVICE data word 111 bits 7:0
^a IDENTIFY (PACKET) DEVICE data words 108-111 contain the world wide name field (see ATA8-ACS).		
^b This 4-bit field is required to be set to 5h (i.e., IEEE Registered) by ATA8-ACS.		

4.2.6 Port names

Port names are not defined in SAS.

NOTE 9 - The SAS addresses used by SAS ports in different SAS domains may be the same (e.g., when a set of phys transmit the same SAS address in the identification sequence but receive different SAS addresses, indicating they are attached to two separate SAS domains) so the SAS address serves as a port identifier (see 4.2.7) rather than a port name.

4.2.7 Port identifiers and SAS port SAS addresses

Each SAS initiator port, SAS target port (e.g., including the STP target port in each STP/SATA bridge), and SAS target/initiator port shall include an NAA IEEE Registered format identifier (see 4.2.2) as its port identifier. The port identifier of a SAS port is called its SAS address. An NAA IEEE Registered format identifier used as a port identifier shall not be used as any other name or identifier (e.g., a device name, port name, or logical unit name (see SAM-4)) except as a port identifier in one or more other SAS domains (see 4.1.4).

Expander ports do not have port identifiers.

SAS ports in end devices report their port identifiers in the IDENTIFY address frame (see 7.8.2). Expander devices containing SAS ports (e.g., SAS ports attached to virtual phys, or STP target ports in STP/SATA bridges) report the port identifiers of those SAS ports in the SMP DISCOVER response (see 10.4.3.9).

NOTE 10 - When a set of SAS phys transmit the same SAS address in the identification sequence but receive different SAS addresses, indicating they are attached to more than one SAS domain, they become part of separate SAS ports in separate domains. Each SAS port shares the same SAS address.

Port identifiers are used as source and destination SAS addresses in the OPEN address frame (see 7.8.3).

Logical units accessed through SSP target ports report SAS target port identifiers through SCSI vital product data (see 10.2.11).

4.2.8 Phy identifiers

Each SAS phy and expander phy shall be assigned an identifier called a phy identifier that is unique within the SAS device and/or expander device. Each SAS logical phy within a SAS phy shall use the same phy identifier. Each expander logical phy within an expander phy shall use the same phy identifier. The phy identifier is used for management functions (see 10.4).

Phy identifiers shall be greater than or equal to 00h and less than 80h, and should be numbered starting with 00h. In an expander device or in a SAS device containing an SMP target port, phy identifiers shall be less than the value of the NUMBER OF PHYS field in the SMP REPORT GENERAL response (see 10.4.3.3). In a SAS device containing an SSP target port, phy identifiers shall be less than the value of the NUMBER OF PHYS field in the Phy Control And Discover mode page (see 10.2.7.5).

4.3 State machines

4.3.1 State machine overview

Figure 29 shows the state machines for SAS devices, their relationships to each other and to the SAS Device, SAS Port, and SAS Phy classes.

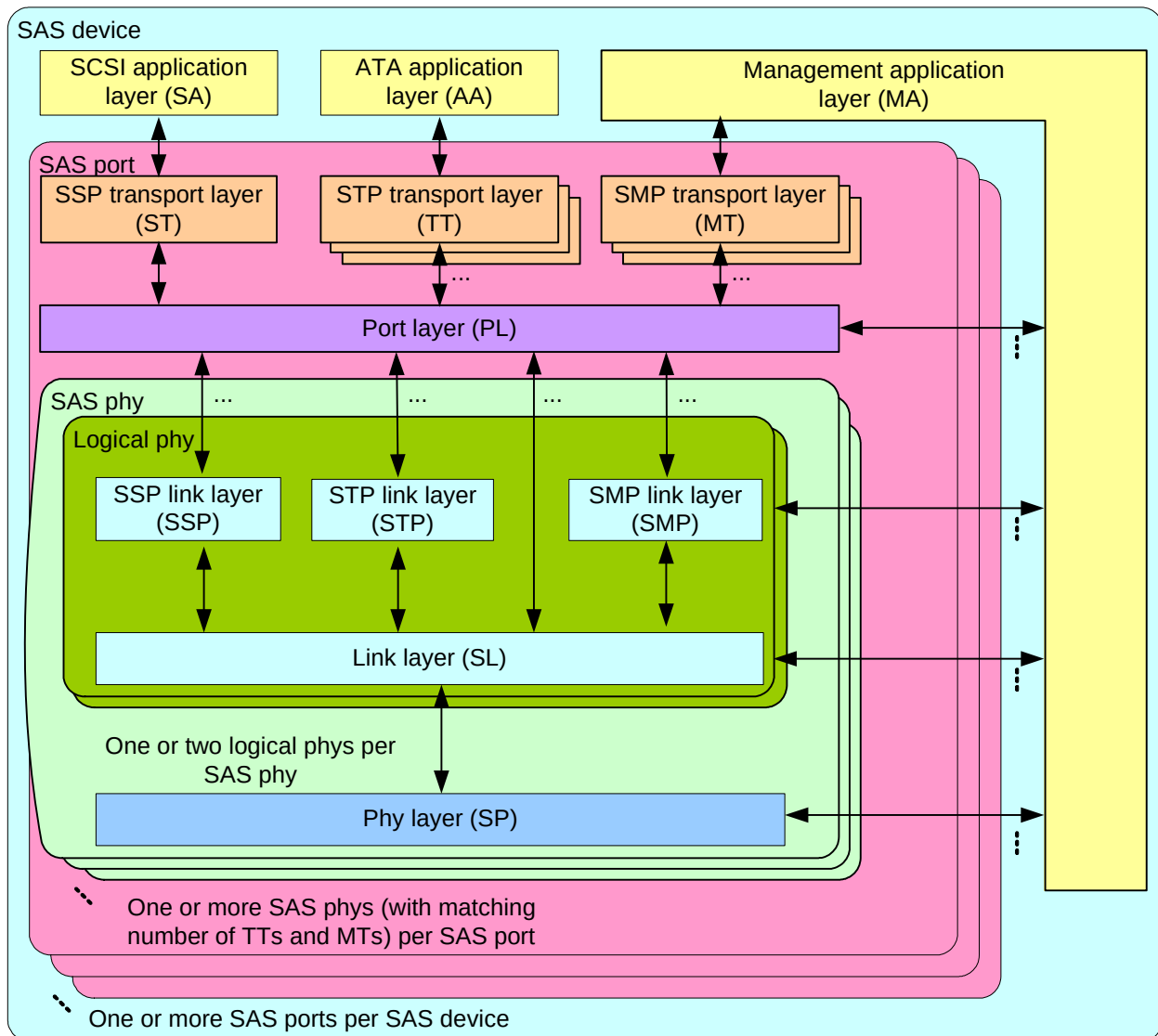
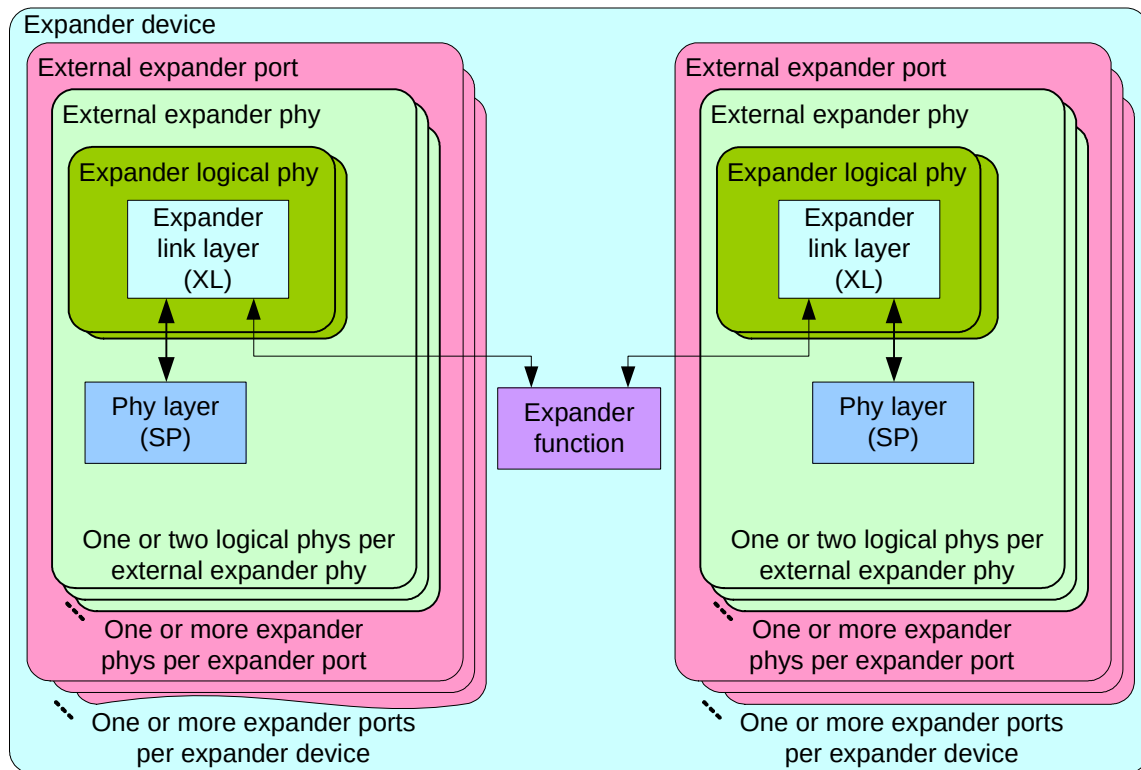


Figure 29 — State machines for SAS devices

Figure 30 shows the state machines for expander devices, their relationships to each other and to the Expander Device, Expander Port, and Expander Phy classes. Expander function state machines are not defined in this standard, but the interface to the expander function is defined in 4.6.6.



NOTE: The expander link layer includes the SL_IR state machine.

Figure 30 — State machines for expander devices

4.3.2 Transmit data path

Figure 31 shows the transmit data path in a SAS phy, showing the relationship between:

- the SP state machine (see 6.8) and the SP transmitter (see 6.8.2);
- multiplexing (see 6.10);
- the SL_IR state machines (see 7.9.4) and the SL_IR transmitter (see 7.9.4.2);
- physical link rate tolerance management (see 7.3);
- the SL state machines (see 7.14) and the SL transmitter (see 7.14.2);
- rate matching (see 7.13); and
- the SSP transmit data path (see figure 32), SMP transmit data path (see figure 33), and STP transmit data path (see figure 34).



Editor's Note 7: Discussion of 07-090 indicates that more specific MUX controls (e.g. state machine signals) would increase understanding, as was done for Link reset sequence complete.

Figure 32 shows the transmit data path for the SSP link layer, including the SSP state machines and the SSP

transmitter (see 7.16.8 and 7.16.8.2), and the communication to the port layer, SSP transport layer, and SCSI application layer. Only the SSP link layer (i.e., not the port, transport, or application layer) transmits dwords.

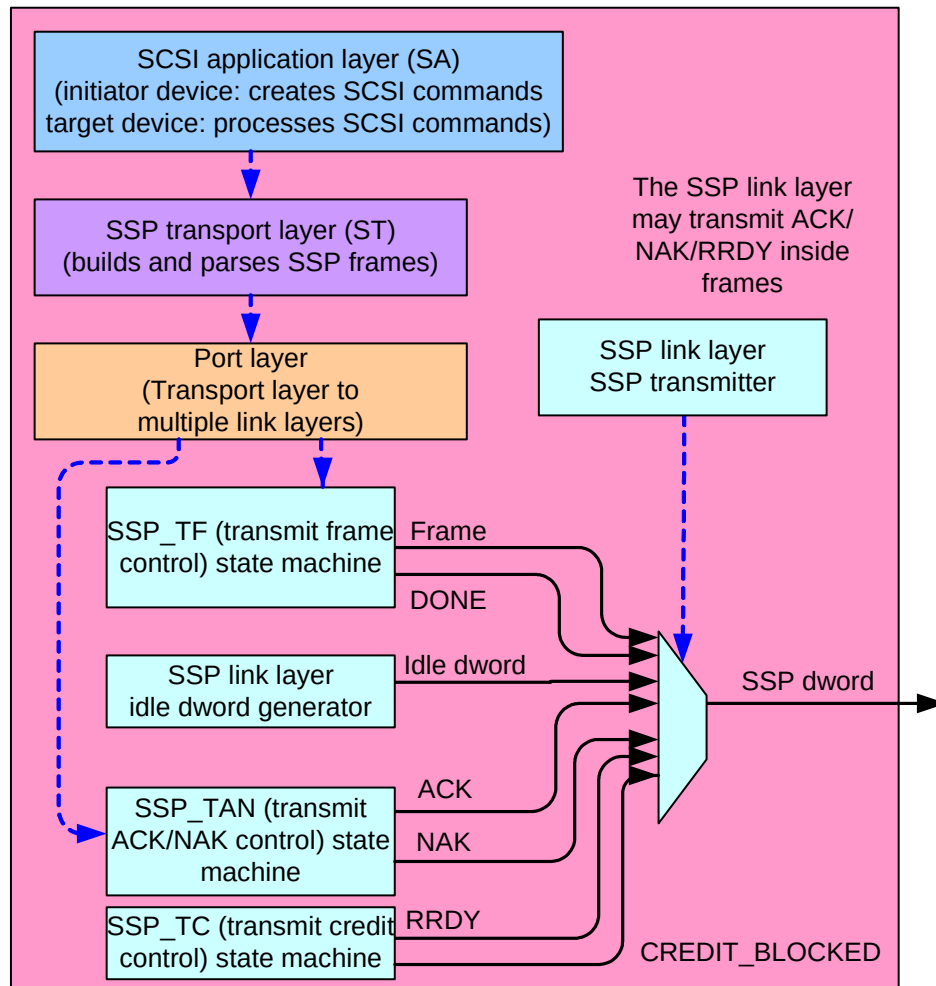


Figure 32 — SSP link, port, SSP transport, and SCSI application layer state machines

Figure 33 shows the transmit data path for the SMP link layer, including the SMP state machines and the SMP

transmitter (see 7.18.5 and 7.18.5.2), and the communication to the port layer, SMP transport layer, and management application layer. Only the SMP link layer (i.e., not the port, transport, or application layer) transmits dwords.

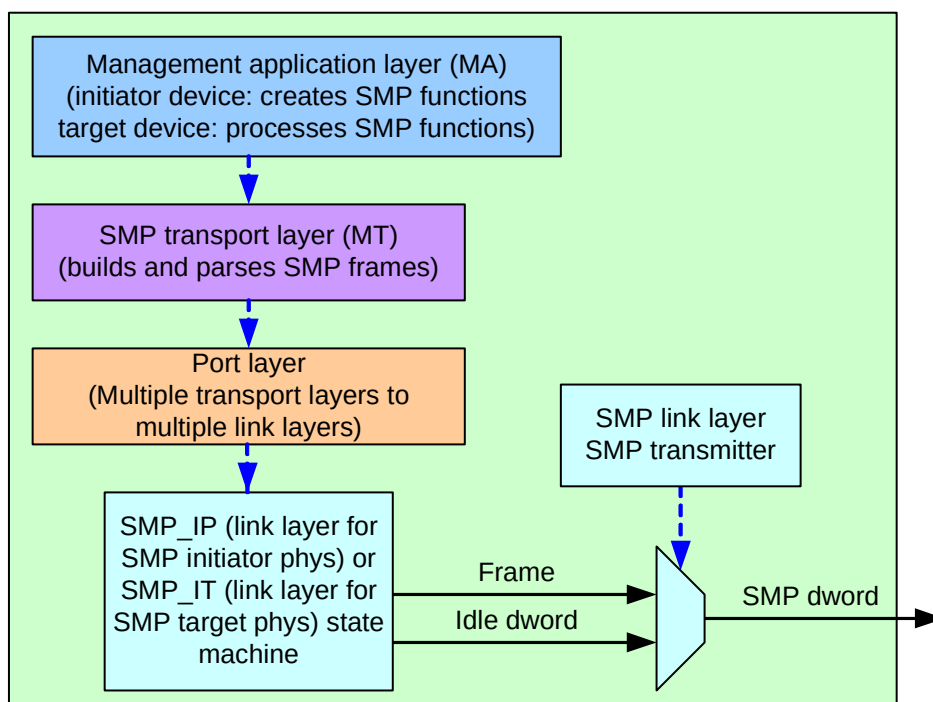


Figure 33 — SMP link, port, SMP transport, and management application layer state machines

Figure 34 shows the transmit data path for the STP link layer, including the SSP state machines and the SSP

transmitter (see 7.17.8), and communication to the port layer, STP transport layer, and ATA application layer. Only the STP link layer (i.e., not the port, transport, or application layer) transmits dwords.

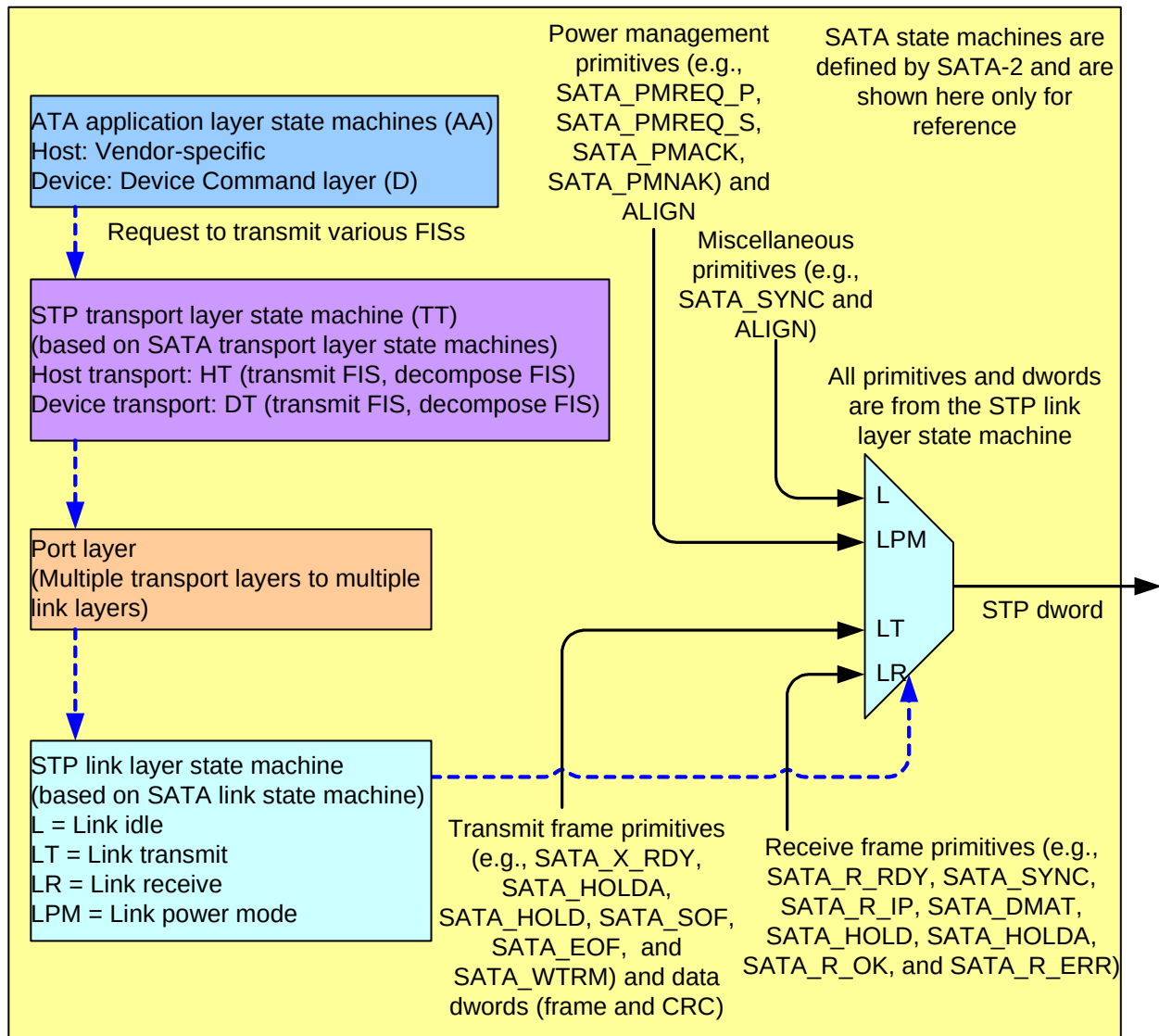


Figure 34 — STP link, port, STP transport, and ATA application layer state machines

Figure 35 shows the transmit data path in an expander phy, showing the relationship between:

- the SP state machine (see 6.8) and the SP transmitter (see 6.8.2);
- multiplexing (see 6.10);
- the SL_IR state machines (see 7.9.4) and the SL_IR transmitter (see 7.9.4.2);
- physical link rate tolerance management (see 7.3); and

e) the XL state machines (see 7.15) and the XL transmitter (see 7.15.2).

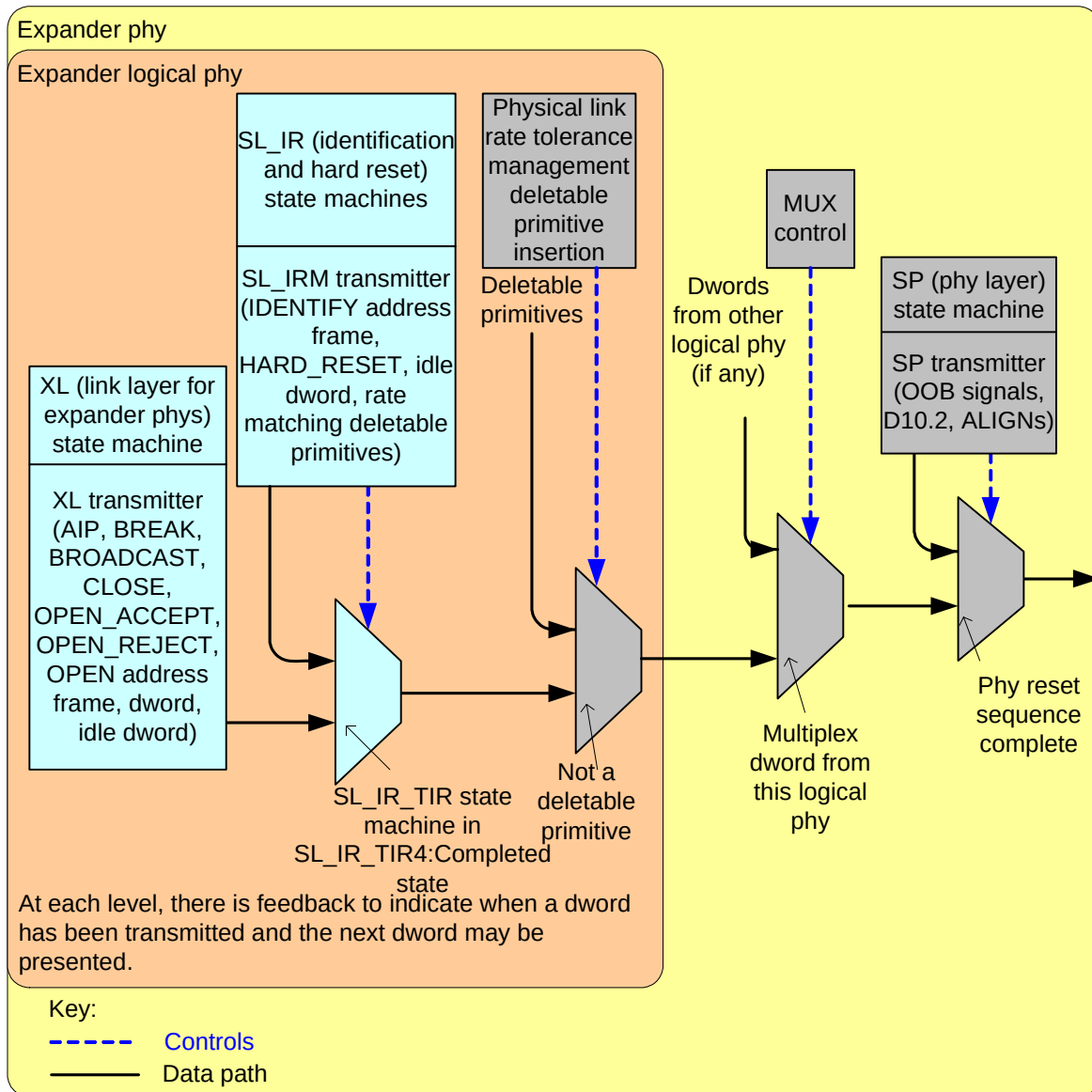


Figure 35 — Transmit data path and state machines in an expander phy

Editor's Note 8: Discussion of 07-090 indicates that more specific MUX controls (e.g. state machine signals) would increase understanding.

4.3.3 Receive data path

The SP_DWS receiver (see 6.9.2) establishes dword synchronization and sends dwords to the SP_DWS state machine (see 6.9) and to the link layer state machine receivers.

If multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), the SP_DWS receiver uses incoming MUX primitives to determine the logical link numbers and route the dwords to the appropriate link layer receivers.

Figure 36 shows the receive data path in a SAS phy, showing the relationship between:

- the SP_DWS state machine (see 6.9) and the SP_DWS receiver (see 6.9.2);
- the SL_IR state machines (see 7.9.4) and the SL_IR receiver (see 7.9.4.2);
- the SL state machines (see 7.14) and the SL receiver (see 7.14.2); and

- d) the SSP receiver (see 7.16.8.2), SMP receiver (see 7.18.5.2), and STP receiver (see 7.17.8).

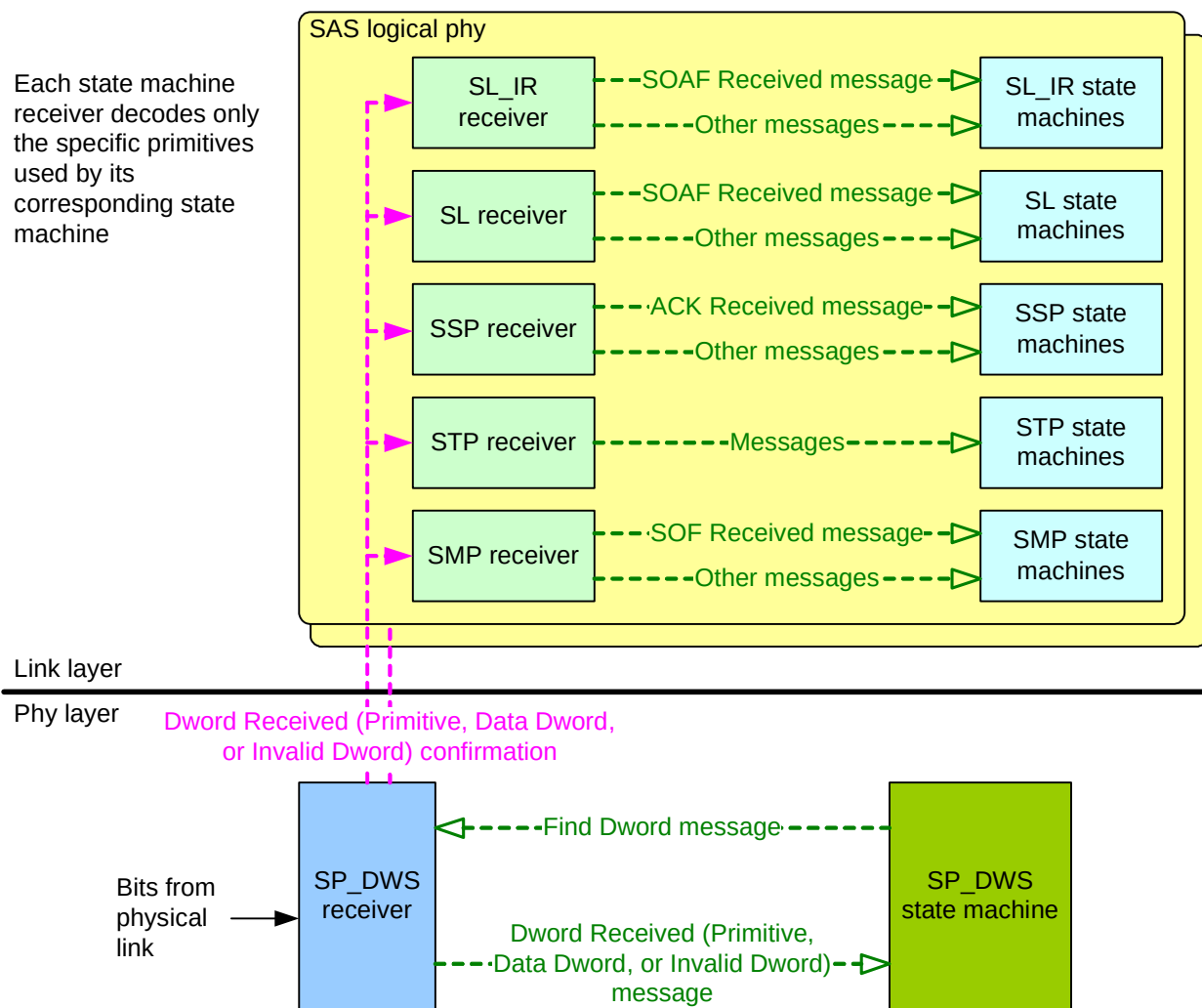


Figure 36 — Receive data path in a SAS phy

Figure 37 shows the receive data path in an expander phy showing the relationship between:

- the SP_DWS state machine (see 6.9) and the SP_DWS receiver (see 6.9.2);
- the SL_IR state machines (see 7.9.4) and the SL_IR receiver (see 7.9.4.2); and
- the XL state machines (see 7.15) and the XL receiver (see 7.15.2).

Each state machine receiver decodes only the specific primitives used by its corresponding state machine

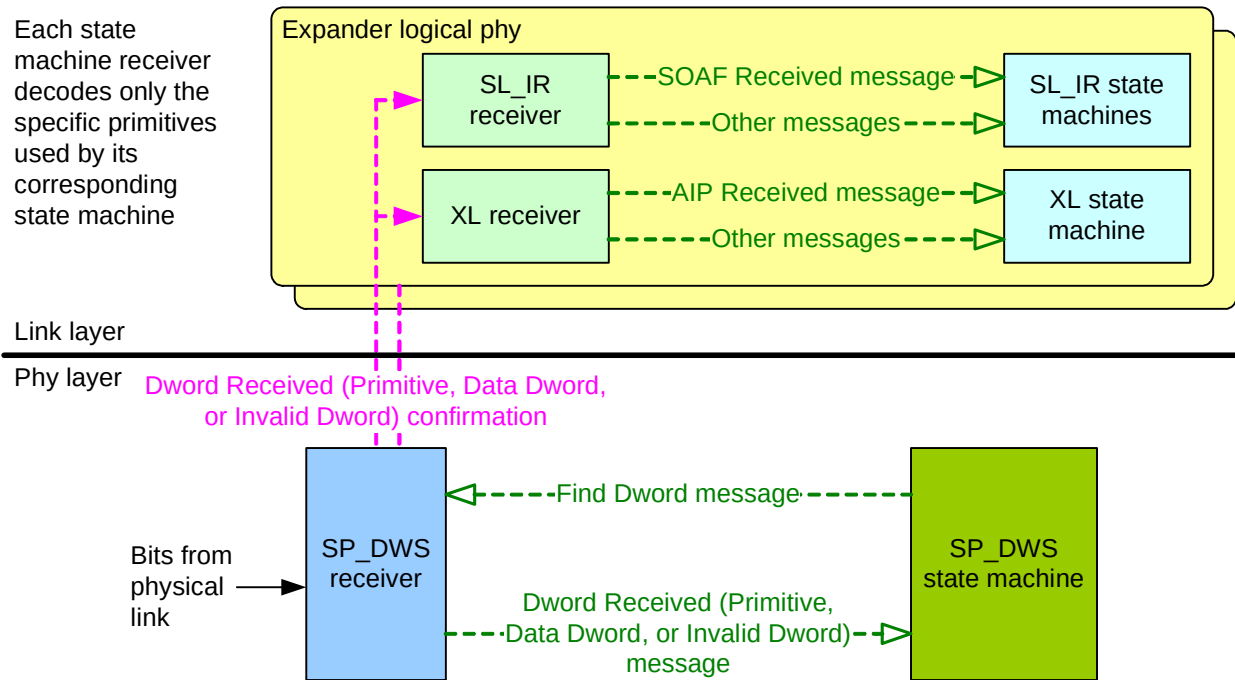


Figure 37 — Receive data path in an expander phy

4.3.4 State machines and SAS Device, SAS Port, and SAS Phy classes

Figure 38 shows which state machines are contained within the SAS Device, SAS Port, and SAS Phy classes.

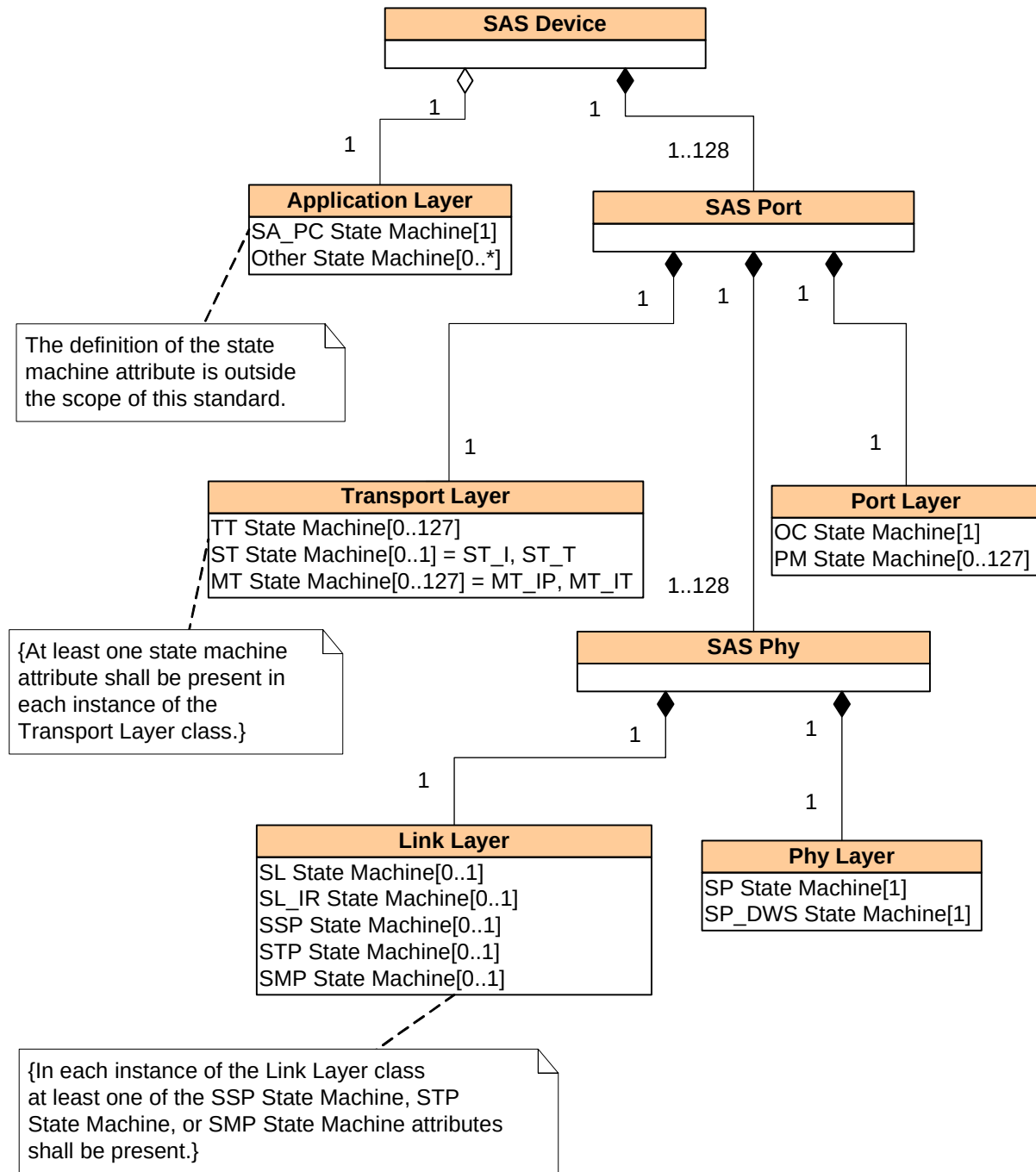


Figure 38 — State machines and SAS Device classes

Figure 39 shows which state machines are contained within the Expander Device, Expander Port, and Expander Phy classes.

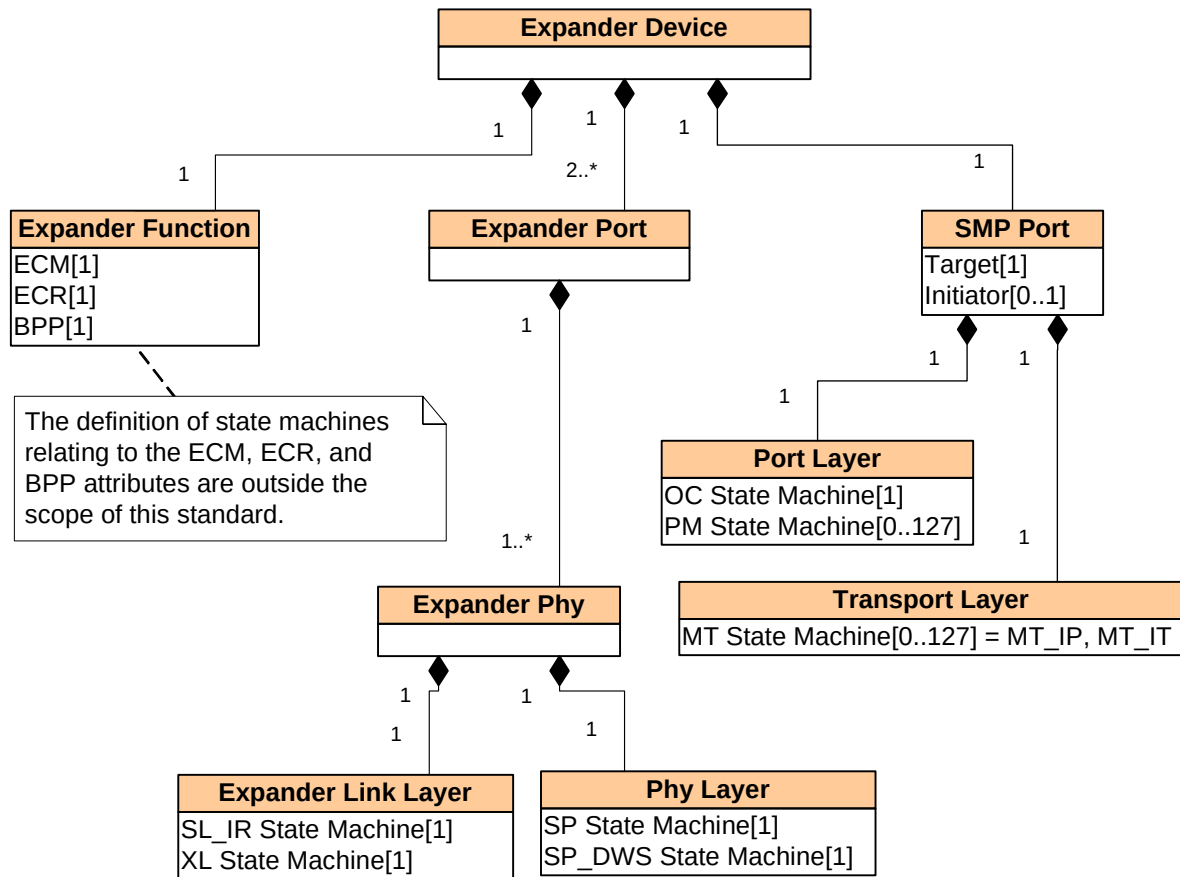


Figure 39 — State machines and Expander Device classes

Editor's Note 9: This figure does not show an SMP phy and associated link layer state machines below the SMP Port since figure 16 does not do so. If the expander allows more than one connection at a time to its SMP target port, then it's effectively a wide SMP port and has multiple phys and link layers. If it only allows one at a time, then it's effectively a narrow SMP port and only has one phy and link layer.

4.4 Resets

4.4.1 Reset overview

Figure 40 illustrates the reset terminology used in this standard:

- link reset sequence;
- phy reset sequence (see 6.7);
- SATA OOB sequence (see 6.7.2.1);
- SATA speed negotiation sequence (see 6.7.2.2);
- SAS OOB sequence (see 6.7.4.1);
- SAS speed negotiation sequence (see 6.7.4.2);
- hard reset sequence (see 7.9); and
- identification sequence (see 7.9).

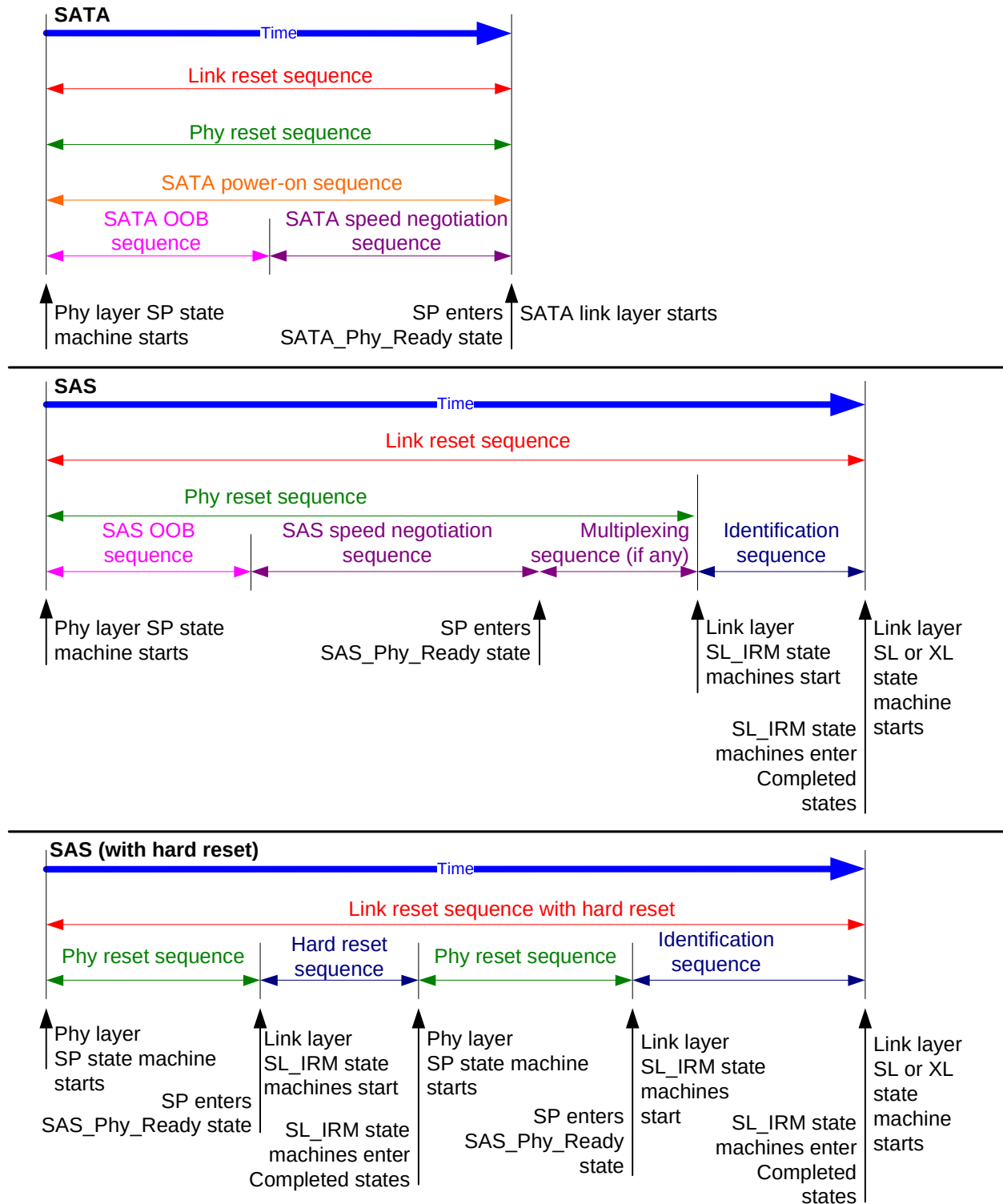


Figure 40 — Reset terminology

The phy reset sequences, including the OOB sequence and speed negotiation sequences, are implemented by the SP state machine and are described in 6.7 and 6.8. The hard reset sequence and identification sequence are implemented by the SL_IR state machine and are described in 7.9.

The link reset sequence has no effect on the transport layer and application layer. The HARD_RESET primitive sequence may be used during the identification sequence to initiate a hard reset. The link reset sequence serves as a hard reset for SATA devices (see SATA-2).

4.4.2 Hard reset

4.4.2.1 Hard reset overview

After the phy reset sequence, if a phy receives a HARD_RESET primitive before an IDENTIFY address frame, it shall be considered a reset event and initiate a hard reset of the port containing that phy.

When a port processes a hard reset, it shall stop transmitting valid dwords on each of the phys contained in that port. Each phy may then participate in new phy reset sequences (e.g., respond to incoming COMINITs) and shall originate a new link reset sequence if one is not detected. The hard reset shall not affect any other ports in the device.

If a SAS device is contained in an expander device, its SSP ports, STP ports, and/or SATA ports shall initiate a hard reset when an SMP PHY CONTROL function with a phy operation of HARD RESET and phy identifier specifying a virtual expander phy attached to such a SAS port is processed (see 10.4.3.27).

4.4.2.2 Additional hard reset processing by SAS ports

If the port processing the hard reset is an SSP port, the hard reset causes a Transport Reset event notification to the SCSI application layer (see 10.2.5), and the SCSI device shall perform the actions defined for hard reset in SAM-4. After processing the hard reset, each logical unit to which the SSP target port has access shall establish a unit attention condition for all SSP initiator ports with the additional sense code set to SCSI BUS RESET OCCURRED (see SAM-4 and SPC-4).

If the port processing the hard reset is an STP port in an STP/SATA bridge, the SATA host port shall originate a link reset sequence.

If the port processing the hard reset is an STP port that is not in an STP/SATA bridge, the STP target device shall perform the actions defined for power-on or hardware reset in ATA8-AAM.

4.4.2.3 Additional hard reset processing by expander ports

If the port processing a hard reset is an expander port, the expander device shall not originate a hard reset sequence on any of its other phys.

If the port processing a hard reset is an expander port, the expander function and other expander ports in the expander device shall not be affected by hard reset. SAS devices contained in the expander device shall not be affected by hard resets received by external expander ports in the expander device.

4.5 I_T nexus loss

When a SAS port receives OPEN_REJECT (NO DESTINATION), OPEN_REJECT (PATHWAY BLOCKED), OPEN_REJECT (RESERVED INITIALIZE 0), OPEN_REJECT (RESERVED INITIALIZE 1), OPEN_REJECT (RESERVED STOP 0), OPEN_REJECT (RESERVED STOP 1), or an open connection timeout occurs in response to a connection request, it shall retry the connection request until:

- a) the connection is established;
- b) for SSP target ports, the time indicated by the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page (see 10.2.7.4) expires; or
- c) the I_T nexus loss timer, if any, expires (see 4.7.1, 8.2.2.1, 10.2.7.4, and 10.4.3.17).

An I_T nexus loss occurs in a SAS port when:

- a) the I_T nexus loss timer expires; or
- b) the SAS port receives an abandon-class OPEN_REJECT (see table 114 in 7.2.5.13).

I_T nexus loss is handled by the port layer state machines (see 8.2.2.3). In some cases, the I_T nexus loss timer is overridden for connection requests through self-configuring expander devices as described in 4.7.1.

If an I_T nexus loss occurs in an SSP port, then the port shall send a Nexus Loss event notification to the SCSI application layer (see 10.2.5) and the SCSI device shall perform the actions defined for I_T nexus loss in SAM-4. If an I_T nexus loss occurs in an SSP initiator port, a SCSI application client should send an I_T NEXUS RESET task management function to the SSP target port when it reestablishes communication.

If an I_T nexus loss occurs in an STP initiator port, then the STP initiator port shall send a Transport Event Notification (Nexus Loss, [Device]) indication to the ATA application client (i.e., create a nexus loss event)(see ATA8-AAM). The ATA application client shall consider any commands for the lost STP target port to be completed with an error.

If an I_T nexus loss occurs in an STP target port, then the ATA device server shall abort all outstanding commands for the lost STP initiator port. If the STP target port is in an STP/SATA bridge, the STP/SATA bridge shall originate a link reset sequence to the SATA device so the ATA device server in the SATA device aborts all outstanding commands.

If an I_T nexus loss occurs in an SMP initiator port, then the port shall stop attempting to establish connections to the lost SMP target port.

If an I_T nexus loss occurs in an initiator port due to I_T nexus loss timer expiration, then a management application client should cause a link reset sequence on the phy(s) attached to the lost target port (e.g, if directly attached, the phys in the initiator port; if attached via expander device(s), the phys in the expander device closest to and attached to the target port).

4.6 Expander device model

4.6.1 Expander device model overview

An expander device shall contain the following:

- a) an expander function containing:
 - A) expander connection manager (ECM)(see 4.6.3);
 - B) expander connection router (ECR)(see 4.6.4); and
 - C) Broadcast propagation processor (BPP)(see 4.6.5);
- b) two or more physical expander phys. For the maximum number of phys, see 4.1.7;
- c) an expander port available per phy; and
- d) an SMP target port and a management device server.

An expander device may contain:

- a) an SMP initiator port and a management application client; and/or
- b) SAS devices with SSP ports, STP ports, and/or SMP ports and their associated device servers and/or application clients.

Figure 41 shows a model of an expander device showing the state machines in each expander port. The internal SMP ports are not shown.

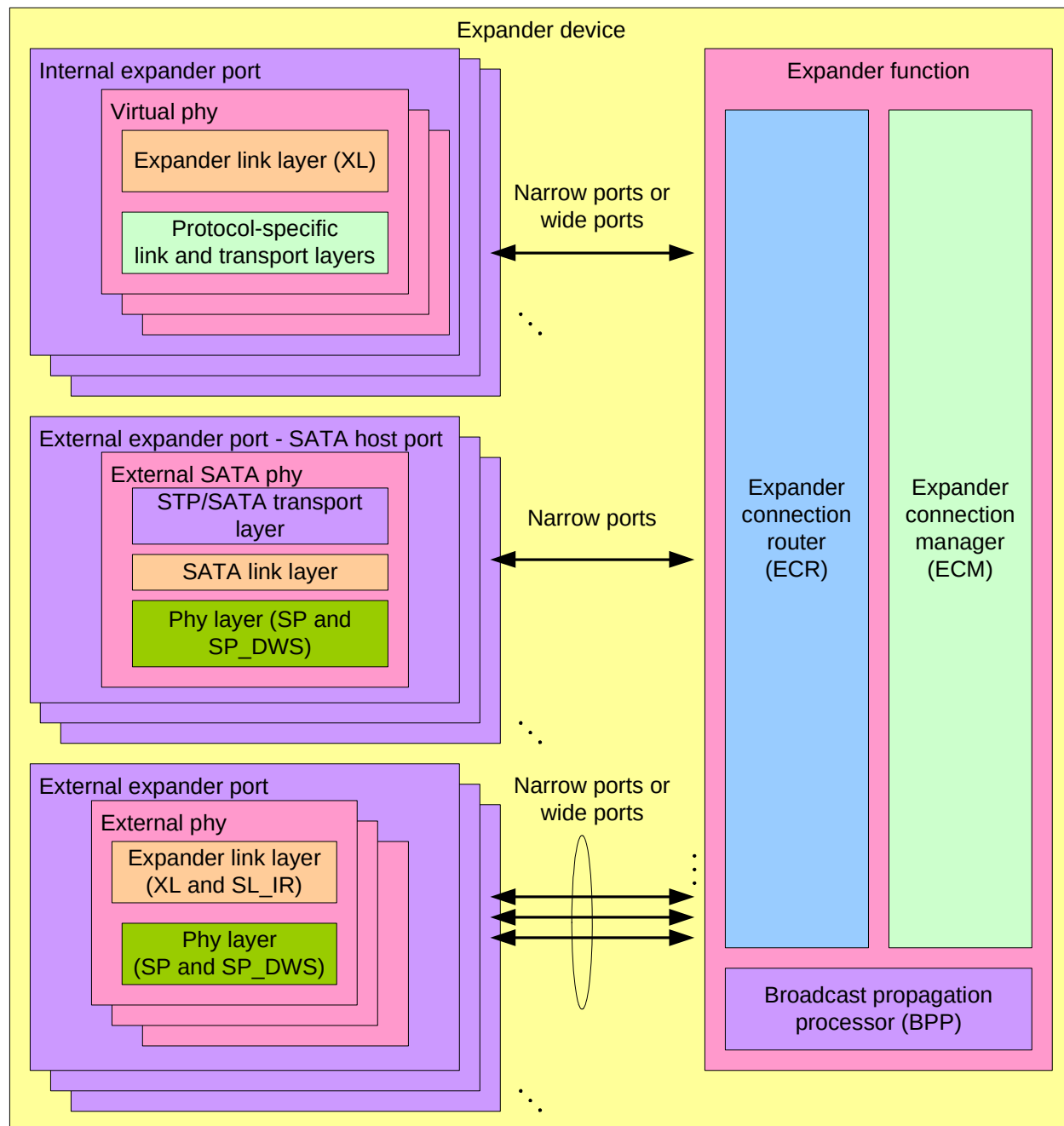


Figure 41 — Expander device model

4.6.2 Expander ports

An external expander port contains one or more physical phys (see 4.1.2). Since each phy in the expander device has the same SAS address, expander ports are created based on the attached SAS addresses (see 4.1.4).

Each phy in an expander port shall have the same routing attribute (see 4.6.7.1). The management device server shall return the same value in the ROUTING ATTRIBUTE field for each phy in an expander port in the SMP DISCOVER response (see 10.4.3.9).

Each phy in an expander port containing phys with table routing attributes in an externally configurable expander device shall have the same number of routing table entries (see 4.6.7.3).

A set of expander phys with table routing attributes in an expander device not supporting table-to-table attachments using the same external connector (see 5.2.3.3) is called an enclosure out port. A set of expander phys with subtractive routing attributes using the same external connector is called an enclosure in port. A set of expander phys with table routing attributes in an expander device supporting table-to-table attachments using the same external connector is called an enclosure universal port.

Each phy in an expander port shall have the same zone phy information (see 4.9.3.1). The zone phy information associated with each of the phys in an expander port is treated as the zoning properties of the expander port.

Each expander logical phy contains an expander link layer with an XL state machine (see 7.15) and an SL_IR state machine (see 7.9.4). The XL state machine in each expander logical phy within an expander port processes connection requests independently of the XL state machines in other expander logical phys.

An internal expander port contains a virtual phy with an expander link layer and a protocol-specific transport layer (e.g., to provide access as an SSP target port to a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-4 and SES-2)).

Each expander device shall include one internal SMP port using the expander device's SAS address.

Any additional internal SAS ports shall be inside SAS devices contained in the expander device, and thus have SAS addresses different from that of the expander device. These SAS ports shall be attached to internal expander ports with virtual phys.

Each STP/SATA bridge shall have a unique SAS address. This SAS address is reported in the ATTACHED SAS ADDRESS field in the SMP DISCOVER response (see 10.4.3.9) for the expander phy containing the STP/SATA bridge (i.e., the expander phy attached to the SATA device or SATA port selector).

4.6.3 Expander connection manager (ECM)

The ECM performs the following functions:

- a) maps a destination SAS address in a connection request to a destination phy using direct, subtractive, or table routing methods;
- b) arbitrates and assigns or denies path resources for connection requests following SAS arbitration and pathway recovery rules; and
- c) configures the ECR.

4.6.4 Expander connection router (ECR)

The ECR routes messages between pairs of expander logical phys as configured by the ECM. Enough routing resources shall be provided to support at least one connection.

When forwarding dwords during a connection from a source phy with a higher physical link rate to a destination phy with a lower physical link rate, rate matching (see 7.13) ensures the dwords are at a connection rate equal to or less than the lower physical link rate. However, the ECR may be requested to forward more dwords than the destination phy is able to accept if:

- a) an invalid dword occurs during a deletable primitive;
- b) an invalid dword occurs during a CLOSE; or
- c) multiple invalid dwords occur during a BREAK.

The ECR may discard dwords if needed and count them as receive elasticity buffer overflows (see 4.11).

When forwarding dwords from a SATA physical link with a higher physical link rate to a SAS logical link with a lower logical link rate, the SATA host port in the STP/SATA bridge shall throttle incoming FISes with SATA_HOLD (see 7.17.2).

NOTE 11 - If SATA_HOLD is deasserted for n dwords, the SATA device sends up to (21+n) more data dwords. SATA_HOLD does not affect primitives (see SATA-2). The STP/SATA bridge may expand or contract repeated and continued primitives without changing their functional meaning.

When forwarding dwords from a SAS logical link with a lower logical link rate to a SATA physical link with a higher physical link rate, the SATA host port in the STP/SATA bridge shall perform a process similar to rate matching (see 7.13) by inserting ALIGN (0) and/or SATA_HOLD on the SATA physical link whenever it underflows.

NOTE 12 - SATA requires that ALIGN 0 be sent in pairs (see SATA-2).

4.6.5 Broadcast propagation processor (BPP)

The BPP receives Broadcasts from each expander logical phy or from the management device server on behalf of an expander logical phy and requests transmission of those Broadcasts on all expander ports except the expander port from which the Broadcast was received.

In a zoning expander device with zoning enabled (see 4.9.2), Broadcasts are forwarded as described in 4.9.5.

4.6.6 Expander device interfaces

4.6.6.1 Expander device interface overview

The expander device arbitrates and routes between expander logical phys. All routing occurs between expander logical phys, not expander ports. The interaction between an XL state machine and the expander function consists of requests, confirmations, indications, and responses. This interaction is called the expander device interface.

Figure 42 describes the interfaces present within an expander device.

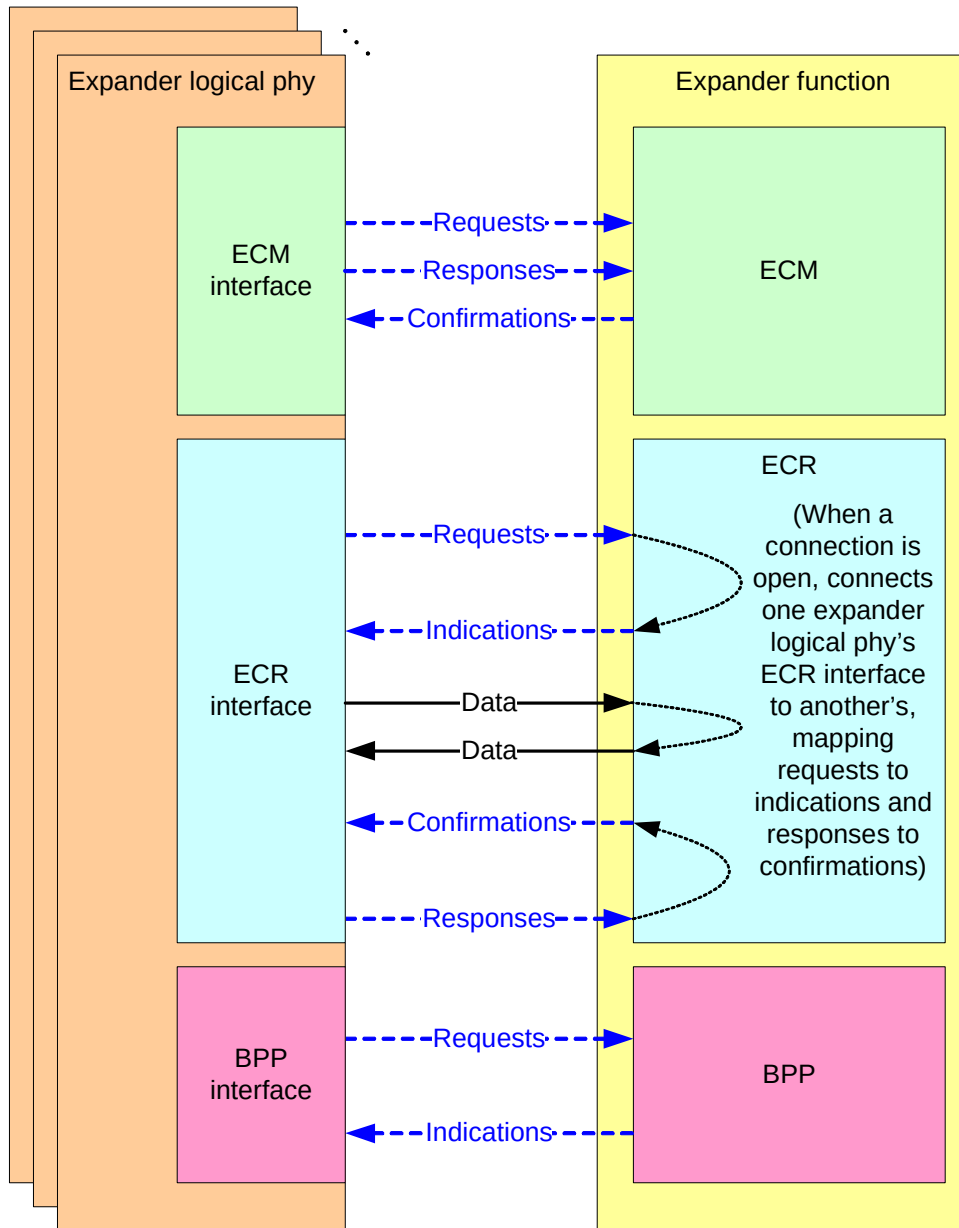


Figure 42 — Expander device interfaces

4.6.6.2 Expander device interfaces detail

Figure 43 shows the interface requests, confirmations, indications, and responses used by an expander device to manage connections.

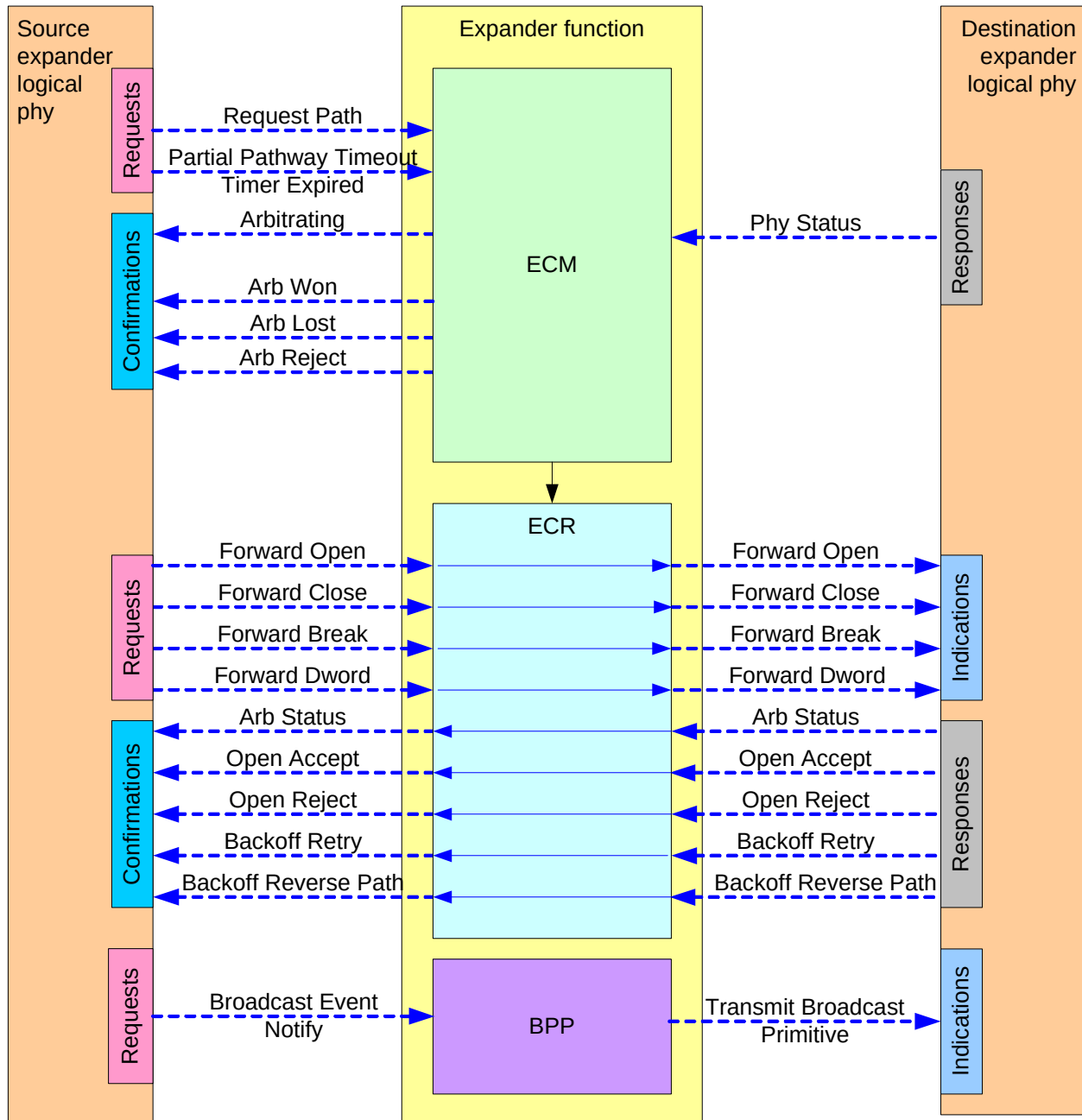


Figure 43 — Expander device interface detail

4.6.6.3 ECM interface

Table 13 describes the requests from an expander logical phy to the ECM. The XL state machine (see 7.15) defines specifically when each request is sent.

Table 13 — Expander logical phy to ECM requests

Request	Description
Request Path (arguments)	Request for a connection.
Partial Pathway Timeout Timer Expired	The Partial Pathway Timeout Timer expired.

Table 14 describes the responses from an expander logical phy to the ECM. The XL state machine (see 7.15) defines specifically when each response is sent.

Table 14 — Expander logical phy to ECM responses

Response	Description
Phy Status (Partial Pathway)	Response meaning that an expander logical phy: a) is being used for an unblocked partial pathway; or b) is waiting on another expander logical phy being used for a partial pathway.
Phy Status (Blocked Partial Pathway)	Response meaning that an expander logical phy: a) is being used for a blocked partial pathway; or b) is waiting on another expander logical phy being used for a blocked partial pathway.
Phy Status (Connection)	Response meaning that an expander logical phy: a) is being used for a connection; or b) is waiting on another expander logical phy being used for a connection.

Table 15 describes the confirmations from the ECM to an expander logical phy. These confirmations are sent in confirmation of a Request Path request. See 7.12.4 for specific definitions for when each confirmation is sent.

Table 15 — ECM to expander logical phy confirmations (part 1 of 2)

Confirmation	Description
Arbitrating (Normal)	Confirmation that the ECM has received the Request Path request.
Arbitrating (Waiting On Partial)	Confirmation that the ECM is waiting on a partial pathway (see 4.1.11).
Arbitrating (Blocked On Partial)	Confirmation that the ECM is waiting on a blocked partial pathway (see 4.1.11).
Arbitrating (Waiting On Connection)	Confirmation that the ECM is waiting for a connection to complete (see 4.1.12).
Arb Won	Confirmation that an expander logical phy has won path arbitration.
Arb Lost	Confirmation that an expander logical phy has lost path arbitration.

Table 15 — ECM to expander logical phy confirmations (part 2 of 2)

Confirmation	Description
Arb Reject (No Destination)	Confirmation that the request is rejected because the expander device is not configuring (see 4.8) and there is no path to the destination.
Arb Reject (Bad Destination)	Confirmation that the request is rejected because the path to the destination maps back to the requesting expander port.
Arb Reject (Connection Rate Not Supported)	Confirmation that the request is rejected because there is a destination port capable of routing to the requested destination SAS address but no phys within the destination port are configured to support the requested connection rate.
Arb Reject (Zone Violation)	Confirmation that the request is rejected because the expander device is not locked and there is a zoning violation (see 4.9.3).
Arb Reject (Pathway Blocked)	Confirmation that the request is rejected because the requesting expander logical phy needs to back off according to SAS pathway recovery rules.
Arb Reject (Retry)	Confirmation that the request is rejected because: a) the expander device is configuring (see 4.7.2) and the ECM would otherwise have returned Arb Reject (No Destination); b) the expander device is locked (see 4.9.6.2) and the ECM would otherwise have returned Arb Reject (Zone Violation); or c) the expander device has reduced functionality (see 4.6.8 and 7.12.4.2.5).

4.6.6.4 ECR interface

Table 16 describes the requests from an expander logical phy to the ECR and the corresponding indications from the ECR to another expander logical phy. The XL state machine (see 7.15) defines specifically when each request is sent.

Table 16 — Expander logical phy to ECR to expander logical phy requests and indications

Request/indication	Description
Forward Open (arguments)	Request/indication to forward an OPEN address frame.
Forward Close	Request/indication to forward a CLOSE.
Forward Break	Request/indication to forward a BREAK.
Forward Dword	Request/indication to forward a dword.

Table 17 describes the responses from an expander logical phy to the ECR and the corresponding confirmations from the ECR to another expander logical phy. These responses are sent in response to a Forward Open indication. The XL state machine (see 7.15) defines specifically when each response is sent.

Table 17 — Expander logical phy to ECR to expander logical phy responses and confirmations

Response/confirmation	Description
Arb Status (Normal)	Confirmation/response that AIP (NORMAL) has been received.
Arb Status (Waiting On Partial)	Confirmation/response that AIP (WAITING ON PARTIAL) has been received.
Arb Status (Waiting On Connection)	Confirmation/response that AIP (WAITING ON CONNECTION) has been received.
Arb Status (Waiting On Device)	Confirmation/response that: a) AIP (WAITING ON DEVICE) has been received; or b) the expander logical phy has completed the forwarding of an OPEN address frame and has entered the XL6:Open_Response_Wait state.
Open Accept	Confirmation/response that OPEN_ACCEPT has been received.
Open Reject	Confirmation/response that OPEN_REJECT has been received.
Backoff Retry	Confirmation/response that: a) a higher priority OPEN address frame has been received (see 7.12.3); and b) the source SAS address and connection rate of the received OPEN address frame are not equal to the destination SAS address and connection rate of the transmitted OPEN address frame.
Backoff Reverse Path	Confirmation/response that: a) a higher priority OPEN address frame has been received (see 7.12.3); and b) the source SAS address and connection rate of the received OPEN address frame are equal to the destination SAS address and connection rate of the transmitted OPEN address frame.

4.6.6.5 BPP interface

Table 18 describes the requests from an expander logical phy to the BPP. Requests from the management device server about SMP ZONED BROADCAST requests received from the SMP target port in zoning

expander devices with zoning enabled are not described. See 4.9.5 for more information on how zoning expander devices with zoning enabled handle Broadcasts.

Table 18 — Expander logical phy to BPP requests

Request	Description
Broadcast Event Notify (Phy Not Ready)	Request to originate a Broadcast (Change) because the expander logical phy's SP state machine transitioned from the SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready state to the SP0:OOB_COMINIT state (see 6.8).
Broadcast Event Notify (SATA Spinup Hold)	Request to originate a Broadcast (Change) because the SATA spinup hold state has been reached (see 6.8 and 6.11) by the expander phy.
Broadcast Event Notify (Identification Sequence Complete)	Request to originate a Broadcast (Change) because the expander logical phy has completed the identification sequence (see 7.9).
Broadcast Event Notify (SATA Port Selector Change)	Request to originate a Broadcast (Change) because the expander phy detected that a SATA port selector appeared or disappeared.
Broadcast Event Notify (Change Received)	Request to forward a Broadcast (Change) because the expander logical phy received a Broadcast (Change). See 7.11 and 7.15.
Broadcast Event Notify (Reserved Change 0 Received)	Request to forward a Broadcast (Reserved Change 0) because the expander logical phy received a Broadcast (Reserved Change 0). See 7.11 and 7.15.
Broadcast Event Notify (Reserved Change 1 Received)	Request to forward a Broadcast (Reserved Change 1) because the expander logical phy received a Broadcast (Reserved Change 1). See 7.11 and 7.15.
Broadcast Event Notify (SES Received)	Request to forward a Broadcast (SES) because the expander logical phy received a Broadcast (SES). See 7.15.
Broadcast Event Notify (Expander Received)	Request to forward a Broadcast (Expander) because the expander logical phy received a Broadcast (Expander). See 7.15.
Broadcast Event Notify (Asynchronous Event Received)	Request to forward a Broadcast (Asynchronous Event) because the expander logical phy received a Broadcast (Asynchronous Event). See 7.15.
Broadcast Event Notify (Reserved 3 Received)	Request to forward a Broadcast (Reserved 3) because the expander logical phy received a Broadcast (Reserved 3). See 7.15.
Broadcast Event Notify (Reserved 4 Received)	Request to forward a Broadcast (Reserved 4) because the expander logical phy received a Broadcast (Reserved 4). See 7.15.

Table 19 describes the indications from the BPP to an expander logical phy. Indications to the management application client to generate SMP ZONED BROADCAST functions from the SMP initiator port in a zoning expander device with zoning enabled are not described. See 4.9.5 for more information on how zoning expander devices with zoning enabled handle Broadcasts.

Table 19 — BPP to expander logical phy indications

Message	Description
Transmit Broadcast (type)	Indication to transmit a BROADCAST with the specified type.

4.6.7 Expander device routing

4.6.7.1 Routing attributes and routing methods

Each expander phy in an expander device shall support one of the following routing attributes:

- a) direct routing attribute;
- b) table routing attribute; or
- c) subtractive routing attribute.

The routing attributes allow the ECM to determine which routing method to use when routing connection requests to the expander logical phys in the expander phy:

- a) the table routing method routes connection requests to attached expander devices using an expander route table;
- b) the subtractive routing method routes unresolved connection requests to an attached expander device; or
- c) the direct routing method routes connection requests to attached end devices, the SMP port of an attached expander device, or SAS devices contained in the expander device.

Table 20 describes the routing methods that the ECM uses based on the routing attributes of an expander phy.

Table 20 — Routing attributes and routing methods

Routing attribute of an expander phy	Routing method used by ECM for the expander phy
Direct	Direct ^a
Table	Direct, if attached to an end device
	Direct, if attached to an expander device, for the SAS address of the expander device
	Table, if attached to an expander device, for SAS addresses beyond the expander device
Subtractive	Direct, if attached to an end device
	Subtractive, if attached to an expander device
^a If attached to an expander device, the ECM is only able to route to the expander device itself through a phy with the direct routing attribute	

An expander device may have zero or more phys with the table routing attribute.

An expander device shall have at most one defined port containing phys with the subtractive routing attribute.

An expander device that does not support table-to-table attachment shall only use phys with the table routing attribute to attach to phys with the subtractive routing attribute in other expander devices.

If multiple phys within an expander device have subtractive routing attributes and are attached to expander devices, they shall attach to phys with identical SAS addresses (i.e., the same expander port).

If multiple phys within an expander device have subtractive routing attributes and are attached to expander devices that do not have identical SAS addresses, the application client that is performing the discover process (see 4.7) shall report an error in a vendor-specific manner.

4.6.7.2 Connection request routing

The ECM shall determine how to route a connection request from a source expander logical phy to a destination expander logical phy in a different expander port if the destination expander logical phy is enabled, operating at a valid logical link rate (e.g., the SMP DISCOVER response reports a NEGOTIATED LOGICAL LINK

RATE field set to G1 (i.e., 8h), G2 (i.e., 9h), or G3 (i.e., Ah)), and not excluded because of zoning (see 4.9.2) using the following precedence:

- 1) route to an expander logical phy with the direct routing attribute or table routing attribute when the destination SAS address matches the attached SAS address;
- 2) route to an expander logical phy with the table routing attribute when the destination SAS address matches an enabled SAS address in the expander route table;
- 3) route to an expander logical phy with the subtractive routing attribute; and
- 4) return an Arb Reject confirmation (see 4.6.6.3) to the source expander logical phy.

If the destination expander logical phy only matches an expander logical phy in the same expander port from which the connection request originated, then the ECM shall return an Arb Reject confirmation (see 4.6.6.3).

If the destination SAS address of a connection request matches a disabled SAS address in an expander route table, then the ECM shall ignore the match.

4.6.7.3 Expander route table

4.6.7.3.1 Expander route table overview

An expander device that supports the table routing method shall contain an expander route table. The expander route table is a structure that provides an association between destination SAS addresses (i.e., routed SAS addresses) and the expander phys to which connection requests to those destination SAS addresses are forwarded.

Zoning expander devices include additional fields in their expander route tables (see 4.9.3.4).

Table 21 defines the types of expander route tables.

Table 21 — Expander route table types

Type	SMP functions to access	Reference
Phy-based	REPORT ROUTE INFORMATION (see 10.4.3.12) and CONFIGURE ROUTE INFORMATION (see 10.4.3.26)	4.6.7.3.2
Expander-based	REPORT EXPANDER ROUTE TABLE LIST (see 10.4.3.16)	4.6.7.3.3

4.6.7.3.2 Phy-based expander route table

Figure 44 shows a representation of a phy-based expander route table.

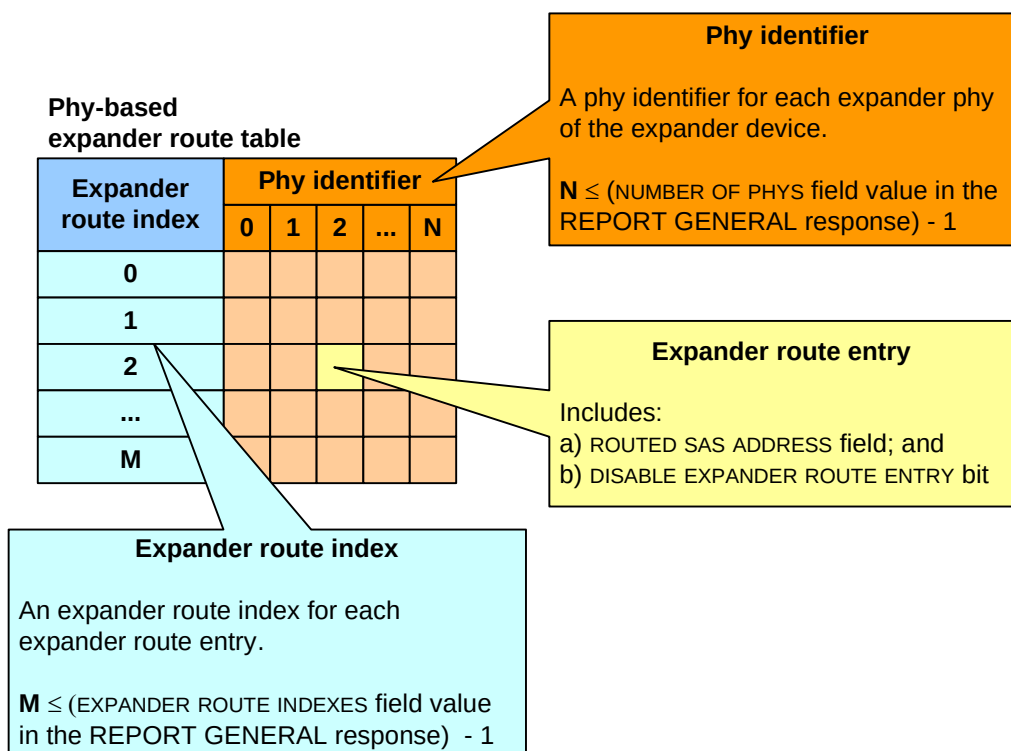


Figure 44 — Phy-based expander route table

For each expander route index and phy identifier combination, the phy-based expander route table contains an expander route entry containing a ROUTED SAS ADDRESS field and a DISABLE EXPANDER ROUTE ENTRY bit.

A management application client may access a specific expander route entry within a phy-based expander route table with the SMP REPORT ROUTE INFORMATION function (see 10.4.3.12) and the SMP CONFIGURE ROUTE INFORMATION function (see 10.4.3.26).

An expander device reports the maximum expander route index in the EXPANDER ROUTE INDEXES field and indicates if the phy-based expander route table is configurable in the CONFIGURABLE ROUTE TABLE bit in the SMP REPORT GENERAL response (see 10.4.3.3).

Each expander route entry shall be disabled after power on.

4.6.7.3.3 Expander-based expander route table

Figure 45 shows a representation of an expander-based expander route table.

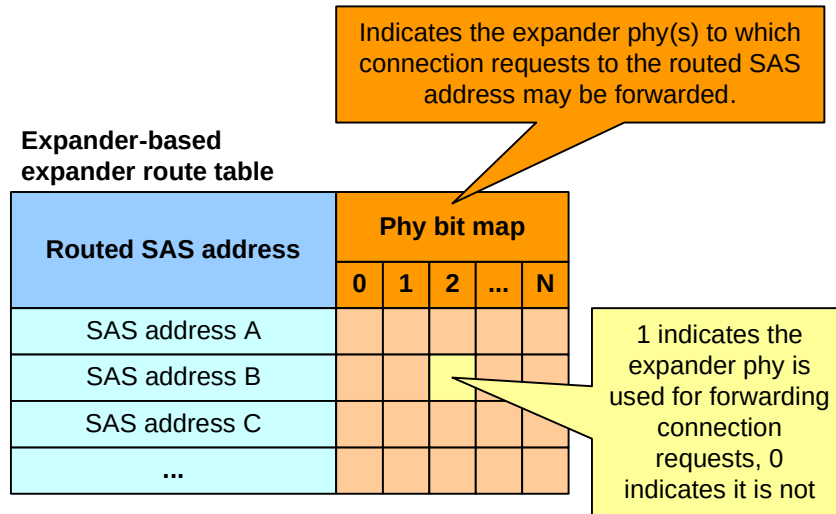


Figure 45 — Expander-based expander route table

For each routed SAS address, the expander-based expander route table contains a phy bit map.

A management application client may access an expander-based expander route table with the SMP REPORT EXPANDER ROUTE TABLE LIST function (see 10.4.3.16).

An expander device reports the size of its expander-based expander route table in the MAXIMUM NUMBER OF ROUTED SAS ADDRESSES field in the SMP REPORT GENERAL response (see 10.4.3.3).

Editor's Note 10: With the expander-based section now separate from the phy-based section, is the meaning of "index" as used in REPORT EXPANDER ROUTE TABLE LIST sufficiently less confusing that it doesn't need to be shown in the figure?

Editor's Note 11: Add a comment that the SAS addresses are not sorted in any particular order

4.6.8 Expander device reduced functionality

An expander device shall originate a Broadcast (Expander) to indicate that it is going to temporarily have reduced functionality for a period of time (e.g., if during a microcode update, it disables the ECM and ECR access to its SMP target port or to one or more expander phys, or if it experiences reduced performance). The maximum period of time that the expander device is going to have reduced functionality is indicated:

- in the REPORT SUPPORTED OPERATION CODES command parameter data (see SPC-4) for the WRITE BUFFER command reported by a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) accessed via an SSP target port contained in the expander device; and
- from the contents of the MAXIMUM REDUCED FUNCTIONALITY TIME field in the REPORT GENERAL response (see 10.4.3.3).

After the expander device originates a Broadcast (Expander) it shall:

- set the REDUCED FUNCTIONALITY bit to one in the REPORT GENERAL response (see 10.4.3.3);
- initialize the reduced functionality delay timer to the value indicated by the INITIAL REDUCED FUNCTIONALITY DELAY field in the REPORT GENERAL response (see 10.4.3.3) and start the reduced functionality delay timer;

- c) wait for the reduced functionality delay timer to expire before reducing any expander functions; and
- d) not stop or restart the reduced functionality delay timer until after the expander device enters the reduced functionality condition.

If the expander device receives a connection request that maps to an expander phy or its SMP target port that is not accessible because of the reduced functionality, then the expander device shall respond with an OPEN REJECT (RETRY) until the operation that caused the expander device to have reduced functionality is complete.

After the operation that caused the expander device to have reduced functionality is complete, the expander device shall:

- 1) set the REDUCED FUNCTIONALITY bit to zero in the REPORT GENERAL response (see 10.4.3.3); and
- 2) originate a Broadcast (Change) or a link reset sequence on each expander phy.

After receiving a Broadcast (Expander), a management application client behind an SMP initiator port should issue a REPORT GENERAL function (see 10.4.3.3) to all expander devices to determine:

- a) the expander devices, if any, that are reducing their functionality (i.e., the REDUCED FUNCTIONALITY bit is set to one in the REPORT GENERAL response)(see 4.6.8); and
- b) the amount of time remaining until the reduced functionality occurs (i.e., the contents of the TIME TO REDUCED FUNCTIONALITY field in the REPORT GENERAL response).

Editor's Note 12: This is not an ideal place for Broadcast (Expander) handling rules, since expander reduced functionality is not the only reason that might appear. Another 4.6.x section (since 4.6 is expander device model) would be better. Another 4.x section might also work (making Broadcast (Expander) handling a peer of the discover process).

Editor's Note 13: Need to mention what the application client is (not) supposed to do while the expander has reduced functionality. Don't try to open connections through the expander during the reduced functionality time. Try to complete all I/Os before that time, so the target doesn't try to open connections back to the initiator during the reduced functionality time.

4.7 Discover process

4.7.1 Discover process overview

Management application clients direct an SMP initiator port to request SMP functions from an SMP target port. Management application clients are located in every SAS initiator device and every self-configuring expander device. A management application client performs a discover process to discover all the SAS devices and expander devices in the SAS domain (i.e., determining their device types, SAS addresses, and supported protocols). A SAS initiator device uses this information to determine SAS addresses to which it is able to establish connections. A self-configuring expander device uses this information to fill in its expander route table.

A management application client performing the discover process shall perform a level-order (i.e., breadth-first) traversal of the SAS domain. The order of traversal shall be to discover:

- 1) the device(s) to which the device containing the management application client is attached;
- 2) if an attached device is an expander device, every device attached to that expander device; and
- 3) for each expander device found, every device attached to that expander device.

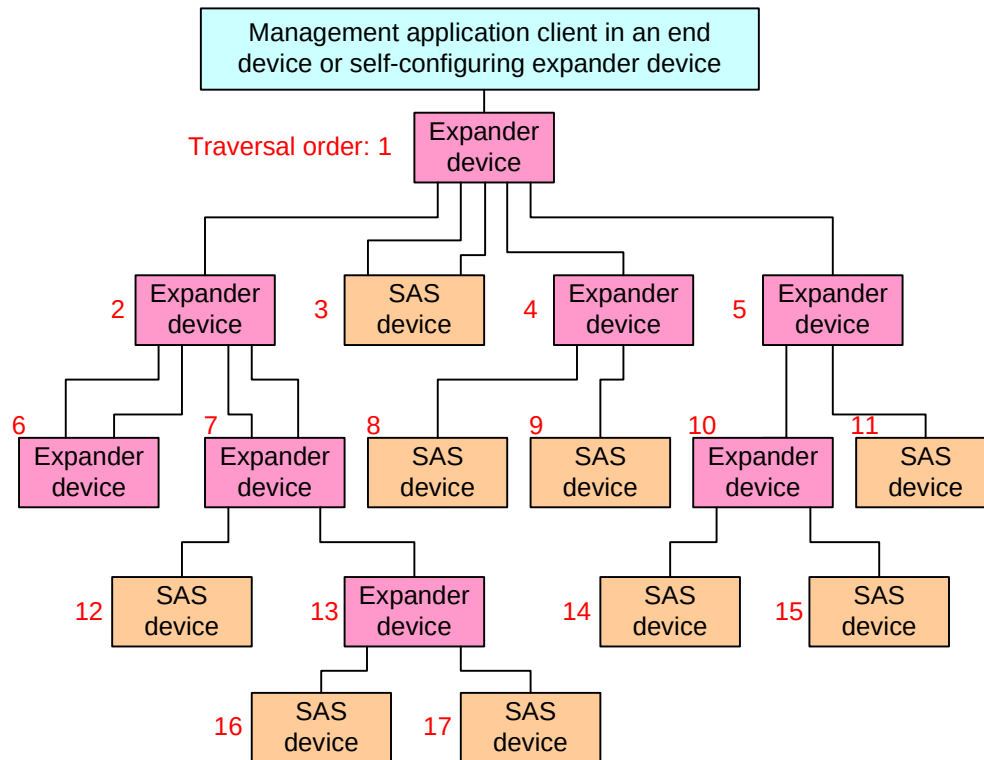
This order is repeated until all expander devices have been traversed. If the management application client discovers an externally configurable expander device that is not located beyond a self-configuring expander device with the CONFIGURES OTHERS bit set to one in the REPORT GENERAL response (see 10.4.3.3) it shall perform the configuration subprocess (see 4.8) to configure the expander route table before attempting to

establish connections with devices attached two levels (see 4.8.4) beyond that externally configurable expander device.

If an end device is directly attached to a self-configuring expander device with the CONFIGURES OTHERS bit set to one in the REPORT GENERAL response (see 10.4.3.3), then the management application client in that end device is not required to perform the configuration subprocess. If all the expander devices in the SAS domain are self-configuring expander devices, then management application clients in end devices are not required to perform the configuration subprocess.

If the management application client is inside a self-configuring expander device, then the process shall be repeated on each expander port.

Figure 46 shows an example of level-order traversal.



Note: Assume that the phy with the lowest phy identifier in each expander device is on the top right, and the remaining phys have increasing phy identifiers assigned in a counter-clockwise direction

Figure 46 — Level-order traversal example

The management application client determines whether an expander device or SAS device is attached at each point in the traversal. For the first device (i.e., the device directly attached), this is determined from the DEVICE TYPE field in the IDENTIFY address frame (see 7.8.2) information received by the phy that the management application client is using. For other devices (i.e., devices not directly attached), this is determined from ATTACHED DEVICE TYPE field the SMP DISCOVER response (see 10.4.3.9).

If an expander device is attached, the management application client shall use the SMP REPORT GENERAL function (see 10.4.3.3) to determine how many phys are in the expander device and then use the SMP DISCOVER function (see 10.4.3.9) and/or the SMP DISCOVER LIST function (see 10.4.3.14) to determine what is attached to each expander phy (e.g., the device type, SAS address, and supported protocol(s)).

NOTE 13 - Expander devices compliant with previous versions of this standard do not implement the SMP DISCOVER LIST function.

If the expander device's EXTERNALLY CONFIGURABLE ROUTE TABLE bit is set to zero in the SMP REPORT GENERAL response, its own management application client shall configure its own expander route table as described in 4.8.

While a self-configuring expander device's CONFIGURING bit is set to one in the SMP REPORT GENERAL response, connection requests for destination ports two or more levels beyond the self-configuring expander device that would otherwise have returned OPEN_REJECT (NO DESTINATION) return OPEN_REJECT (RETRY) instead (see 4.6.6.3 and 4.7.2).

If a SAS device is attached, the discover process is not required to obtain any more information about the SAS device. Additional discovery software may access that SAS device, however:

- a) if the SAS device supports an SMP target port, the management application client may use SMP functions (e.g., REPORT GENERAL and REPORT MANUFACTURER INFORMATION) to determine additional information about the SAS target device;
- b) if the SAS device supports an SSP target port, a SCSI application client may transmit SCSI commands (e.g., INQUIRY and REPORT LUNS) to determine additional information about the SCSI target device; and
- c) if the end device supports an STP target port, an ATA application client may transmit ATA commands (e.g., IDENTIFY DEVICE and IDENTIFY PACKET DEVICE) to determine additional information about the ATA device.

The result of the discover process is that the management application client has the necessary information (e.g., the device type, SAS address, and supported protocol(s)) to communicate with each SAS device and expander device in the SAS domain and each externally configurable expander device is configured with the necessary expander route entries to allow routing of connection requests through the SAS domain.

If the discover process occurs and any phy within the expander device is in the process of a link reset sequence resulting from an SMP PHY CONTROL function (see 10.4.3.27) phy operation of LINK RESET or HARD RESET, then the management device server shall set the NEGOTIATED PHYSICAL LINK RATE field (see table 349) to RESET_IN_PROGRESS in the SMP DISCOVER response (see 10.4.3.9).

The discover process may be aborted prior to completion and restarted if there is an indication that it may be based on incorrect information (e.g., reception of a Broadcast (Change) or a change in the EXPANDER CHANGE COUNT field returned in an SMP response).

Annex L contains an example implementation of how a management application client may perform the discover process.

4.7.2 Discover process in a self-configuring expander device

The management application client of a self-configuring expander device shall configure:

- a) the expander routing table in that expander device; and
- b) the expander routing table in each externally configurable expander device in the SAS domain that is not located behind another self-configuring expander device.

When a self-configuring expander device receives a Broadcast (Change) the management application client shall start the discover process using the expander port that received the Broadcast (Change). If a change to the expander route table is identified then the management device server shall set its CONFIGURING bit to one in the SMP REPORT GENERAL response (see 10.4.3.3).

If zoning is enabled, the management application client in a self-configuring expander device shall use the SMP DISCOVER response (see 10.4.3.9) or SMP DISCOVER LIST response (see 10.4.3.14) values to set the zone group values in the zoning expander route table for all SAS addresses in the zoning expander route table (see 4.9.3.4).

The management application client shall set the CONFIGURING bit to zero when the discover process is complete. When the CONFIGURING bit changes from one to zero:

- a) a zoning expander device with zoning enabled shall originate a Broadcast (Change) on each expander port that has access to the expander port through which the discover process was performed based on the zone permission table; and

- b) an expander device with zoning disabled shall originate a Broadcast (Change) on each expander port other than the one through which the discover process was performed.

When a Broadcast (Change) has been received by a self-configuring expander device, all previously valid SAS addresses shall continue to be routable until they are determined to be no longer valid. When a change has been identified, all unaffected SAS addresses shall continue to be routable.

While the CONFIGURING bit is set to one, the expander device shall return OPEN_REJECT (RETRY) for any connection requests that would otherwise have resulted in OPEN_REJECT (NO DESTINATION) (see 4.6.6.3).

The management application client in a self-configuring expander device shall maintain self-configuration status for the last vendor-specific number of errors encountered during self-configuration and should maintain at least one self-configuration status per phy. The management device server shall assign descriptors to the statuses sequentially starting at 0001h and shall return the descriptors in the SMP REPORT SELF-CONFIGURATION STATUS response (see 10.4.3.5). The management device server shall return the index of the last self-configuration status descriptor in the SMP REPORT GENERAL response (see 10.4.3.3), the SMP REPORT SELF-CONFIGURATION STATUS response (see 10.4.3.5), and the SMP DISCOVER LIST response (see 10.4.3.14). The management device server shall wrap the index to 0001h when the highest supported descriptor index has been used.

The management device server shall support self-configuration status descriptor indexes from 0001h to FFFFh. The actual number of self-configuration status descriptors that the management device server maintains for retrieval with the REPORT SELF-CONFIGURATION STATUS request is vendor specific and is indicated by the MAXIMUM NUMBER OF STORED SELF-CONFIGURATION STATUS DESCRIPTORS field defined in the REPORT GENERAL response (see 10.4.3.3). The volatility of these stored descriptors is vendor specific. The management device server shall replace the least recently recorded self-configuration status descriptor with a new one once the number of recorded descriptors exceeds the value indicated by the MAXIMUM NUMBER OF STORED SELF-CONFIGURATION STATUS DESCRIPTORS field.

4.7.3 Enabling multiplexing

A management application client may configure multiplexing (see 6.10) in expander devices. Self-configuring expander devices may configure multiplexing for their own phys. The choice of whether or not to enable multiplexing on a physical link is vendor-specific.

If the SAS domain contains all 6 Gbps target phys, then the management application clients should disable multiplexing on every phy.

If the SAS domain contains all 3 Gbps target phys, then the management application clients should:

- a) multiplex each 6 Gbps physical link into two 3 Gbps logical links; and
- b) not multiplex 3 Gbps physical links.

If the SAS domain contains all 1,5 Gbps target phys, then the management application client should:

- a) multiplex each 6 Gbps physical link into two 3 Gbps logical links; and
- b) multiplex each 3 Gbps physical link into two 1,5 Gbps logical links.

NOTE 14 - Rate matching is used for 1,5 Gbps connections carried on 3 Gbps logical links.

4.8 Configuration subprocess

4.8.1 Configuration subprocess overview

As part of the discover process (see 4.7), when it discovers externally configurable expander device, the management application client performs the configuration subprocess to configure the expander routing table in that externally configurable expander device with SAS addresses discovered two levels beyond each table routing phy in that externally configurable expander device. A single discover process performs the configuration subprocess at least once per externally configurable expander device.

The routing table in an expander device needs to be configured before connections are able to be established with devices attached two levels beyond that expander device.

4.8.2 Allowed topologies

If the management application client detects an externally configurable expander device phy with the table routing attribute attached to an expander device phy with either the direct routing attribute or the table routing attribute, it shall report an error in a vendor-specific manner.

If the management application client detects an overflow of an expander route index, it shall report an error in a vendor-specific manner.

If the route table optimization (see 4.8.3) is disabled and the management application client detects an expander route entry that references the SAS address of the expander device itself (i.e., self-reference), it shall disable the expander route entry by setting the `DISABLE EXPANDER ROUTE ENTRY` bit to one in the `SMP CONFIGURE ROUTE INFORMATION` request (see 10.4.3.26). The management application client shall disable each expander route entry in the expander route table by setting the `DISABLE EXPANDER ROUTE ENTRY` bit to one in the `SMP CONFIGURE ROUTE INFORMATION` request (see 10.4.3.26) for each expander phy that has its attached device type set to 000b (i.e., no device attached).

If the management application client detects a port that the configuration subprocess has not already configured with a SAS address it has already found attached to another expander device, it should use the `SMP PHY CONTROL` function (see 10.4.3.27) to disable all the expander phys attached to that SAS address except for phys in the expander device with the lowest SAS address. Figure 47 shows some illegal topologies.

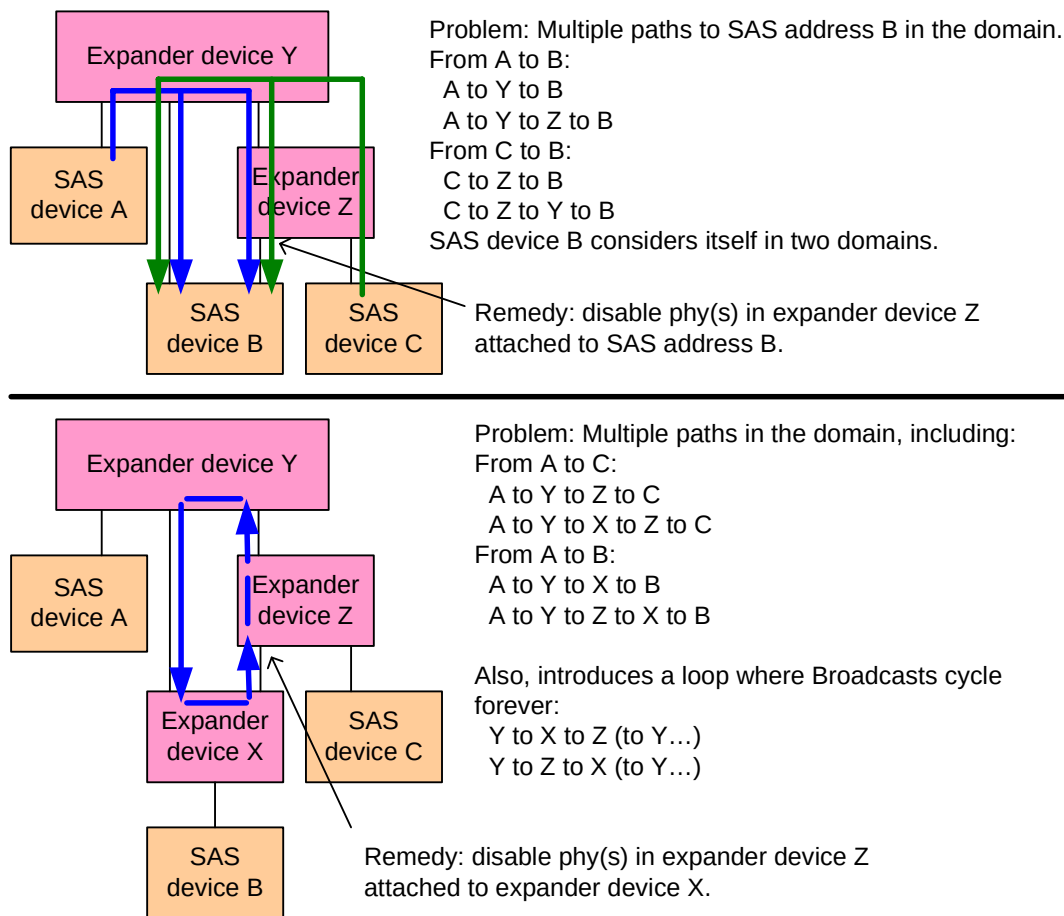


Figure 47 — Examples of invalid topologies

Annex L contains an example implementation of how a management application client may perform the discover process and configuration subprocess.

4.8.3 Route table optimization

The management application client shall support a route table optimization that reduces the number of entries required in an expander route table in an externally configurable expander device. The method used to enable and disable the route table optimization is vendor specific.

If the route table optimization is enabled, then the management application client shall exclude discovered SAS addresses from the expander route table when any of the following conditions are met:

- a) in the SMP DISCOVER response (see 10.4.3.9) for the discovered phy:
 - A) the FUNCTION RESULT field is set to a non-zero value (i.e., not SMP FUNCTION ACCEPTED);
 - b) in the SMP DISCOVER response for the discovered phy:
 - A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
 - B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table); and
 - C) the ATTACHED DEVICE TYPE field is set to zero (i.e., no device attached);
 - c) in the SMP DISCOVER response for the discovered phy:
 - A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
 - B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);
 - C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device or expander device); and
 - D) the ATTACHED SAS ADDRESS field contains the SAS address of the expander device being configured (i.e. a self-referencing address);
 - d) in the SMP DISCOVER response for the discovered phy:
 - A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
 - B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);
 - C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device or expander device); and
 - D) the ATTACHED SAS ADDRESS field contains the SAS address of a device directly attached to the expander device being configured;
- or
- e) in the SMP DISCOVER response for the discovered phy:
 - A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
 - B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);
 - C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device or expander device); and
 - D) the ATTACHED SAS ADDRESS field contains a SAS address that already exists in the expander route table.

If the discovered SAS address being included in the expander route table is for a device that is currently not attached (i.e., the ATTACHED DEVICE TYPE field is set to zero (i.e., no device attached) and the ROUTE ATTRIBUTE field is set to 0h (i.e., direct)), then the entry shall be inserted with the ROUTED SAS ADDRESS field set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit set to one (see 10.4.3.26).

If route table optimization is disabled, then all SAS addresses shall be qualified for insertion in the expander route table.

If the management application client supports route table optimization, then the management application client should provide a vendor-specific method for initiating a check of the resulting expander route tables. The check should be performed under the following situations:

- a) when an I_T nexus loss occurs for a destination port that is expected to be present;
- b) when a discover process has been completed;
- c) when another SMP initiator port is discovered in the SAS domain; or
- d) when a self-configuring expander device is discovered in the SAS domain.

If the management application client detects an inconsistency in the expander route tables when route table optimization is enabled (e.g., detects entries that appear to have been filled in by a discover process with route table optimization disabled), then the management application client shall report an error in a vendor-specific manner and shall disable route table optimization. The management application client should then re-initiate a discover process with route table optimization disabled.

4.8.4 Expander route index order

The expander route table shall be configured for each expander phy that has a table routing attribute.

If the phy is not attached to an expander device, every expander route entry for that phy shall be disabled (i.e., the ROUTED SAS ADDRESS field shall be set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit shall be set to one).

If the phy is attached to an expander device, the expander route table shall be configured for that phy as follows. For purposes of configuring the expander route table for that phy, the expander devices attached to the expander phy are assigned levels:

- 1) the expander device in which the expander route table is being configured is level 0;
- 2) the attached expander device is considered level 1;
- 3) devices attached to the level 1 expander device, except for the level 0 expander device, are considered level 2;
- 4) devices attached to level 2 expander devices, except for level 1 expander devices, are considered level 3; and
- 5) for each n greater than 3, devices attached to level n-1 expander devices, except for level n-2 expander devices, are considered level n.

The expander route table for each expander phy shall be configured starting from expander route index 0 by level (e.g., if there are three levels, then all level 1 entries first, then all level 2 entries, then all level 3 entries) up to the value of the EXPANDER ROUTE INDEXES field reported by the SMP REPORT GENERAL function (see 10.4.3.3).

Assuming the level 1 expander device has expander phys attached to N phys with qualified SAS addresses (see 4.8.3), the first N entries shall be used for those SAS addresses in expander phy order (i.e., the addresses attached to lower expander phy numbers first).

For each of the level 2 devices that:

- a) is an expander device attached to M phys with qualified SAS addresses; and
- b) is attached to an expander phy in the level 1 expander device with the table routing attribute,

the next M entries shall be used for the level 2 expander device's qualified SAS addresses in expander phy order (i.e., lower phy numbers first).

This process shall repeat for all levels of expander devices.

SAS addresses of devices attached beyond expander phys that are attached table-to-table shall not be included in the expander route table. The SAS address of the first expander device that is attached table-to-table shall be included and the SAS address of every device attached beyond that expander device shall not be included.

NOTE 15 - Not including those SAS addresses provides compatibility with management application clients compliant with previous versions of this standard. End devices in SAS domains containing externally configurable expander devices and table-to-table attachments may not be able to establish connections to each other. End devices in SAS domains containing only self-configuring expander devices compliant with this standard (i.e., supporting table-to-table attachments) are able to establish connections to any other end device.

Figure 48 shows an example of a route table that does not include SAS addresses beyond a table-to-table attachment.

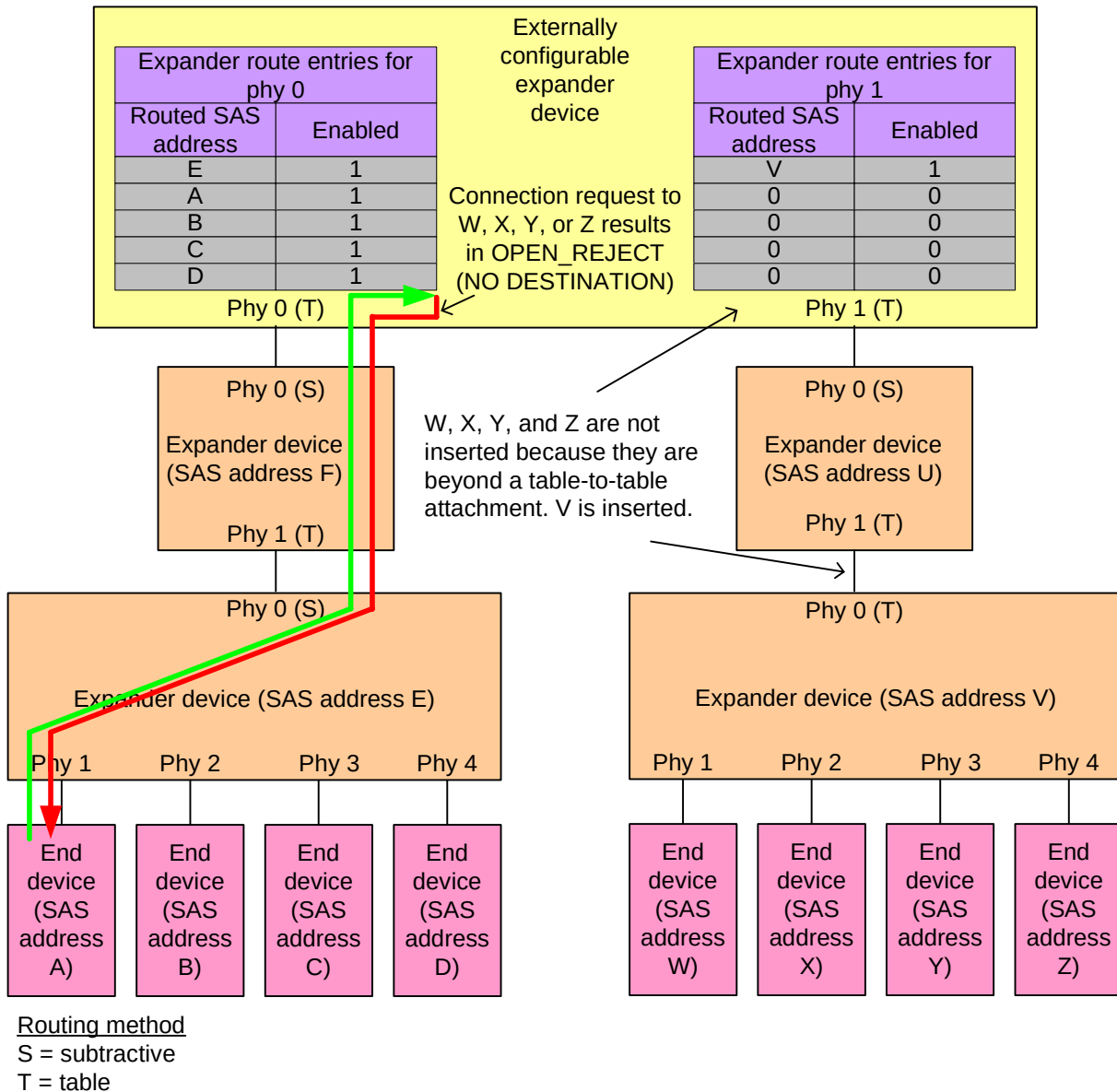


Figure 48 — Externally configurable expander device and table-to-table attachment

After the expander route table has been configured with entries for all levels of expander devices, all remaining expander route entries, if any, shall be disabled (i.e., the ROUTED SAS ADDRESS field shall be set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit shall be set to one). The management application client is not required to disable entries if the topology of expander devices has not changed.

Figure 49 shows a portion of a SAS domain, where phy A in expander device R is being configured.

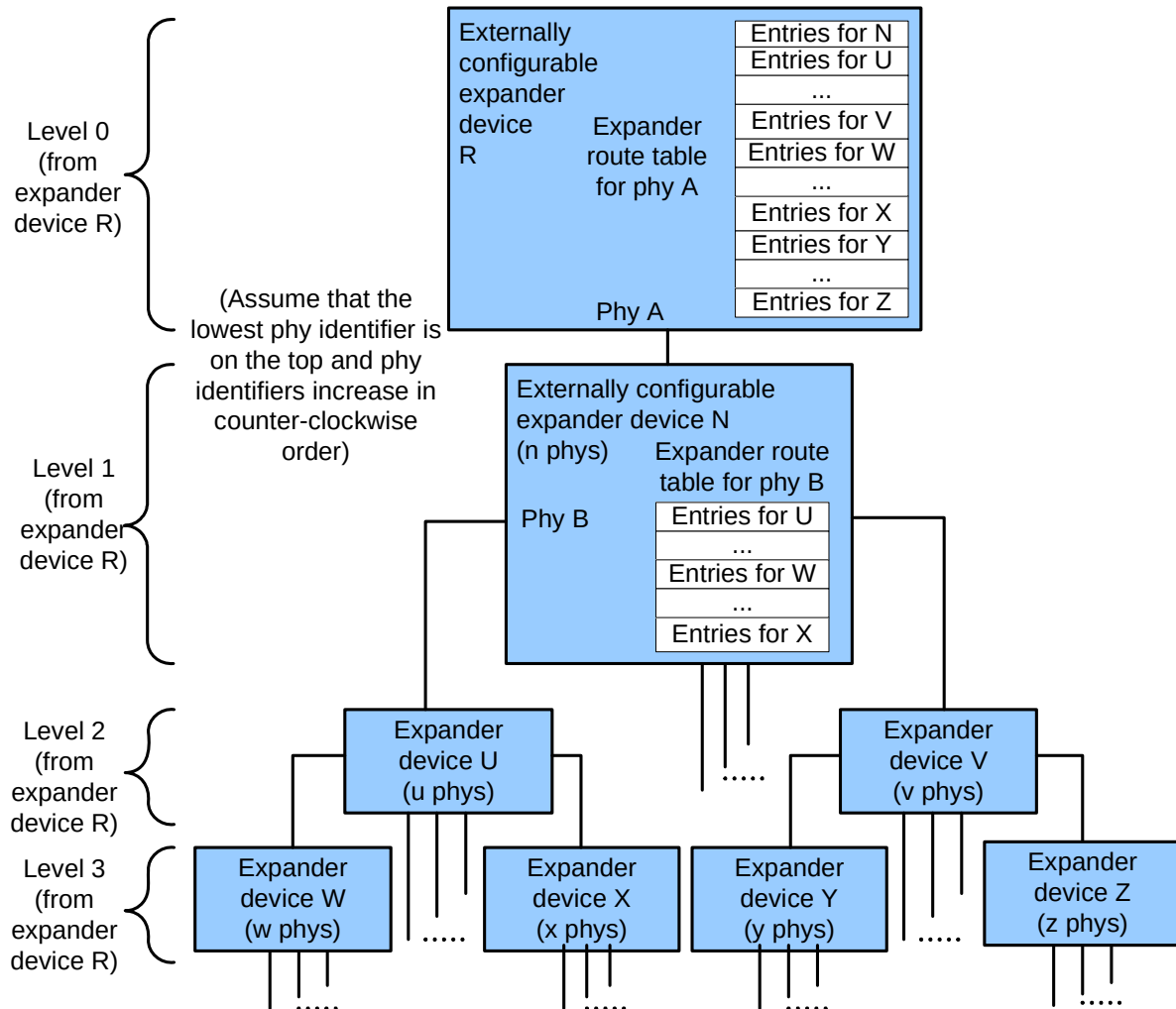


Figure 49 — Expander route index levels example

Table 22 shows how the expander route table is configured for externally configurable expander device R phy A in figure 49.

Table 22 — Expander route table levels for externally configurable expander device R phy A

Expander route index	Expander route entry contents
Level 1 (from device R) entries	
0 .. ($\leq n$ entries)	Qualified SAS addresses attached to expander device N
Level 2 (from device R) entries	
($\leq u$ entries)	Qualified SAS addresses attached to expander device U
...	...additional qualified SAS addresses for expander devices at level 2...
($\leq v$ entries)	Qualified SAS addresses attached to expander device V
Level 3 (from device R) entries	
($\leq w$ entries)	Qualified SAS addresses attached to expander device W
...	...additional qualified SAS addresses for expander devices at level 3...
($\leq x$ entries)	Qualified SAS addresses attached to expander device X
($\leq y$ entries)	Qualified SAS addresses attached to expander device Y
...	...additional qualified SAS addresses for expander devices at level 3...
($\leq z$ entries)	Qualified SAS addresses attached to expander device Z
Entries for additional levels	
...	...
Disabled entries	
...	...

Table 23 shows how the expander route table is configured for externally configurable expander device N phy B in figure 49.

Table 23 — Expander route table levels for externally configurable expander device N

Expander route index	Expander route entry contents
Level 1 (from device N) entries	
($\leq u$ entries)	Qualified SAS addresses attached to expander device U
Level 2 (from device N) entries	
($\leq w$ entries)	Qualified SAS addresses attached to expander device W
...	...additional qualified SAS addresses for expander devices at level 2...
($\leq x$ entries)	Qualified SAS addresses attached to expander device X
Entries for additional levels	
...	...
Disabled entries	
...	...

Figure 50 shows an example topology.

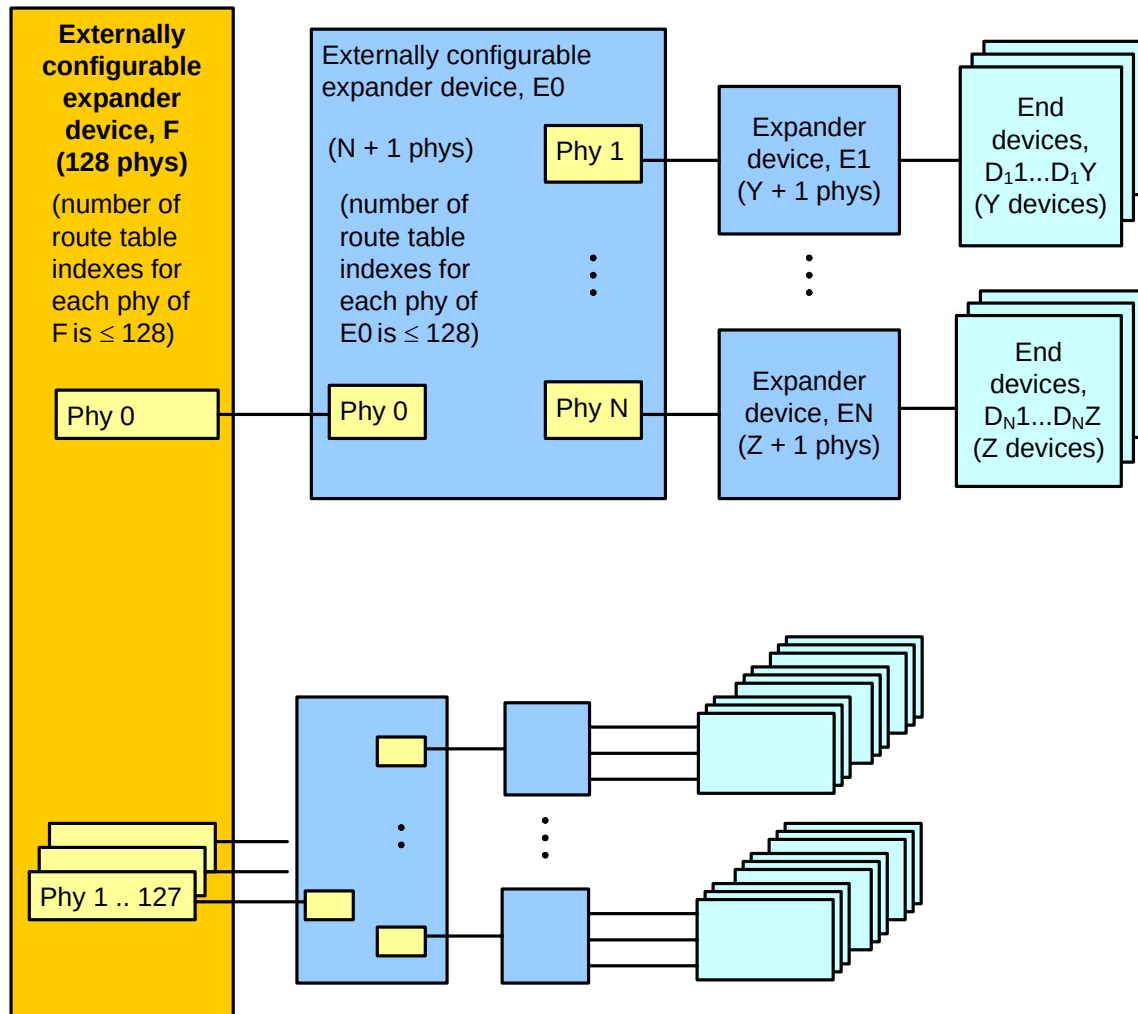


Figure 50 — Expander route index order example

Table 24 shows the expander route index order for externally configurable expander device E0 phy 1 in figure 50.

Table 24 — Expander route entries for externally configurable expander device E0 phy 1

Expander route index	Expander route entry contents
Level 1 entries	
0	SAS address (e.g., D_{11}) of the device attached to phy 1 of expander device E1
1	SAS address (e.g., D_{12}) of the device attached to phy 2 of expander device E1
...	...
$Y - 1$	SAS address (e.g., D_{1Y}) of the device attached to phy Y of expander device E1
Level 2 and beyond: no entries	
Disabled entries	

Table 25 shows the expander route index order for externally configurable expander device F phy 0 in figure 50.

Table 25 — Expander route entries for externally configurable expander device F phy 0

Expander route index	Expander route entry contents
Level 1 entries	
0	SAS address (e.g., E1) of the device attached to phy 1 of expander device E0
...	...additional qualified SAS addresses for expander device E0...
N - 1	SAS address (e.g., EN) of the device attached to phy N of expander device E0
Level 2 entries	
N	SAS address (e.g., D ₁ 1) of the device attached to phy 1 of expander device E1
...	...additional qualified SAS addresses for expander device E1...
	SAS address (e.g., D ₁ Y) of the device attached to phy Y of expander device E1
...	...additional qualified SAS addresses for expander devices E2 through EN-1...
	SAS address (e.g., D _N 1) of the device attached to phy 1 of expander device EN
...	...additional qualified SAS addresses for expander device EN...
	SAS address (e.g., D _N Z) of the device attached to phy Z of expander device EN
Level 3 and beyond: no entries since all devices attached to E1 through EN, except for E0, are end devices	
Disabled entries	

4.9 Zoning

4.9.1 Zoning overview

SAS zoning is implemented by a set of zoning expander devices with zoning enabled that define a zoned portion of a service delivery subsystem (ZPSDS). The zoning expander devices control whether a phy is permitted to participate in a connection to another phy.

Figure 51 shows an example of one ZPSDS in a SAS domain.

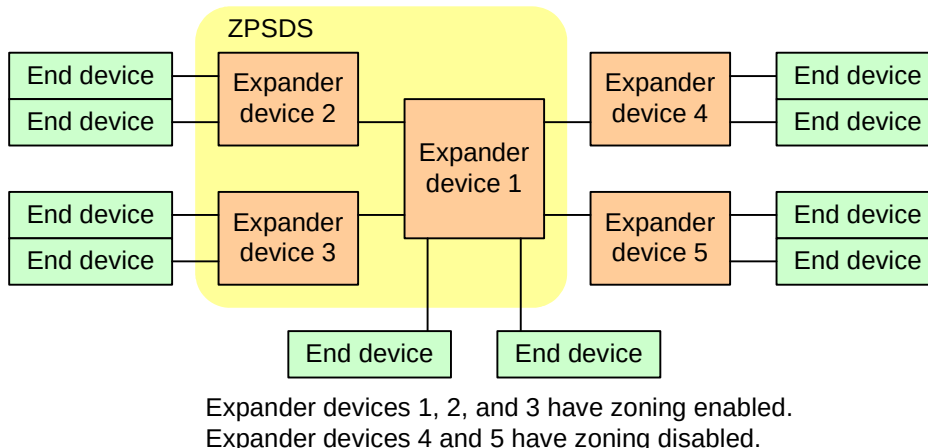
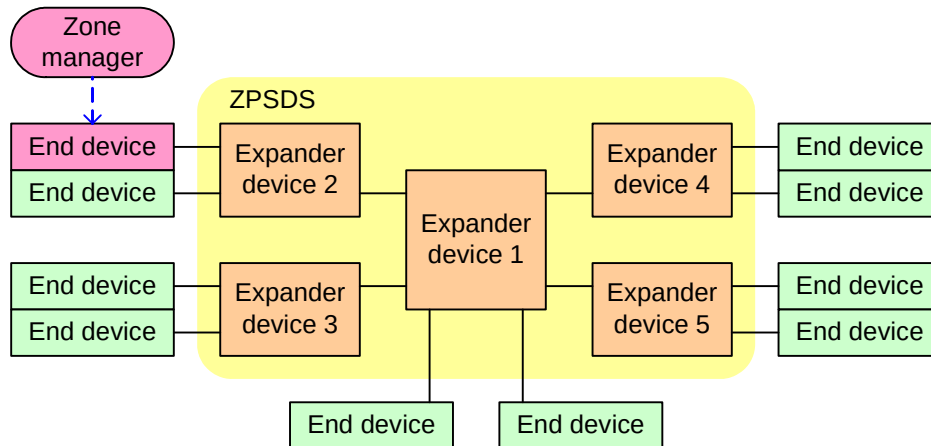


Figure 51 — One ZPSDS example

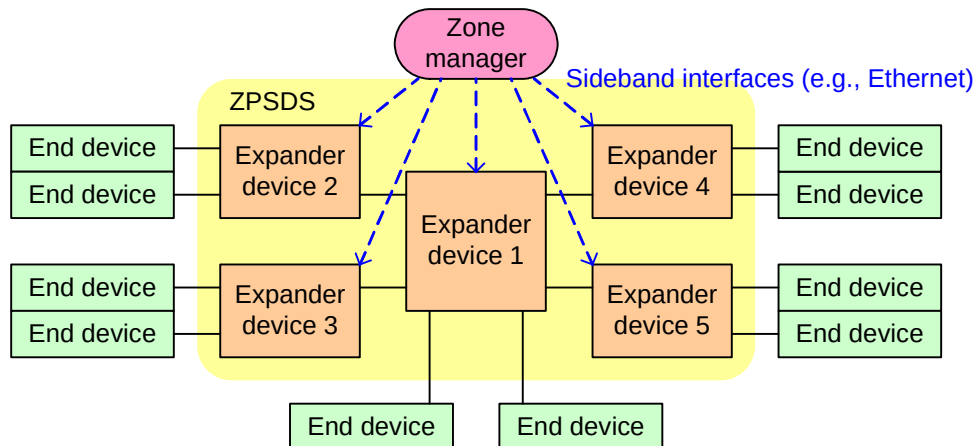
A ZPSDS has a zone manager responsible for its configuration. The zone manager may have access to an end device with a SAS port whose zone group has access to zone group 2, or may have access to one or more zoning expander devices through a sideband interface (e.g., Ethernet) outside the scope of this standard. The SAS address reported for a sideband zone manager is zero.

Figure 52 shows examples of zone manager locations in a SAS domain.

Zone manager attached to an end device with a SAS port whose zone group has access to zone group 2



Zone manager attached directly to the expander devices in the ZPSDS



Zone manager attached directly to one expander device in the ZPSDS

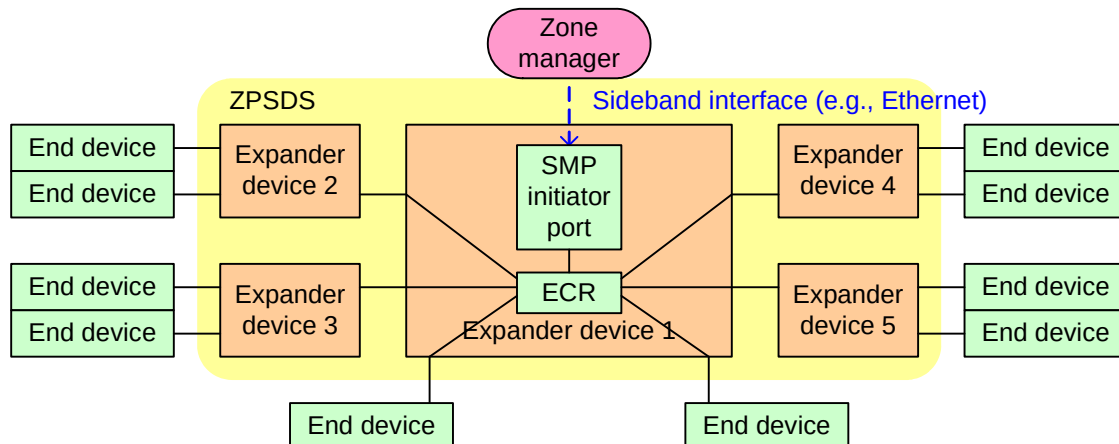


Figure 52 — Zone manager location examples

There may be any number of non-overlapping ZPSDSes in a service delivery subsystem, particularly as a SAS domain is being reconfigured (e.g., as a user is attaching enclosures together). A SAS domain with more than one ZPSDS should be transitory. A ZPSDS may encompass some or all of a service delivery subsystem.

Figure 53 shows an example of three ZPSDSes in a SAS domain.

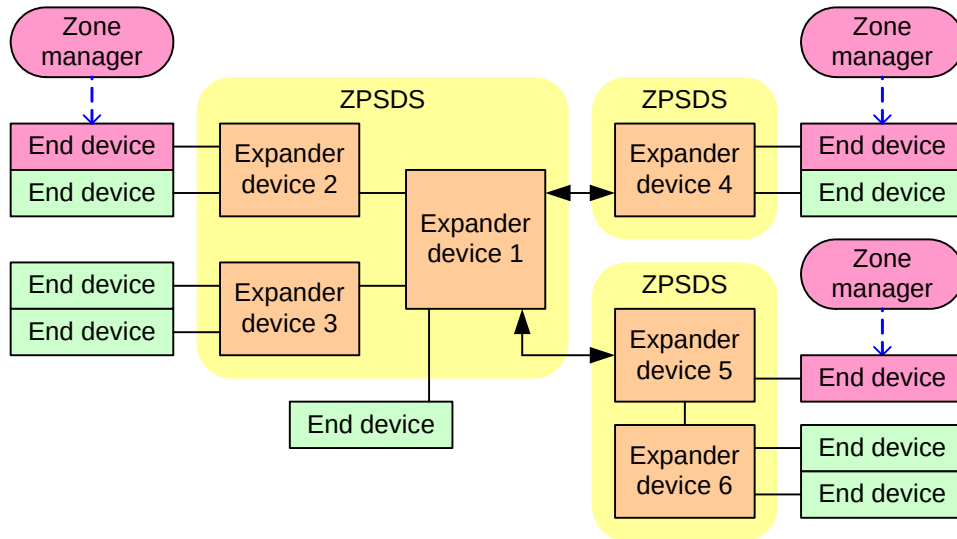


Figure 53 — Three ZPSDSes example

The zone manager assigns zone groups (see 4.9.3.2) to all zoning expander phys inside the ZPSDS. There are 128 or 256 zone groups numbered 0 through 127 or 255. All phys in a wide port shall be assigned to the same zone group. Zone groups are assigned to zoning expander phys as part of the zone phy information (see 4.9.3.1) and are stored along with SAS addresses in the zoning expander route table (see 4.9.3.4). The zone groups assigned in one ZPSDS have no relationship to the zone groups assigned in another ZPSDS.

The zone manager shall assign each zoning expander phy attached to another zoning expander phy inside a ZPSDS to zone group 1. The zone manager shall assign each zoning expander phy on the boundary of the ZPSDS (i.e., with the `INSIDE ZPSDS` bit set to zero) to a zone group. All phys in the SAS domain beyond that boundary zoning expander phy are considered to be in the same zone group as that zoning expander phy.

Each zoning expander device contains a zone permission table (see 4.9.3.3) that controls whether a connection is allowed between phys based on their zone groups. As defined in 4.9.3.5, a requested connection shall only be established if the zone permission table indicates that access between the zone group of the source port and the zone group of the target port is allowed.

The zoning expander route table (see 4.9.3.4) is an extended version of the expander route table (see 4.6.7.3) that also includes the zone group of each SAS address.

A zoning expander device may support physical presence detection and/or a zone manager password to allow management access. The zone manager password is 32 bytes long and is specified in table 26.

Table 26 — Zone manager password

Code	Name	Description
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000h	ZERO	Well-known value that provides access to any zone manager that presents it
FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFFh	DISABLED	Well-known value that does not provide access to any zone manager (e.g., zone manager password usage is disabled)
All others		Random value, providing access only to a zone manager that presents the correct value

The expander device:

- a) shall maintain a current value;
- b) shall maintain a shadow value;
- c) may maintain a saved value; and
- d) shall have a default value,

for each the following settings:

- a) zoning enabled;
- b) the zone permission table;
- c) zone phy information; and/or
- d) the zone manager password.

Support or lack of support for saved values for one setting does not imply support or lack of support for saved values for any other setting (e.g., the expander device may maintain a saved value for zoning enabled but not for the zone permission table).

For each setting, after power on or expander reduced functionality, the expander device shall set the current value to the saved value, if any, or the default value, if there is no saved value.

Table 27 describes the reasons for which a zoning expander device accepts the SMP ZONE LOCK function after power on.

Table 27 — Zoning expander device zoning configuration after power on

Value after power on			Reason(s) that the SMP ZONE LOCK function is accepted		
Zoning enabled	Zone permission table saved	Zone manager password	Physical presence asserted	SMP initiator port has access to zone group 2	ZONE MANAGER PASSWORD field in the ZONE LOCK request set to the zone manager password
no	yes or no	ZERO	yes	no	yes
		DISABLED			no
		All others			yes
yes	no	ZERO		no	yes
		DISABLED			no
		All others			yes
	yes	ZERO		yes	yes
		DISABLED			no
		All others			yes

4.9.2 Zoning expander device requirements

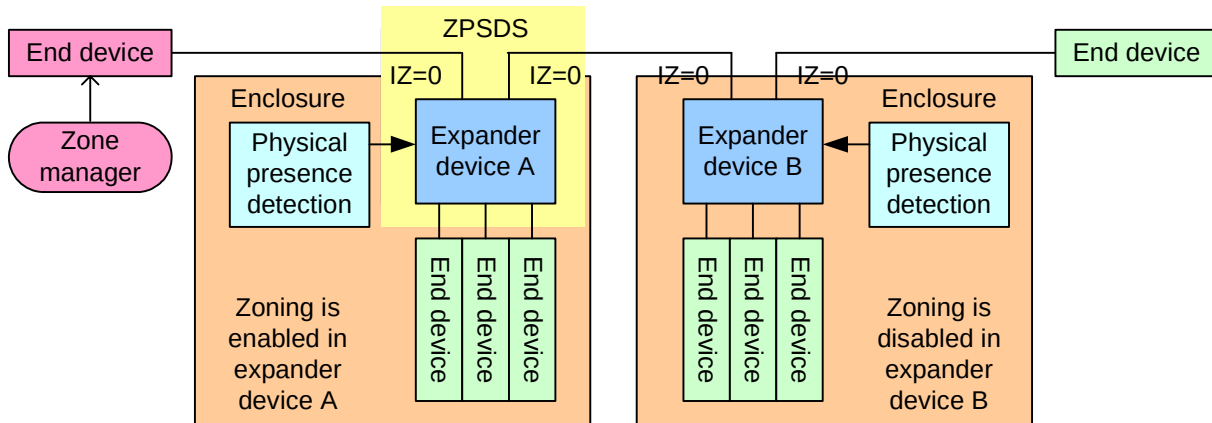
In addition to the requirements for expander devices described in 4.6, a zoning expander device shall:

- contain a zoning expander route table (see 4.9.3.4);
- contain current and shadow zone permission tables that supports 128 or 256 zone groups (see 4.9.3.3);
- contain current and shadow zone phy information for each phy;
- if zoning is enabled, allow or deny connection requests based on the active zone permission table (see 4.9.3.5);
- set the ZONING SUPPORTED bit to one in its SMP REPORT GENERAL response (see 10.4.3.3);
- support the ZONING ENABLED bit in the SMP REPORT GENERAL response;
- support the zone lock inactivity timer;
- be self-configuring;
- contain an SMP initiator port (see 4.6.1); and
- support zoning-related SMP functions.

A zoning expander device may include physical presence detection to allow locking (see 4.9.6.2). The definition of physical presence detection is vendor-specific.

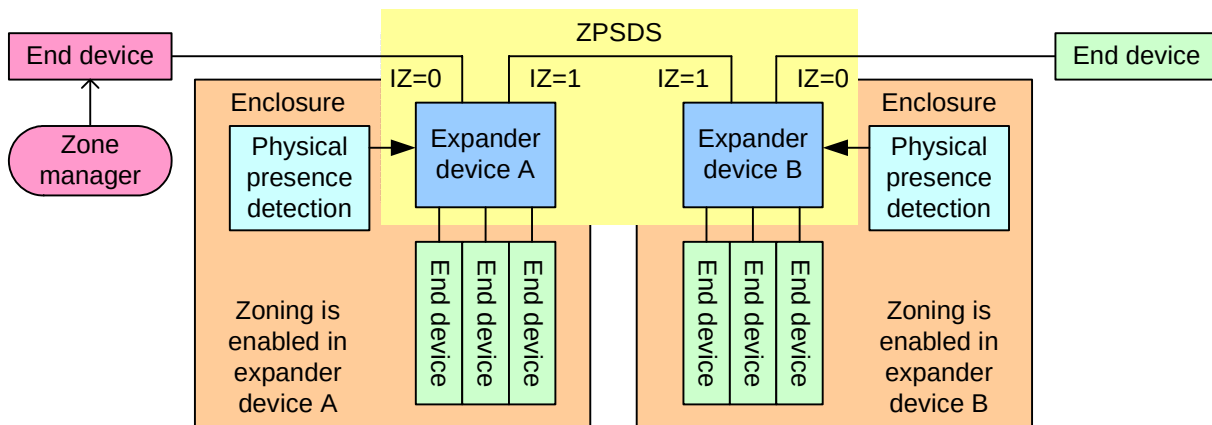
Figure 54 shows an example of two enclosures with physical presence detection where zoning is enabled in the expander device in the left enclosure, but is not enabled in the expander devices in the right enclosure. The zone manager is able to configure zoning in zoning expander device A because the zone group of its SMP initiator port has access to zone group 2. However, it is not able to enable or configure zoning in

expander device B unless physical presence is asserted or it presents the correct zone manager password for that expander device.



The ZPSDS is extended as shown below after the zone manager on the left:

- 1) uses the ZONE LOCK function to lock expander devices A and B with physical presence asserted in expander B or providing the correct zone manager password;
- 2) uses the CONFIGURE ZONE PHY INFORMATION function and the CONFIGURE ZONE PERMISSION TABLE function to configure each expander device, including setting the INSIDE ZPSDS bit to one in each expander phy that is going to be inside the ZPSDS;
- 3) uses the ENABLE DISABLE ZONING function to enable zoning in expander device B;
- 4) uses the ZONE ACTIVATE function to activate the changes; and
- 5) uses the ZONE UNLOCK function to unlock expander devices A and B.

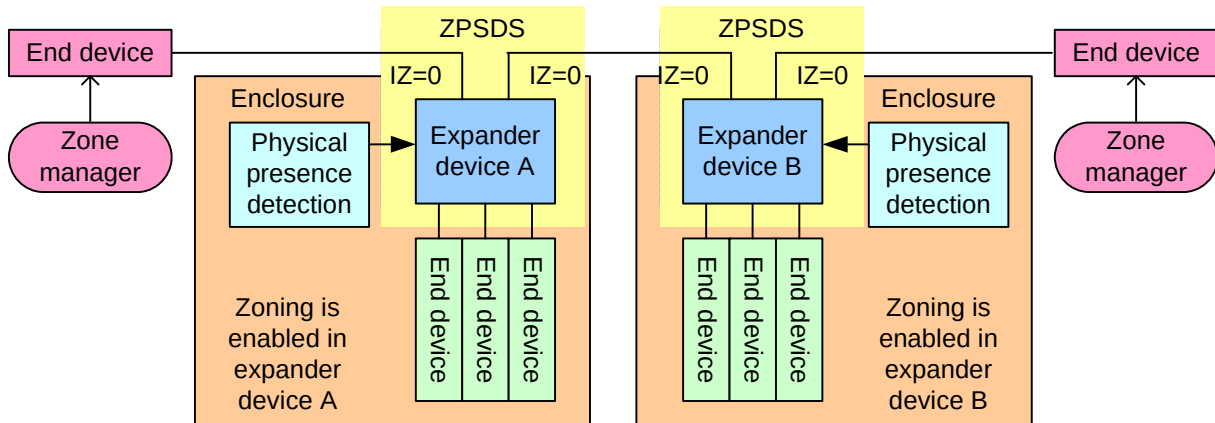


IZ = INSIDE ZPSDS bit in the zone phy information

Figure 54 — Extending a ZPSDS example

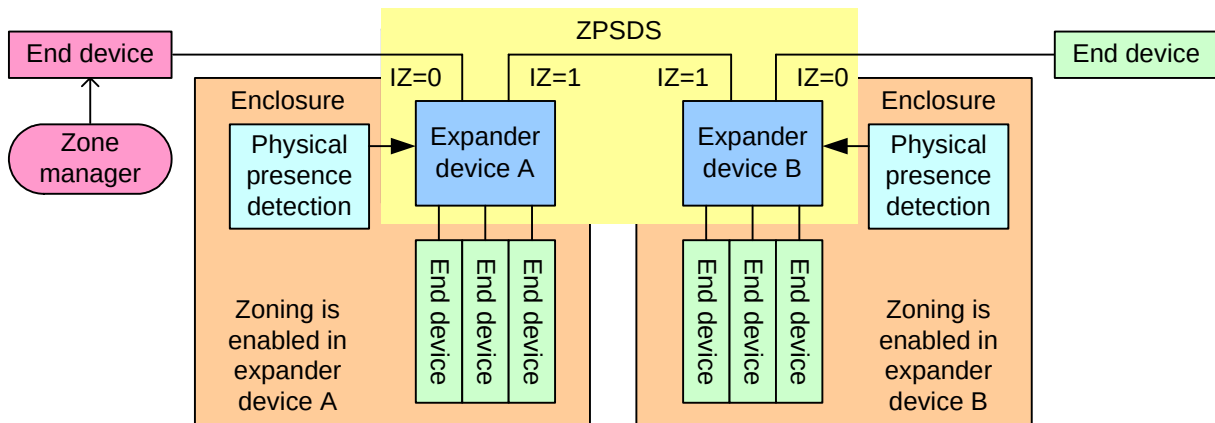
Figure 55 shows an example of two enclosures with physical presence detection where zoning is enabled in both the expander devices. The zone manager is able to configure zoning in zoning expander device A because the zone group of its SMP initiator port has access to zone group 2. However, it is not able to

configure zoning in expander device B unless physical presence is asserted or it presents the correct zone manager password for that expander device.



The ZPSDS is extended as shown below after the zone manager on the left:

- 1) uses the ZONE LOCK function to lock expander devices A and B with physical presence asserted in expander B, since it may not have access to zone group 2 in the right ZPSDS;
- 2) uses the CONFIGURE ZONE PHY INFORMATION function and the CONFIGURE ZONE PERMISSION TABLE function to configure each expander device, including setting the INSIDE ZPSDS bit to one in each expander phy that is going to be inside the ZPSDS;
- 3) uses the ENABLE DISABLE ZONING function to enable zoning in expander device B;
- 4) uses the ZONE ACTIVATE function to activate the changes; and
- 5) uses the ZONE UNLOCK function to unlock expander devices A and B.



IZ = INSIDE ZPSDS bit in the zone phy information

Figure 55 — Overtaking a ZPSDS example

4.9.3 Zoning operation

4.9.3.1 Zone phy information

Each phy of a zoning expander device shall support the following zone phy information fields:

- a) INSIDE ZPSDS bit;
- b) REQUESTED INSIDE ZPSDS bit;
- c) INSIDE ZPSDS PERSISTENT bit;
- d) ZONE GROUP PERSISTENT bit; and
- e) ZONE GROUP field.

Each phy of a zoning expander device shall support the zone phy information fields defined in table 28.

Table 28 — Zone phy information

Field	Description	Recommended default
INSIDE ZPSDS bit	Indicates if the phy is inside or on the boundary of a ZPSDS	N/A ^a
REQUESTED INSIDE ZPSDS bit	Used to establish the boundary of the ZPSDS	0
INSIDE ZPSDS PERSISTENT bit	Used to determine the value of the INSIDE ZPSDS bit after a link reset sequence	0
ZONE GROUP PERSISTENT bit	Used to determine the zone group of the phy after a link reset sequence when the INSIDE ZPSDS bit is set to zero	0
ZONE GROUP field	The zone group to which the phy belongs	00h
^a The INSIDE ZPSDS bit is determined from the values exchanged during the link reset sequence.		

Table 29 lists the usage of the current values of the zone phy information fields.

Table 29 — Zone phy information usage

Field	Transmitted in IDENTIFY address frame ^a	Indicated in DISCOVER function and DISCOVER LIST function ^b	Attached value indicated in DISCOVER function ^c	Programmable with the CONFIGURE ZONE PHY INFORMATION function ^d	Changeable by the expander device after a link reset sequence ^e
INSIDE ZPSDS bit	No	Yes	No	No	Yes
REQUESTED INSIDE ZPSDS bit	Yes	Yes	Yes	Yes	Yes
INSIDE ZPSDS PERSISTENT bit	Yes	Yes	Yes	Yes	No
ZONE GROUP PERSISTENT bit	No	Yes	No	Yes	No
ZONE GROUP field	No	Yes	No	Yes	Yes
^a Defined in the IDENTIFY address frame (see 7.8.2). ^b Defined in the DISCOVER response (see 10.4.3.9) and the DISCOVER LIST response SHORT FORMAT descriptor (see 10.4.3.14.4). ^c Defined in the DISCOVER response (see 10.4.3.9). ^d Defined in the zone phy configuration descriptor (see 10.4.3.24.3). The saved values are also programmable with this function. ^e See table 33 in 4.9.4.					

All phys in an expander port shall have the same zone phy information.

The expander device shall preserve the zone phy information while zoning is disabled and there is no power loss and no expander device reduced functionality (see 4.6.8).

The INSIDE ZPSDS bit indicates if the phy is inside or on the boundary of a ZPSDS. An INSIDE ZPSDS bit set to zero indicates that the phy is attached to an end device, an expander device that does not support zoning, or a zoning expander device with zoning disabled, or a zoning expander device with zoning enabled that is outside the ZPSDS (i.e., is in another ZPSDS). An INSIDE ZPSDS bit set to one indicates that the phy is

attached to a zoning expander device with zoning enabled and is thus inside a ZPSDS. The INSIDE ZPSDS bit only changes following a link reset sequence (see 4.9.4), based on:

- a) the REQUESTED INSIDE ZPSDS bit;
- b) the REQUESTED INSIDE ZPSDS bit received in the incoming IDENTIFY address frame (see 7.8.2);
- c) the INSIDE ZPSDS PERSISTENT bit; and
- d) the INSIDE ZPSDS PERSISTENT bit received in the incoming IDENTIFY address frame.

The REQUESTED INSIDE ZPSDS bit is used to establish the boundary of the ZPSDS. The REQUESTED INSIDE ZPSDS bit is used to determine the values of other zone phy information fields after a link reset sequence (see 4.9.4).

The INSIDE ZPSDS PERSISTENT bit is used to determine the value of the INSIDE ZPSDS bit after a link reset sequence (see 4.9.4).

The ZONE GROUP field contains the zone group to which the phy belongs. The zone group of the SMP initiator port and SMP target port in a zoning expander device shall be 01h. 4.9.3.2 defines more about zone groups.

The ZONE GROUP PERSISTENT bit is used to determine the method of determining the zone group of the phy after a link reset sequence when the INSIDE ZPSDS bit is set to zero (see 4.9.4).

4.9.3.2 Zone groups

The zone groups are defined in table 30.

Table 30 — Zone groups

Zone group	Configurable in zone permission table ^a	Description
0	No	Phys in zone group 0 have access to phys in zone group 1 and do not have access to phys in other zone groups.
1	No	Phys in zone group 1 have access to phys in all zone groups.
2	Yes	Phys in zone group 2 have access to phys in the zone groups indicated by the zone permission table. A management device server in a zoning expander device with zoning enabled only allows management application clients using phys in zone groups with access to zone group 2 to perform the following SMP functions: a) CONFIGURE GENERAL (see 10.4.3.17); and b) ZONE LOCK (see 10.4.3.20) and SMP zone configuration functions (see 4.9.6.1) performed while the zoning expander device is locked. A management device server in a zoning expander device with zoning enabled only allows management application clients to perform certain SMP phy-based control and configuration functions (e.g., PHY CONTROL, PHY TEST FUNCTION, and CONFIGURE PHY EVENT) if the zone group of the management application client's phy has access to zone group 2 or the zone group of the specified phy.
3	Yes	Phys in zone group 3 have access to phys in the zone groups indicated by the zone permission table. A management device server in a zoning expander device with zoning enabled only allows management application clients using a phy in a zone group with access to zone group 3 to perform certain SMP zoning-related functions (i.e., ZONED BROADCAST (see 10.4.3.19)).
4 to 7	Reserved	
8 to 255	Yes	Phys in zone groups 8 through 255 have access to phys in the zone groups indicated by the zone permission table.
^a A zone group defined as configurable is able to be changed with the SMP CONFIGURE ZONE PERMISSION TABLE function (see 10.4.3.25).		

4.9.3.3 Zone permission table

The zone permission table specifies access permission between zone groups. If a bit in the zone permission table is set to one then connection requests shall be permitted between phys in the zone groups. If a bit in the zone permission table is set to zero then connection requests between phys in the zone groups shall be rejected with OPEN_REJECT (ZONE VIOLATION) or OPEN_REJECT (RETRY) as described in 4.9.3.5.

The zone permission table structure is shown in table 31.

Table 31 — Zone permission table

Destination zone group (i.e., d)	Source zone group (i.e., s) ^{a b}				
	0	1	2 to 3	4 to 7	8 to (z-1) ^c
0	0	1	0	0	0
1	1	1	1	1	1
2 to 3	0	1	ZP[s = 2 to 3, d = 2 to 3]	Reserved	ZP[s = 8 to (z-1), d = 2 to 3]
4 to 7	0	1	Reserved	Reserved	Reserved
8 to (z-1) ^c	0	1	ZP[s = 2 to 3, d = 8 to (z-1)]	Reserved	ZP[s = 8 to (z-1), d = 8 to (z-1)]
^a Shading identifies configurable zone groups. ^b All reserved ZP bits shall be set to zero (e.g., bits ZP[4 to 7, 4 to (z-1)] are set to zero). ^c The number of zone groups (i.e., z) is reported in NUMBER OF ZONE GROUPS field in the REPORT GENERAL response (see 10.4.3.3).					

A ZP[s, d] bit set to one specifies that the source zone group (i.e., s) has permission to access the destination zone group (i.e., d). A ZP[s, d] bit set to zero specifies that the source zone group (i.e., s) does not have permission to access the destination zone group (i.e., d).

If ZP[s, d] is set to a value, ZP[d,s] shall be set to the same value.

The zoning expander device shall preserve the zone permission table while zoning is disabled and may or may not preserve the zone permission table through power loss. If the zoning expander device preserves that zoning is enabled and does not preserve the zone permission table, it shall set the zone permission table to grant minimal permissions on power on as specified in table 32.

Table 32 — Zone permission table granting minimal permissions

Destination zone group (i.e., d)	Source zone group (i.e., s) ^{a b}				
	0	1	2 to 3	4 to 7	8 to (z-1) ^c
0	0	1	0	0	0
1	1	1	1	1	1
2 to 3	0	1	0	Reserved	0
4 to 7	0	1	Reserved	Reserved	Reserved
8 to (z-1) ^c	0	1	0	Reserved	0
^a Shading identifies configurable zone groups. ^b All reserved ZP bits shall be set to zero (e.g., bits ZP[4 to 7, 4 to (z-1)] are set to zero). ^c The number of zone groups (i.e., z) is reported in NUMBER OF ZONE GROUPS field in the REPORT GENERAL response (see 10.4.3.3).					

If a zone manager enables zoning on zoning capable expander devices that report different values in the NUMBER OF ZONE GROUPS field in the REPORT GENERAL response (see 10.4.3.3)(e.g., some support 128 and others support 256), then the zone manager shall:

- configure all zoning enabled expander devices contained within the ZPSDS to use the highest commonly supported number of zone groups (e.g., 128);

- b) configure the zone phy information in all the zoning expander devices to set each phy to a zone group less than the highest commonly supported number of zone groups; and
- c) configure the zone permission table in all the zoning expander devices to set each entry to zero that is higher than the highest commonly supported number of zone groups.

4.9.3.4 Zoning expander route table

A zoning expander route table is an expander-based expander route table (see 4.6.7.3) that is able to hold the zone group of each routed SAS address.

Figure 56 shows a representation of the zoning expander route table.

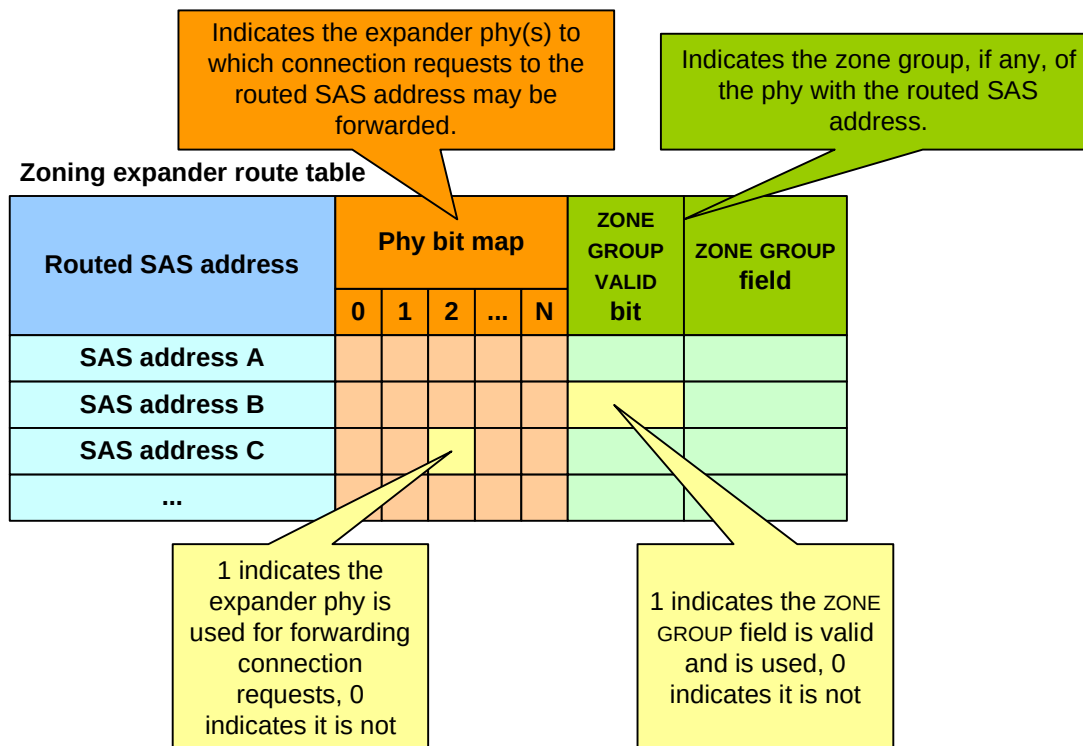


Figure 56 — Zoning expander route table

The zoning expander route table:

- a) shall include discovered SAS addresses for each expander phy with the ROUTING ATTRIBUTE field set to 2h (i.e., table) in the DISCOVER response; and
- b) may include discovered SAS addresses for expander phys with the ROUTING ATTRIBUTE field set to 1h (i.e., subtractive) in the DISCOVER response as long as they do not prevent inclusion of SAS addresses for expander phys with the ROUTING ATTRIBUTE field set to 2h (i.e., table). The determination of which such SAS addresses to include is vendor-specific.

The total number of routed SAS addresses shall not exceed the value indicated in the MAXIMUM NUMBER OF ROUTED SAS ADDRESSES field in the REPORT GENERAL response.

4.9.3.5 Source zone group and destination zone group determination

When a zoning expander device with zoning enabled receives an OPEN address frame (see 7.8.3):

- a) the zone group of the source port (i.e., s) is identified as defined in table 33; and
- b) the zone group of the destination port (i.e., d) is identified as defined in table 34.

If the ZP[s, d] bit is set to one then access between the phys shall be permitted and the zoning expander device shall perform the ECM arbitration procedure. If the ZP[s, d] bit is set to zero then access between the

phys is not permitted and the zoning expander device shall transmit an OPEN_REJECT in response to the connection request as follows:

- a) OPEN_REJECT (RETRY) if the zoning expander device is locked; and
- b) OPEN_REJECT (ZONE VIOLATION) if the zoning expander device is unlocked.

Zoning expander devices with zoning enabled shall follow the rules in table 33 to determine the zone group of the source port.

Table 33 — Source zone group determination

INSIDE ZPSDS bit of the expander phy that received the OPEN address frame	Source zone group
0	Zone group of the receiving expander phy
1	Source zone group specified by the SOURCE ZONE GROUP field in the received OPEN address frame

Zoning expander devices with zoning enabled shall follow the rules in table 34 to determine the zone group of the destination port.

Table 34 — Destination zone group determination

Routing method of the destination expander phy	Destination zone group
Direct	Zone group of the destination expander phy
Subtractive	If the ZONE GROUP VALID bit is set to one, then the zone group stored in the zoning expander route table for the destination SAS address. If the ZONE GROUP VALID bit is set to zero, then the zone group of the destination expander phy (i.e., the subtractive routing phy).
Table	If the ZONE GROUP VALID bit is set to one, then the zone group stored in the zoning expander route table for the destination SAS address. If the ZONE GROUP VALID bit is set to zero, then the zone group of the destination expander phy (i.e., the table routing phy).

Figure 57 shows an example of a ZPSDS.

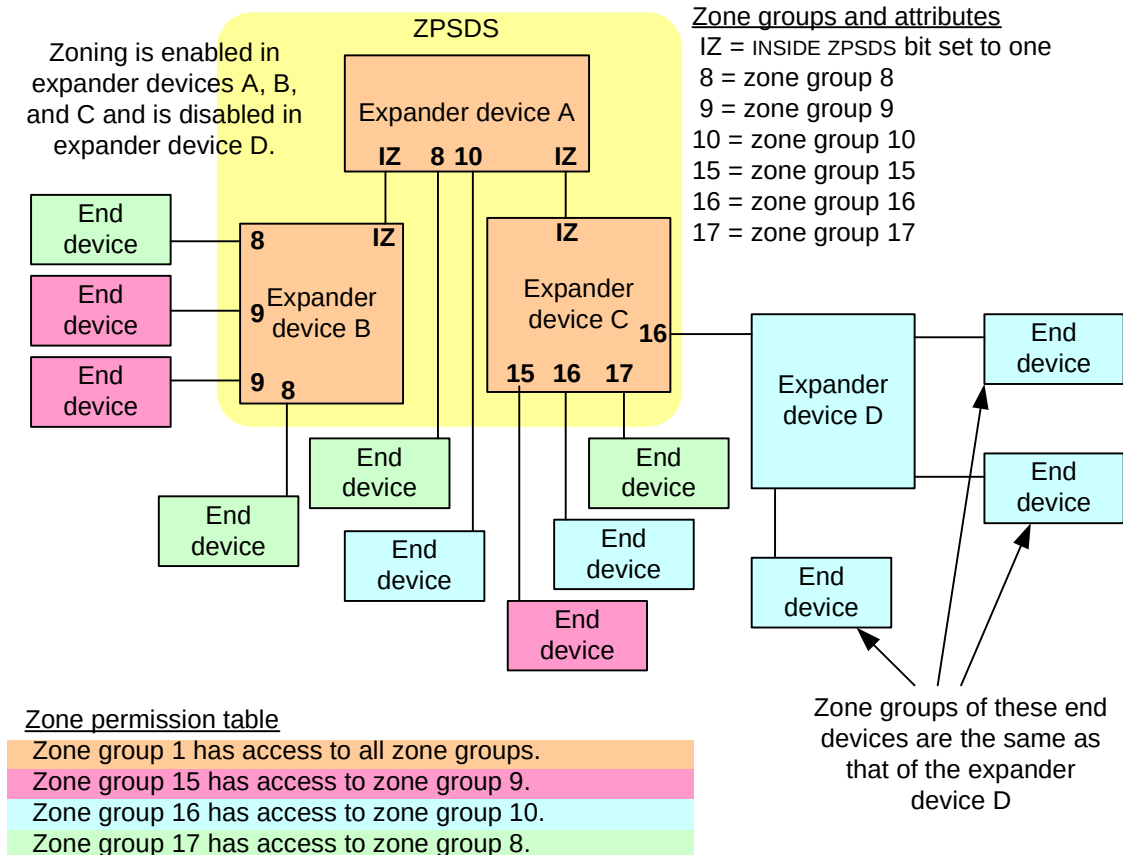


Figure 57 — Example of a ZPSDS

4.9.4 Zone phy information and link reset sequences

At the completion of a link reset sequence (see 4.4), if a SATA device is attached to an expander phy, the zoning expander device with zoning enabled shall set the INSIDE ZPSDS bit to zero for that expander phy.

At the completion of a link reset sequence, if a SATA device is not attached to an expander phy, the zoning expander device with zoning enabled shall update the current zone phy information fields as defined in table 35 based on:

- the REQUESTED INSIDE ZPSDS bit and the INSIDE ZPSDS PERSISTENT bit in the zone phy information (i.e., the bits transmitted in the outgoing IDENTIFY address frame (see 7.8.2)); and
- the REQUESTED INSIDE ZPSDS bit and INSIDE ZPSDS PERSISTENT bit received in the incoming IDENTIFY address frame.

Table 35 — Zone phy information fields after a link reset sequence

REQUESTED INSIDE ZPSDS bit		INSIDE ZPSDS PERSISTENT bit		Zone phy information field changes
Transmitted	Received	Transmitted	Received	
0	0 or 1	0 or 1	0 or 1	The zoning expander device shall set the INSIDE ZPSDS bit to zero.
1	0			
1	1	0	0	If the SAS address received in the IDENTIFY address frame during the identification sequence is different from the SAS address prior to the completion of the link reset sequence, then the zoning expander device shall: a) set the REQUESTED INSIDE ZPSDS bit to zero; and b) set the INSIDE ZPSDS bit to zero.
		0	1	
		1	0	If the SAS address received in the IDENTIFY address frame during the identification sequence is the same as the SAS address prior to the completion of the link reset sequence, then the zoning expander device shall: a) set the INSIDE ZPSDS bit to one; and b) set the ZONE GROUP field to 01h.
		1	1	The zoning expander device shall: a) set the INSIDE ZPSDS bit to one; and b) set the ZONE GROUP field to 01h.

If the ZONE GROUP PERSISTENT bit is set to one, then a link reset sequence (see 4.4) shall not cause the zone group of an expander phy to change unless the INSIDE ZPSDS bit changes from zero to one as specified in table 35. If the ZONE GROUP PERSISTENT bit is set to zero, then table 36 specifies events based on the initial condition of an expander phy that shall cause a zoning expander device with zoning enabled to change the ZONE GROUP field of the expander phy to its reset value (i.e., the saved value, if any, or the default value (e.g., 00h) if there is no saved value).

Table 36 — Events that cause the ZONE GROUP field to be reset when the ZONE GROUP PERSISTENT bit set to zero

Initial condition	Event after the initial condition is established
Completed link reset sequence with a SAS device attached	A subsequent link reset sequence completes and: - the SAS address received in the IDENTIFY address frame (see 7.8.2) during the identification sequence is different from the SAS address prior to the completion of the link reset sequence; or - a SATA device is attached.
Completed link reset sequence with a SATA device attached	Either: - A subsequent link reset sequence completes and: - a hot-plug timeout (see 6.7.5) occurred between the time of the initial condition and the time the link reset sequence completed; - the zoning expander device has detected the possibility that a new SATA device has been inserted. The method of detection is outside the scope of this standard (e.g., an enclosure services process reports a change in the ELEMENT STATUS CODE field in the Device or Array Device element (see SES-2), or a change in the WORLD WIDE NAME field in the attached SATA device's IDENTIFY (PACKET) DEVICE data (see ATA8-ACS)); or - a SAS phy or expander phy is attached; - or - The expander phy is disabled with the SMP PHY CONTROL function (see 10.4.3.27) DISABLE phy operation.

4.9.5 Broadcast processing in a zoning expander device with zoning enabled

The BPP determines the source zone group(s) of the Broadcast as follows:

- if the BPP receives a Broadcast Event Notify message from an expander logical phy (i.e., a zoning expander logical phy received a BROADCAST), the Broadcast has a single source zone group set to the zone group of that expander phy; and
- if the BPP receives a message from the management device server indicating that it received an SMP ZONED BROADCAST request from an SMP initiator port that has access to zone group 3, the Broadcast has each of the source zone groups specified in the SMP ZONED BROADCAST request.

The BPP forwards the Broadcast to each expander port other than the one on which the Broadcast was received (i.e., the expander port that received the BROADCAST or SMP ZONED BROADCAST request) if:

- the broadcast is not a Broadcast (Zone Activate) and any of the source zone groups have access to the zone group of the expander port;
- the broadcast is a Broadcast (Zone Activate), the BPP is in a locked zoning expander device, the INSIDE ZPSDS bit is set to one, and the source zone group has access to zone group 2; or
- the broadcast is a Broadcast (Zone Activate), the BPP is not in a locked zoning expander device, and any of the source zone groups have access to the zone group of the expander port.

To forward a Broadcast to an expander port:

- if the expander port's INSIDE ZPSDS bit is set to one, the BPP shall request that the SMP initiator port establish a connection on at least one phy in the expander port to the SMP target port of the attached expander device and transmit an SMP ZONED BROADCAST request specifying the source zone group(s); and

- b) if the expander port's INSIDE ZPSDS bit is set to zero, the BPP shall send a Transmit Broadcast message to at least one phy in the expander port, causing it to transmit a BROADCAST.

4.9.6 Zone configuration

4.9.6.1 Zone configuration overview

Zoning expander devices implement a lock to coordinate zoning configuration by zone manager(s).

There are four steps in the zone configuration process:

- 1) lock (see 4.9.6.2);
- 2) load (see 4.9.6.3);
- 3) activate (see 4.9.6.4); and
- 4) unlock (see 4.9.6.5).

The management device server in a zoning expander device only accepts SMP zone configuration function requests, SMP ZONE ACTIVATE requests, and SMP ZONE UNLOCK requests when it is locked, and only accepts SMP zone configuration function requests from the zone manager that locked the zoning expander device (i.e., the active zone manager). SMP zone configuration functions change zoning expander shadow values. When changes are complete, the zone manager activates the changes and the zoning expander device sets the zoning expander active values equal to the zoning expander shadow values. The zone manager then unlocks the zoning expander devices.

For a ZPSDS to function correctly, all zoning expander devices are required to have identical values in their zone permission tables. To change zone permission tables, a zone manager device locks all zoning expander devices in a ZPSDS.

To change zone phy information, a zone manager locks only the zoning expander devices containing the phys to be changed.

When a zoning expander device with zoning disabled is being added to a ZPSDS (see figure 54 in 4.9.1) or two or more ZPSDSes are being merged (see figure 55 in 4.9.1), the zone manager locks all of the zoning expander devices that are to be included in the final ZPSDS. The zone manager configures the zone phy information in each zoning expander device (e.g., sets the REQUESTED INSIDE ZPSDS bit to one for phys inside the final ZPSDS) and configures all of the zone permission tables to be identical.

If the zone lock inactivity timer expires then the zoning expander device performs the unlock step. The zoning expander device is unlocked and the zoning expander shadow values are not activated.

4.9.6.2 Lock step

The lock step ensures that the same zone manager locks each zoning expander device. A zone manager sends the SMP ZONE LOCK request (see 10.4.3.20) to lock a zoning expander device. A zoning expander device is locked when the ZONE LOCKED bit is set to one in the SMP REPORT GENERAL response and after the SAS address of the zone management server device has been stored. The management device server in a locked zoning expander device processes SMP zone configuration function requests, SMP ZONE ACTIVATE requests, and SMP ZONE UNLOCK requests.

If more than one zone manager attempts to lock a group of zoning expander devices, the following rules ensure that any concurrent requests are resolved:

- 1) If the first SMP ZONE LOCK response received by a zone manager has the FUNCTION RESULT field set to ZONE LOCK VIOLATION (see 10.4.3.2), then the group of zoning expander devices is locked by another zone manager and the zone manager should originate no further requests until it receives a Broadcast (Change);
- 2) If at least one SMP ZONE LOCK request is successful and at least one other response has:
 - A) the FUNCTION RESULT field set to ZONE LOCK VIOLATION (see 10.4.3.2); and
 - B) the CONFIGURING bit set to one (see 4.9.6.3),

then a zoning expander device is locked and being configured by another zone manager. The zone manager that failed to lock the zoning expander devices should unlock all zoning expander devices

that it has locked. When a Broadcast (Change) event is received, then the zone manager should retry the lock step; and

- 3) If at least one SMP ZONE LOCK request is successful and at least one other response has:
 - A) the FUNCTION RESULT field set to ZONE LOCK VIOLATION (see 10.4.3.2); and
 - B) the CONFIGURING bit set to zero,

then another zone manager has locked at least one zoning expander device in the group of zoning expander devices and the zone manager shall evaluate the ACTIVE ZONE MANAGER SAS ADDRESS field in the SMP ZONE LOCK response:

- A) if the returned SAS address has a lower numeric value than the SMP port SAS address of the zone manager, then the zone manager with the higher numeric value SAS address shall repeat the SMP ZONE LOCK request to all zoning expander devices that it has not already locked until all required zoning expander devices are locked, or until a Broadcast (Change) is received; or
- B) if the returned ACTIVE ZONE MANAGER SAS ADDRESS field has a higher numeric value than the SMP port SAS address of the zone manager, then the zone manager with the lower numeric value SAS address shall originate an SMP ZONE UNLOCK request to unlock all zoning expander devices that it locked.

The lock step is complete after a zone manager receives a successful SMP ZONE LOCK response from all required zoning expander devices.

4.9.6.3 Load step

The load step stores SMP zone configuration information as zoning expander shadow values. A zoning expander device only processes SMP zone configuration function requests originated by the active zone manager when it is locked.

The SMP zone configuration functions are:

- a) SMP CONFIGURE ZONE PHY INFORMATION (see 10.4.3.24);
- b) SMP CONFIGURE ZONE PERMISSION TABLE (see 10.4.3.25); and
- c) SMP ENABLE DISABLE ZONING (see 10.4.3.18).

After a locked zoning expander device processes any SMP zone configuration function request, it sets the CONFIGURING bit to one in the SMP REPORT GENERAL response (see 10.4.3.3).

SMP zone configuration functions change the zoning expander shadow values and do not affect the zoning expander active values. The zoning expander shadow values become zoning expander active values during the activate step (see 4.9.6.4).

If the active zone manager receives a response to an SMP zone configuration function with the FUNCTION RESULT field set to ZONE LOCK VIOLATION (see 10.4.3.2), then it should unlock all locked zoning expander devices.

The load step may be skipped when a locked zoning expander device is unlocked:

- a) by a zone manager with a higher SAS address during the lock step (see 4.9.6.2); or
- b) because the zone lock inactivity timer expires.

4.9.6.4 Activate step

The activate step copies the zoning expander shadow register values to the zoning expander active values. The active zone manager issues one of the following:

- a) a Broadcast (Zone Activate) (see 4.1.13); or
- b) an SMP ZONE ACTIVATE request (see 10.4.3.21) to all locked zoning expander devices.

After a locked zoning expander device receives a Broadcast (Zone Activate) or processes an SMP ZONE ACTIVATE request, then the zoning expander device shall set the zoning expander active values equal to the zoning expander shadow values.

If the active zone manager receives an SMP ZONE ACTIVATE response with the FUNCTION RESULT field set to ZONE LOCK VIOLATION (see 10.4.3.2), then it should unlock all locked zoning expander devices.

The activate step may be skipped when a locked zoning expander device is unlocked:

- a) by a zone manager with a higher SAS address during the lock step (see 4.9.6.2); or
- b) because the zone lock inactivity timer expires.

4.9.6.5 Unlock step

The unlock step ensures that the active zone manager unlocks the locked zoning expander devices, or if the zone manager fails then the zone lock inactivity timer expires and the zoning expander devices unlock.

If the active zone manager originated Broadcast (Zone Activate), then it sends an SMP ZONE UNLOCK request (see 10.4.3.22) with the `ACTIVATE REQUIRED` bit set to one to each of the locked zoning expander devices. This ensures that each zoning expander device has performed the activate step before it performs the unlock step. If it receives an SMP ZONE UNLOCK response with the `FUNCTION RESULT` field set to `NOT ACTIVATED` (see 10.4.3.2), then the zone manager retries the SMP ZONE UNLOCK request a vendor-specific number of times, then originates an SMP ZONE ACTIVATE request to each locked zoning expander device.

If the active zone manager originated SMP ZONE ACTIVATE request(s), then after all the SMP ZONE ACTIVATE functions have successfully completed it sends an SMP ZONE UNLOCK request with the `ACTIVATE REQUIRED` bit set to zero to each of the locked zoning expander devices. If it receives an SMP ZONE UNLOCK response with the `FUNCTION RESULT` field set to `BUSY` (see 10.4.3.2), then the zone manager retries the SMP ZONE UNLOCK request.

When the SMP ZONE UNLOCK request is successful or the zone lock inactivity timer expires, then the zoning expander device is unlocked and shall:

- a) set the `ZONE LOCKED` bit to zero in the SMP REPORT GENERAL response (see 10.4.3.3);
- b) set the `CONFIGURING` bit to zero in the SMP REPORT GENERAL response;
- c) if the zone lock timer expired, originate a Broadcast (Change) from zone group 1; and
- d) if the management device server processed an SMP ZONE UNLOCK request, originate a Broadcast (Change) (see 7.11) from either:
 - A) each zone group whose zone permission table entries or zone phy information has changed; or
 - B) zone group 1.

When all SMP ZONE UNLOCK requests are successful the configuration process is complete.

4.9.6.6 Zone lock inactivity timer

The zone lock inactivity timer is supported by all zoning expander devices. The use of a timer ensures that if the zone manager disappears without performing the unlock step that all locked zoning expander devices are unlocked.

When a zoning expander device processes an SMP ZONE LOCK request (see 10.4.3.20) then the zone lock inactivity timer default value is set to the value of the `ZONE LOCK INACTIVITY TIME LIMIT` field.

The zone lock inactivity timer is initialized and started if the default value is non-zero and:

- a) the zoning expander device completes processing of any SMP zone configuration function request or SMP ZONE ACTIVATE request while the `ZONE LOCKED` bit is set to one in the SMP REPORT GENERAL response (see 10.4.3.3); or
- b) the zoning expander device completes processing of a successful SMP ZONE LOCK request.

The zone lock inactivity timer is stopped if:

- a) the `ZONE LOCK INACTIVITY TIME LIMIT` field is set to zero in an SMP ZONE LOCK request; or
- b) the `ZONE LOCKED` bit is set to zero in the SMP REPORT GENERAL response (e.g., an SMP ZONE UNLOCK request (see 10.4.3.22) is processed, or the zone lock inactivity timer expires).

If the zone lock inactivity timer expires then the zoning expander device:

- a) sets the `ZONE LOCKED` bit to zero in the SMP REPORT GENERAL response;
- b) sets the `CONFIGURING` bit to zero in the SMP REPORT GENERAL response; and
- c) sends Broadcast (Change) on all ports.

If the zone lock inactivity timer expires while the zoning expander device is processing an SMP configuration function then the zoning expander device may complete the request successfully or return a function result of ZONE LOCK VIOLATION.

4.9.6.7 Enable a zoning expander device

If a zoning expander device has the ZONING SUPPORTED bit set to one and the ZONING ENABLED bit set to zero in the REPORT GENERAL response (see 10.4.3.3), then a zone manager configures the zoning expander device using the zone configuration process. This ensures that the zone permission table is the same in all zoning expander devices inside the ZPSDS.

Changes made by the SMP ENABLE DISABLE ZONING function sent by the active zone manager become active during the activate step (see 4.9.6.4).

4.10 Phy test functions

The optional Protocol-Specific diagnostic page for SAS (see 10.2.9.2) provide methods for an application client to enable and disable a phy test function (e.g., transmission of the CJTPAT) for a selected phy in a SAS target device with an SSP target port. The optional SMP PHY TEST FUNCTION function (see 10.4.3.28) provides similar methods for expander devices and SAS target devices with SMP target ports.

The application client sends a SEND DIAGNOSTIC command with the Protocol-Specific diagnostic page or an SMP PHY TEST FUNCTION function specifying the phy in the SAS target device that is to perform the phy test function and the phy test function to be performed. If the phy test function requires a specific phy test pattern and/or phy test pattern physical link rate, then it also specifies the phy test pattern and phy test pattern physical link rate.

The SEND DIAGNOSTIC command may be sent through any SSP target port to any logical unit in the SAS target device that contains the phy that is to perform the phy test function.

For the SEND DIAGNOSTIC command, the phy shall begin the specified phy test function after the SSP target port receives an ACK for the RESPONSE frame transmitted in response to the SEND DIAGNOSTIC command that requested the phy test function. For the SMP PHY TEST FUNCTION function, the phy shall begin the specified phy test function after the SMP target port transmits the SMP response frame.

Once a SAS phy has begun performing a phy test function, it shall ignore its receiver. To stop a SAS phy from performing a phy test function, an application client sends a SEND DIAGNOSTIC command or an SMP PHY TEST FUNCTION function to a SAS phy in the SAS target device that is not performing a phy test function requesting a phy test function of 00h (i.e., STOP). If no such phy is available, the phy test function only stops on power loss.

Editor's Note 14: 07-119 proposes the far-end retimed loopback mode be added again (dropped early in SAS-1 development) and makes additional changes like adding an address frame to trigger phy test functions (like SATA's BIST Activate FIS).

4.11 Phy events

Phys shall count the following events using saturating counters and report them in the Protocol-Specific Port log page (see 10.2.8.1) and/or the SMP REPORT PHY ERROR LOG function (see 10.4.3.10):

- a) invalid dwords received;
- b) dwords received with running disparity errors;
- c) loss of dword synchronization; and
- d) phy reset problems.

NOTE 16 - This standard also defines wrapping counters that count those same events (see table 37).

The saturating counters are each up to 4 bytes wide.

Phys may also count certain events (e.g., elasticity buffer overflows) using wrapping counters and record peak values for certain events (e.g., the longest connection time) using peak value detectors, reporting them in the Protocol-Specific Port log page (see 10.2.8.1), SMP REPORT PHY EVENT function (see 10.4.3.13), and/or the SMP REPORT PHY EVENT LIST function (see 10.4.3.15). The wrapping counters and peak value detectors are each 4 bytes wide. Peak value detectors trigger Broadcast (Expander) under certain circumstances (see 7.2.5.5).

For phys not controlled by SMP target ports, the number of additional events monitored and which events to monitor is vendor-specific.

For phys controlled by SMP target ports, the number of additional events that are simultaneously monitored is vendor-specific, but the SMP CONFIGURE PHY EVENT function (see 10.4.3.29) allows the events to count/record to be specified.

The management device server shall maintain phy events for the last vendor-specific number of events and should maintain at least one phy event per phy. The management device server shall assign descriptors to the events sequentially starting at 0001h and shall return the descriptors in the SMP REPORT PHY EVENT LIST response (see 10.4.3.15). The management device server shall return the index of the descriptor for the last phy event in the SMP REPORT GENERAL response (see 10.4.3.3), the SMP REPORT PHY EVENT LIST response (see 10.4.3.15), and the SMP DISCOVER LIST response (see 10.4.3.14). The management device server shall wrap the index to 0001h when the highest supported descriptor index has been used.

The management device server shall support phy event list descriptor (see 10.4.3.15.4) indexes from 0001h to FFFFh. The actual number of phy event list descriptors that the management device server maintains for retrieval with the REPORT PHY EVENT LIST request is vendor specific and is indicated by the MAXIMUM NUMBER OF STORED PHY EVENT LIST DESCRIPTORS field defined in the REPORT GENERAL response (see 10.4.3.3). The volatility of these stored descriptors is vendor specific. The management device server shall replace the least recently recorded phy event list descriptor with a new one once the number of recorded descriptors exceeds the value indicated by the MAXIMUM NUMBER OF STORED PHY EVENT LIST DESCRIPTORS field.

The PHY EVENT SOURCE field, defined in table 37, is used in the Protocol-Specific Port log page (see 10.2.8.1), the REPORT PHY EVENT function (see 10.4.3.13), the REPORT PHY EVENT LIST function (see 10.4.3.15), and the CONFIGURE PHY EVENT function (see 10.4.3.29) and indicates or specifies the type of phy event in the accompanying PHY EVENT field.

Table 37 — PHY EVENT SOURCE field (part 1 of 4)

Code	Name	Type ^a	Description
00h	No event	N/A	No event. The PHY EVENT field is not valid.
01h	Invalid dword count ^b	WC	Number of invalid dwords (see 3.1.112) that have been received outside of phy reset sequences (i.e., between when the SP state machine (see 6.8) sends a Phy Layer Ready (SAS) confirmation or Phy Layer Ready (SATA) confirmation and when it sends a Phy Layer Not Ready confirmation to the link layer).
02h	Running disparity error count ^b	WC	Number of dwords containing running disparity errors (see 6.2) that have been received outside of phy reset sequences.
03h	Loss of dword synchronization count ^b	WC	Number of times the phy has restarted the link reset sequence because it lost dword synchronization (i.e., the SP state machine transitioned from SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)).
^a The Type column indicates the source type: a) WC = wrapping counter b) PVD = peak value detector c) N/A = not applicable ^b This standard also defines a saturating counter that counts this event (see 10.2.8.1 and 10.4.3.10).			

Table 37 — PHY EVENT SOURCE field (part 2 of 4)

Code	Name	Type ^a	Description
04h	Phy reset problem count ^b	WC	Number of times a phy reset problem has occurred (see 6.7.4.2.4).
05h	Elasticity buffer overflow count	WC	Number of times the phy's receive elasticity buffer (see 7.3) has overflowed outside of phy reset sequences (e.g., because it did not receive a sufficient number of deletable primitives).
06h	Received ERROR count	WC	Number of times the phy received an ERROR primitive.
07h - 1Fh	Reserved for phy layer-based phy events		
20h	Received address frame error count	WC	Number of times the phy detected an invalid address frame (see 7.8) (e.g., because of a CRC error).
21h	Transmitted abandon-class OPEN_REJECT count	WC	Number of times the phy received an OPEN address frame and transmitted an abandon-class OPEN_REJECT (see 7.2.5.13). In expander devices, forwarded OPEN_REJECTs shall not be counted.
22h	Received abandon-class OPEN_REJECT count	WC	Number of times the phy originated an OPEN address frame and received an abandon-class OPEN_REJECT (see 7.2.5.13). In expander devices, OPEN_REJECTs in response to forwarded OPEN address frames shall not be counted.
23h	Transmitted retry-class OPEN_REJECT count	WC	Number of times the phy received an OPEN address frame and transmitted a retry-class OPEN_REJECT (see 7.2.5.13). In expander devices, forwarded OPEN_REJECTs shall not be counted.
24h	Received retry-class OPEN_REJECT count	WC	Number of times the phy originated an OPEN address frame and received a retry-class OPEN_REJECT (see 7.2.5.13). In expander devices, OPEN_REJECTs in response to forwarded OPEN address frames shall not be counted.
25h	Received AIP (WAITING ON PARTIAL) count	WC	Number of times the phy received an AIP (WAITING ON PARTIAL) or AIP (RESERVED WAITING ON PARTIAL). In expander devices, forwarded AIPs shall be counted.
26h	Received AIP (WAITING ON CONNECTION) count	WC	Number of times the phy received an AIP (WAITING ON CONNECTION). In expander devices, forwarded AIPs shall be counted.
27h	Transmitted BREAK count	WC	Number of times the phy transmitted a BREAK that was not a response to a BREAK it received (e.g., a Close Timeout was detected by the SL state machine interfacing to the SMP target port).
28h	Received BREAK count	WC	Number of times the phy received a BREAK that was not a response to a BREAK that it transmitted.
^a The Type column indicates the source type: a) WC = wrapping counter b) PVD = peak value detector c) N/A = not applicable ^b This standard also defines a saturating counter that counts this event (see 10.2.8.1 and 10.4.3.10).			

Table 37 — PHY EVENT SOURCE **field** (part 3 of 4)

Code	Name	Type ^a	Description
29h	Break Timeout count	WC	Number of times the phy transmitted a BREAK and did not receive a BREAK or BREAK_REPLY in response (e.g., as detected by the XL state machine and/or the SL state machine interfacing to the SMP target port).
2Ah	Connection count	WC	Number of connections in which the phy was involved.
2Bh	Peak transmitted pathway blocked count	PVD	Peak value of a PATHWAY BLOCKED COUNT field in an OPEN address frame transmitted by the phy. Since the maximum value of the PATHWAY BLOCKED COUNT field is FFh, only byte 3 of the PHY EVENT field is used.
2Ch	Peak transmitted arbitration wait time	PVD	Peak value of an ARBITRATION WAIT TIME field in an OPEN address frame transmitted by the phy. Since the maximum value of the ARBITRATION WAIT TIME field is FFFFh, only bytes 2 and 3 of the PHY EVENT field are used.
2Dh	Peak arbitration time	PVD	Peak time in microseconds after transmitting an OPEN address frame that the phy has waited for connection response (e.g., OPEN_ACCEPT or OPEN_REJECT).
2Eh	Peak connection time	PVD	The peak duration, in microseconds, of any connection in which the phy was involved.
2Fh - 3Fh	Reserved for SAS arbitration-related phy information		
40h	Transmitted SSP frame count	WC	Number of SSP frames transmitted.
41h	Received SSP frame count	WC	Number of SSP frames received.
42h	Transmitted SSP frame error count	WC	Number of times the phy was used in a connection involving the SSP target port, transmitted an SSP frame, and received a NAK or an ACK/NAK timeout.
43h	Received SSP frame error count	WC	Number of times the phy was used in a connection involving the SSP target port, detected an invalid SSP frame, and transmitted a NAK (CRC ERROR) (e.g., because of a CRC error).
44h	Transmitted CREDIT_BLOCKED count	WC	Number of times the phy transmitted a CREDIT_BLOCKED.
45h	Received CREDIT_BLOCKED count	WC	Number of times the phy received a CREDIT_BLOCKED.
46h - 4Fh	Reserved for SSP-related phy events		
^a The Type column indicates the source type: a) WC = wrapping counter b) PVD = peak value detector c) N/A = not applicable ^b This standard also defines a saturating counter that counts this event (see 10.2.8.1 and 10.4.3.10).			

Table 37 — PHY EVENT SOURCE field (part 4 of 4)

Code	Name	Type ^a	Description
50h	Transmitted SATA frame count	WC	Number of STP or SATA frames transmitted.
51h	Received SATA frame count	WC	Number of STP or SATA frames received.
52h	SATA flow control buffer overflow count	WC	Number of times the phy's STP flow control buffer (see 7.17.2) has overflowed (e.g., because it received more data dwords than allowed after transmitting SATA_HOLD during an STP connection). In an expander device, this count should be maintained in the expander phy transmitting the SATA_HOLD and receiving the data dwords, but may be maintained in the expander phy receiving the SATA_HOLD and transmitting the data dwords.
53h - 5Fh	Reserved for STP and SATA-related phy events		
60h	Transmitted SMP frame count	WC	Number of SMP frames transmitted.
61h	Received SMP frame count	WC	Number of SMP frames received.
62h	Reserved for SMP-related phy events		
63h	Received SMP frame error count	WC	Number of times the phy was used to access the SMP target port and the SMP target port detected an invalid SMP frame and transmitted a BREAK (e.g., because of a CRC error).
64h - 6Fh	Reserved for SMP-related phy events		
70h - CFh	Reserved		
D0h - FFh	Vendor specific		
^a The Type column indicates the source type: a) WC = wrapping counter b) PVD = peak value detector c) N/A = not applicable ^b This standard also defines a saturating counter that counts this event (see 10.2.8.1 and 10.4.3.10).			

5 Physical layer

5.1 Physical layer overview

The physical layer defines:

- a) passive interconnect (e.g., connectors and cable assemblies); and
- b) transmitter and receiver device electrical characteristics.

Within this standard, references to connector gender use the terms plug and receptacle as equivalent to the terms free and fixed, respectively, that may be used in the references that define the connectors. Fixed and free terminology has no relationship to the application of the connector.

5.2 Passive interconnect

5.2.1 SATA connectors and cable assemblies

Figure 58 shows a schematic representation of the connectors and cables defined by SATA (see SATA-2). A SATA host is analogous to a SAS initiator device and a SATA device is analogous to a SAS target device.

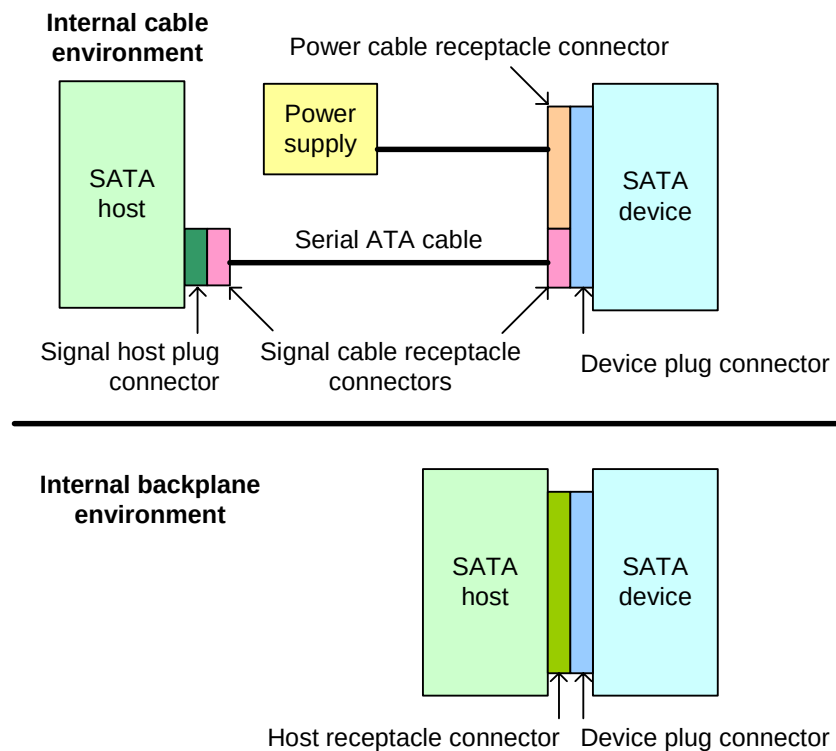


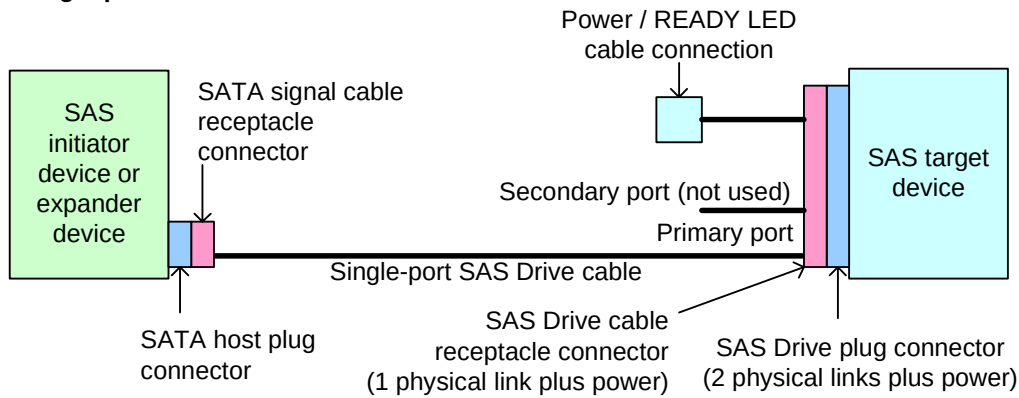
Figure 58 — SATA connectors and cables

5.2.2 SAS connectors and cables

This standard defines SAS Drive cable, SAS Drive backplane, SAS internal cable, and SAS external cable environments.

Figure 59 shows a schematic representation of the SAS Drive cable environments.

Single-port SAS Drive cable environment



Dual-port SAS Drive cable environment

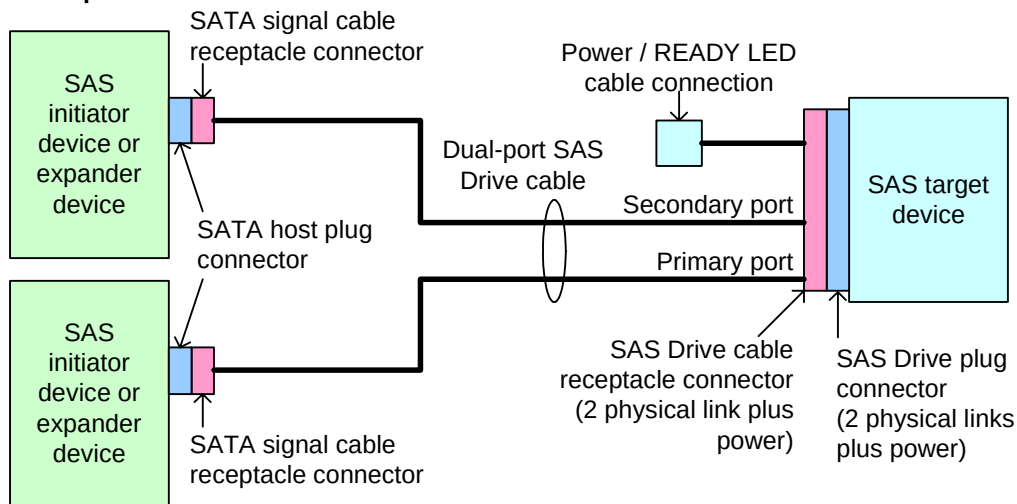


Figure 59 — SAS Drive cable environments

Figure 60 shows a schematic representation of the SAS Drive backplane environment.

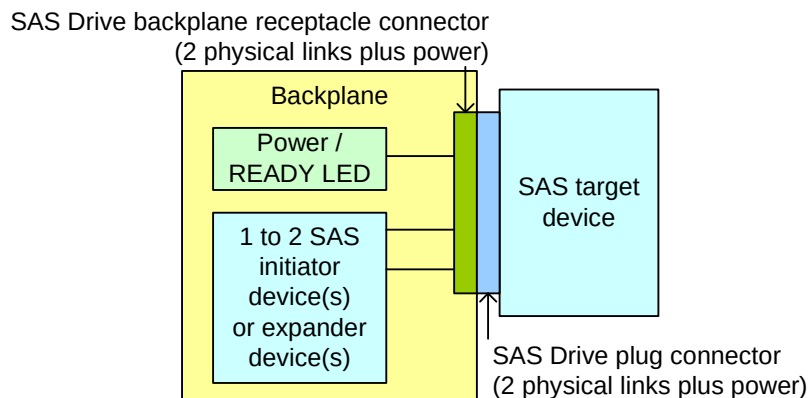


Figure 60 — SAS Drive backplane environment

Figure 61 shows a schematic representation of the SAS external cable environment.

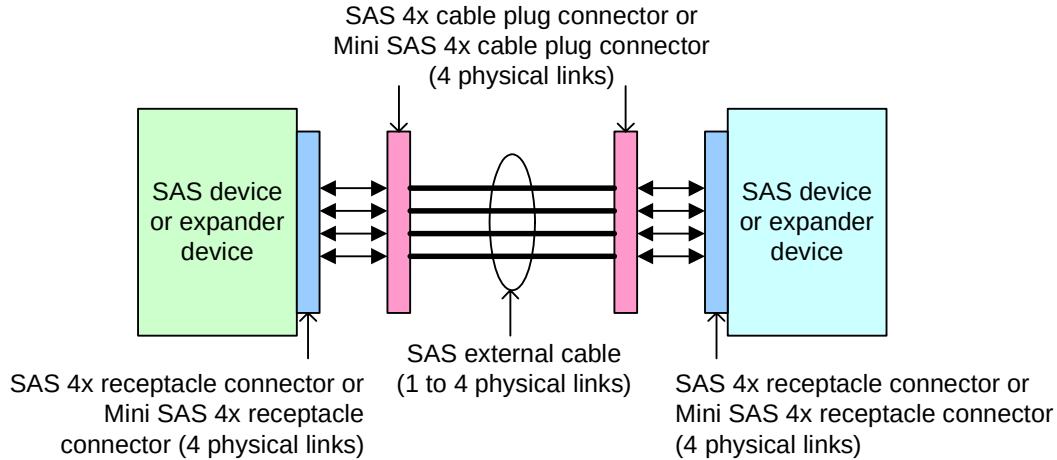


Figure 61 — SAS external cable environment

Figure 62 shows a schematic representation of the SAS internal cable environment attaching a controller to a backplane using a SAS internal symmetric cable.

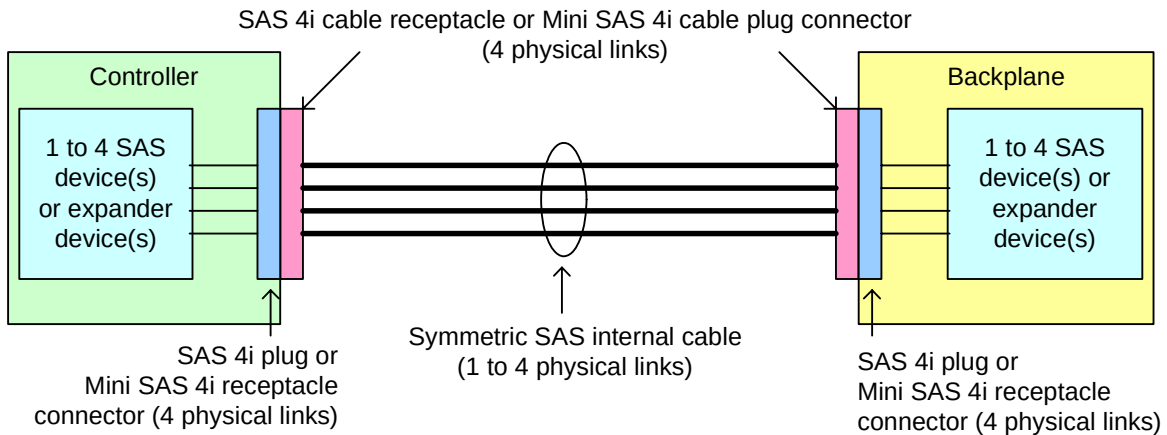


Figure 62 — SAS internal symmetric cable environment - controller to backplane

A SAS internal symmetric cable provides one to four physical links, and may be used as any combination of wide links and narrow links (see 4.1.4) using those physical links.

Figure 63 shows a schematic representation of the SAS internal cable environment attaching a controller to a controller using a SAS internal symmetric cable. Two controllers may also be attached together with a SAS internal symmetric cable.

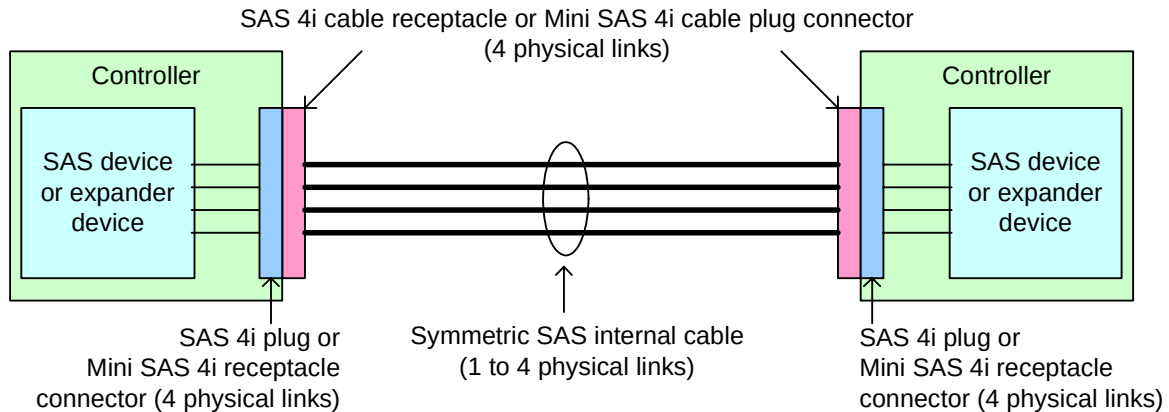


Figure 63 — SAS internal symmetric cable environment - controller to controller

Figure 64 shows a schematic representation of the SAS internal cable environment using a SAS controller-based fanout cable.

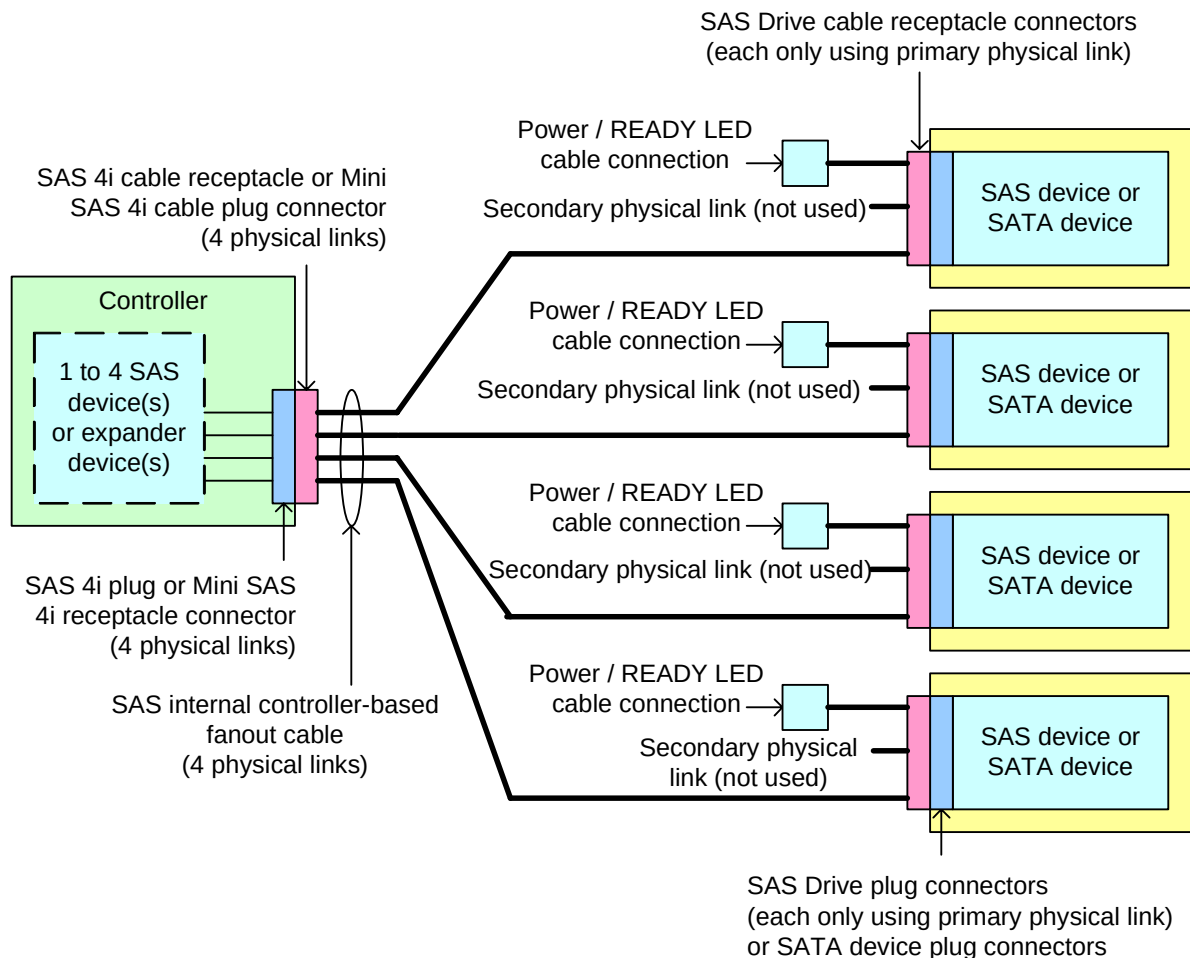


Figure 64 — SAS internal controller-based fanout cable environment

Figure 65 shows a schematic representation of the SAS internal cable environment using a SAS backplane-based fanout cable.

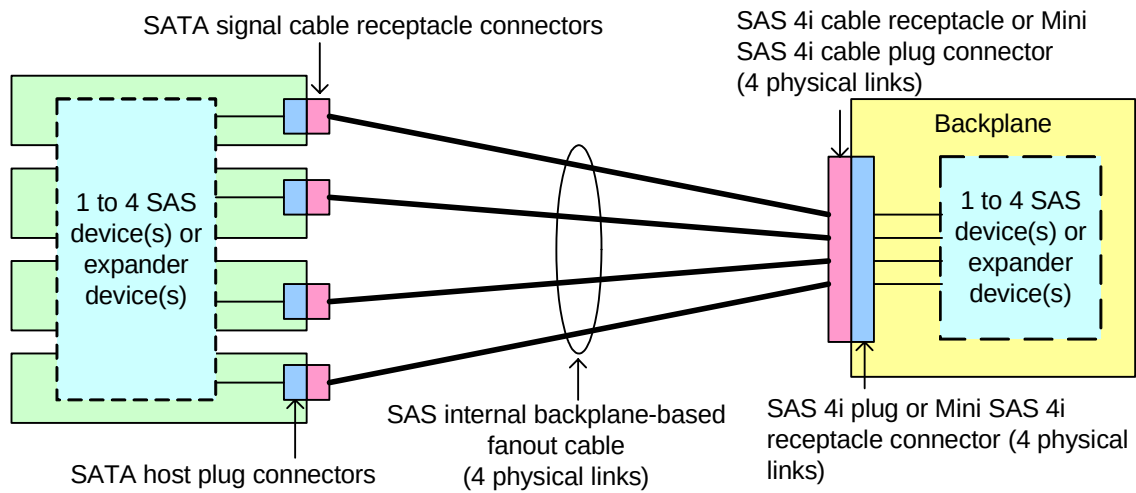


Figure 65 — SAS internal backplane-based fanout cable environment

5.2.3 Connectors

5.2.3.1 Connectors overview

Table 38 summarizes the connectors defined in this standard.

Table 38 — Connectors

Type of connector	Physical links	Reference	Attaches to		
			Type of connector	Physical links	Reference
SATA internal connectors used by SAS					
SATA signal cable receptacle	1	SATA-2	SATA host plug	1	SATA-2
SATA host plug	1	SATA-2	SATA signal cable receptacle	1	SATA-2
SATA device plug	1	SATA-2	SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2
			SAS Drive backplane receptacle	2	5.2.3.2.1.3
SAS internal connectors - SAS Drive connectors					
SAS Drive plug	2	5.2.3.2.1.1	SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2
			SAS Drive backplane receptacle	2	5.2.3.2.1.3
SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2	SAS Drive plug	2	5.2.3.2.1.1
			SATA device plug	1	SATA-2
SAS Drive backplane receptacle	2	5.2.3.2.1.3	SAS Drive plug	2	5.2.3.2.1.1
			SATA device plug	1	SATA-2
SAS internal connectors - other					
SAS 4i cable receptacle	4	5.2.3.2.2.1	SAS 4i plug	4	5.2.3.2.2.2
SAS 4i plug	4	5.2.3.2.2.2	SAS 4i cable receptacle	4	5.2.3.2.2.1
Mini SAS 4i cable plug	4	5.2.3.2.3.1	Mini SAS 4i receptacle	4	5.2.3.2.3.2
Mini SAS 4i receptacle	4	5.2.3.2.3.2	Mini SAS 4i cable plug	4	5.2.3.2.3.1
SAS external connectors					
SAS 4x cable plug	4	5.2.3.3.1.1	SAS 4x receptacle	4	5.2.3.3.1.2
SAS 4x receptacle	4	5.2.3.3.1.2	SAS 4x cable plug	4	5.2.3.3.1.1
Mini SAS 4x cable plug	4	5.2.3.3.2.1	Mini SAS 4x receptacle	4	5.2.3.3.2.2
Mini SAS 4x receptacle	4	5.2.3.3.2.2	Mini SAS 4x plug	4	5.2.3.3.2.1

A SAS icon (see Annex M) should be placed on or near each SAS connector.

5.2.3.2 SAS internal connectors

5.2.3.2.1 SAS Drive connectors

5.2.3.2.1.1 SAS Drive plug connector

The SAS Drive plug connector is the Device Free (Plug) connector defined in SFF-8482.

See SFF-8223, SFF-8323, and SFF-8523 for the SAS Drive plug connector locations on common form factors.

Figure 66 shows the SAS Drive plug connector.

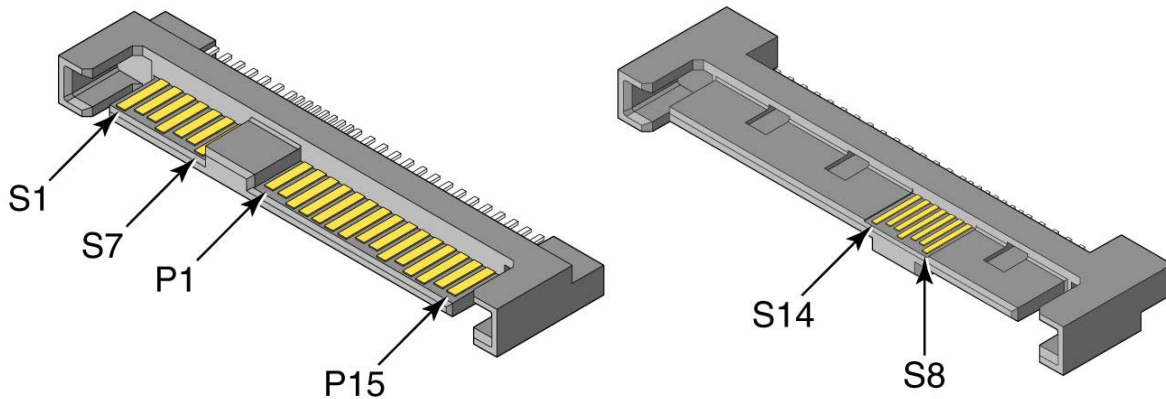


Figure 66 — SAS Drive plug connector

Table 39 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive plug connector.

5.2.3.2.1.2 SAS Drive cable receptacle connector

The SAS Drive cable receptacle connector is the Internal Cable Fixed (Receptacle) connector defined in SFF-8482.

The single-port version attaches to:

- a SAS Drive plug connector, providing contact for the power pins and only the primary physical link;
- a SATA device plug connector, providing contact for the power pins and the primary physical link.

Figure 67 shows the single-port version of the SAS Drive cable receptacle connector.

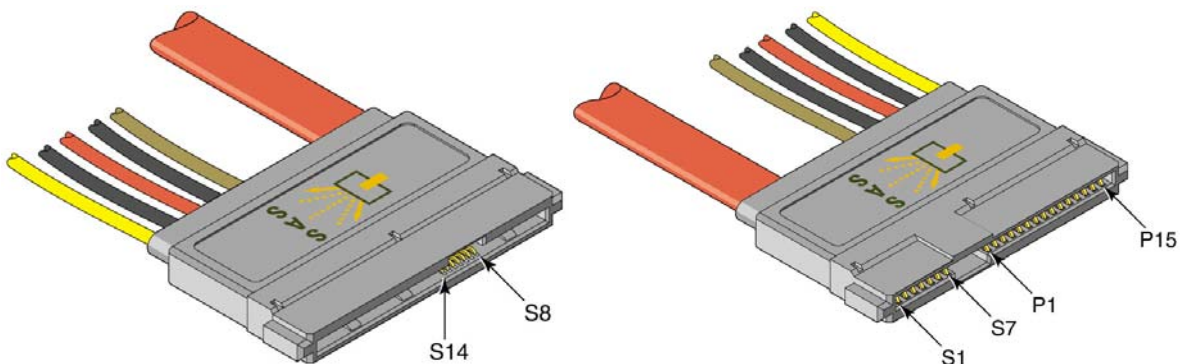


Figure 67 — Single-port SAS Drive cable receptacle connector

The dual-port version attaches to:

- a SAS Drive plug connector, providing contact for the power pins and only the primary physical link;

- b) a SAS Drive plug connector, providing contact for the power pins and both the primary and secondary physical links; or
- c) a SATA device plug connector, providing contact for the power pins and the primary physical link.

Figure 68 shows the dual-port version of the SAS Drive cable receptacle connector.

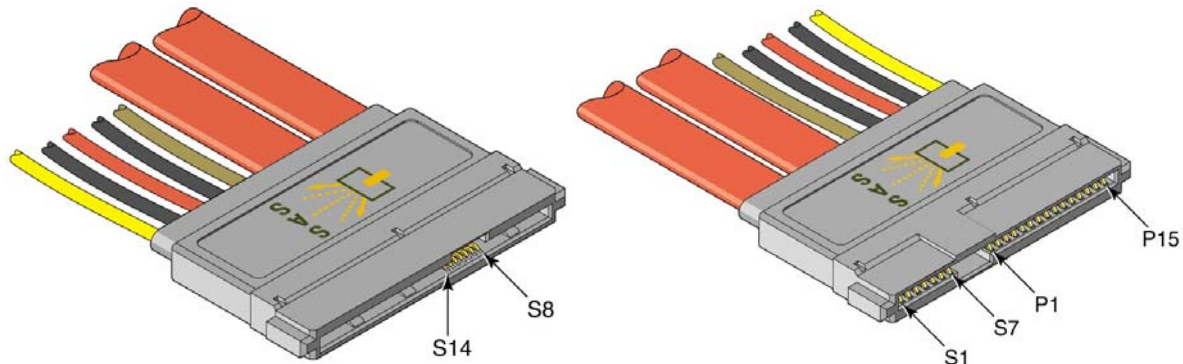


Figure 68 — Dual-port SAS Drive cable receptacle connector

Table 39 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive cable receptacle connector. The secondary physical link (i.e., pins S8 through S14) is not supported by the single-port internal cable receptacle.

5.2.3.2.1.3 SAS Drive backplane receptacle connector

The SAS Drive backplane receptacle connector is the Backplane Fixed (Receptacle) connector defined in SFF-8482.

The SAS Drive backplane receptacle connector attaches to:

- a) a SAS Drive plug connector, providing contact for the power pins and only the primary physical link;
- b) a SAS Drive plug connector, providing contact for the power pins and both primary and secondary physical links; or
- c) a SATA device plug connector, providing contact for the power pins and the primary physical link.

Figure 69 shows the SAS Drive backplane receptacle connector.

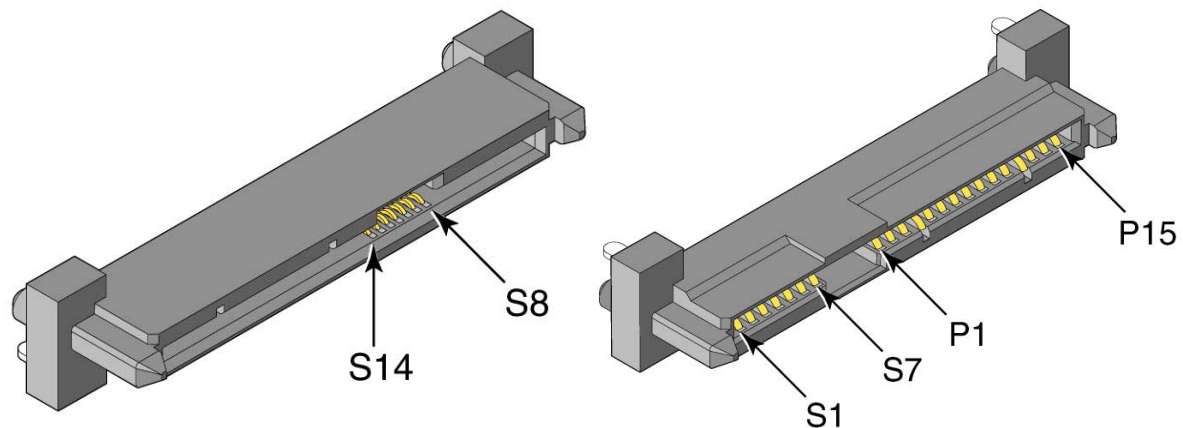


Figure 69 — SAS Drive backplane receptacle connector

Table 39 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive backplane receptacle connector.

5.2.3.2.1.4 SAS Drive connector pin assignments

Table 39 defines the SAS target device pin assignments for the SAS Drive plug connector (see 5.2.3.2.1.1), the SAS Drive cable receptacle connector (see 5.2.3.2.1.2), and the SAS Drive backplane receptacle connector (see 5.2.3.2.1.3). The TP+, TP-, RP+, and RP- signals are used by the primary physical link. The TS+, TS-, RS+, and RS- signals are used by the secondary physical link, if any.

SAS Drive plug connector pin assignments, except for the addition of the secondary physical link when present, are in the same locations as they are in a SATA device plug connector (see SATA-2).

Table 39 — SAS Drive connector pin assignments

Segment	Pin	Backplane receptacle	SAS Drive plug and SAS Drive cable receptacle
Primary signal segment	S1	SIGNAL GROUND	
	S2	TP+	RP+
	S3	TP-	RP-
	S4	SIGNAL GROUND	
	S5	RP-	TP-
	S6	RP+	TP+
	S7	SIGNAL GROUND	
Secondary signal segment ^a	S8	SIGNAL GROUND	
	S9	TS+	RS+
	S10	TS-	RS-
	S11	SIGNAL GROUND	
	S12	RS-	TS-
	S13	RS+	TS+
	S14	SIGNAL GROUND	
Power segment ^b	P1	V ₃₃ ^c	
	P2	V ₃₃ ^c	
	P3	V ₃₃ , precharge ^c	
	P4	GROUND	
	P5	GROUND	
	P6	GROUND	
	P7	V ₅ , precharge ^c	
	P8	V ₅ ^c	
	P9	V ₅ ^c	
	P10	GROUND	
	P11	READY LED ^d	
	P12	GROUND	
	P13	V ₁₂ , precharge ^c	
	P14	V ₁₂ ^c	
	P15	V ₁₂ ^c	
^a S8 through S14 are no-connects on single-port implementations.			
^b Backplane receptacle connectors and SAS Drive cable receptacle connectors provide V ₃₃ , V ₅ , and V ₁₂ . SAS Device plug connectors receive V ₃₃ , V ₅ , and V ₁₂ .			
^c Behind a SAS Drive plug connector, the precharge pin and each corresponding voltage pin shall be connected together on the SAS target device (e.g., the V ₅ , precharge pin P7 is connected to the two V ₅ pins P8 and P9).			
^d Electrical characteristics for READY LED are defined in 5.4 and signal behavior is defined in 10.4.1. SATA devices use P11 for activity indication and staggered spin-up disable and have different electrical characteristics (see SATA-2).			

5.2.3.2.2 SAS 4i connectors

5.2.3.2.2.1 SAS 4i cable receptacle connector

The SAS 4i cable receptacle connector is the 4 Lane Cable Receptacle (fixed) with Backshell connector defined in SFF-8484.

Figure 70 shows the SAS 4i cable receptacle connector.

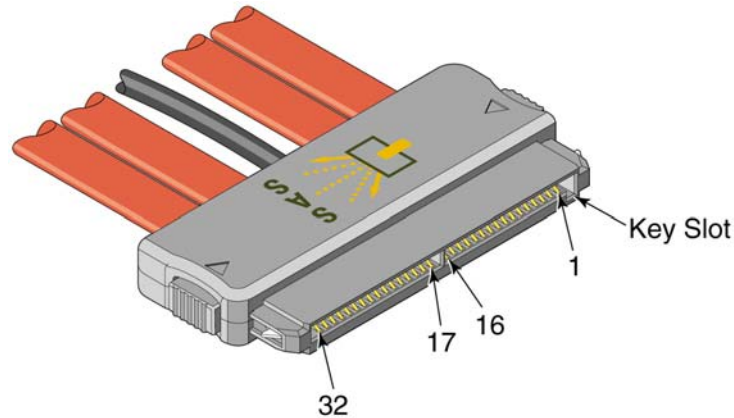


Figure 70 — SAS 4i cable receptacle connector

Table 40 and table 41 (see 5.2.3.2.2.3) define the pin assignments for the SAS 4i cable receptacle connector.

5.2.3.2.2.2 SAS 4i plug connector

The SAS 4i plug connector is the 4 Lane Vertical Plug (free) or 4 Lane R/A Plug (free) connector defined in SFF-8484.

Figure 71 shows the SAS 4i plug connector.

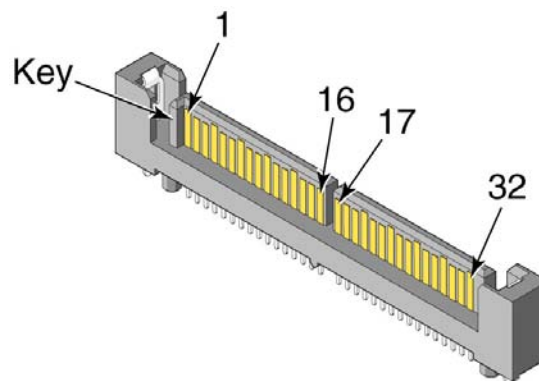


Figure 71 — SAS 4i plug connector

Table 40 and table 41 (see 5.2.3.2.2.3) define the pin assignments for the SAS 4i plug connector.

5.2.3.2.2.3 SAS 4i connector pin assignments

Table 40 defines the pin assignments for SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) and SAS 4i plug connectors (see 5.2.3.2.2.2) for controller applications using one, two, three, or four of the physical links.

Table 40 — Controller SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a			
	One	Two	Three	Four
Rx 0+	2	2	2	2
Rx 0-	3	3	3	3
Tx 0-	5	5	5	5
Tx 0+	6	6	6	6
Rx 1+	N/C	8	8	8
Rx 1-	N/C	9	9	9
Tx 1-	N/C	11	11	11
Tx 1+	N/C	12	12	12
Sideband 0	14	14	14	14
Sideband 1	15	15	15	15
Sideband 2	16	16	16	16
Sideband 3	17	17	17	17
Sideband 4	18	18	18	18
Sideband 5	19	19	19	19
Rx 2+	N/C	N/C	21	21
Rx 2-	N/C	N/C	22	22
Tx 2-	N/C	N/C	24	24
Tx 2+	N/C	N/C	25	25
Rx 3+	N/C	N/C	N/C	27
Rx 3-	N/C	N/C	N/C	28
Tx 3-	N/C	N/C	N/C	30
Tx 3+	N/C	N/C	N/C	31
SIGNAL GROUND	1, 4, 7, 10, 13, 20, 23, 26, 29, 32			
^a N/C = not connected				

The use of the sideband signals by a controller is vendor-specific. One implementation of the sideband signals by a controller is an SGPIO initiator interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

Table 41 defines the pin assignments for SAS 4i plug connectors (see 5.2.3.2.2.1) and SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) for backplane applications using one, two, three, or four of the physical links.

Table 41 — Backplane SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a			
	One	Two	Three	Four
Rx 3+	N/C	N/C	N/C	2
Rx 3-	N/C	N/C	N/C	3
Tx 3-	N/C	N/C	N/C	5
Tx 3+	N/C	N/C	N/C	6
Rx 2+	N/C	N/C	8	8
Rx 2-	N/C	N/C	9	9
Tx 2-	N/C	N/C	11	11
Tx 2+	N/C	N/C	12	12
Sideband 5	14	14	14	14
Sideband 4	15	15	15	15
Sideband 3	16	16	16	16
Sideband 2	17	17	17	17
Sideband 1	18	18	18	18
Sideband 0	19	19	19	19
Rx 1+	N/C	21	21	21
Rx 1-	N/C	22	22	22
Tx 1-	N/C	24	24	24
Tx 1+	N/C	25	25	25
Rx 0+	27	27	27	27
Rx 0-	28	28	28	28
Tx 0-	30	30	30	30
Tx 0+	31	31	31	31
SIGNAL GROUND	1, 4, 7, 10, 13, 20, 23, 26, 29, 32			
^a N/C = not connected				

The use of the sideband signals by a backplane is vendor-specific. One implementation of the sideband signals by a backplane is an SGPIO target interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

5.2.3.2.3 Mini SAS 4i connectors

5.2.3.2.3.1 Mini SAS 4i cable plug connector

The Mini SAS 4i cable plug connector is the free (plug) 36-circuit Unshielded Compact Multilane connector defined in SFF-8087 and SFF-8086.

Figure 72 shows the Mini SAS 4i cable plug connector.

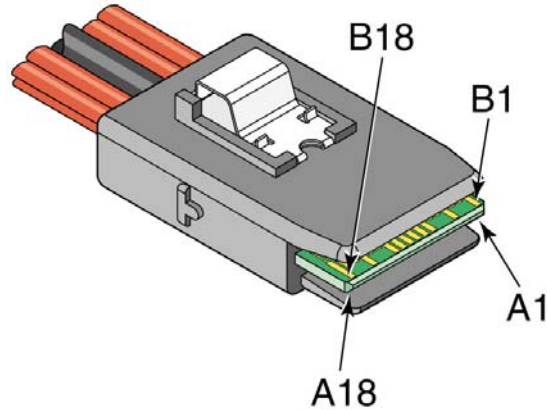


Figure 72 — Mini SAS 4i cable plug connector

Table 42 and table 43 (see 5.2.3.2.3.3) define the pin assignments for the Mini SAS 4i cable plug connector.

5.2.3.2.3.2 Mini SAS 4i receptacle connector

The Mini SAS 4i receptacle connector is the fixed (receptacle) 36-circuit Unshielded Compact Multilane connector defined in SFF-8087 and SFF-8086.

Figure 73 shows the Mini SAS 4i receptacle connector.

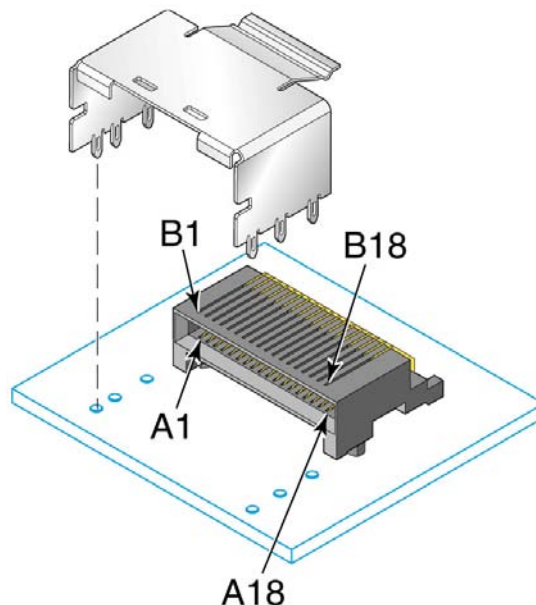


Figure 73 — Mini SAS 4i receptacle connector

Table 42 and table 43 (see 5.2.3.2.3.3) define the pin assignments for the Mini SAS 4i receptacle connector.

5.2.3.2.3.3 Mini SAS 4i connector pin assignments

Table 42 defines the pin assignments for Mini SAS 4i plug connectors (see 5.2.3.2.3.1) and Mini SAS 4i cable receptacle connectors (see 5.2.3.2.3.2) for controller applications using one, two, three, or four of the physical links.

Table 42 — Controller Mini SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a				Mating level
	One	Two	Three	Four	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Sideband 7	A8	A8	A8	A8	First
Sideband 3	A9	A9	A9	A9	
Sideband 4	A10	A10	A10	A10	
Sideband 5	A11	A11	A11	A11	
Rx 2+	N/C	N/C	A13	A13	Third
Rx 2-	N/C	N/C	A14	A14	
Rx 3+	N/C	N/C	N/C	A16	
Rx 3-	N/C	N/C	N/C	A17	
Tx 0+	B2	B2	B2	B2	Third
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Sideband 0	B8	B8	B8	B8	First
Sideband 1	B9	B9	B9	B9	
Sideband 2	B10	B10	B10	B10	
Sideband 6	B11	B11	B11	B11	
Tx 2+	N/C	N/C	B13	B13	Third
Tx 2-	N/C	N/C	B14	B14	
Tx 3+	N/C	N/C	N/C	B16	
Tx 3-	N/C	N/C	N/C	B17	
SIGNAL GROUND	A1, A4, A7, A12, A15, A18, B1, B4, B7, B12, B15, B18				First
^a N/C = not connected					

The use of the sideband signals by a controller is vendor-specific. One implementation of the sideband signals by a controller is an SGPIO initiator interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

Table 43 defines the pin assignments for Mini SAS 4i plug connectors (see 5.2.3.2.3.1) and Mini SAS 4i cable receptacle connectors (see 5.2.3.2.3.2) for backplane applications using one, two, three, or four of the physical links.

Table 43 — Backplane Mini SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a				Mating level
	One	Two	Three	Four	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Sideband 0	A8	A8	A8	A8	First
Sideband 1	A9	A9	A9	A9	
Sideband 2	A10	A10	A10	A10	
Sideband 6	A11	A11	A11	A11	
Rx 2+	N/C	N/C	A13	A13	Third
Rx 2-	N/C	N/C	A14	A14	
Rx 3+	N/C	N/C	N/C	A16	
Rx 3-	N/C	N/C	N/C	A17	
Tx 0+	B2	B2	B2	B2	Third
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Sideband 7	B8	B8	B8	B8	First
Sideband 3	B9	B9	B9	B9	
Sideband 4	B10	B10	B10	B10	
Sideband 5	B11	B11	B11	B11	
Tx 2+	N/C	N/C	B13	B13	Third
Tx 2-	N/C	N/C	B14	B14	
Tx 3+	N/C	N/C	N/C	B16	
Tx 3-	N/C	N/C	N/C	B17	
SIGNAL GROUND	A1, A4, A7, A12, A15, A18, B1, B4, B7, B12, B15, B18				First
^a N/C = not connected					

The use of the sideband signals by a backplane is vendor-specific. One implementation of the sideband signals by a backplane is an SGPIO target interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

5.2.3.3 SAS external connectors

5.2.3.3.1 SAS 4x connectors

5.2.3.3.1.1 SAS 4x cable plug connector

The SAS 4x cable plug connector is the 4X free (plug) connector with jack screws defined in SFF-8470. The SAS 4x cable plug connector shall not include keys and may include key slots. Key slots for the SAS 4x cable plug connector are not defined by this standard.

Figure 74 shows the SAS 4x cable plug connector.

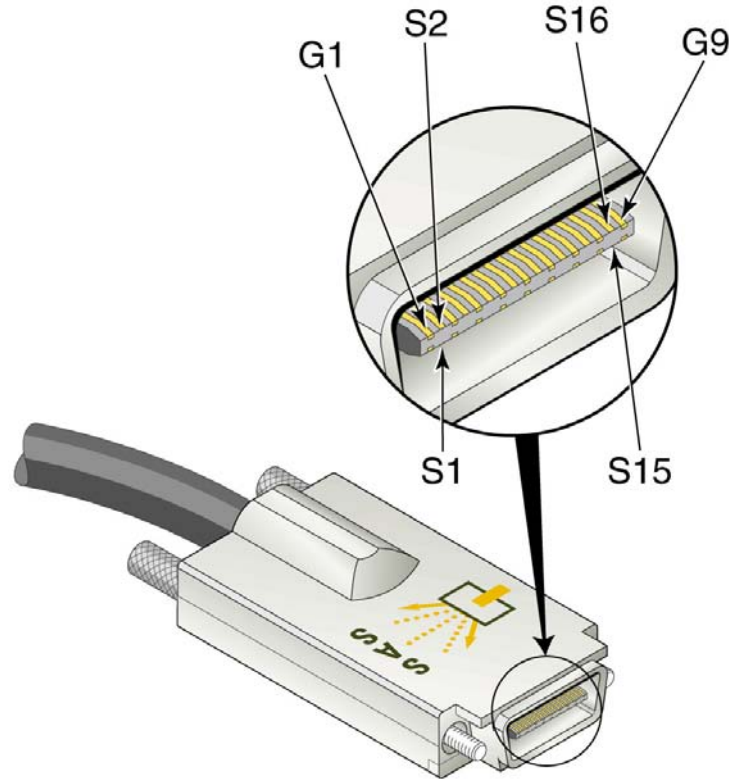


Figure 74 — SAS 4x cable plug connector

Table 46 (see 5.2.3.3.1.3) defines the pin assignments for the SAS 4x cable plug connector.

Table 44 defines the icons that should be placed on or near SAS 4x cable plug connectors.

Table 44 — SAS 4x cable plug connector icons

Use	Icon
End of a SAS external cable that attaches to an end device, an enclosure out port, or an enclosure universal port	Diamond
End of a SAS external cable that attaches to an end device, an enclosure in port, or an enclosure universal port	Circle

5.2.3.3.1.2 SAS 4x receptacle connector

The SAS 4x receptacle connector is the 4X fixed (receptacle) connector with jack screws defined in SFF-8470. The SAS 4x receptacle connector shall not include keys and may include key slots. Key slots for the SAS 4x receptacle connector are not defined by this standard.

A SAS 4x receptacle connector may be used by one or more SAS devices (e.g., one SAS device using physical links 0 and 3, another using physical link 1, and a third using physical link 2).

A SAS 4x receptacle connector shall be used by no more than one expander device at a time, and all physical links shall be used by the same expander port (i.e., all the expander phys shall have the same routing attribute (e.g., subtractive or table) (see 4.6.2)).

Figure 75 shows the SAS 4x receptacle connector.

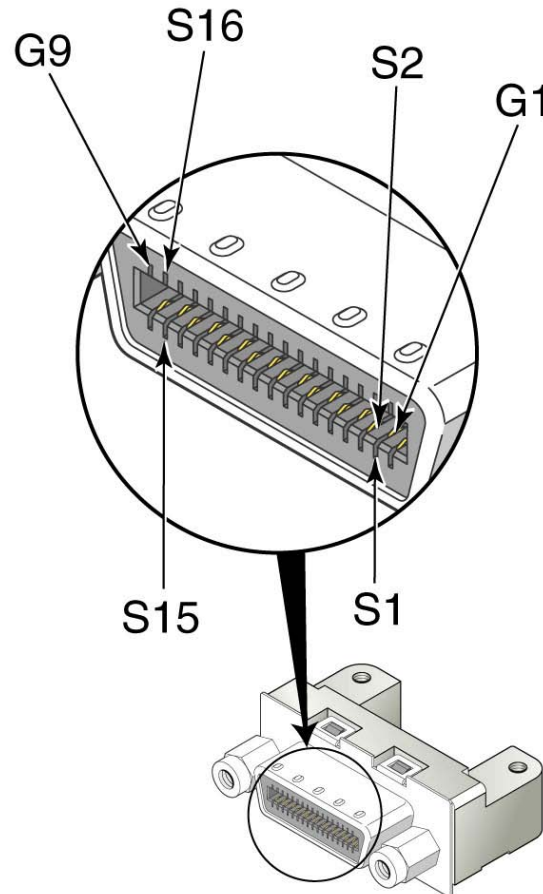


Figure 75 — SAS 4x receptacle connector

Table 46 (see 5.2.3.3.1.3) defines the pin assignments for the SAS 4x receptacle connector.

Table 45 defines the icons that should be placed near SAS 4x receptacle connectors.

Table 45 — SAS 4x receptacle connector icons

Use	Icons
Enclosure out port (see 4.6.2)	Diamond
End device or enclosure universal port	Diamond and circle
Enclosure in port (see 4.6.2)	Circle

5.2.3.3.1.3 SAS 4x connector pin assignments

Table 46 defines the pin assignments for SAS 4x cable plug connectors (see 5.2.3.3.1.1) and SAS 4x receptacle connectors (see 5.2.3.3.1.2) for applications using one, two, three, or four of the physical links.

Table 46 — SAS 4x connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a			
	One	Two	Three	Four
Rx 0+	S1	S1	S1	S1
Rx 0-	S2	S2	S2	S2
Rx 1+	N/C	S3	S3	S3
Rx 1-	N/C	S4	S4	S4
Rx 2+	N/C	N/C	S5	S5
Rx 2-	N/C	N/C	S6	S6
Rx 3+	N/C	N/C	N/C	S7
Rx 3-	N/C	N/C	N/C	S8
Tx 3-	N/C	N/C	N/C	S9
Tx 3+	N/C	N/C	N/C	S10
Tx 2-	N/C	N/C	S11	S11
Tx 2+	N/C	N/C	S12	S12
Tx 1-	N/C	S13	S13	S13
Tx 1+	N/C	S14	S14	S14
Tx 0-	S15	S15	S15	S15
Tx 0+	S16	S16	S16	S16
SIGNAL GROUND	G1 - G9			
CHASSIS GROUND	Housing			
^a N/C = not connected				

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the connector when used in a cable assembly.

5.2.3.3.2 Mini SAS 4x connectors

5.2.3.3.2.1 Mini SAS 4x cable plug connector

The Mini SAS 4x cable plug connector is the free (plug) 26-circuit Shielded Compact Multilane connector defined in SFF-8088 and SFF-8086.

Figure 76 shows the Mini SAS 4x cable plug connector.

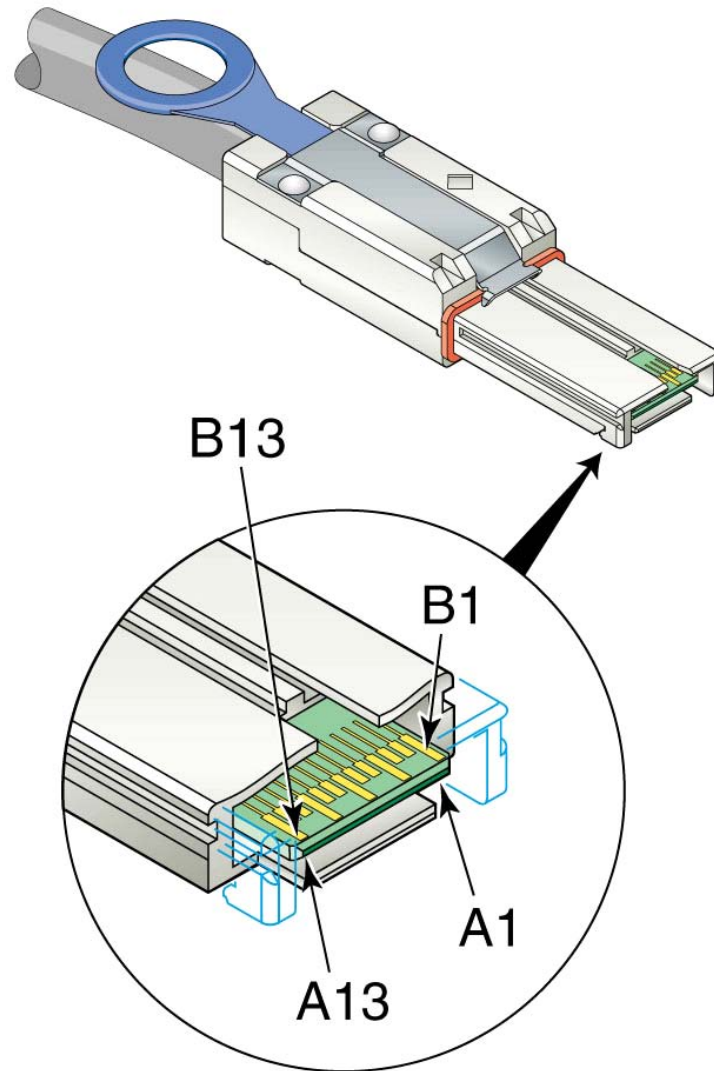


Figure 76 — Mini SAS 4x cable plug connector

If constructed with a pull tab as shown in figure 76, the pull tab should use PANTONE 279 C (i.e., light blue coated).

Table 49 (see 5.2.3.3.2.3) defines the pin assignments for the Mini SAS 4x cable plug connector.

Mini SAS 4x cable plug connectors shall include key slots to allow attachment to Mini SAS 4x receptacle connectors (see 5.2.3.3.2.2) with matching keys.

Table 47 defines the icons that shall be placed on or near Mini SAS 4x cable plug connectors and the key slot positions (see SFF-8088) that shall be used by Mini SAS 4x cable plug connectors.

Table 47 — Mini SAS 4x cable plug connector icons and key slot positions

Use	Icon	Key slot positions	Reference
End of a SAS external cable that attaches to an end device, an enclosure out port, or an enclosure universal port	Diamond	2, 4	Figure 77
End of a SAS external cable that attaches to an end device, an enclosure in port, or an enclosure universal port	Circle	4, 6	Figure 78

Figure 77 shows the key slots on the Mini SAS 4x cable plug connector that attaches to an end device or an enclosure universal port (see figure 80 in 5.2.3.3.2.2), or an enclosure out port (see figure 81 in 5.2.3.3.2.2).

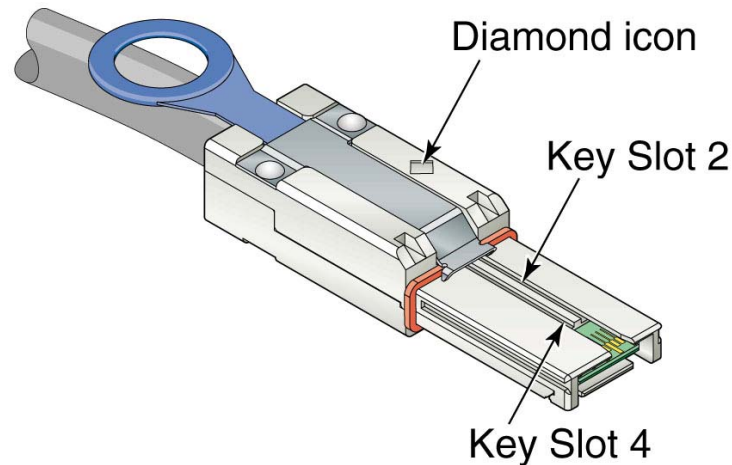


Figure 77 — Mini SAS 4x cable plug connector that attaches to an enclosure out port

Figure 78 shows the key slots on the Mini SAS 4x cable plug connector that attaches to an end device or an enclosure universal port (see figure 80 in 5.2.3.3.2.2), or an enclosure in port (see figure 82 in 5.2.3.3.2.2).

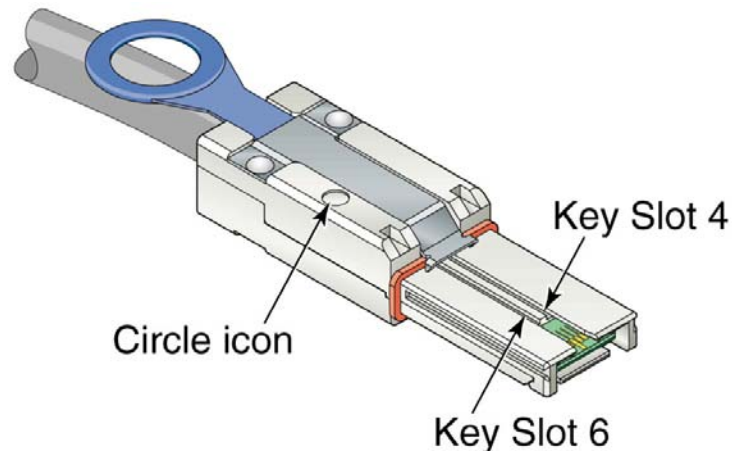


Figure 78 — Mini SAS 4x cable plug connector that attaches to an enclosure in port

5.2.3.3.2.2 Mini SAS 4x receptacle connector

The Mini SAS 4x receptacle connector is the fixed (receptacle) 26-circuit Shielded Compact Multilane connector defined in SFF-8088 and SFF-8086.

A Mini SAS 4x receptacle connector may be used by one or more SAS devices (e.g., one SAS device using physical links 0 and 3, another using physical link 1, and a third using physical link 2).

A Mini SAS 4x receptacle connector shall be used by no more than one expander device at a time, and all physical links shall be used by the same expander port (i.e., all the expander phys shall have the same routing attribute (e.g., subtractive or table) (see 4.6.2)).

Figure 79 shows the Mini SAS 4x receptacle connector.

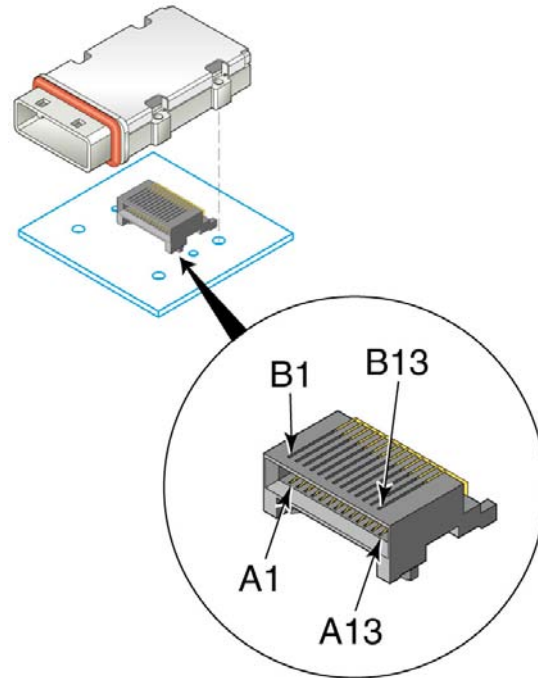


Figure 79 — Mini SAS 4x receptacle connector

Table 49 (see 5.2.3.3.2.3) defines the pin assignments for the Mini SAS 4x receptacle connector.

Mini SAS 4x receptacle connectors shall include keys to prevent attachment to Mini SAS 4x cable plug connectors (see 5.2.3.3.2.1) without matching key slots.

Table 48 defines the icons that shall be placed on or near Mini SAS 4x receptacle connectors and the key positions (see SFF-8088) that shall be used by Mini SAS 4x receptacle connectors.

Table 48 — Mini SAS 4x receptacle connector icons and key positions

Use	Icons	Key position	Reference
End device or enclosure universal port	Diamond and circle	4	Figure 80
Enclosure out port (see 4.6.2)	Diamond	2	Figure 81
Enclosure in port (see 4.6.2)	Circle	6	Figure 82

Figure 80 shows the key on a Mini SAS 4x receptacle connector used by end devices. This connector may be attached to the Mini SAS 4x cable plug shown in figure 77 or the Mini SAS 4x cable plug shown in figure 78 (see 5.2.3.3.2.1).

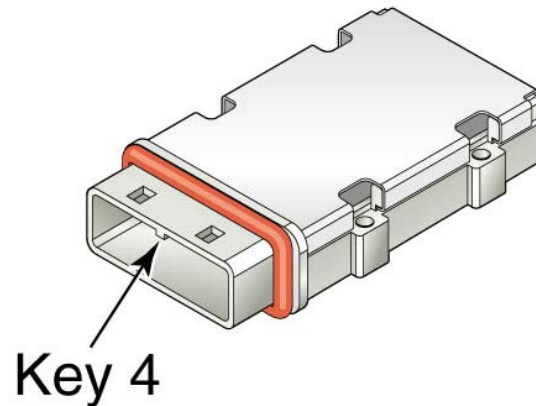


Figure 80 — Mini SAS 4x receptacle connector - end device or enclosure universal port

Figure 81 shows the key on a Mini SAS 4x receptacle connector used by an enclosure out port. This connector may be attached to the Mini SAS 4x cable plug shown in figure 77 (see 5.2.3.3.2.1).

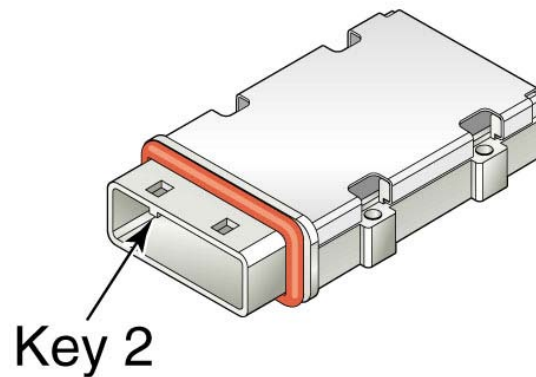


Figure 81 — Mini SAS 4x receptacle connector - enclosure out port

Figure 82 shows the key on a Mini SAS 4x receptacle connector used by an enclosure in port. This connector may be attached to the Mini SAS 4x cable plug shown in figure 78 (see 5.2.3.3.2.1).

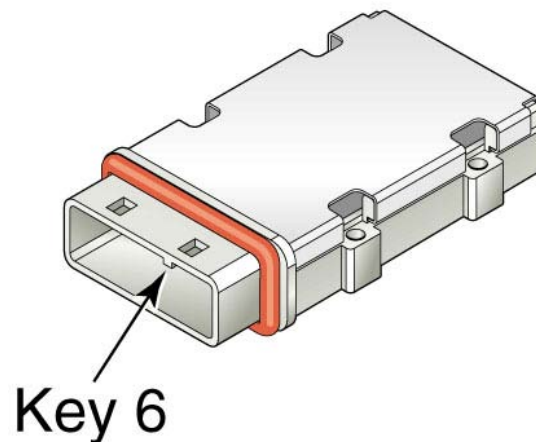


Figure 82 — Mini SAS 4x receptacle connector - enclosure in port

5.2.3.3.2.3 Mini SAS 4x connector pin assignments

Table 49 defines the pin assignments for Mini SAS 4x cable plug connectors (see 5.2.3.3.2.1) and Mini SAS 4x receptacle connectors (see 5.2.3.3.2.2) for applications using one, two, three, or four of the physical links.

Table 49 — Mini SAS 4x connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly ^a				Mating level
	One	Two	Three	Four	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Rx 2+	N/C	N/C	A8	A8	
Rx 2-	N/C	N/C	A9	A9	
Rx 3+	N/C	N/C	N/C	A11	
Rx 3-	N/C	N/C	N/C	A12	
Tx 0+	B2	B2	B2	B2	
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Tx 2+	N/C	N/C	B8	B8	
Tx 2-	N/C	N/C	B9	B9	
Tx 3+	N/C	N/C	N/C	B11	
Tx 3-	N/C	N/C	N/C	B12	
SIGNAL GROUND	A1, A4, A7, A10, A13 B1, B4, B7, B10, B13				First
CHASSIS GROUND	Housing				N/A
^a N/C = not connected					

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the connector when used in a cable assembly.

5.2.4 Cable assemblies

5.2.4.1 SAS internal cable assemblies

5.2.4.1.1 SAS Drive cable assemblies

There are two types of SAS Drive cable assemblies:

- Single-port SAS Drive cable assembly; and
- Dual-port SAS Drive cable assembly.

Both SAS Drive cable assemblies shall use:

- a SAS Drive cable receptacle connector (see 5.2.3.2.1.2) on the SAS target device end; and

- b) a SATA signal cable receptacle connector (see SATA-2) on the SAS initiator device or expander device end.

The power and READY LED signal connection is vendor specific.

A SAS initiator device shall use a SATA host plug connector (see SATA-2) for connection to a SAS Drive cable assembly. The signal assignment for the SAS initiator device or expander device with this connector shall be the same as that defined for a SATA host (see SATA-2).

Figure 83 shows the Single-port SAS Drive cable assembly.

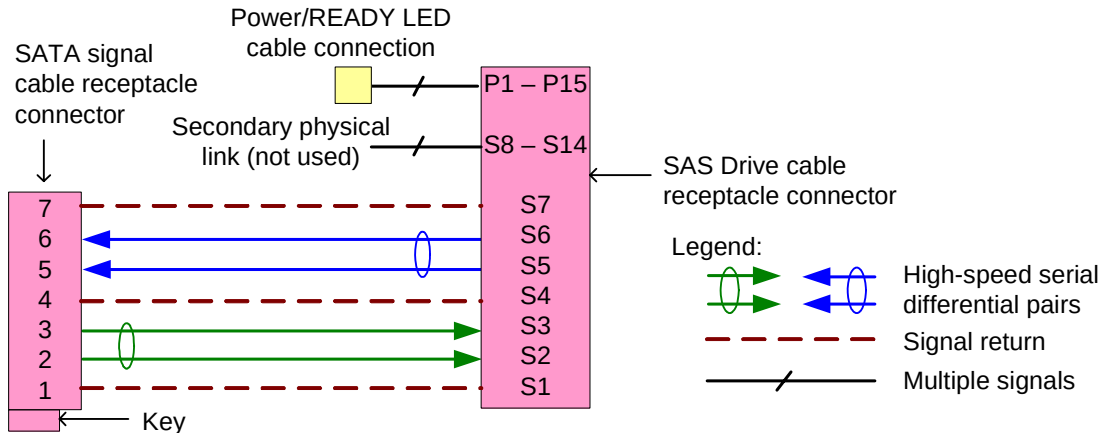


Figure 83 — Single-port SAS Drive cable assembly

Figure 84 shows the Dual-port SAS Drive cable assembly.

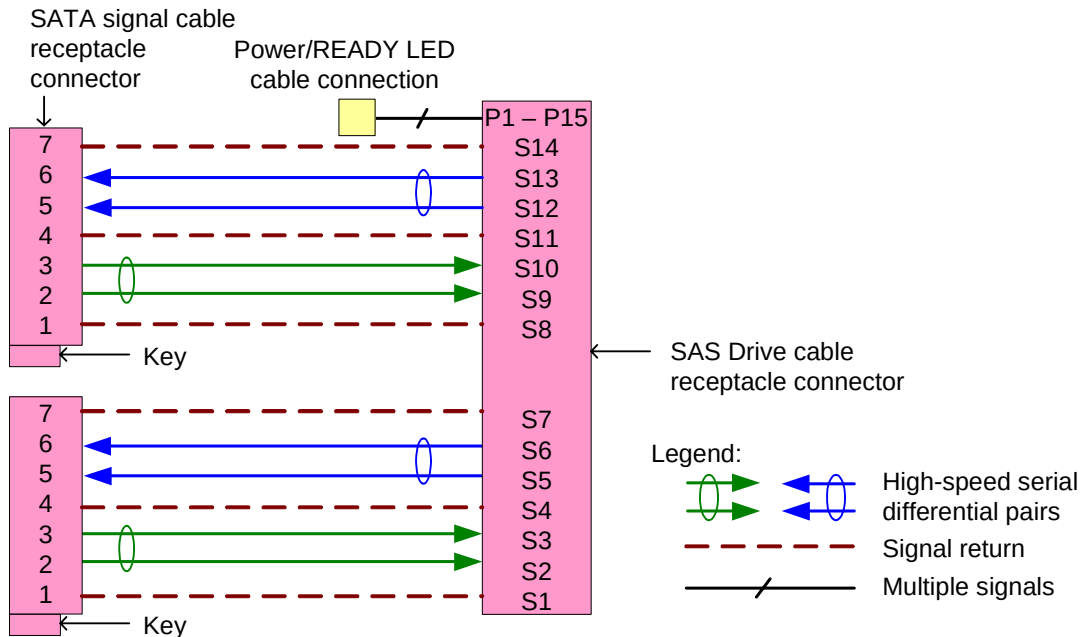


Figure 84 — Dual-port SAS Drive cable assembly

5.2.4.1.2 SAS internal symmetric cable assemblies

5.2.4.1.2.1 SAS internal symmetric cable assemblies overview

There are several types of SAS internal symmetric cable assemblies:

- a) SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) on each end (see 5.2.4.1.2.2);

- b) Mini SAS 4i cable plug connectors (see 5.2.3.3.2.1) on each end (see 5.2.4.1.2.3); and
- c) a SAS 4i cable receptacle connector on one end and a Mini SAS 4i cable plug connector on the other end, with vendor-specific sidebands (see 5.2.4.1.2.4);
- d) a SAS 4i cable receptacle connector on the controller end and a Mini SAS 4i cable plug connector on the backplane end, with sidebands supporting SGPIO (see 5.2.4.1.2.5); and
- e) a Mini SAS 4i cable receptacle connector on the controller end and a SAS 4i cable plug connector on the backplane end, with sidebands supporting SGPIO (see 5.2.4.1.2.6).

In a SAS internal symmetric cable assembly, the Tx signals on one end shall be connected to Rx signals on the other end (e.g., a Tx + of one connector shall connect to an Rx + of the other connector). SAS internal symmetric cable assemblies should be labeled to indicate how many physical links are included (e.g., 1X, 2X, 3X, and 4X on each connector's housing).

5.2.4.1.2.2 SAS internal symmetric cable assembly - SAS 4i

Figure 85 shows the SAS internal symmetric cable assembly with SAS 4i cable receptacle connectors at each end.

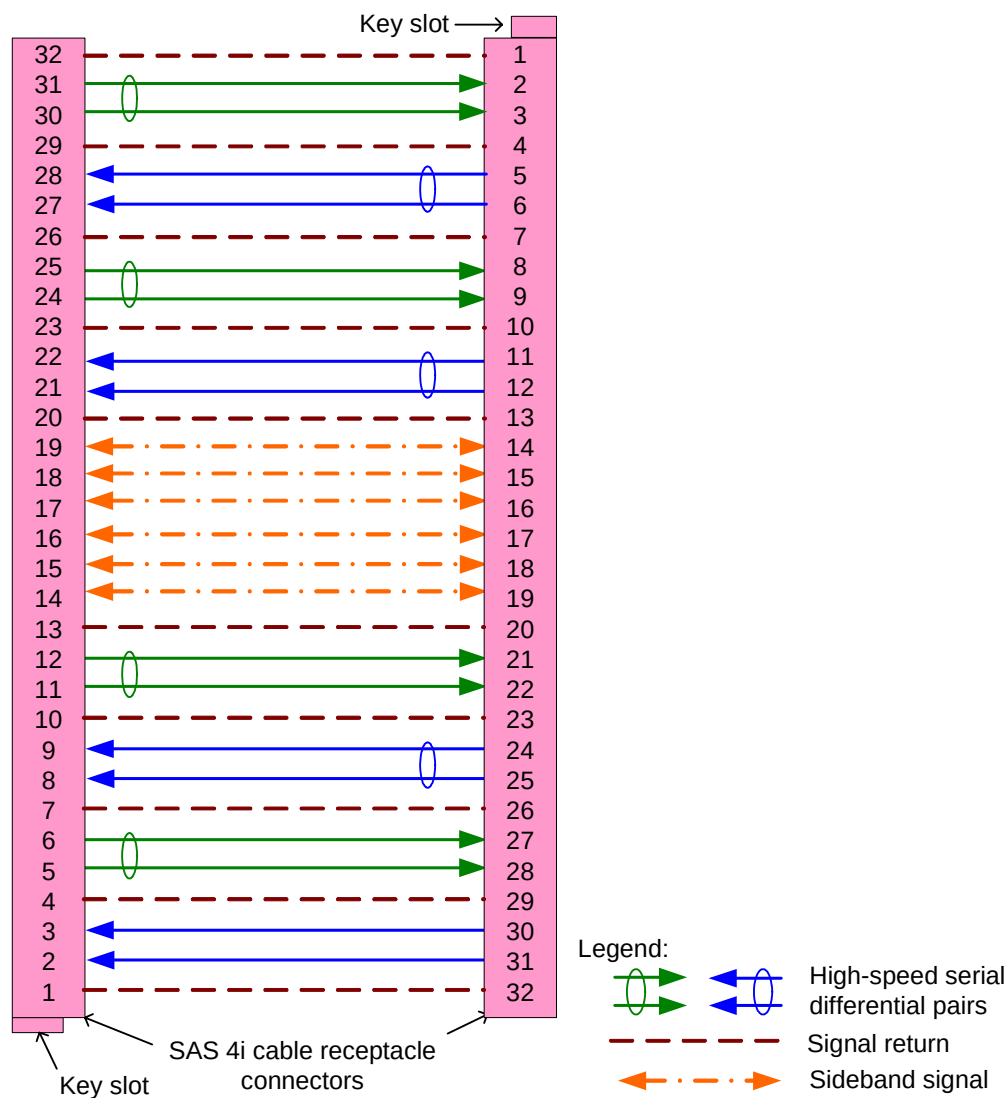


Figure 85 — SAS internal symmetric cable assembly - SAS 4i

In addition to the signal return connections shown in figure 85, the cable assembly may connect one or more of the signal returns together.

For controller-to-backplane applications, the cable assembly may support one to four physical links. SIDEBAND signals on the controller are attached to the corresponding SIDEBAND signals on the backplane (e.g., SIDEBAND0 of the controller is attached to SIDEBAND0 of the backplane).

For controller-to-controller applications, the cable assembly shall support all four physical links and the controllers should use all four physical links, because one controller's physical links 0 and 1 are attached the other controller's physical links 3 and 2, respectively. If both controllers use one or two physical links starting with physical links 0, communication is not possible. If both controllers use physical links 0, 1, and 2, then only communication over physical links 1 and 2 is possible. SIDEBAND signals on one controller are not attached to their corresponding SIDEBAND signals on the other controller (e.g., SIDEBAND0 of one controller is attached to SIDEBAND5 of the other controller).

5.2.4.1.2.3 SAS internal symmetric cable assembly - Mini SAS 4i

Figure 86 shows the SAS internal cable assembly with Mini SAS 4i cable plug connectors at each end.

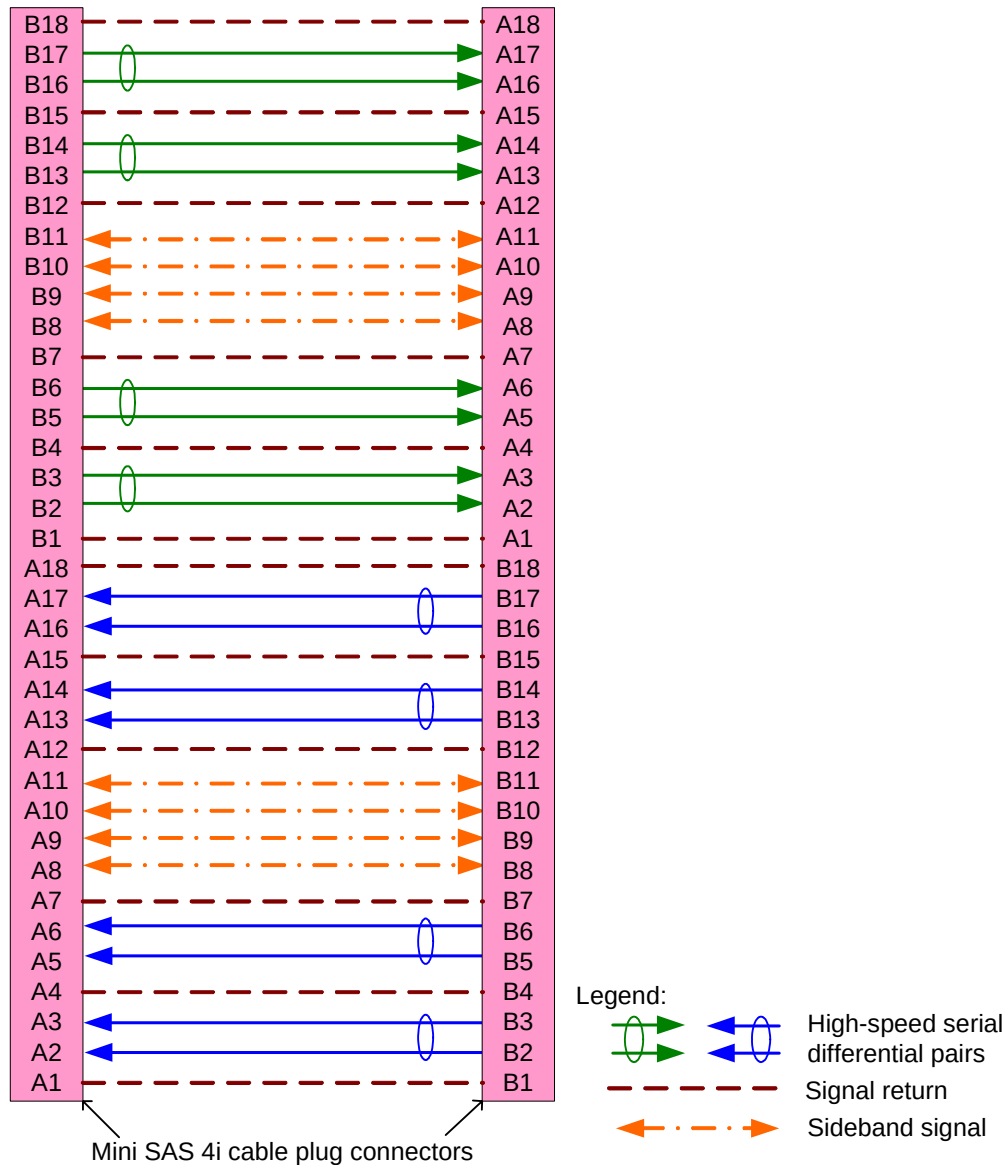


Figure 86 — SAS internal symmetric cable assembly - Mini SAS 4i

In addition to the signal return connections shown in figure 86, the cable assembly may connect one or more of the signal returns together.

The cable assembly may support one to four physical links.

For controller-to-backplane applications, SIDEBAND signals on the controller are attached to the corresponding SIDEBAND signals on the backplane (e.g., SIDEBAND0 of the controller is attached to SIDEBAND0 of the backplane).

For controller-to-controller applications, SIDEBAND signals on one controller are not attached to their corresponding SIDEBAND signals on the other controller (e.g., SIDEBAND0 of one controller is attached to SIDEBAND6 of the other controller).

5.2.4.1.2.4 SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i with vendor-specific sidebands

Figure 87 shows the SAS internal symmetric cable assembly with a SAS 4i cable receptacle connector at one end and a Mini SAS 4i cable plug connector at the other end, with vendor-specific sidebands.

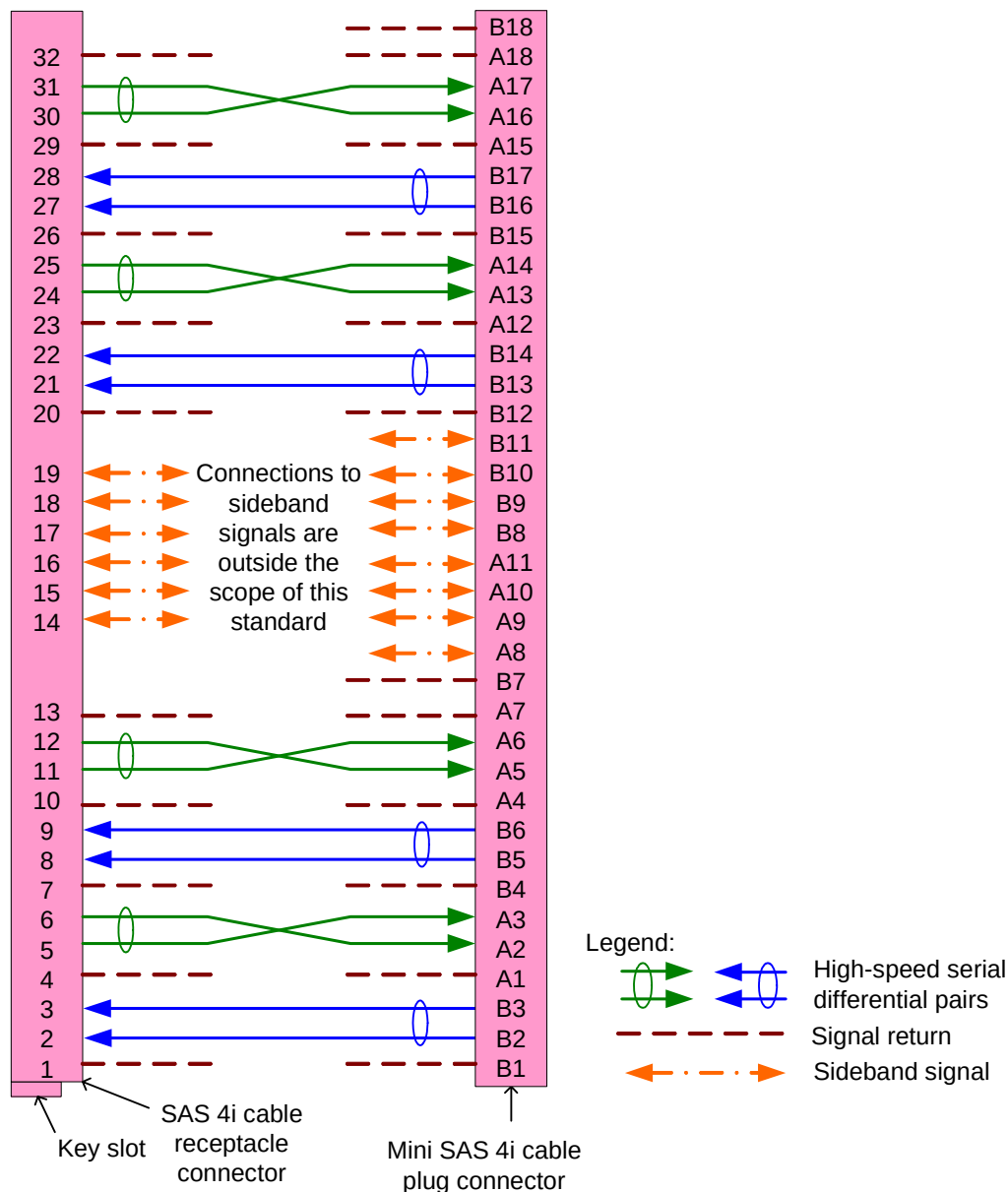


Figure 87 — SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i with vendor-specific sidebands

NOTE 17 - The cable assembly needs different SIDEBAND signal routing based on whether the controller or backplane is using the SAS 4i connector.

The cable assembly shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

For controller-to-backplane applications with the SAS 4i cable receptacle connector on the controller end, the cable assembly may support one to four physical links.

For controller-to-controller applications, the cable assembly may support one to four physical links.

For controller-to-backplane applications with the Mini SAS 4i cable receptacle connector on the controller end, the cable assembly shall support all four physical links and the controller should use all four physical links, because the controller's physical links 0, 1, 2, and 3 are attached to the backplane's physical links 3, 2, 1, and 0, respectively. If both controllers use one or two physical links starting with physical links 0, communication is not possible. If both controllers use physical links 0, 1, and 2, then only communication over physical links 1 and 2 is possible.

5.2.4.1.2.5 SAS internal symmetric cable assembly - SAS 4i controller to Mini SAS 4i backplane with SGPIO

Figure 88 shows the SAS internal symmetric cable assembly with a SAS 4i cable receptacle connector at the controller end and a Mini SAS 4i cable plug connector at the backplane end, with sidebands connected to support SGPIO (see SFF-8485).

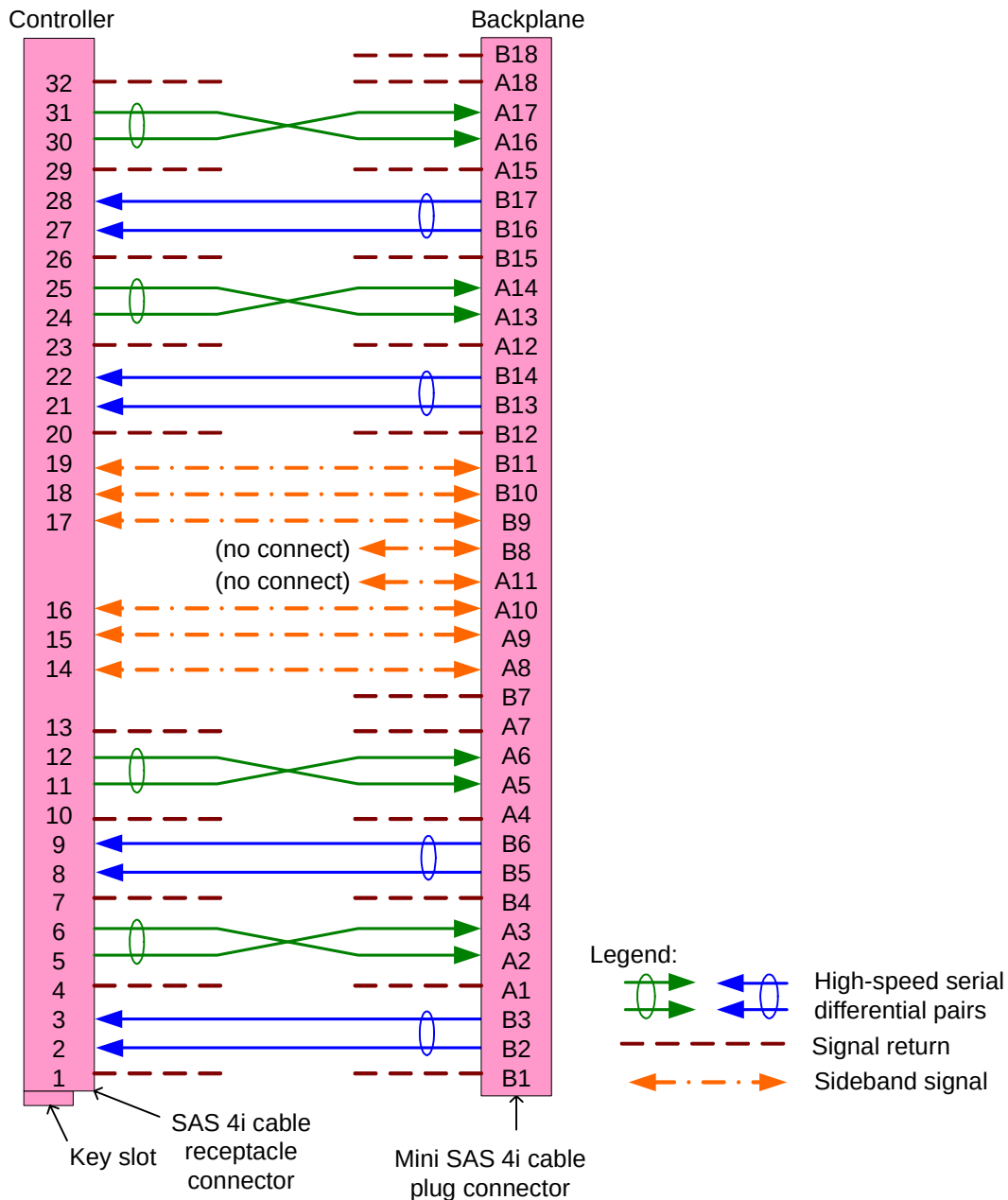


Figure 88 — SAS internal symmetric cable assembly - SAS 4i controller to Mini SAS 4i backplane with SGPIO

The cable assembly shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

The cable assembly may support one to four physical links.

5.2.4.1.2.6 SAS internal symmetric cable assembly - Mini SAS 4i controller to SAS 4i backplane with SGPIO

Figure 89 shows the SAS internal symmetric cable assembly with a Mini SAS 4i cable receptacle connector at the controller end and a SAS 4i cable plug connector at the backplane end, with sidebands connected to support SGPIO (see SFF-8485).

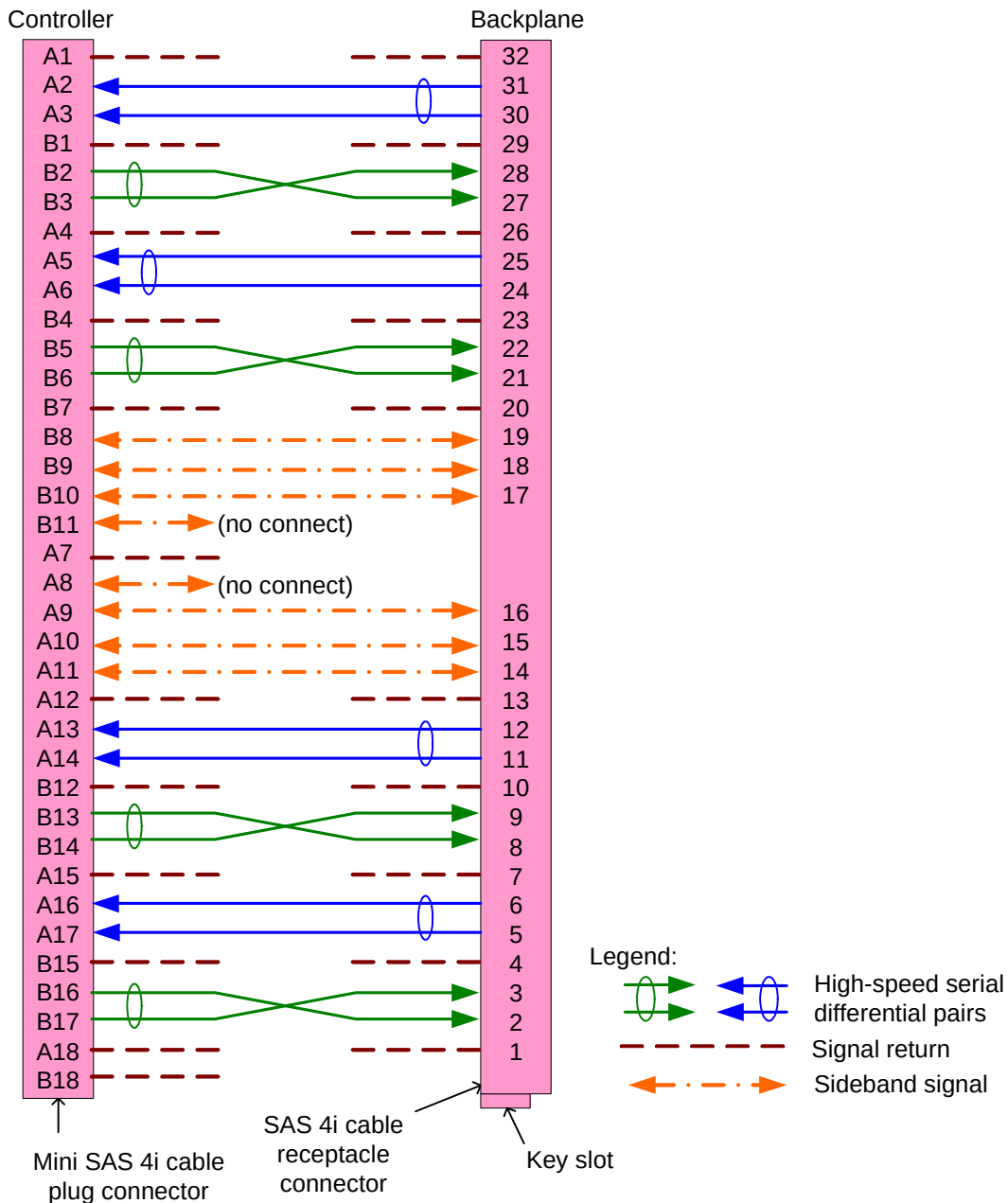


Figure 89 — SAS internal symmetric cable assembly - Mini SAS 4i controller to SAS 4i backplane with SGPIO

The cable assembly shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

The cable assembly may support one to four physical links.

5.2.4.1.3 SAS internal fanout cable assemblies

5.2.4.1.3.1 SAS internal fanout cable assemblies overview

There are several types of SAS internal fanout cable assemblies:

- a) SAS internal controller-based fanout cable assemblies (see 5.2.4.1.3.2) with:
 - A) a SAS 4i cable receptacle connector on one end (i.e., the controller end) and four SAS Drive cable receptacle connectors on the other end (i.e., the backplane end); and
 - B) a Mini SAS 4i cable plug connector on one end (i.e., the controller end) and four SAS Drive cable receptacle connectors on the other end (i.e., the backplane end);and
- b) SAS internal backplane-based fanout cable assemblies (see 5.2.4.1.3.3):
 - A) four SATA signal cable receptacle connectors on one end (i.e., the controller end) and a SAS 4i cable receptacle connector on the other end (i.e., the backplane end); and
 - B) four SATA signal cable receptacle connectors on one end (i.e., the controller end) and a Mini SAS 4i cable plug connector on the other end (i.e., the backplane end).

In a SAS internal fanout symmetric cable assembly, the Tx signals on one end shall be connected to Rx signals on the other end (e.g., a Tx + of one connector shall connect to an Rx + of the other connector).

5.2.4.1.3.2 SAS internal controller-based fanout cable assemblies

Figure 90 shows the SAS internal controller-based fanout cable assembly with a SAS 4i cable receptacle connector at the controller end.

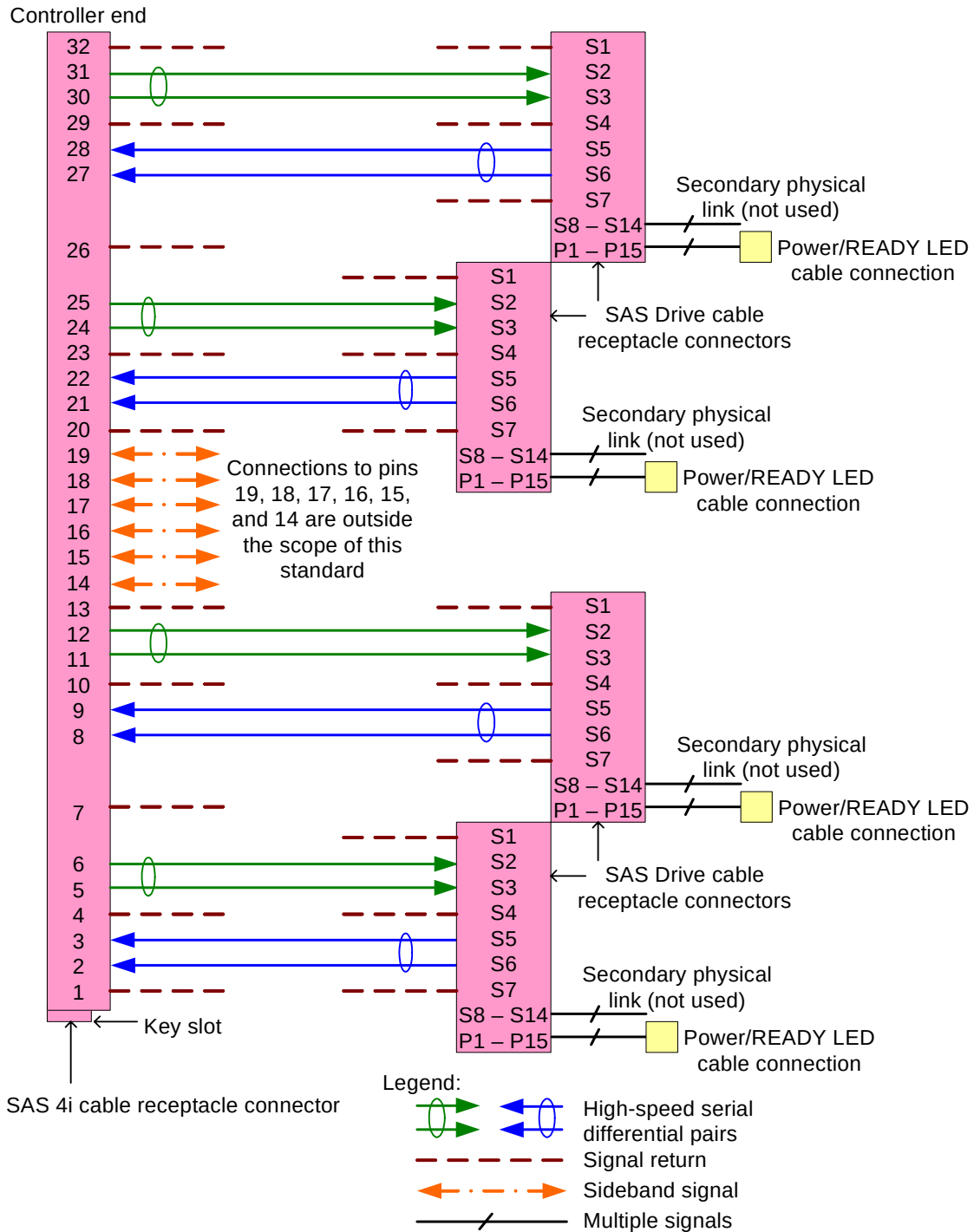


Figure 90 — SAS internal controller-based fanout cable assembly - SAS 4i

The cable assembly shown in figure 90 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

Figure 91 shows the SAS internal controller-based fanout cable assembly with a Mini SAS 4i cable plug connector at the controller end.

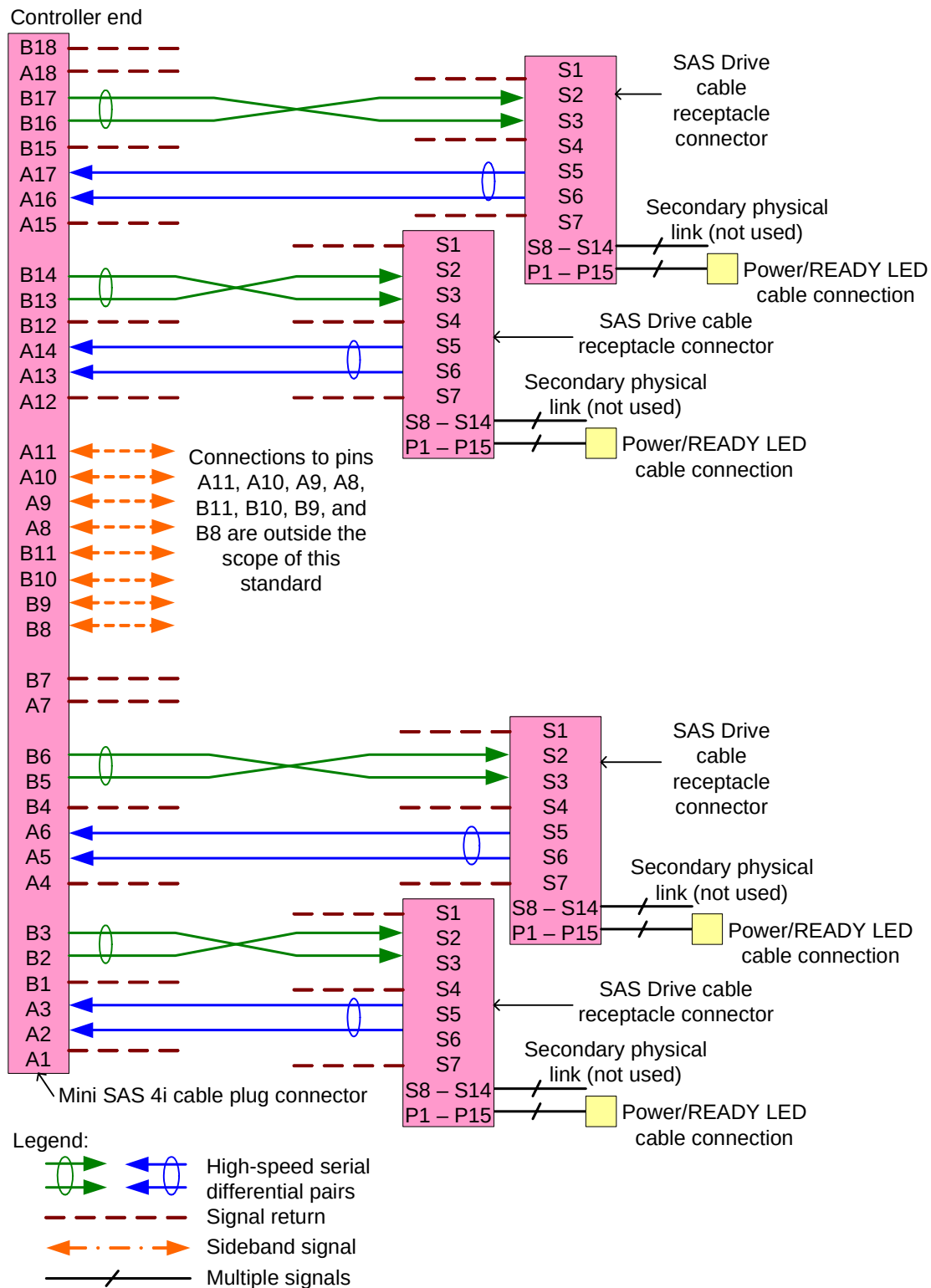


Figure 91 — SAS internal controller-based fanout cable assembly - Mini SAS 4i

The cable assembly shown in figure 91 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.4.1.3.3 SAS internal backplane-based fanout cable assemblies

Figure 92 shows the SAS internal backplane-based fanout cable assembly with the SAS 4i cable receptacle connector.

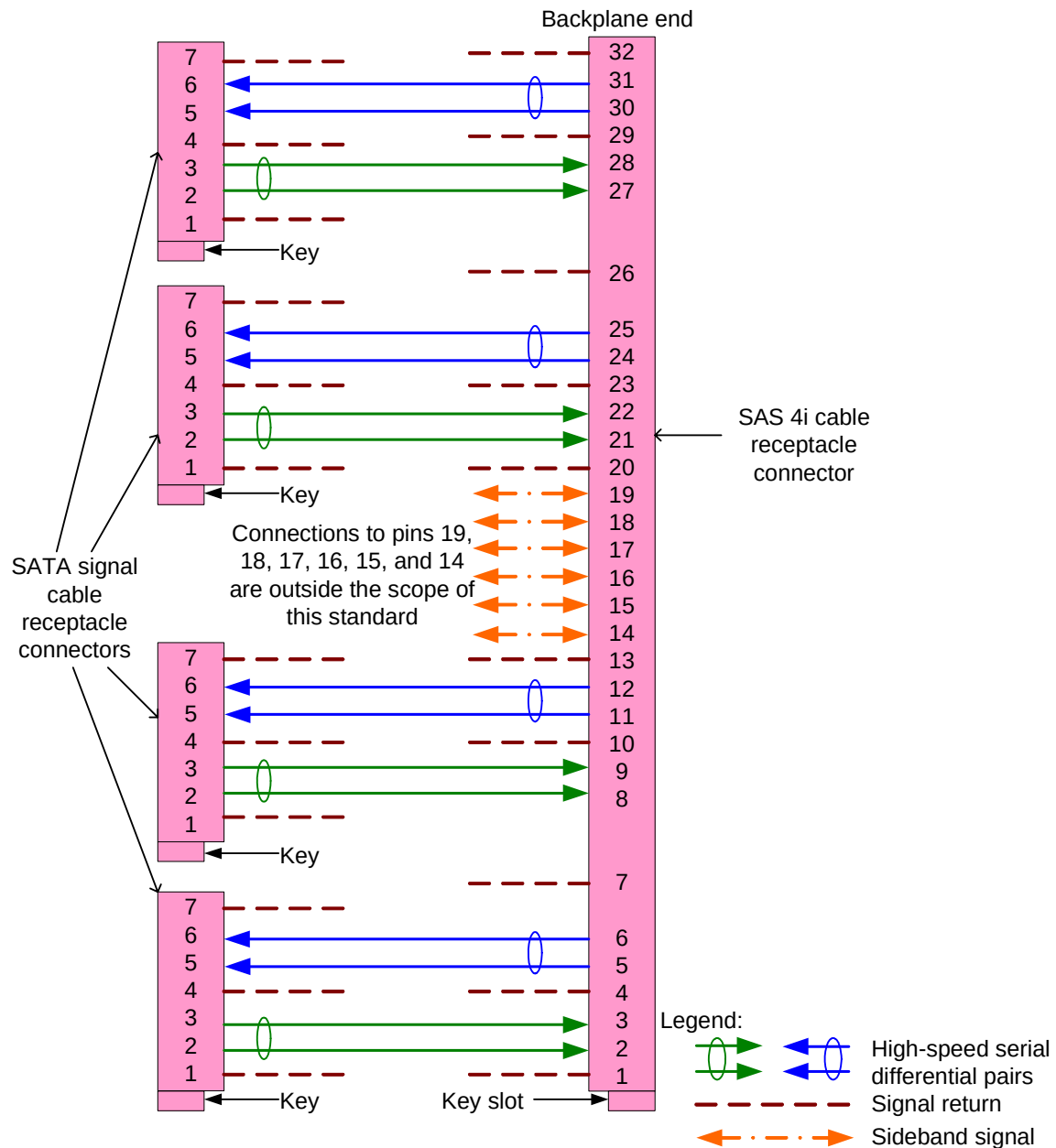


Figure 92 — SAS internal backplane-based fanout cable assembly - SAS 4i

The cable assembly shown in figure 92 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

Figure 93 shows the SAS internal backplane-based fanout cable assembly with the Mini SAS 4i cable receptacle connector.

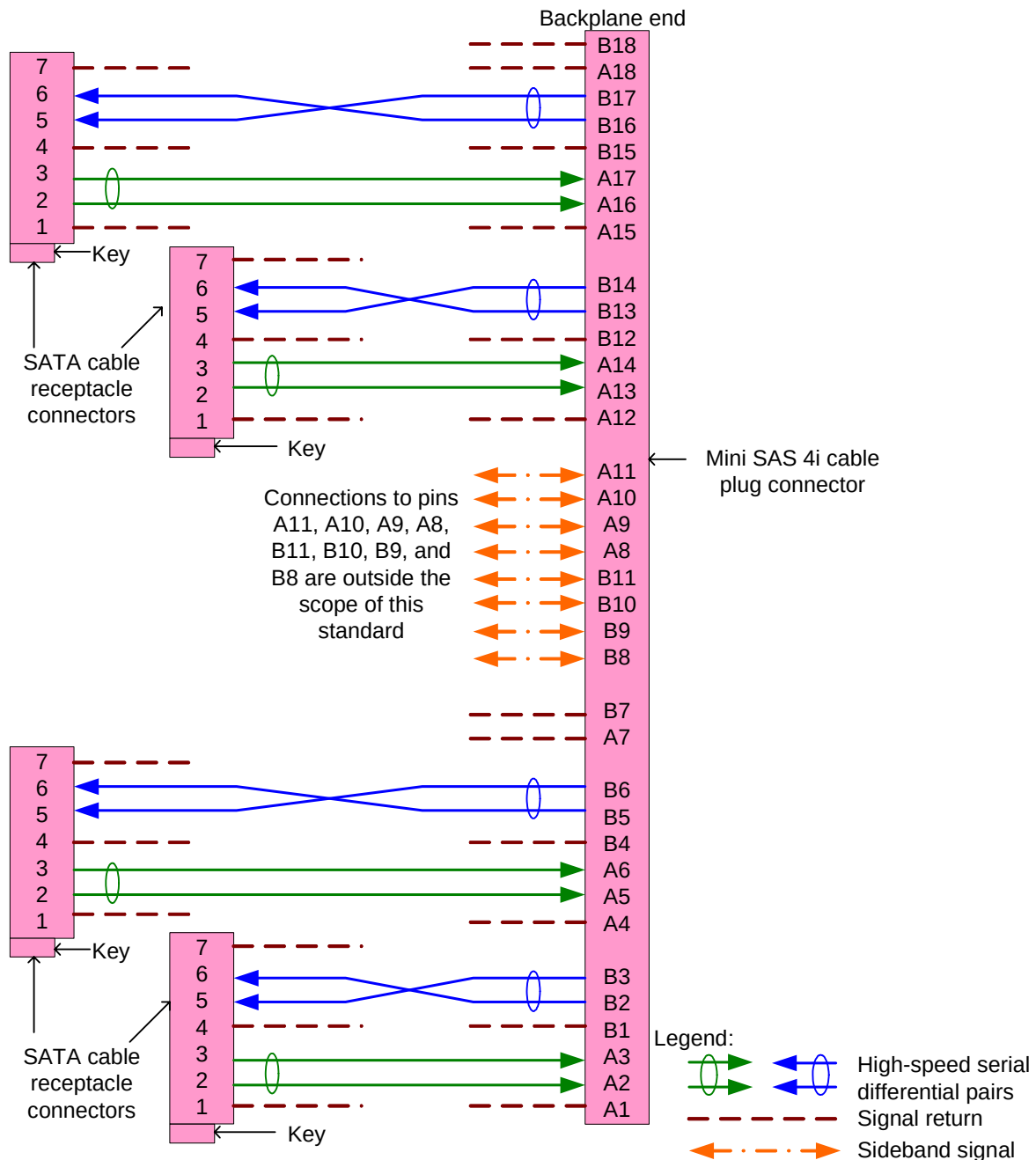


Figure 93 — SAS internal backplane-based fanout cable assembly - Mini SAS 4i

The cable assembly shown in figure 93 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.4.2 SAS external cable assemblies

5.2.4.2.1 SAS external cable assemblies overview

There are several types of SAS external cable assemblies:

- SAS 4x cable plug connector (see 5.2.3.3.1.1) at each end (see 5.2.4.2.2);
- Mini SAS 4x cable plug connector (see 5.2.3.3.2.1) at each end (see 5.2.4.2.3); and

- c) SAS 4x cable plug connector at one end and Mini SAS 4x cable plug connector at the other end (see 5.2.4.2.4).

SAS external cable assemblies do not include power or the READY LED signal.

Although the connector always supports four physical links, a SAS external cable assembly may support one, two, three, or four physical links. SAS external cable assemblies should be labeled to indicate how many physical links are included (e.g., 1X, 2X, 3X, and 4X on each connector's housing)

The Tx signals on one end shall be connected to the corresponding Rx signals of the other end (e.g., Tx 0+ of one connector shall be connected to Rx 0+ of the other connector).

Signal returns shall not be connected to CHASSIS GROUND in the cable assembly.

In addition to the SAS icon (see Annex M), additional icons are defined for external connectors to guide users into making compatible attachments (i.e., not attaching expander device table routing phys to expander device table routing phys in externally configurable expander devices, which is not allowed by this standard).

Connectors that have one or more matching icons are intended to be attached. Connectors that do not have a matching icon should not be attached together.

One end of the SAS external cable assembly shall support being attached to an end device, an enclosure out port, or an enclosure universal port. The other end of the SAS external cable assembly shall support being attached to an end device, an enclosure in port, or an enclosure universal port. If a SAS 4x cable plug connector is used, it should include icons as defined in 5.2.3.3.1.1. If a Mini SAS 4x cable plug connector is used, it shall include icons and key slots as defined in 5.2.3.3.2.1.

5.2.4.2.2 SAS external cable assembly - SAS 4x

Figure 94 shows the SAS external cable assembly with SAS 4x cable plug connectors at each end.

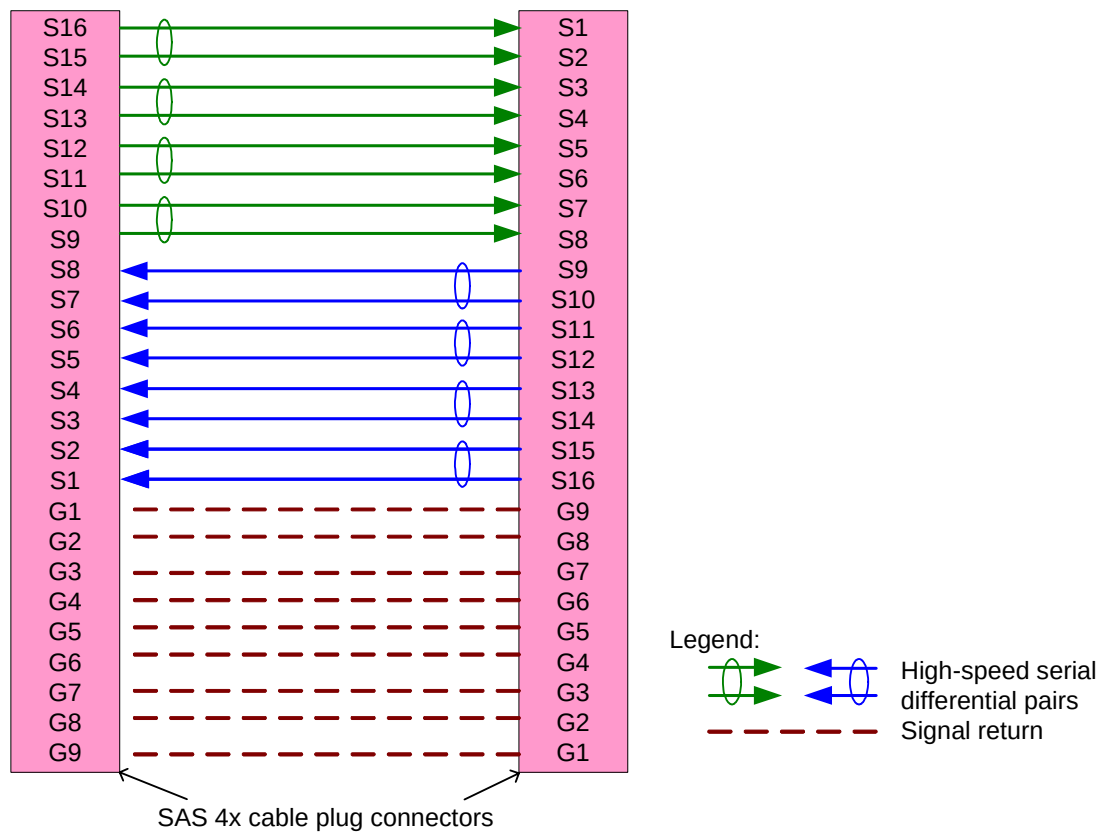


Figure 94 — SAS external cable assembly - SAS 4x

In addition to the signal return connections shown in figure 94, the cable assembly may connect one or more of the signal returns together.

5.2.4.2.3 SAS external cable assembly - Mini SAS 4x

Figure 95 shows the SAS external cable assembly with Mini SAS 4x cable plug connectors at each end.

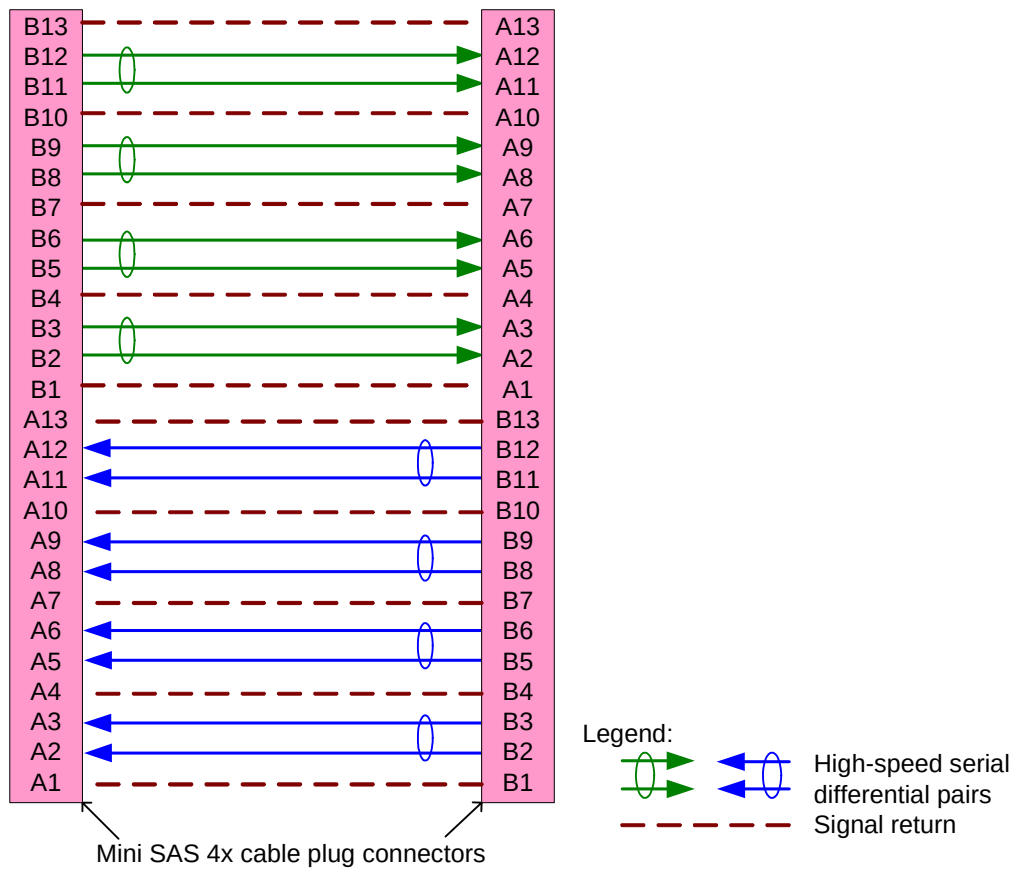


Figure 95 — SAS external cable assembly - Mini SAS 4x

In addition to the signal return connections shown in figure 95, the cable assembly may connect one or more of the signal returns together.

Figure 96 shows the icons and key slots in the SAS external cable assembly with Mini SAS 4x cable plug connectors at each end.

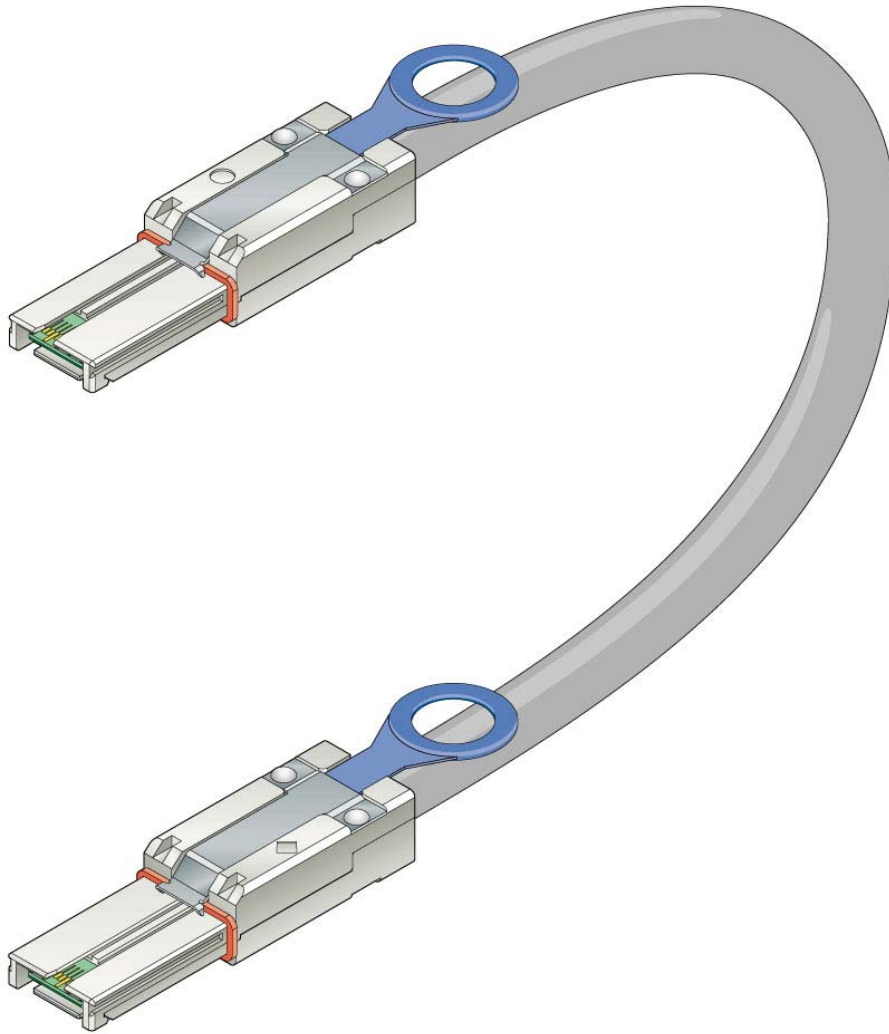


Figure 96 — SAS external cable assembly with Mini SAS 4x cable plug connectors

Although the topology is supported by this standard, a SAS external cable assembly with Mini SAS 4x connectors on each end that attaches an enclosure in port to another enclosure in port is not defined by this standard.

5.2.4.2.4 SAS external cable assembly - SAS 4x to Mini SAS 4x

Figure 97 shows the SAS external cable assembly with a SAS 4x cable plug connector at one end and a Mini SAS 4x cable plug connector at the other end.

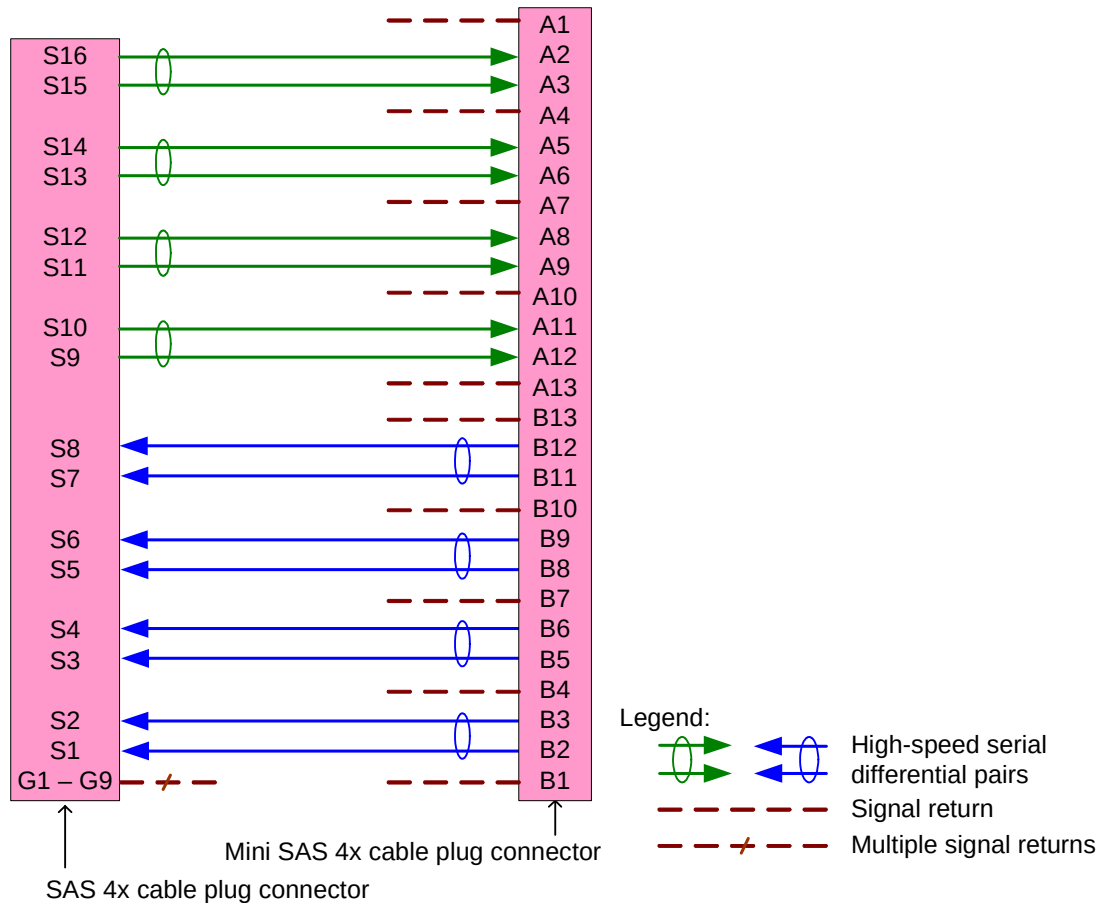


Figure 97 — SAS external cable assembly - SAS 4x to Mini SAS 4x

The cable assembly shown in figure 97 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.5 Backplanes

SAS backplane designs should follow the recommendations in SFF-8460.

5.2.6 Cable assembly and backplane specifications

Table 50 defines requirements for internal cable assemblies using SAS Drive connectors and backplanes.

Table 50 — Requirements for internal cable assemblies using SAS Drive connectors and backplanes

Requirement ^{a, b}	Units	1,5 Gbps	3 Gbps
Bulk cable or backplane: ^{c, d}			
Differential impedance	ohm	100 ± 10	
Maximum differential impedance imbalance ^e	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors: ^{d, f}			
Differential impedance	ohm	100 ± 15	
Maximum differential impedance imbalance ^e	ohm	5	
^a All measurements are made through mated connector pairs.			
^b The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included.			
^c The impedance measurement identifies the impedance mismatches present in the bulk cable or backplane when terminated in its characteristic impedance. This measurement excludes mated connectors at both ends of the bulk cable or backplane, when present, but includes any intermediate connectors or splices.			
^d Where the bulk cable or backplane has an electrical length of > 4 ns the procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the impedance.			
^e The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.			
^f The mated connectors measurement applies only to the mated connector pair at each end, as applicable.			

Table 51 defines requirements for SAS internal cable assemblies using SAS 4i cable receptacle connectors or Mini SAS 4i cable plug connectors.

Table 51 — Requirements for SAS internal cable assemblies using SAS 4i or Mini SAS 4i

Requirement ^{a, b}	Units	1,5 Gbps	3 Gbps
Bulk cable: ^{c, d}			
Differential impedance	ohm	100 ± 10	
Maximum differential impedance imbalance ^e	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors: ^{d, f}			
Differential impedance	ohm	100 ± 15	
Maximum differential impedance imbalance ^e	ohm	5	
Cable assembly: ^g			
Maximum insertion loss ^h	dB	6	
Maximum intra-pair skew ^{i, j}	ps	10	
^a All measurements are made through mated connector pairs. ^b The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included. ^c The impedance measurement identifies the impedance mismatches present in the bulk cable when terminated in its characteristic impedance. This measurement excludes mated connectors at both ends of the bulk cable, when present, but includes any intermediate connectors or splices. ^d Where the bulk cable has an electrical length of > 4 ns, the procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the impedance. ^e The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals. ^f The mated connectors measurement applies only to the mated connector pair at each end, as applicable. ^g The internal cable assembly is part of a TxRx connection that complies with the requirements for intra-enclosure compliance points defined in 5.3. ^h The range for this frequency domain measurement is 10 MHz to 4 500 MHz. ⁱ The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result. ^j The procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the intra-pair skew.			

Table 52 defines additional requirements for cable assemblies using SAS 4i cable receptacle connectors.

Table 52 — Additional requirements for cable assemblies using SAS 4i

Requirement ^{a, b, c, d}	Units	1,5 Gbps	3 Gbps
Maximum near-end crosstalk on any of the following (adjacent) signal pairs: Rx 0/Tx 0, Tx 0/Rx 1, Rx 1/Tx 1, Rx 2/Tx 2, Tx 2/Rx 3, and Rx 3/Tx 3	dB	-33	
Maximum near-end crosstalk on any of the following signal pairs: Rx 0/Rx 1, Rx 0/Tx 1, Tx 0/Tx 1, Rx 2/Rx 3, Rx 2/Tx 3, and Tx 2/Tx 3	dB	-45	
Maximum near-end crosstalk on all other signal pairs	dB	-50	
^a All measurements are made through mated connector pairs. ^b The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included. ^c The range for this frequency domain measurement is 10 MHz to 4 500 MHz. ^d The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

Table 53 defines additional requirements for cable assemblies using Mini SAS 4i cable plug connectors.

Table 53 — Additional requirements for cable assemblies using Mini SAS 4i

Requirement ^{a, b, c, d}	Units	1,5 Gbps	3 Gbps
Maximum near-end crosstalk for each receive pair	dB	-26	
^a All measurements are made through mated connector pairs. ^b Determine all valid aggressor/victim near-end crosstalk transfer modes. Over the complete frequency range of this measurement, determine the sum of the crosstalk transfer ratios, measured in the frequency domain, of all crosstalk transfer modes. To remove unwanted bias due to test fixture noise, magnitudes less than -50 dB (e.g., -60 dB) at all frequencies may be ignored. The following equation details the summation process of the four valid near-end crosstalk sources. All NEXT values expressed in dB format in a passive transfer network shall have negative dB magnitude. $\text{TotalNEXT}(f) = 10 \times \log \sum_{1}^{4} 10^{(\text{NEXT}(f)/10)}$ ^c The range for this frequency domain measurement is 10 MHz to 4 500 MHz. ^d The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

Table 54 defines requirements for external cable assemblies.

Table 54 — Requirements for external cable assemblies

Requirement ^{a, b}	Units	1,5 Gbps	3 Gbps
Bulk cable: ^{c, d}			
Differential impedance	ohm	100 ± 5	
Maximum differential impedance imbalance ^e	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors:			
Differential impedance ^{f, d}	ohm	100 ± 10	
Cable assembly: ^g			
Maximum insertion loss	See 5.3.3		
Maximum rise time ^{h, i}	ps	150	
Maximum ISI ^j	ps	60	
Maximum intra-pair skew ^{h, k}	ps	50	
^a All measurements are made through mated connector pairs. ^b The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included. ^c The impedance measurement identifies the impedance mismatches present in the bulk cable when terminated in its characteristic impedance. This measurement excludes mated connectors at both ends of the bulk cable, when present, but includes any intermediate connectors or splices. ^d Where the bulk cable has an electrical length of > 4 ns, the procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the impedance. ^e The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals. ^f The mated connectors measurement applies only to the mated connector pair at each end, as applicable. ^g The external cable assembly is part of a TxRx connection that complies with the requirements for inter-enclosure compliance points as defined in 5.3. ^h The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result. ⁱ Connect the TDR step impulse response generators to the near end of the cable assembly and measure the output rise time at the far end. The input rise time shall be no higher than 35 ps. ^j Measured DJ at the far end of the cable assembly under test using a lone bit pattern (see table 306 in 10.2.9.2) at 3 Gbps. ^k The procedure detailed in SFF-8410, or an equivalent procedure, shall be used to determine the intra-pair skew.			

Table 55 defines additional requirements for external cable assemblies using SAS 4x cable plug connectors.

Table 55 — Additional requirements for external cable assemblies using SAS 4x

Requirement ^{a, b, c}	Units	1,5 Gbps	3 Gbps
Frequency domain measurement range 10 MHz to 2 250 MHz:			
Maximum near-end crosstalk from any single aggressor pair offset by one position (i.e., adjacent)(e.g., Rx 0/Rx 1)	dB	-30	
Maximum near-end crosstalk from any single aggressor pair offset by two positions (e.g., Rx 0/Rx 2)	dB	-36	
Maximum near-end crosstalk from any single aggressor pair offset by more than two positions (e.g., Rx 0/Rx 3)	dB	-40	
Frequency domain measurement range 2 250 Mhz to 4 500 MHz:			
Maximum near-end crosstalk from any single aggressor pair offset by one position (i.e., adjacent)(e.g., Rx 0/Rx 1)	dB	-24	
Maximum near-end crosstalk from any single aggressor pair offset by two positions (e.g., Rx 0/Rx 2)	dB	-30	
Maximum near-end crosstalk from any single aggressor pair offset by more than two positions (e.g., Rx 0/Rx 3)	dB	-40	
^a All measurements are made through mated connector pairs. ^b The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included. ^c The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

Table 56 defines additional requirements for external cable assemblies using Mini SAS 4x cable plug connectors.

Table 56 — Additional requirements for external cable assemblies using Mini SAS 4x

Requirement ^{a, b, c, d}	Units	1,5 Gbps	3 Gbps
Maximum near-end crosstalk for each receive pair	dB	-26	
^a All measurements are made through mated connector pairs. ^b Determine all valid aggressor/victim near-end crosstalk transfer modes. Over the complete frequency range of this measurement, determine the sum of the crosstalk transfer ratios, measured in the frequency domain, of all crosstalk transfer modes. To remove unwanted bias due to test fixture noise, magnitudes less than -50 dB (e.g., -60 dB) at all frequencies may be ignored. The following equation details the summation process of the four valid near-end crosstalk sources. All NEXT values expressed in dB format in a passive transfer network shall have negative dB magnitude. $\text{TotalNEXT}(f) = 10 \times \log \sum_{1}^{4} 10^{(\text{NEXT}(f)/10)}$ ^c The range for this frequency domain measurement is 10 MHz to 4 500 MHz. ^d The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

5.3 Transmitter and receiver device electrical characteristics

5.3.1 Compliance points

Signal behavior at separable connectors requires compliance with signal characteristics defined by this standard only if the connectors are identified as compliance points by the supplier of the parts that contain the candidate compliance point.

Signal characteristics for compliance points are measured at physical positions called probe points in a test load (see 5.3.2). Measurements at the probe points in a test load approximate measurements at the compliance point in the actual TxRx connection. Some components in the test load may be de-embedded as described in B.4.

Table 57 lists the compliance points.

Table 57 — Compliance points

Compliance point	Type	Description
IT	intra-enclosure (i.e., internal)	The signal from a transmitter device (see 3.1.280), as measured at probe points in a test load attached with an internal connector.
IR	intra-enclosure (i.e., internal)	The signal going to a receiver device (see 3.1.179), as measured at probe points in a test load attached with an internal connector.
CT	inter-enclosure (i.e., cabinet)	The signal from a transmitter device, as measured at probe points in a test load attached with an external connector.
CR	inter-enclosure (i.e., cabinet)	The signal going to a receiver device, as measured at probe points in a test load attached with an external connector.

Figure 98 shows the locations of the CT and CR compliance points using a SAS 4x or Mini SAS 4x cable assembly, and shows how two of the compliance points are tested using test loads (see 5.3.2).

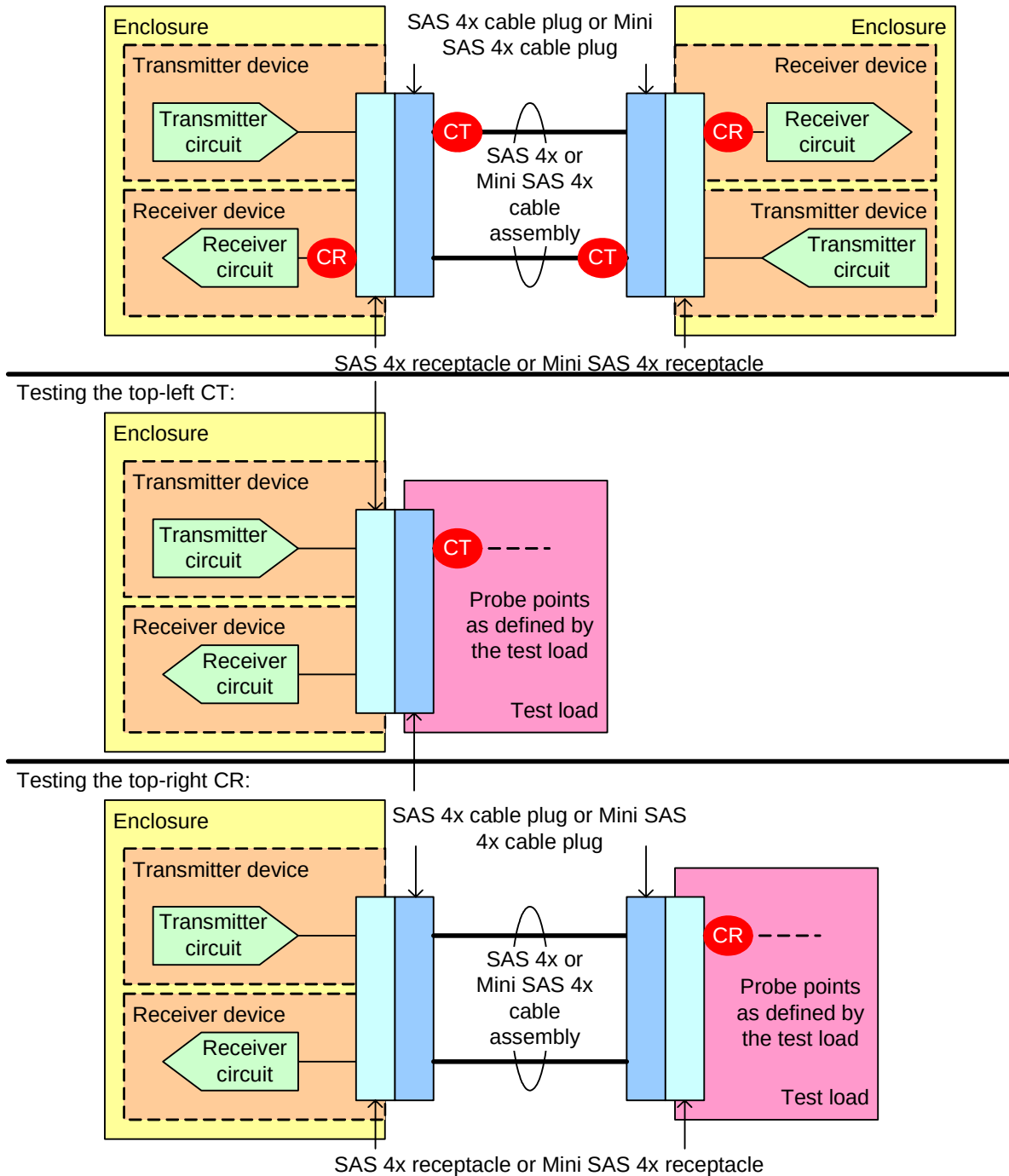


Figure 98 — SAS 4x and Mini SAS 4x cable assembly CT and CR compliance points

Figure 99 shows the locations of the IT and IR compliance points using a backplane with a SAS Drive backplane receptacle (see 5.2.3.2.1.3) that is not using SATA, and shows how the compliance points are tested using test loads (see 5.3.2).

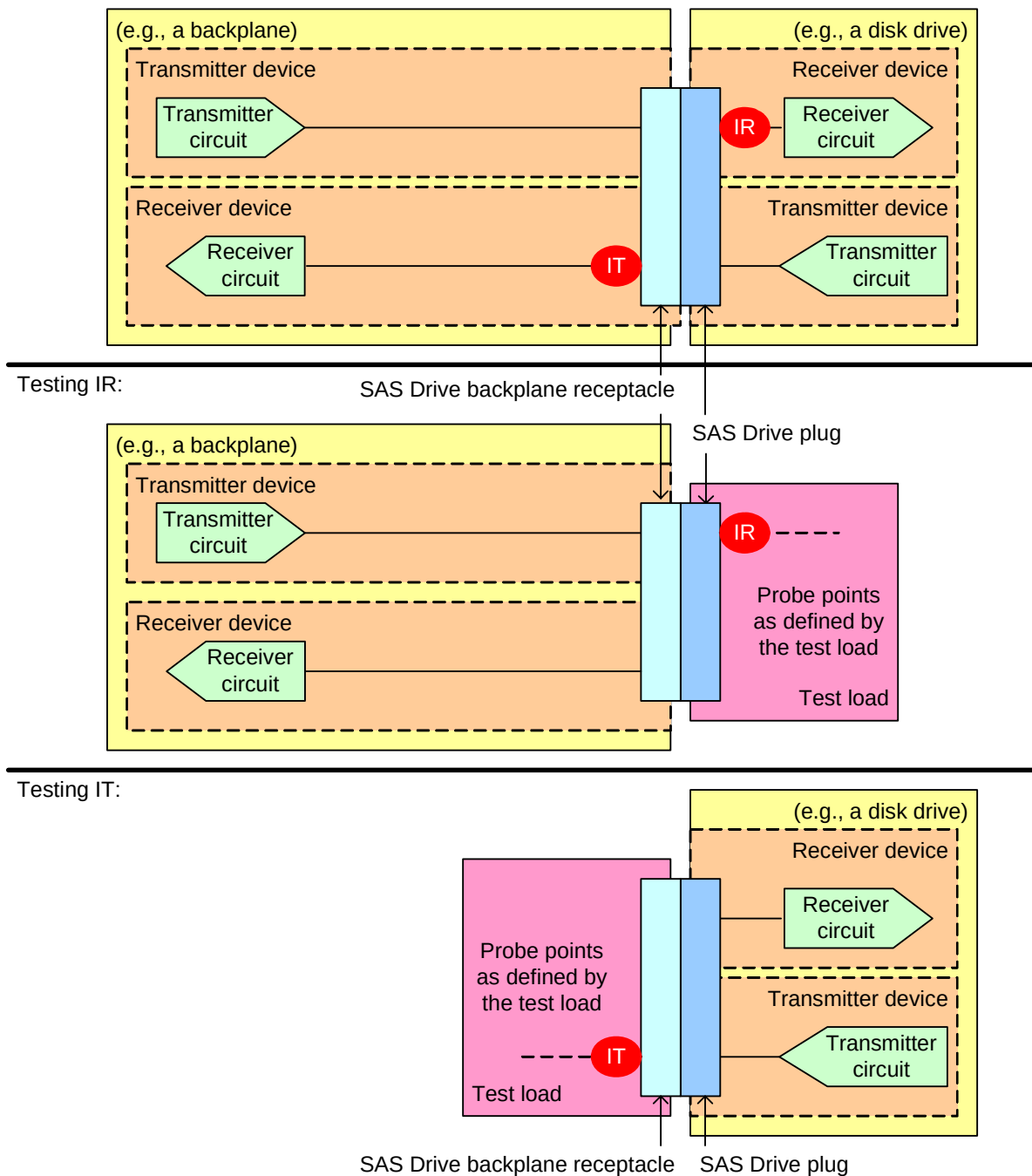


Figure 99 — Backplane IT and IR compliance points

If the backplane supports SATA, there are no IT or IR compliance points. SATA defines the signal characteristics that the SATA phy delivers and that the SAS backplane is required to deliver to the SATA device, as shown in figure 100.

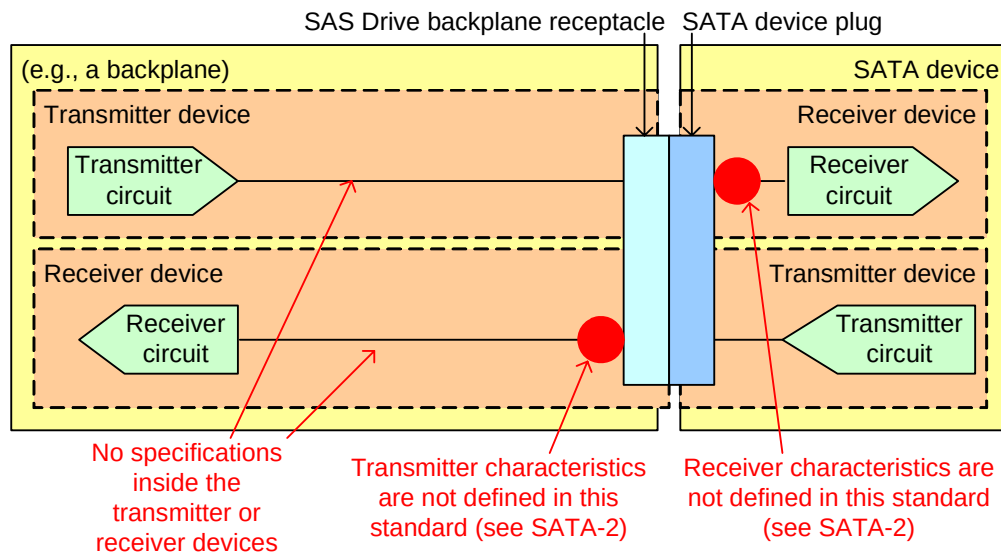


Figure 100 — Backplane compliance points with SATA phy attached

Figure 101 shows the locations of the IT and IR compliance points using a SAS 4i or Mini SAS 4i cable assembly, and shows how two of the compliance points are tested using test loads (see 5.3.2).

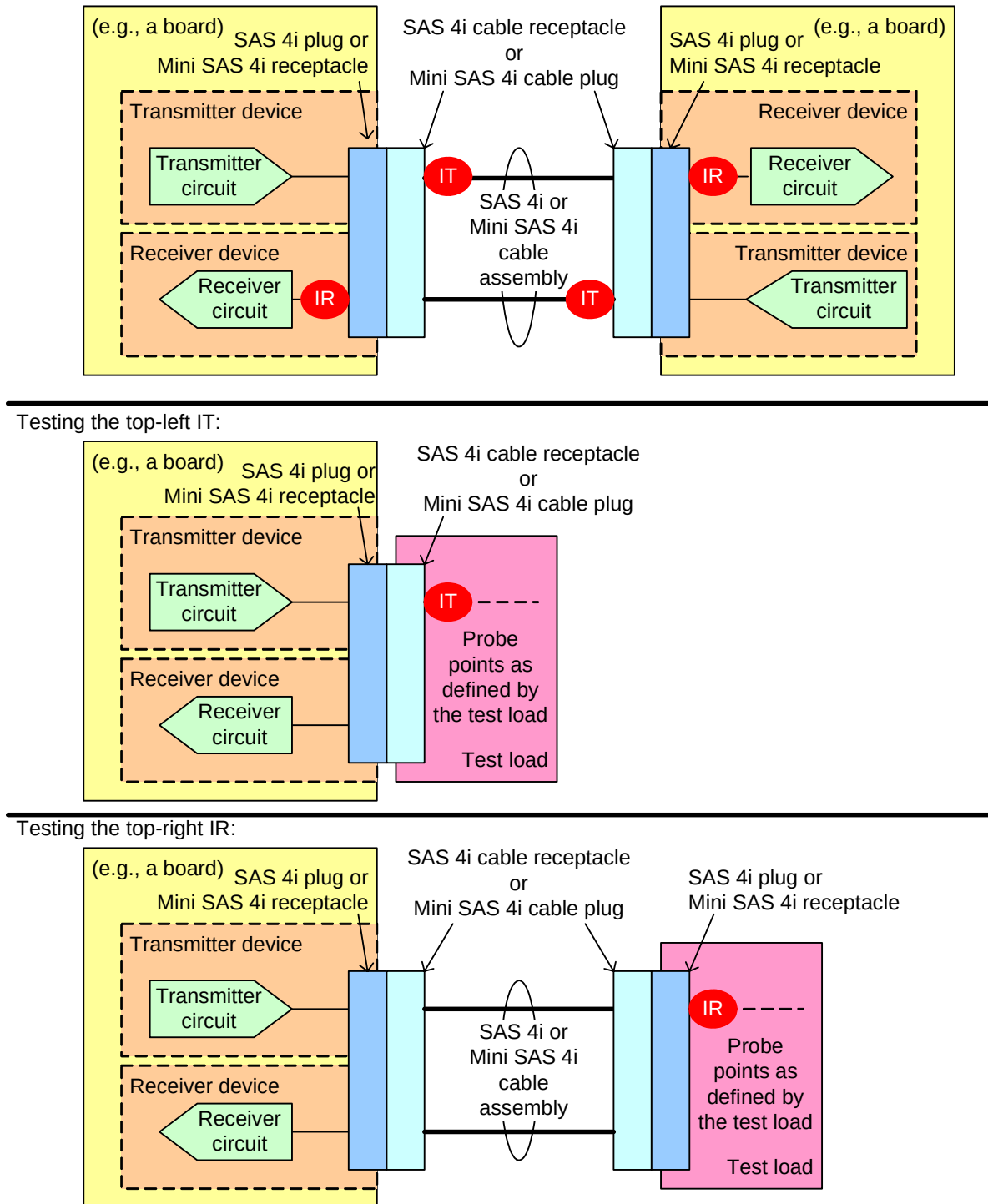


Figure 101 — SAS 4i and Mini SAS 4i cable assembly IT and IR compliance points

Figure 102 shows the locations of the IT and IR compliance points using a SAS 4i cable and a backplane, where the backplane is not attached to a SATA device, and shows how two of the compliance points are tested using test loads (see 5.3.2).

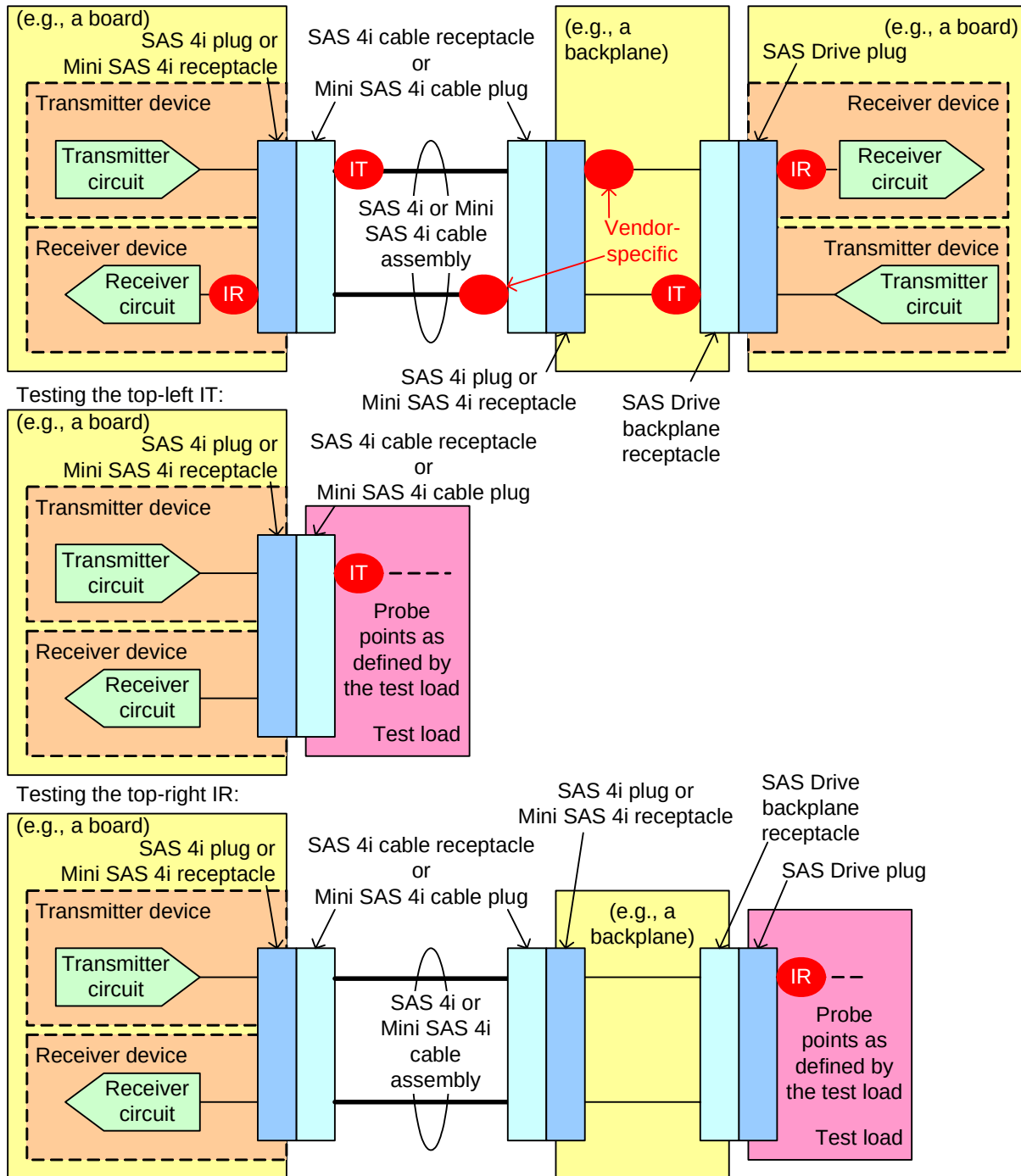


Figure 102 — SAS 4i and Mini SAS 4i cable and backplane IT and IR compliance points

Figure 103 shows the locations of the IT and IR compliance points using a SAS 4i cable and a backplane, where the backplane supports being attached to a SATA device. There are no IT and IR compliance points at the SAS Drive backplane receptacle connector when a SATA device is attached; SATA defines the signal characteristics that the SATA device delivers and that the SAS backplane is required to deliver to the SATA device. There are compliance points at the SAS 4i connector, however.

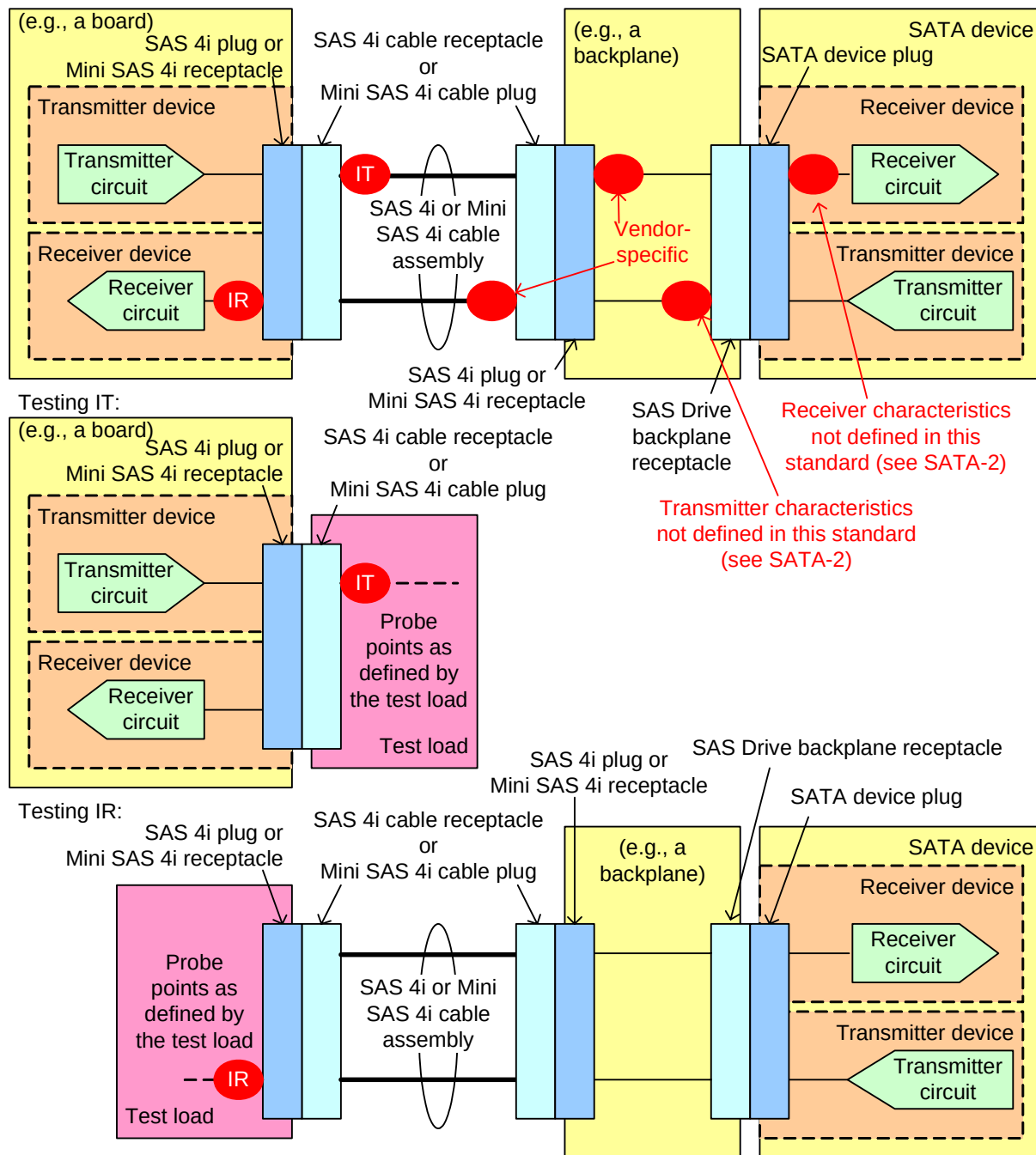


Figure 103 — Internal cable and backplane IT and IR compliance points with SATA device attached

Figure 104 shows the locations of the IT and IR compliance points using an internal cable. It also shows how two of the compliance points are tested using test loads (see 5.3.2).

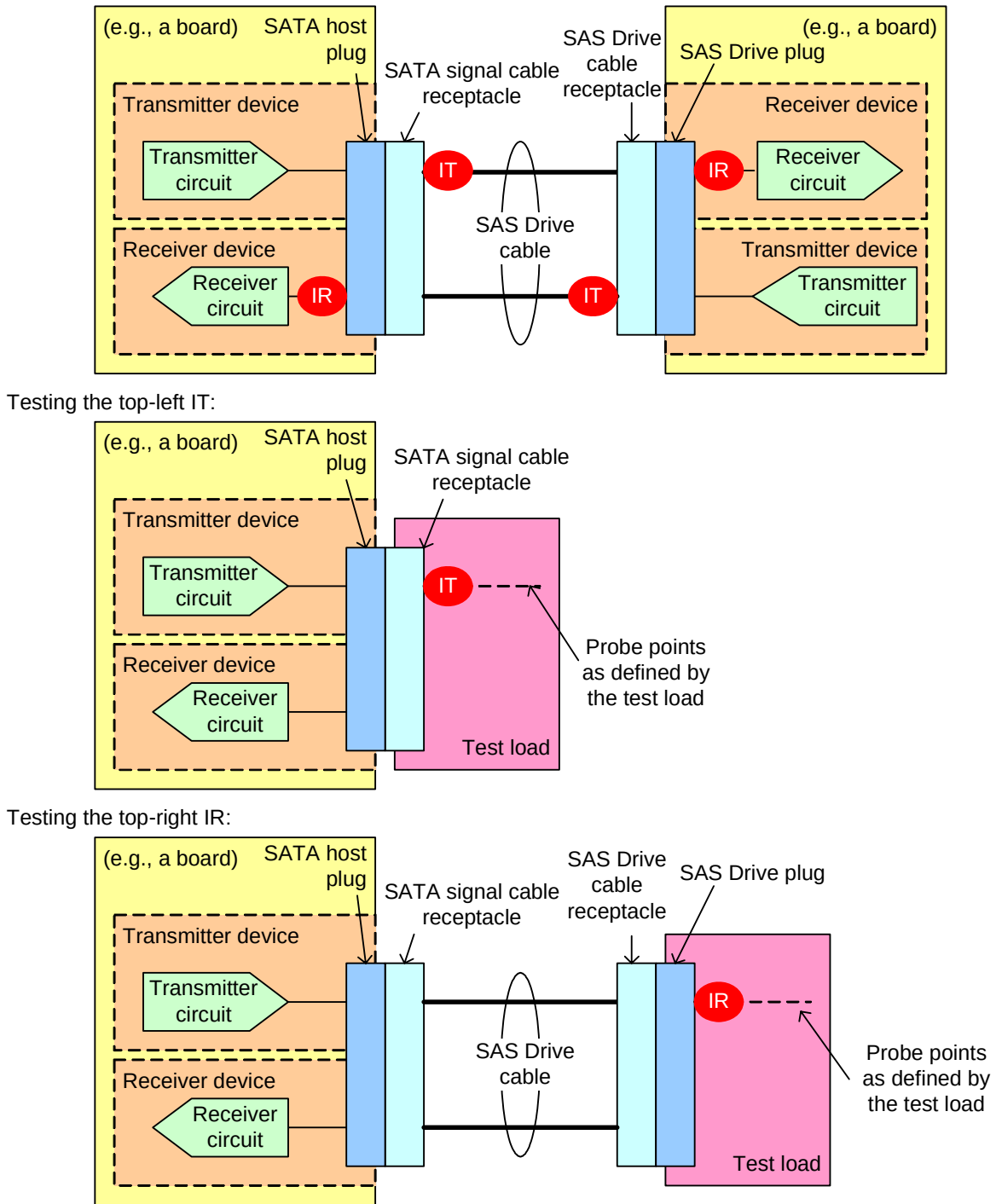


Figure 104 — Internal cable IT and IR compliance points

5.3.2 Test loads

5.3.2.1 Test loads overview

A test load methodology is used for the specification of transmitter device signal output characteristics (see 5.3.6.2 and 5.3.6.3) and delivered signal characteristics (see 5.3.7.2). This methodology specifies the signal as measured at specified probe points in specified test loads.

The test loads used by the methodology are:

- zero-length test load (see 5.3.2.2): used for testing transmitter device compliance points and receiver device compliance points;
- transmitter compliance transfer function (TCTF) test load (see 5.3.2.3): used for testing transmitter device compliance points; and
- low-loss TCTF test load (see 5.3.2.3): used for testing transmitter device compliance points when SATA devices using Gen2i levels (see SATA-2) are supported and the SAS receiver device does not support the signal levels received through a full TCTF test load (see 5.3.2.3).

Physical positions denoted as probe points identify the position in the test load where the signal properties are measured, but do not imply that physical probing is used for the measurement. Physical probing may be disruptive to the signal and should not be used unless verified to be non-disruptive.

5.3.2.2 Zero-length test load

Figure 105 shows the zero-length test load as used for testing a transmitter device compliance point.

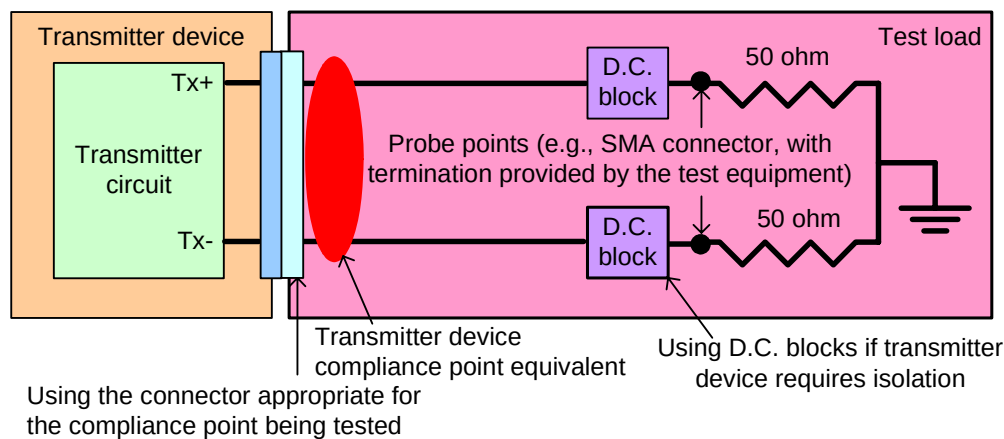


Figure 105 — Zero-length test load for transmitter device compliance point

Figure 106 shows the zero-length test load as used for testing a receiver device compliance point.

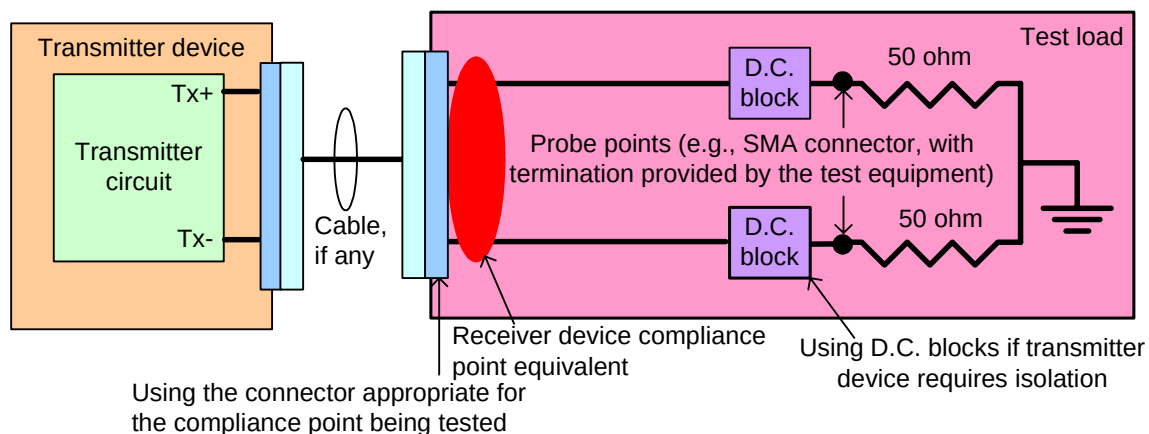


Figure 106 — Zero-length test load for receiver device compliance point

Figure 105 and figure 106 show ideal designs. Actual designs include:

- insertion loss between the compliance and probe points; and
- return loss due one or more impedance mismatches between the compliance point and 50 ohm termination points.

Not shown are non-ideal effects of the test equipment raw measurements (e.g., additional insertion and return loss). For de-embedding methods to remove non-ideal effects see Annex B.

Usage of fixturing and test equipment shall comply with the requirements defined in this subclause. The requirements in this subclause include the combined effects of the fixturing and test equipment.

The zero-length test load, including all fixturing and instrumentation required for the measurement, shall comply with the following equations:

For $50 \text{ MHz} < f \leq 6,0 \text{ GHz}$:

$$|S_{DD21}(f)| \geq -20 \log_{10}(e) \times ((1,0 \times 10^{-6} \times f^{0,5}) + (2,8 \times 10^{-11} \times f) + (5,3 \times 10^{-21} \times f^2)) - 0,2 \text{ dB}$$

$$|S_{DD11}(f)| \leq -15 \text{ dB}$$

Figure 107 shows the allowable $|S_{DD21}(f)|$ of a zero-length test load and the $|S_{DD21}(f)|$ of a sample zero-length test load.

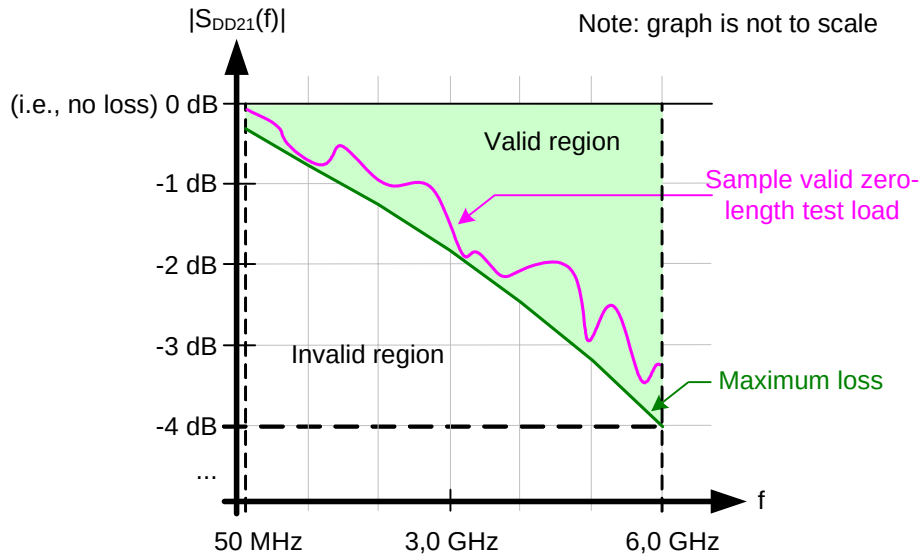


Figure 107 — Zero-length test load S_{DD21}

NOTE 18 - The zero-length test load performance specifications defined in this subclause were not required for previous versions of this standard.

5.3.2.3 TCTF test load

Figure 108 shows the TCTF test load.

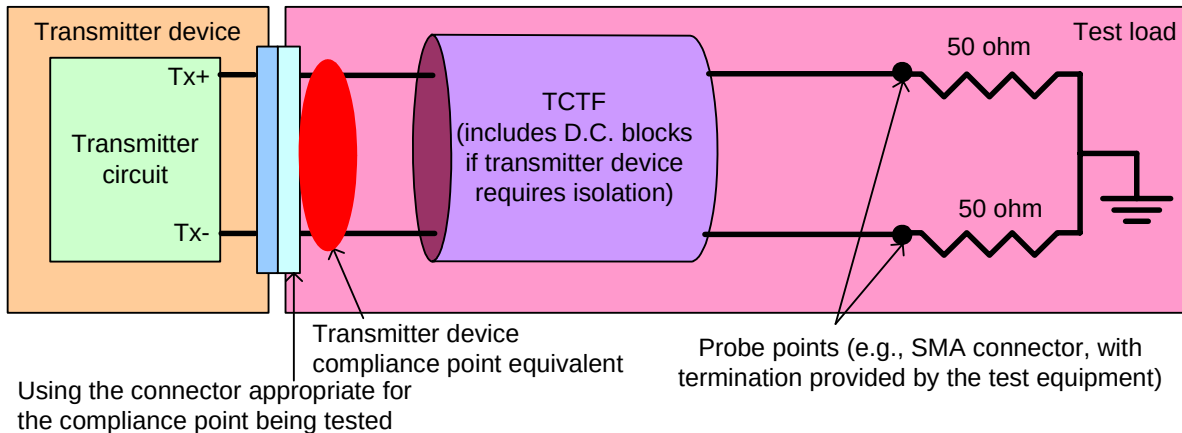


Figure 108 — TCTF test load

The TCTF test load shall meet the requirements in table 50 (see 5.2.6). The nominal impedance shall be the target impedance.

The TCTF is defined by a set of S-parameters (see B.9). Only the magnitude of $S_{DD21}(f)$ (i.e., insertion loss) is specified by this standard.

For testing a 3 Gbps transmitter device at IT, the TCTF test load shall comply with the following equations:

For $50 \text{ MHz} < f \leq 3,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((6,5 \times 10^{-6} \times f^{0,5}) + (2,0 \times 10^{-10} \times f) + (3,3 \times 10^{-20} \times f^2)) \text{ dB}$$

and for $3,0 \text{ GHz} < f \leq 5,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -10,9 \text{ dB}$$

and, specifying a minimum ISI loss:

$$(|S_{DD21}(f = 300 \text{ MHz})| \text{ of TCTF test load}) - (|S_{DD21}(f = 1\,500 \text{ MHz})| \text{ of TCTF test load}) > 3,9 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$
 f signal frequency in Hz

Figure 109 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at IT at 3 Gbps.

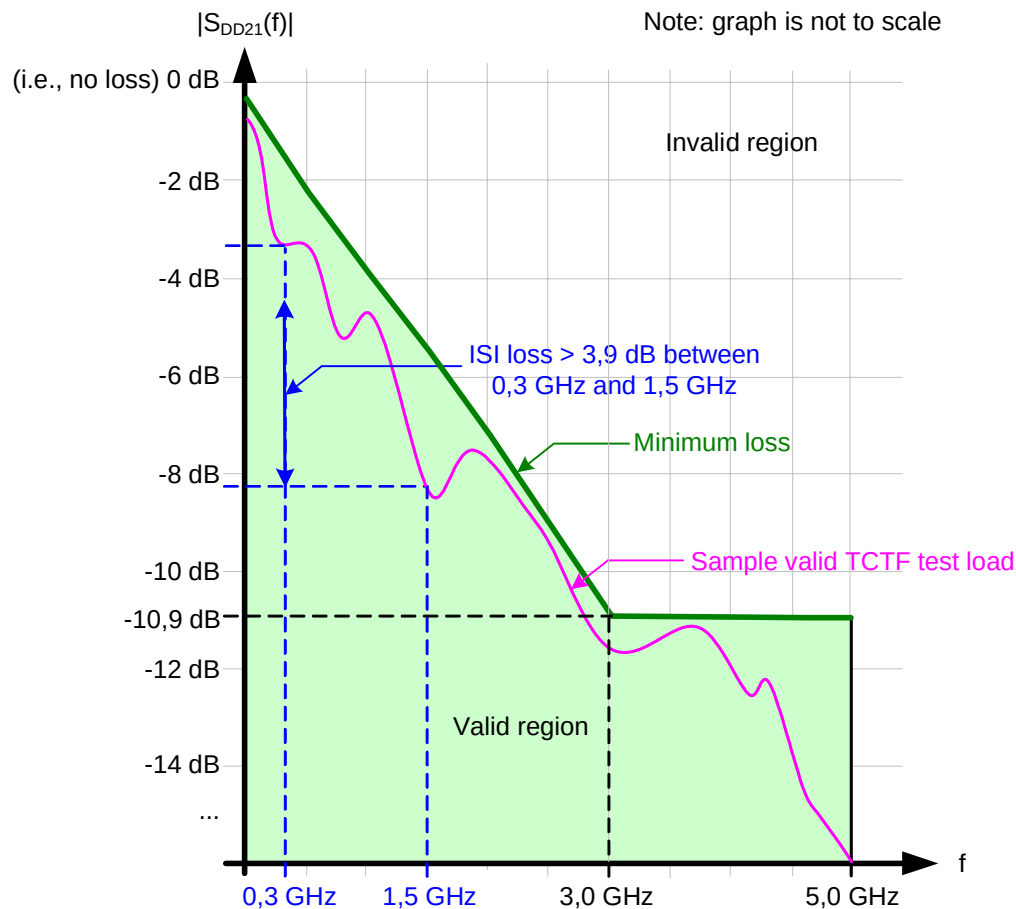


Figure 109 — TCTF test load S_{DD21} and ISI loss requirements at IT at 3 Gbps

For testing a 3 Gbps transmitter device at CT, the TCTF test load shall comply with the following equations:

For $50 \text{ MHz} < f \leq 3,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((1,7 \times 10^{-5} \times f^{0,5}) + (1,0 \times 10^{-10} \times f)) \text{ dB}$$

and for $3,0 \text{ GHz} < f \leq 5,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -10,7 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 300 \text{ MHz})| - |S_{DD21}(f = 1500 \text{ MHz})| > 3,9 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f signal frequency in Hz

Figure 110 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at CT at 3 Gbps.

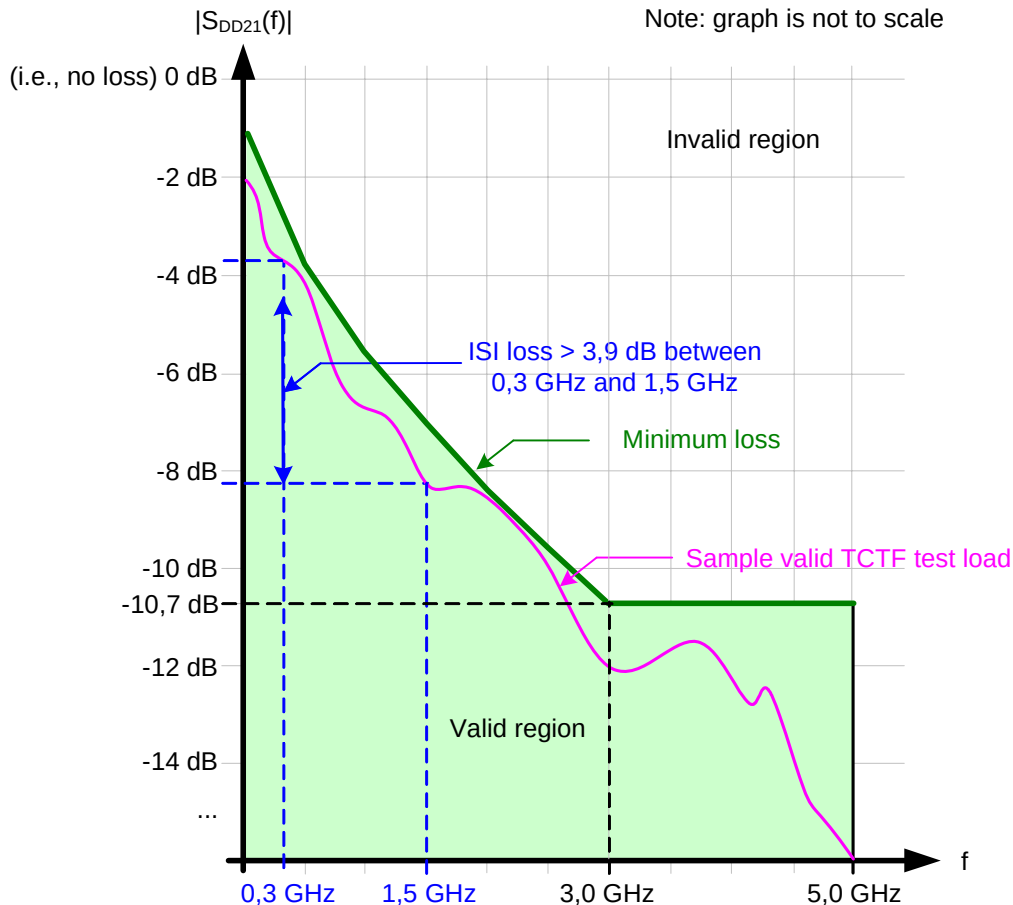


Figure 110 — TCTF test load S_{DD21} and ISI loss requirements at CT at 3 Gbps

For testing a 1,5 Gbps transmitter device at IT, the TCTF test load shall comply with the following equations:

For $50 \text{ MHz} < f \leq 1,5 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((6,5 \times 10^{-6} \times f^{0,5}) + (2,0 \times 10^{-10} \times f) + (3,3 \times 10^{-20} \times f^2)) \text{ dB}$$

and for $1,5 \text{ GHz} < f \leq 5,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -5,4 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 150 \text{ MHz})| - |S_{DD21}(f = 750 \text{ MHz})| > 2,0 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f signal frequency in Hz

Figure 111 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at IT at 1,5 Gbps.

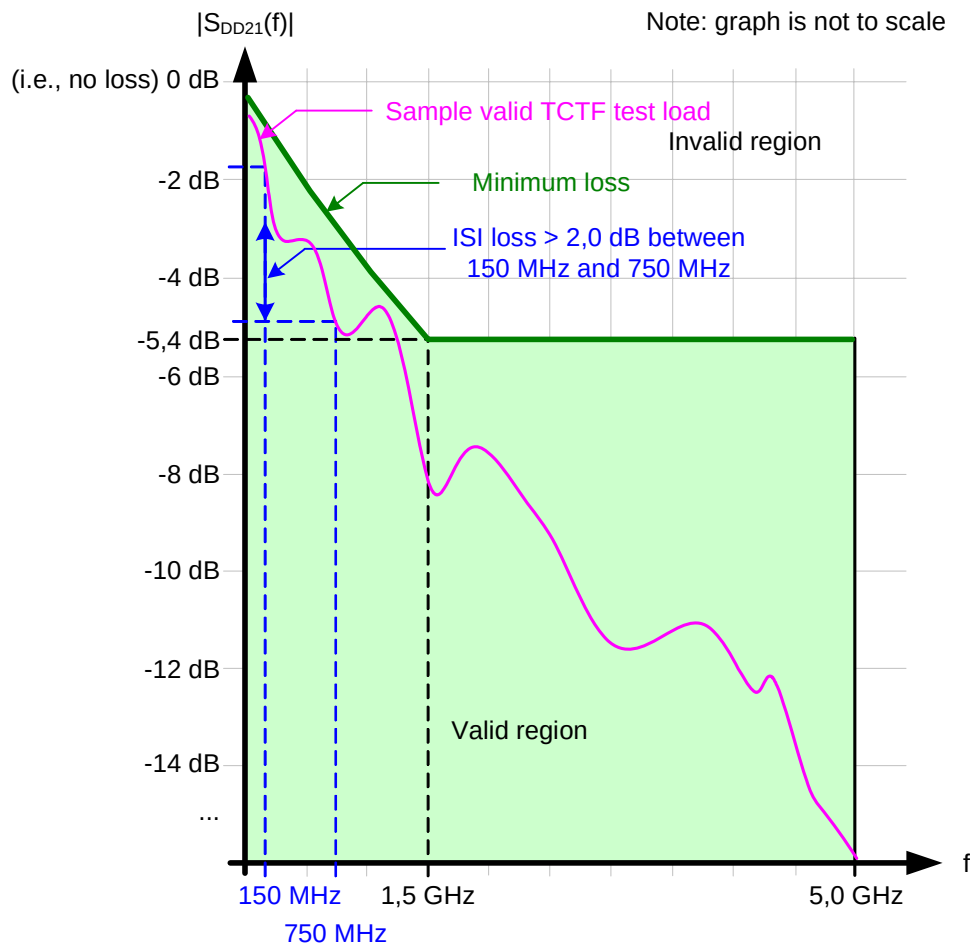


Figure 111 — TCTF test load S_{DD21} and ISI loss requirements at IT at 1,5 Gbps

For testing a 1,5 Gbps transmitter device at CT, the TCTF test load shall comply with the following equations:

For $50 \text{ MHz} < f \leq 1,5 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((1,7 \times 10^{-5} \times f^{0,5}) + (1,0 \times 10^{-10} \times f)) \text{ dB}$$

and for $1,5 \text{ GHz} < f \leq 5,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -7,0 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 150 \text{ MHz})| - |S_{DD21}(f = 750 \text{ MHz})| > 2,0 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$
 f signal frequency in Hz

Figure 112 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at CT at 1,5 Gbps.

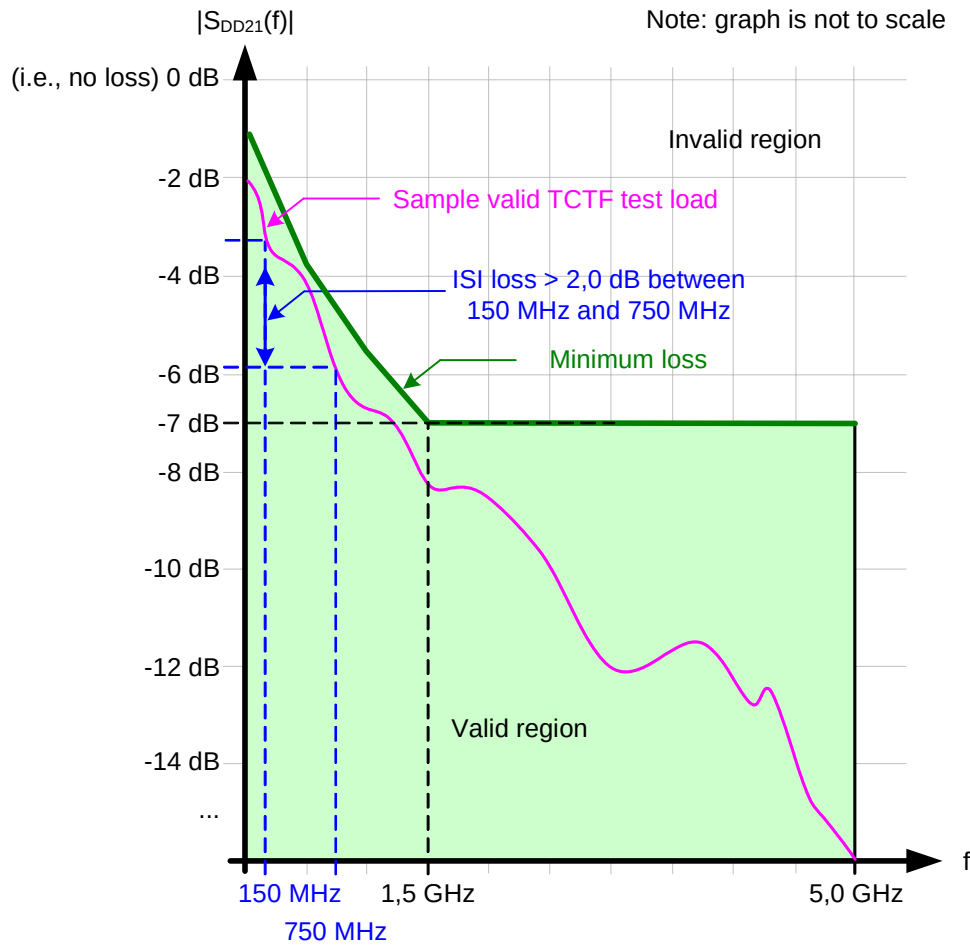


Figure 112 — TCTF test load S_{DD21} and ISI loss requirements at CT at 1,5 Gbps

5.3.2.4 Low-loss TCTF test load

Figure 113 shows the low-loss TCTF test load.

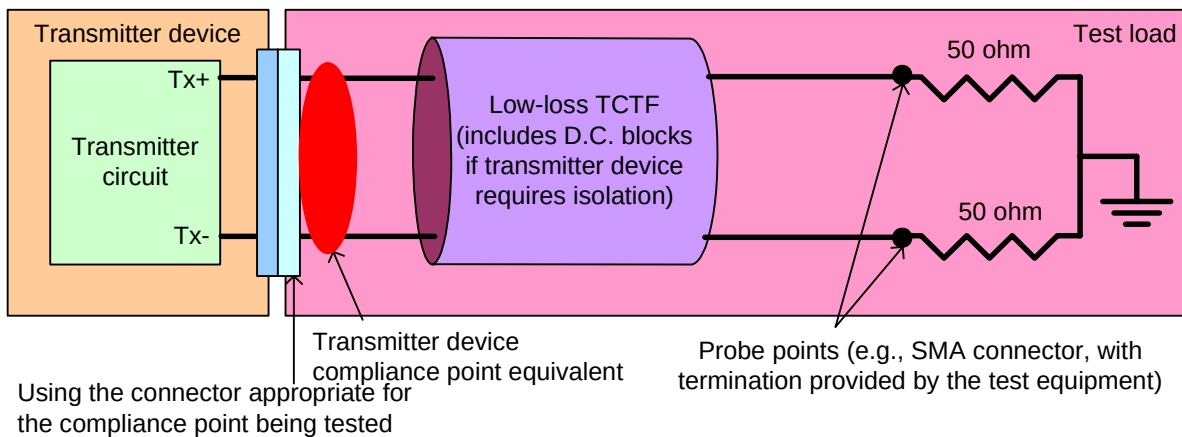


Figure 113 — Low-loss TCTF test load

The low-loss TCTF test load shall meet the requirements in table 50 (see 5.2.6). The nominal impedance shall be the target impedance.

The low-loss TCTF is defined by a set of S-parameters (see B.9). Only the magnitude of $S_{DD21}(f)$ (i.e., insertion loss) is specified by this standard.

The low-loss TCTF test load shall comply with the following equations:

For $50 \text{ MHz} < f \leq 3,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((2,2 \times 10^{-6} \times f^{0,5}) + (6,9 \times 10^{-11} \times f) + (1,1 \times 10^{-20} \times f^2)) \text{ dB}$$

for $3,0 \text{ GHz} < f \leq 5,0 \text{ GHz}$:

$$|S_{DD21}(f)| \leq -3,8 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 300 \text{ MHz})| - |S_{DD21}(f = 1\,500 \text{ MHz})| > 1,3 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f signal frequency in Hz

Editor's Note 15: -3.8 dB should probably be -3.7 dB to match the equation results at 3 GHz.

Figure 114 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a low-loss TCTF test load and the $|S_{DD21}(f)|$ of a sample low-loss TCTF test load at 3 Gbps.

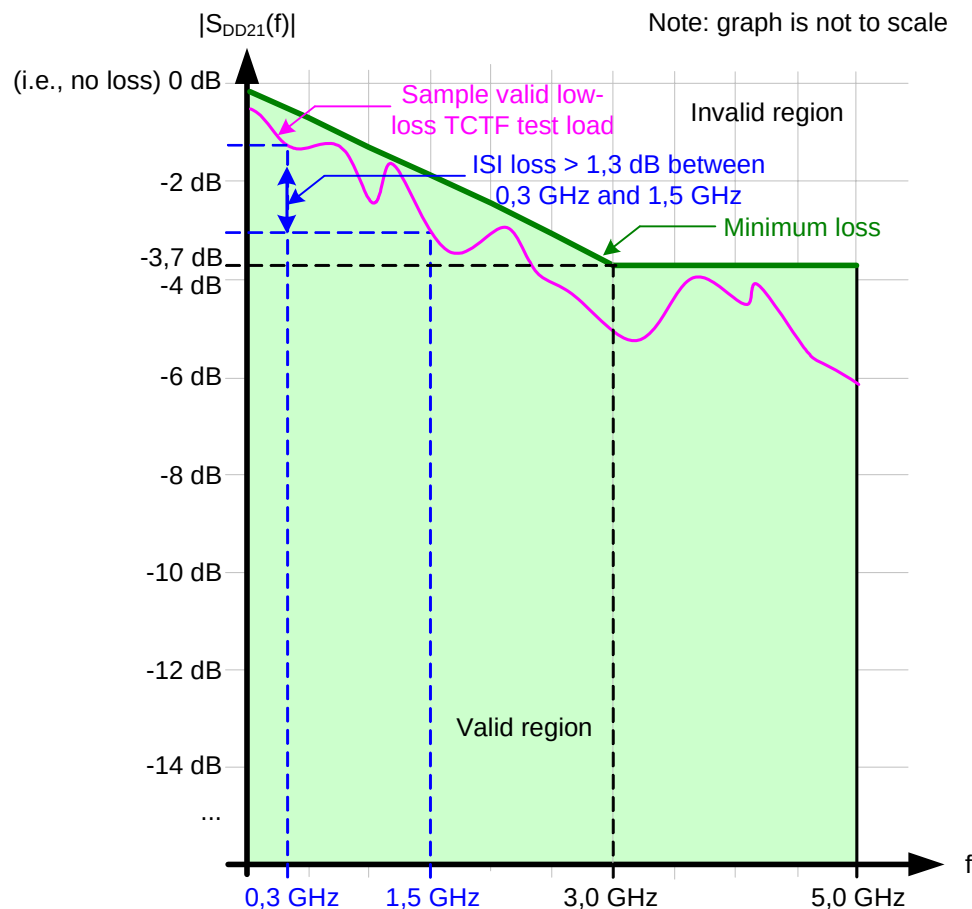


Figure 114 — Low-loss TCTF test load S_{DD21} and ISI loss requirements at 3 Gbps

5.3.3 General electrical characteristics

A TxRx connection is the complete simplex signal path between the transmitter circuit (see 3.1.278) and receiver circuit (see 3.1.178).

A TxRx connection segment is that portion of a TxRx connection delimited by separable connectors or changes in conductive material.

This standard defines the electrical requirements of the signal at the compliance points IT, IR, CT, and CR in a TxRx connection. Each compliant phy shall be compatible with these electrical requirements to allow interoperability within a SAS environment.

Each TxRx connection shall support a bit error ratio (BER) that is less than 10^{-12} (i.e., fewer than one bit error per 10^{12} bits). The parameters specified in this standard support meeting this requirement under all conditions including the minimum input and output amplitude levels.

Each TxRx connection shall be designed such that its loss characteristics are less than:

- the loss of the TCTF test load plus ISI at 3 Gbps (see figure 112 in 5.3.2.3) over the frequency range of 50 MHz to 3 000 MHz; or
- the loss of the low-loss TCTF test load plus ISI at 3 Gbps (see figure 114 in 5.3.2.4) over the frequency range of 50 MHz to 3 000 MHz, if the system supports SATA devices using Gen2i levels (see SATA-2) but the receiver device does not support SATA Gen2i levels through the TCTF test load.

Each TxRx connection shall meet the delivered signal specifications in table 64 (see 5.3.7.2).

NOTE 19 - A TxRx connection is constructed from multiple components. It is possible that a TxRx connection does not meet the delivered signal requirements of table 64 (see 5.3.7.2) when the combined losses and noise introduced by those components is considered, even if each individual component is compliant with the requirements of this standard. Such a TxRx connection is not compliant with this standard.

For external cable assemblies, these electrical requirements are consistent with using good quality passive cable assemblies constructed with shielded twinaxial cable with 24 gauge solid wire up to 6 meters in length.

Each TxRx connection segment shall comply with the impedance requirements detailed in 5.2.6 for the conductive material from which they are formed. An equalizer network, if present, shall be considered part of the TxRx connection.

TxRx connections shall be applied only to homogenous ground applications (e.g., between devices within an enclosure or rack, or between enclosures interconnected by a common ground return or ground plane).

Table 58 defines the general electrical characteristics.

Table 58 — General electrical characteristics

Characteristic	Units	1,5 Gbps (i.e., G1)	3 Gbps (i.e., G2)
Physical link rate (nominal)	MBps	150	300
Bit rate (nominal)	Mbaud	1 500	3 000
Unit interval (UI)(nominal)	ps	666,6	333,3
Differential TxRx connection impedance (nominal)	ohm	100	
Maximum A.C. coupling capacitor ^a	nF	12	
Maximum noise during OOB idle time ^b	mV(P-P)	120	
^a The coupling capacitor value for A.C. coupled transmit and receive pairs. A.C. coupling requirements for transmitter devices are described in 5.3.6.1. A.C. coupling requirements for receiver devices are described in 5.3.7.1.			
^b With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3 Gbps), no signal level during the idle time shall exceed the specified maximum differential amplitude.			

Editor's Note 16: Define f_{baud} in this table for use by some 6 Gbps equations, and ensure all uses of "bit rate", "baud", and "baud rate" are consistent and logic. Watch wherever the number 1667 is

used.

Table 59 defines the transmitter device general electrical characteristics.

Table 59 — General transmitter device electrical characteristics

Characteristic	Units	1,5 Gbps	3 Gbps
Physical link rate long-term stability at IT and CT	ppm	± 100	
Physical link rate SSC modulation at IT and CT	ppm	See table 68 and table 69 in 5.3.8.2	
Maximum transmitter device transients ^a	V	± 1,2	
Transmitter device source termination:			
Differential impedance ^b	ohm	60 min/115 max	
Maximum differential impedance imbalance ^{b, c}	ohm	5	
Common-mode impedance ^b	ohm	15 min/40 max	
^a See 5.3.4 for transient test circuits and conditions. ^b All transmitter device termination measurements are made through mated connector pairs. ^c The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.			

Table 60 defines the receiver device general electrical characteristics.

Table 60 — Receiver device general electrical characteristics

Characteristic	Units	1,5 Gbps	3 Gbps
Physical link rate long-term tolerance at IR if SATA is not supported	ppm	± 100	
Physical link rate long-term tolerance at IR if SATA is supported	ppm	± 350	
Physical link rate SSC modulation tolerance at IR	ppm	See table 70 in 5.3.8.3	
Maximum receiver device transients ^a	V	± 1,2	
Minimum Receiver A.C. common-mode voltage tolerance V _{CM} ^b	mV(P-P)	150	
Receiver A.C. common-mode frequency tolerance range F _{CM} ^b	MHz	2 to 200	
Receiver device termination:			
Differential impedance ^{c, d, e}	ohm	100 ± 15	
Maximum differential impedance imbalance ^{c, d, e, f}	ohm	5	
Maximum receiver termination time constant ^{c, d, e}	ps	150	100
Common-mode impedance ^{c, d}	ohm	20 min/40 max	
^a See 5.3.4 for transient test circuits and conditions.			
^b Receiver devices shall tolerate sinusoidal common-mode noise components within the peak-to-peak amplitude (V _{CM}) and the frequency range (F _{CM}).			
^c All receiver device termination measurements are made through mated connector pairs.			
^d The receiver device termination impedance specification applies to all receiver devices in a TxRx connection and covers all time points between the connector nearest the receiver device, the receiver device, and the transmission line terminator. This measurement shall be made from that connector.			
^e At the time point corresponding to the connection of the receiver device to the transmission line, the input capacitance of the receiver device and its connection to the transmission line may cause the measured impedance to fall below the minimum impedances specified in this table. With impedance measured using amplitude in units of ρ (i.e., the reflection coefficient, a dimensionless unit) and duration in units of time, the area of the impedance dip caused by this capacitance is the receiver termination time constant. The receiver termination time constant shall not be greater than the values shown in this table.			
An approximate value for the receiver termination time constant is given by the product of the amplitude of the dip in units of ρ and the width of the dip in units of time, as measured at the half amplitude point. The amplitude is defined as the difference in the reflection coefficient between the reflection coefficient at the nominal impedance and the reflection coefficient at the minimum impedance point.			
The value of the receiver device excess input capacitance is given by the following equation:			
$C = \frac{\text{receiver termination time constant}}{(R0 \parallel RR)}$			
where (R0 ∥ RR) is the parallel combination of the transmission line characteristic impedance and termination resistance at the receiver device.			
^f The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.			

5.3.4 Transmitter and receiver device transients

Transients may occur at transmitter devices or receiver devices as a result of changes in supply power conditions or mode transitions.

A mode transition is an event that may result in a measurable transient due to the response of the transmitter device or receiver device. The following conditions constitute a mode transition:

- enabling or disabling driver circuitry;
- enabling or disabling receiver common-mode circuitry;
- hot plug event;
- adjusting driver amplitude;
- enabling or disabling pre-emphasis (i.e., de-emphasis); and
- adjusting terminator impedance.

Transmitter device transients are measured at nodes V_P and V_N with respect to GROUND on the test circuit shown in figure 115 during all power state and mode transitions. Receiver device transients are measured at nodes V_P and V_N with respect to GROUND on the test circuit shown in figure 116 during all power state and mode transitions. Test conditions shall include power supply power on and power off conditions, voltage sequencing, and mode transitions.

Figure 115 shows the test circuit attached to IT or CT to test transmitter device transients.

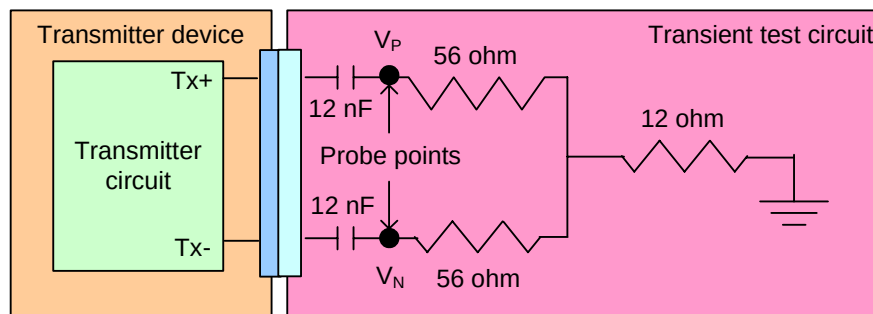


Figure 115 — Transmitter device transient test circuit

Figure 116 shows the test circuit attached to IR or CR to test receiver device transients.

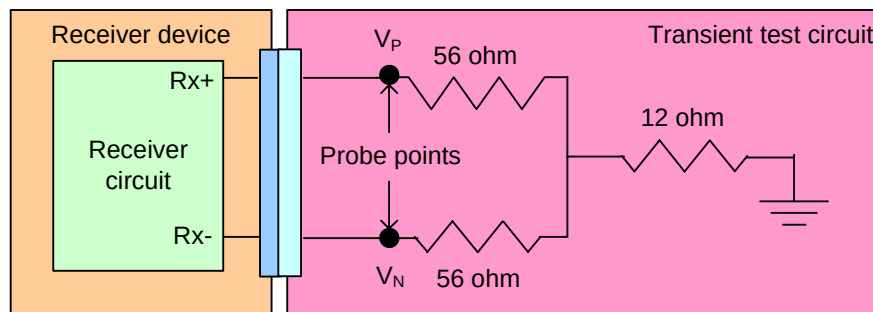


Figure 116 — Receiver device transient test circuit

5.3.5 Eye masks

5.3.5.1 Eye masks overview

The eye masks shown in this subclause shall be interpreted as graphical representations of the voltage and time limits of the signal. The eye mask boundaries define the eye contour of the 10^{-12} jitter population at all signal levels. Current equivalent time sampling oscilloscope technology is not practical for measuring compliance to this eye contour. See MJSQ for methods that are suitable for verifying compliance to these eye masks.

5.3.5.2 Transmitter device eye mask

Figure 117 describes the eye mask used for testing the signal output of the transmitter device at IT, CT, IR, and CR. This eye mask applies to jitter after the application of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of $((\text{bit rate}) / 1\,667)$.

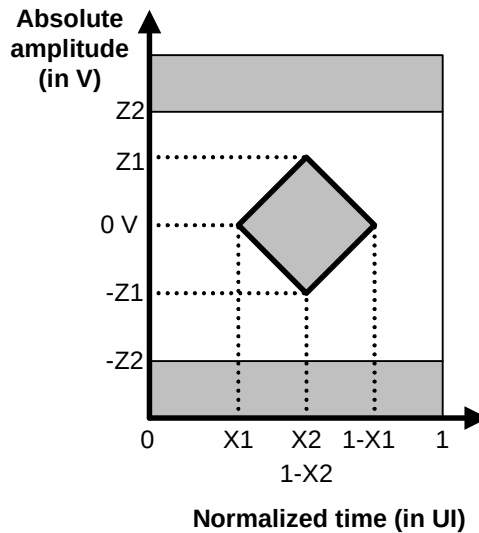


Figure 117 — Transmitter device eye mask

Verifying compliance with the limits represented by the transmitter device eye mask should be done with reverse channel traffic present in order that the effects of crosstalk are taken into account.

5.3.5.3 Receiver device eye mask

Figure 118 describes the eye mask used for testing the signal delivered to the receiver device at IR and CR. The signal shall be measured using a jitter timing reference (e.g., a golden PLL) that approximates a single pole (i.e., 20 dB per decade) low-pass filter with corner frequency of $((\text{bit rate}) / 1\,667)$. This requirement accounts for the low frequency tracking properties and response time of the CDRs in receiver devices.

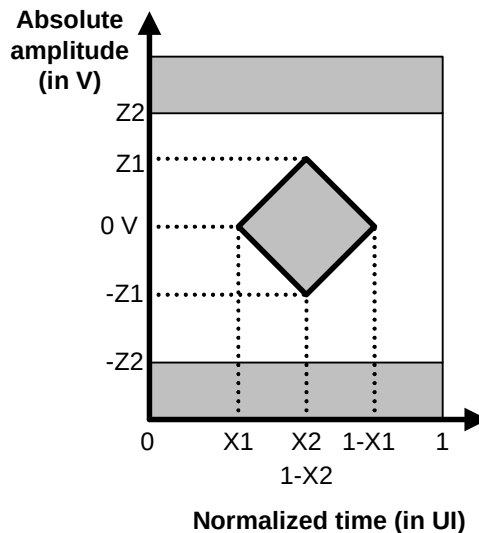


Figure 118 — Receiver device eye mask

Verifying compliance with the limits represented by the receiver device eye mask should be done with reverse channel traffic present in order that the effects of crosstalk are taken into account.

5.3.5.4 Receiver device jitter tolerance eye mask

Figure 119 describes the eye mask used to test the jitter tolerance of the receiver device at IR and CR. Figure 119 shall be constructed using the following values:

- X2 and Z2 shall be the values for the delivered signal listed in table 64 (see 5.3.7.2);
- X1_{OP} shall be half the value of TJ for maximum delivered jitter listed in table 65 (see 5.3.7.3); and
- X1_{TOL} shall be half the value of TJ for receiver device jitter tolerance listed in table 66 (see 5.3.7.4), for applied sinusoidal jitter frequencies above ((bit rate) / 1 667).

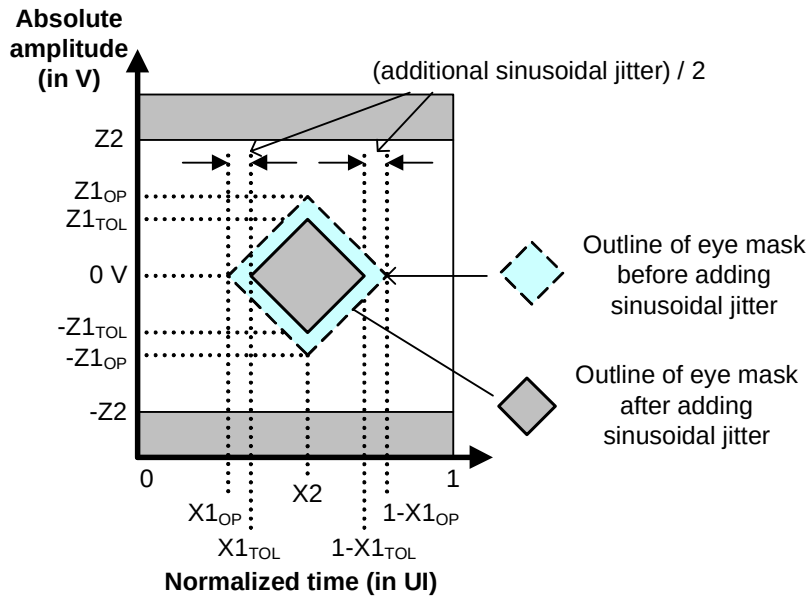


Figure 119 — Deriving a receiver device jitter tolerance eye mask

The leading and trailing edge slopes of the receiver device eye mask in figure 118 (see 5.3.5.3) shall be preserved. As a result, the amplitude value of Z1 is less than that given for the delivered signal in table 64 (see 5.3.7.2), and Z1_{TOL} and Z1_{OP} shall be defined from those slopes by the following equation:

$$Z1_{TOL} = Z1_{OP} \times \frac{X2 - \left(\frac{ASJ}{2}\right) - X1_{OP}}{X2 - X1_{OP}}$$

where:

Z1 _{TOL}	is the value for Z1 to be used for the receiver device jitter tolerance eye mask
Z1 _{OP}	is the Z1 value for the delivered signal in table 64
X1 _{OP}	is the X1 value for the delivered signal in table 64
X2	is the X2 value for the delivered signal in table 64
ASJ	is the additional sinusoidal jitter defined in figure 120

The X1 points in the receiver device jitter tolerance eye mask (see figure 119) are greater than the X1 points in the receiver device eye mask (see figure 118) due to the addition of sinusoidal jitter.

Figure 120 defines the applied sinusoidal jitter.

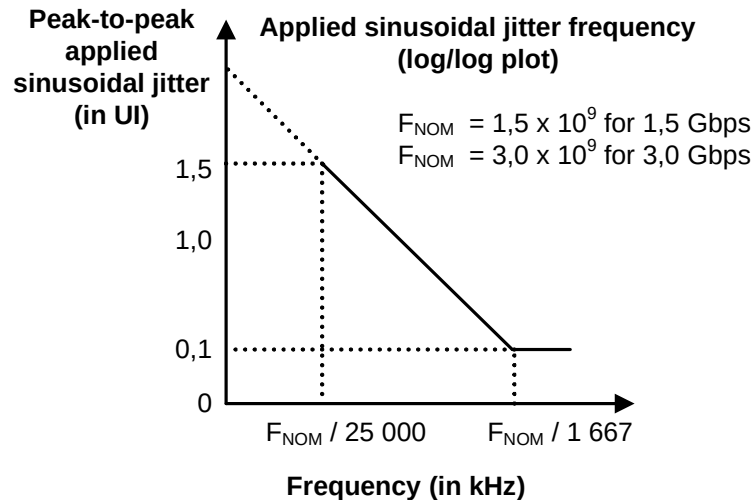


Figure 120 — Applied sinusoidal jitter

CJTPAT shall be used for all jitter testing unless otherwise specified. Annex A defines the required pattern on the physical link and provides information regarding special considerations for running disparity (see 6.2) and scrambling (see 7.6).

5.3.6 Transmitter device characteristics

5.3.6.1 Transmitter device characteristics overview

A.C. coupling requirements for transmitter devices are as follows:

- transmitter devices using inter-enclosure TxRx connections (i.e., attached to CT compliance points) shall be A.C. coupled to the interconnect through a transmission network;
- transmitter devices using intra-enclosure TxRx connections (i.e., attached to IT compliance points) that support SATA shall be A.C. coupled to the interconnect through a transmission network; and
- transmitter devices using intra-enclosure TxRx connections (i.e., attached to IT compliance points) that do not support SATA may be A.C. or D.C. coupled.

Transmitter devices may or may not incorporate pre-emphasis (i.e., de-emphasis) and other forms of compensation. The transmitter device shall use the same settings (e.g., pre-emphasis and voltage swing) with both the zero-length test load and the appropriate TCTF test load.

See B.5 for a methodology for measuring transmitter device signal output.

5.3.6.2 Transmitter device signal output characteristics as measured with the zero-length test load

Table 61 specifies the signal output characteristics for the transmitter device as measured with the zero-length test load (see 5.3.2.2) attached at a transmitter device compliance point (i.e., IT or CT). All specifications are based on differential measurements.

Table 61 — Transmitter device signal output characteristics as measured with the zero-length test load at transmitter device compliance points IT and CT

Signal characteristic ^a	Units	1,5 Gbps	3 Gbps
Maximum intra-pair skew ^b	ps	20	15
Maximum transmitter device off voltage ^c	mV(P-P)	50	
Maximum rise/fall time ^d	ps	273	137
Minimum rise/fall time ^d	ps	67	
Maximum transmitter output imbalance ^e	%	10	
OOB offset delta ^f	mV	± 25	
OOB common-mode delta ^g	mV	± 50	
^a All tests in this table shall be performed with zero-length test load (see 5.3.2.2).			
^b The intra-pair skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern (see table 306 in 10.2.9.2) on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Tx+ and Tx- signals. Intra-pair skew is defined as the time difference between the means of the midpoint crossing times of the Tx+ signal and the Tx- signal.			
^c The transmitter device off voltage is the maximum A.C. voltage measured at compliance points IT and CT when the transmitter is unpowered or transmitting D.C. idle (e.g., during idle time of an OOB signal).			
^d Rise/fall times are measured from 20 % to 80 % of the transition with a repeating 0101b pattern (see table 306 in 10.2.9.2) on the physical link.			
^e The maximum difference between the V+ and V- A.C. RMS transmitter device amplitudes measured with CJTPAT (see A.2) into the zero-length test load shown in figure 105 (see 5.3.2.2), as a percentage of the average of the V+ and V- A.C. RMS amplitudes.			
^f The maximum difference in the average differential voltage (D.C. offset) component between the burst times and the idle times of an OOB signal.			
^g The maximum difference in the average of the common-mode voltage between the burst times and the idle times of an OOB signal.			

5.3.6.3 Transmitter device signal output characteristics as measured with each test load

Table 62 specifies the signal output characteristics for the transmitter device as measured with each test load (i.e., the zero-length test load (see 5.3.2.2) and either the TCTF test load (see 5.3.2.3) or the low-loss TCTF test load (see 5.3.2.4)) attached at a transmitter device compliance point (i.e., IT or CT). All specifications are based on differential measurements.

Table 62 — Transmitter device signal output characteristics as measured with each test load at transmitter device compliance points IT and CT

Signal characteristic	Units	IT		CT	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Maximum jitter (see figure 117 in 5.3.5.2) ^a	N/A	See table 65 in 5.3.7.3			
Maximum peak to peak voltage (i.e., 2 × Z2 in figure 117) if SATA is not supported	mV(P-P)	1 600		1 600	
Maximum peak to peak voltage (i.e., 2 × Z2 in figure 117) if SATA is supported	mV(P-P)	see SATA-2 ^e		N/A	
Minimum eye opening (i.e., 2 × Z1 in figure 117), if SATA is not supported	mV(P-P)	325	275	275	
Minimum eye opening (i.e., 2 × Z1 in figure 117), if SATA is supported	mV(P-P)	see SATA-2 ^e		N/A	
Half of maximum jitter (i.e., X1 in figure 117) ^b	UI	0,275			
Center of bit time (i.e., X2 in figure 117)	UI	0,50			
Maximum intra-pair skew ^c	ps	80	75	80	75
Maximum voltage (non-operational)	mV(P-P)	2 000			
Minimum OOB burst amplitude ^d , if SATA is not supported	mV(P-P)	240 ^f			
Minimum OOB burst amplitude ^d , if SATA is supported	mV(P-P)	240 mV ^{e f}		N/A	
^a The value for X1 applies at a total jitter probability of 10⁻¹². At this level of probability, direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter requirements. ^b The value for X1 shall be half the value of TJ for maximum delivered jitter listed in table 65. The test or analysis shall include the effects of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667). ^c The intra-pair skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern (see table 306 in 10.2.9.2) on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Tx+ and Tx- signals. Intra-pair skew is defined as the time difference between the means of the midpoint crossing times of the Tx+ signal and the Tx- signal at the probe points. ^d With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3 Gbps), each signal level during the OOB burst shall exceed the specified minimum differential amplitude before transitioning to the opposite bit value or before termination of the OOB burst. ^e Amplitude measurement methodologies of SATA and this standard differ. Under conditions of maximum rise/fall time and jitter, eye diagram methodologies used in this standard may indicate less signal amplitude than the technique specified by SATA-2. Implementors of designs supporting SATA are required to ensure interoperability and should perform additional system characterization with an eye diagram methodology using SATA devices. ^f The OOB burst contains 1,5 Gbps D24.3 characters or ALIGN (0) primitives (see 6.6 and SATA-2).					

5.3.6.4 Transmitter device maximum jitter

Table 63 defines the maximum jitter the transmitter device shall deliver as measured with each test load (i.e., the zero-length test load (see 5.3.2.2) and either the TCTF test load (see 5.3.2.3) or the low-loss TCTF test load (see 5.3.2.4)) at a transmitter device compliance point (i.e., IT or CT).

Table 63 — Transmitter device maximum jitter as measured with each test load at transmitter device compliance points IT and CT

Signal characteristic ^{a, b}	Units	IT		CT	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Deterministic jitter (DJ) ^d	UI	0,35			
Total jitter (TJ) ^{c, d, e}	UI	0,55			
^a All DJ and TJ values are level 1 (see MJSQ). ^b The values for jitter in this table are measured at the average signal amplitude point. ^c TJ is specified at a CDF level of 10⁻¹². ^d The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.2. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ. ^e If TJ received at any point is less than the maximum allowed, then the jitter distribution of the signal is allowed to be asymmetric. The TJ plus the magnitude of the asymmetry shall not exceed the allowed maximum TJ. The numerical difference between the average of the peaks with a BER < 10⁻¹² and the average of the individual events is the measure of the asymmetry. Jitter peak-to-peak measured < (maximum TJ - Asymmetry).					

5.3.6.5 Transmitter device signal output levels for OOB signals

Transmitter devices supporting SATA shall use SATA Gen1i or Gen2i signal output levels (see SATA-2) during the first OOB sequence (see 6.7) after a power on or hard reset. If the phy does not receive COMINIT within a hot-plug timeout (see 6.7.5), the transmitter device shall increase its transmit levels to the SAS signal output levels specified in table 61 (see 5.3.6) and table 62 (see 5.3.6.3) and perform the OOB sequence again. If no COMINIT is received within a hot-plug timeout of the second OOB sequence, the transmitter device shall initiate another OOB sequence using SATA Gen1i or Gen2i signal output levels. The transmitter device shall continue alternating between transmitting COMINIT using SATA Gen1i or Gen2i signal output levels and transmitting COMINIT with SAS signal output levels until the phy receives COMINIT.

If the phy both transmits and receives COMSAS (i.e., a SAS phy or expander phy is attached), the transmitter device shall set its transmit levels to the SAS signal output levels. If it had been using SATA Gen1i or Gen2i signal output levels, this mode transition (i.e., output voltage change) may result in a transient (see 5.3.4) during the idle time between COMSAS and the SAS speed negotiation sequence (see 6.7.4.2).

If the transmitter device is using SAS signal output levels and the phy does not receive COMSAS (i.e., a SATA phy is attached), the transmitter device shall set its transmit levels to the SATA Gen1i or Gen2i signal output levels and restart the OOB sequence.

Transmitter devices that do not support SATA shall transmit OOB signals using SAS signal output levels.

Editor's Note 17: mention that the selected signal output level is used for the speed negotiation sequence and beyond

5.3.7 Receiver device characteristics

5.3.7.1 Receiver device characteristics overview

A.C. coupling requirements for receiver devices are as follows:

- a) all receiver devices (i.e., attached to IR or CR compliance points) shall be A.C. coupled to the interconnect through a receive network.

The receive network shall terminate the TxRx connection by a 100 ohm equivalent impedance as specified in table 50 (see 5.2.6).

The receiver device shall operate within the required BER (see 5.3.3) when a signal with valid voltage and timing characteristics is delivered to the receiver device compliance point from a nominal 100 ohm source. The received signal shall be considered valid if it meets the voltage and timing limits specified in table 64 (see 5.3.7.2).

Additionally, the receiver device shall operate within the required BER (see 5.3.3) when the signal has additional sinusoidal jitter present as specified in table 66 (see 5.3.7.4) with the common-mode signal V_{CM} as specified in table 60 (see 5.3.3). Jitter tolerance for receiver device compliance points is illustrated in figure 119 (see 5.3.5.4). Figure 119 assumes that any external interference occurs prior to the point at which the test is applied. When testing the jitter tolerance capability of a receiver device, the additional 0,1 UI of sinusoidal jitter may be reduced by an amount proportional to the actual externally induced interference between the application point of the test and the input to the receiver device. The additional jitter reduces the eye opening in both voltage and time.

See B.8 for a methodology for measuring receiver device signal tolerance.

5.3.7.2 Delivered signal (receiver device signal tolerance) characteristics

Table 64 specifies the requirements of the signal delivered by the system with the zero-length test load (see 5.3.2.2) at the receiver device compliance point (i.e., IR or CR). These imply the required signal tolerance characteristics of the receiver device.

Table 64 — Delivered signal characteristics as measured with the zero length test load at receiver device compliance points IR and CR (part 1 of 2)

Signal characteristic	Units	IR		CR	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Maximum peak to peak voltage (i.e., $2 \times Z2$ in figure 118) if a SATA phy is not attached	mV(P-P)	1 600		1 600	
Maximum peak to peak voltage (i.e., $2 \times Z2$ in figure 118) if a SATA phy is attached	mV(P-P)	see SATA-2 ^e		N/A	
Minimum eye opening (i.e., $2 \times Z1$ in figure 118), if a SATA phy is not attached	mV(P-P)	325	275	275	
Minimum eye opening (i.e., $2 \times Z1$ in figure 118), if a SATA phy using Gen1i or Gen1x levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	225 ^e	N/A	N/A	
Minimum eye opening (i.e., $2 \times Z1$ in figure 118), if a SATA phy using Gen2i levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	N/A	175 ^e	N/A	
Minimum eye opening (i.e., $2 \times Z1$ in figure 118), if a SATA phy using Gen2x levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	N/A	275 ^e	N/A	
Minimum eye opening (i.e., $2 \times Z1$ in figure 118), if a SATA phy is attached and the interconnect is characterized with the low-loss TCTF test load (see 5.3.2.4)	mV(P-P)	275 ^e		N/A	

Table 64 — Delivered signal characteristics as measured with the zero length test load at receiver device compliance points IR and CR (part 2 of 2)

Signal characteristic	Units	IR		CR	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Jitter tolerance (see figure 119 in 5.3.5.4) ^a	N/A	See table 66 in 5.3.7.4			
Half of maximum jitter (i.e., X1 in figure 118) ^b	UI	0,275			
Center of bit time (i.e., X2 in figure 118)	UI	0,50			
Maximum intra-pair skew ^c	ps	80	75	80	75
Maximum voltage (non-operational)	mV(P-P)	2 000			
Minimum OOB burst amplitude ^d , if SATA is not supported	mV(P-P)	240 ^f			
Minimum OOB burst amplitude ^d , if SATA is supported	mV(P-P)	225 ^f		N/A	
^a The value for X1 applies at a total jitter probability of 10⁻¹². At this level of probability direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter requirements. ^b The value for X1 shall be half the value given for TJ in table 65. The test or analysis shall include the effects of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667). ^c The intra-pair skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern (see table 306 in 10.2.9.2) on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Rx+ and Rx- signals. Intra-pair skew is defined as the time difference between the means of the midpoint crossing times of the Rx+ signal and the Rx- signal at the probe points. ^d With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3 Gbps), to be detected as an OOB burst, each signal level during the OOB burst shall exceed the specified minimum differential amplitude before transitioning to the opposite bit value or before termination of the OOB burst. ^e Amplitude measurement methodologies of SATA and this standard differ. Under conditions of maximum rise/fall time and jitter, eye diagram methodologies used in this standard may indicate less signal amplitude than the technique specified by SATA-2. Implementors of designs supporting attachment to SATA devices are required to ensure interoperability and should perform additional system characterization with an eye diagram methodology using SATA devices. ^f The OOB burst contains either 1,5 Gbps D24.3 characters, 1,5 Gbps ALIGN (0) primitives, or 3 Gbps ALIGN 0 primitives (see 6.6 and SATA-2).					

5.3.7.3 Maximum delivered jitter

Table 65 defines the maximum jitter the system shall deliver to the receiver device at the receiver device compliance point (i.e., IR or CR).

Table 65 — Maximum delivered jitter at receiver device compliance points IR and CR

Signal characteristic ^{a, b}	Units	IR		CR	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Deterministic jitter (DJ) ^d	UI	0,35			
Total jitter (TJ) ^{c, d, e}	UI	0,55			
^a All DJ and TJ values are level 1 (see MJSQ). ^b The values for jitter in this table are measured at the average signal amplitude point. ^c TJ is specified at a CDF level of 10⁻¹². ^d The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.2. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ. ^e If TJ received at any point is less than the maximum allowed, then the jitter distribution of the signal is allowed to be asymmetric. The TJ plus the magnitude of the asymmetry shall not exceed the allowed maximum TJ. The numerical difference between the average of the peaks with a BER < 10⁻¹² and the average of the individual events is the measure of the asymmetry. Jitter peak-to-peak measured < (maximum TJ - Asymmetry).					

Editor's Note 18: Since IR and CR are the same, why not merge those columns?

5.3.7.4 Receiver device jitter tolerance

Table 66 defines the amount of jitter the receiver device shall tolerate at the receiver device compliance point (i.e., IR or CR). Receiver device jitter testing shall be performed with the maximum (i.e., slowest) rise/fall times, minimum signal amplitude, and maximum total jitter, and should be performed with normal activity in the receiver device (e.g., with other transmitter circuits and receiver circuits on the same board as the receiver device performing normal activity).

Table 66 — Receiver device jitter tolerance at receiver device compliance points IR and CR

Signal characteristic	Units	IR		CR	
		1,5 Gbps	3 Gbps	1,5 Gbps	3 Gbps
Applied sinusoidal jitter (SJ) ^b	UI	0,10 ^c	0,10 ^d	0,10 ^c	0,10 ^d
Deterministic jitter (DJ) ^{a, h}	UI	0,35 ^f	0,35 ^g	0,35 ^f	0,35 ^g
Total jitter (TJ) ^{a, e, h}	UI	0,65			
^a All DJ and TJ values are level 1 (see MJSQ).					
^b The jitter values given are normative for a combination of applied SJ, DJ, and TJ that receiver devices shall be able to tolerate without exceeding the required BER (see 5.3.3). Receiver devices shall tolerate applied SJ of progressively greater amplitude at lower frequencies, according to figure 120 (see 5.3.5.4), with the same DJ and RJ levels as were used in the high frequency sweep.					
^c Applied sinusoidal swept frequency: 900 kHz to the minimum of 5 MHz and $(3,75 \times 2^{(\text{generation} - 1)} \text{ MHz})$ (e.g., 5 MHz for 1,5 Gbps and 7,5 MHz for 3 Gbps).					
^d Applied sinusoidal swept frequency: 1 800 kHz to the minimum of 5 MHz and $(3,75 \times 2^{(\text{generation} - 1)} \text{ MHz})$ (e.g., 5 MHz for 1,5 Gbps and 7,5 MHz for 3 Gbps).					
^e No value is given for RJ. For compliance with this standard, the actual RJ amplitude shall be the value that brings TJ to the stated value at a probability of 10^{-12} . The additional 0,1 UI of applied SJ is added to ensure the receiver device has sufficient operating margin in the presence of external interference.					
^f The measurement bandwidth shall be 900 kHz to 750 MHz.					
^g The measurement bandwidth shall be 1 800 kHz to 1 500 MHz.					
^h The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.3. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ.					

Editor's Note 19: Since IR and CR are the same, why not merge those columns?

5.3.8 Spread spectrum clocking (SSC)

5.3.8.1 Spread spectrum clocking overview

Spread spectrum clocking (SSC) is the technique of modulating the operating frequency of a transmitted signal to reduce the measured peak amplitude of radiated emissions.

Phys transmit with SSC as defined in 5.3.8.2 and receive with SSC as defined in 5.3.8.3.

NOTE 20 - Phys compliant with previous versions of this standard do not transmit with SSC. Phys compliant with previous versions of this standard that do not support being attached to SATA devices were not required to receive with SSC.

Table 67 defines the SSC modulation types.

Table 67 — SSC modulation types

SSC modulation type	Maximum SSC frequency deviation ^a
Center-spreading	+2 300 / -2 300 ppm
None	+0 / -0 ppm
Down-spreading	+0 / -2 300 ppm
SATA down-spreading ^b	+0 / -5 000 ppm
^a This is in addition to the physical link rate long-term stability and tolerance defined in table 59 and table 60 (see 5.3.3). ^b This is only used as a receiver parameter.	

Editor's Note 20: SATA may be changing its rules to allow any long-term frequency in the range of +350/-5350 ppm around nominal. For example, a phy could transmit at -4800 ppm only, or sweep from -5200 ppm to -3000 ppm but never reach nominal. (Hopefully the receiver still honors +350 ppm, because there's no way for the other phy to know) This disagrees with how footnote a) is used to add SSC frequency deviation to the long-term stability. SAS could follow suit, or just require that for reception from SATA devices and depend on the fact that "SATA down-spreading" is defined by SATA to provide the details.

A phy may be transmitting with a different SSC modulation type than it is receiving (e.g., a phy is transmitting with center-spreading while it is receiving with down-spreading).

If the SSC modulation type is not none, the phy shall transmit within the specified maximum SSC frequency deviation with an SSC modulation frequency that is a minimum of 30 kHz and a maximum of 33 kHz.

The SSC modulation profile (e.g., triangular or exponential) is vendor-specific, but is intended to provide the maximum amount of electromagnetic interference (EMI) reduction. For center-spreading, the average amount of up-spreading (i.e., > 0 ppm) in the SSC modulation profile shall be the same as the average amount of down-spreading (i.e., < 0 ppm). The amount of asymmetry in the SSC modulation profile shall be less than 288 ppm.

NOTE 21 - 288 ppm is the rate of deletable primitives that are left over after accounting for the physical link rate long-term stability. It is calculated as the deletable primitive rate defined in previous versions of this standard of 1/2 048 (i.e., 488 ppm) minus the width between the extremes of the physical link rate long-term stability of +100/-100 ppm (i.e., 200 ppm).

SSC-induced high-frequency jitter is included in the deterministic jitter (DJ) and consequently in total jitter (TJ) at the transmitter output. The jitter is measured after the application of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667).

Editor's Note 21: A more sophisticated attenuation model will be included in SAS-2 to accommodate SSC.

5.3.8.2 Transmitter SSC modulation

A SAS phy transmits with the SSC modulation types defined in table 68.

Table 68 — SAS phy transmitter SSC modulation

SSC modulation type ^a	Requirement	Condition
None	Mandatory	When attached to a phy that does not support SSC
None or down-spreading ^b	Optional	When attached to a phy that supports SSC
^a SAS phys compliant with previous versions of this standard were only allowed to transmit with an SSC modulation type of none. ^b A SAS device (e.g., disk drive) that attaches to a SAS domain in a location (e.g., a drive bay) that also supports a SATA device may always transmit with down-spreading (i.e., +0 / -2 300 ppm) on all its phys, since phys that support being attached to SATA devices have always been required to tolerate SATA down-spreading (i.e., +0 / -5 000 ppm) (see table 70 in 5.3.8.3).		

Editor's Note 22: Table footnote b) disagrees with the rule in 6.7.4.2.3.2 that says a phy "shall disable SSC during SNW-1, SNW-2, and Final-SNW." The intent was that this footnote overrides that section. This footnote should be reconsidered; it may be better to eliminate this exception and just require all SAS phys to properly negotiate SSC.

An expander phy transmits with the SSC modulation types defined in table 69.

Table 69 — Expander phy transmitter SSC modulation types

SSC modulation type ^a	Requirement	Condition
None	Mandatory	When attached to a SAS phy or expander phy that does not support SSC
None or center-spreading	Optional	When attached to a SAS phy or expander phy that supports SSC
None or down-spreading	Optional	When attached to a SATA phy
^a Expander phys compliant with previous versions of this standard were only allowed to transmit with an SSC modulation type of none.		

5.3.8.3 Receiver SSC modulation tolerance

SAS phys and expander phys support (i.e., tolerate) receiving with SSC modulation types defined in table 70.

Table 70 — Receiver SSC modulation tolerance

SSC modulation type ^{a b}	Requirement	Phys that tolerate
None	Mandatory	All phys
Center-spreading	Optional ^c	All phys
Down-spreading	Optional ^c	All phys
SATA down-spreading	Mandatory	Phys that support being attached to SATA phys
^a This is in addition to the physical link rate long-term tolerance defined in table 60 (see 5.3.3). ^b Phys compliant with previous versions of this standard that do not support being attached to SATA devices were only required to tolerate an SSC modulation type of none. Phys compliant with previous versions of this standard that support being attached to SATA devices were only required to tolerate SSC modulation types of none and SATA down-spreading. ^c If either the SSC modulation type of center-spreading or down-spreading is supported, both shall be supported.		

5.3.8.4 Expander device center-spreading tolerance buffer

Expander devices supporting the SSC modulation type of center-spreading shall support a center-spreading tolerance buffer for each connection with the buffer size defined in table 71 to hold any dwords it receives during the up-spreading portion(s) of the SSC modulation period that it is unable to forward because the ECR and/or the transmitting expander phy is slower than the receiving expander phy and because the dword stream does not include enough deletable primitives. The expander device unloads the center-spreading tolerance buffer during the down-spreading portion(s) of the SSC modulation period when the receiving expander phy is slower than the ECR and the transmitting expander phy.

Table 71 — Expander device center-spreading tolerance buffer

Physical link rate	Minimum buffer size
6 Gbps	14 dwords
3 Gbps	8 dwords
1,5 Gbps	4 dwords

NOTE 22 - The minimum buffer size is based on the number of dwords that may be transmitted during half of the longest allowed SSC modulation period (i.e., half of the period indicated by 30 kHz) at the maximum physical link rate (i.e., +2 400 ppm) minus the number that may be transmitted at the minimum physical link rate (i.e., -2 400 ppm). This accounts for forwarding dwords in a connection that originated from a phy compliant with previous versions of this standard (i.e., a phy with an SSC modulation type of none and inserting deletable primitives at a rate supporting only the long-term frequency stability).

Figure 121 shows an example of center-spreading tolerance buffer usage.

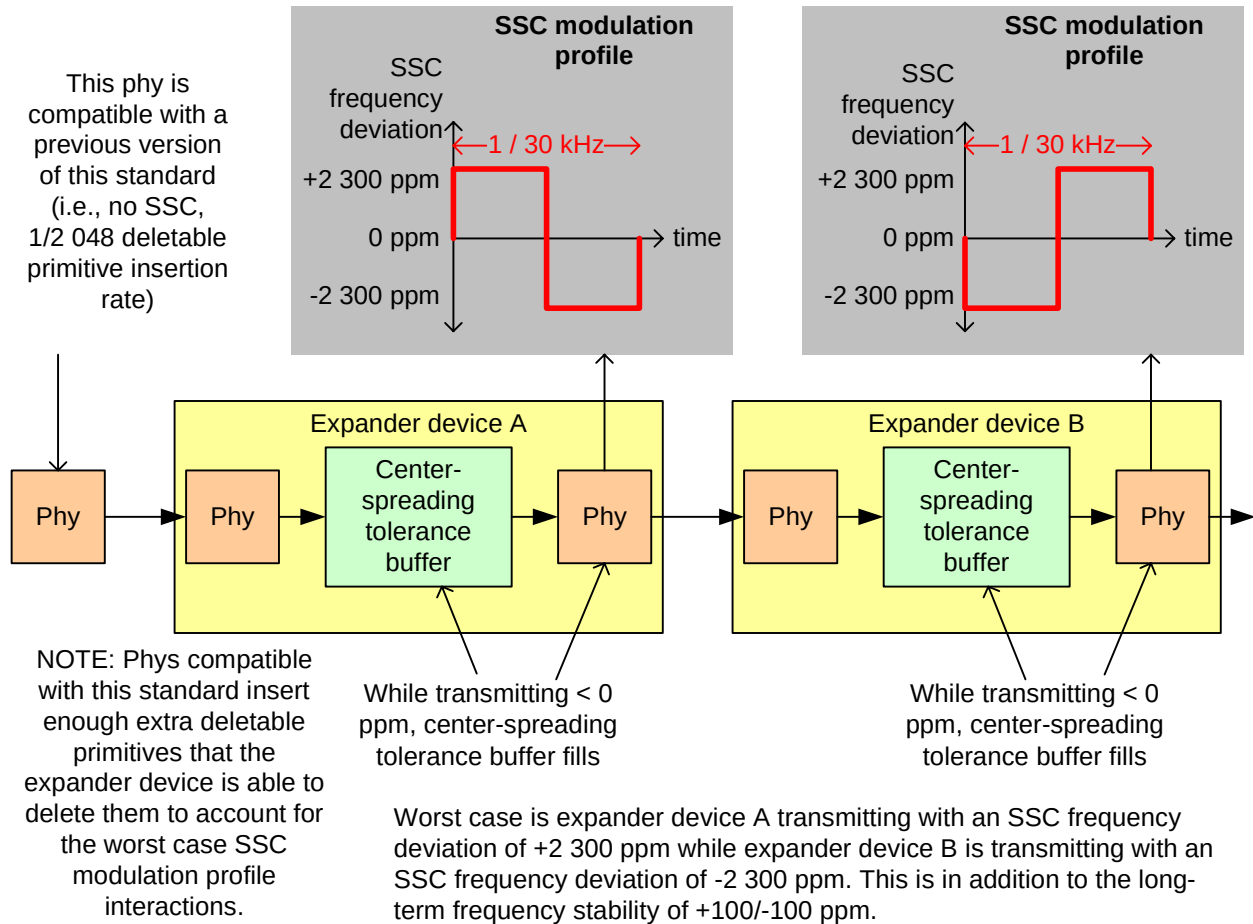


Figure 121 — Center-spreading tolerance buffer

5.3.9 Non-tracking clock architecture

Transceivers shall be designed with a non-tracking clock architecture (i.e., the receive clock derived from the bit stream received by the receiver device shall not be used as the transmit clock by the transmitter device).

Receiver devices that support SATA shall tolerate clock tracking by the SATA device. Receiver devices that do not support SATA are not required to tolerate clock tracking by the SATA device.

5.4 READY LED signal electrical characteristics

A SAS target device uses the READY LED signal to activate an externally visible LED that indicates the state of readiness and activity of the SAS target device.

All SAS target devices using the SAS Drive plug connector (see 5.2.3.2.1.1) shall support the READY LED signal.

The READY LED signal is designed to pull down the cathode of an LED using an open collector or open drain transmitter circuit. The LED and the current limiting circuitry shall be external to the SAS target device.

Table 72 describes the output characteristics of the READY LED signal.

Table 72 — Output characteristics of the READY LED signal

State	Test condition	Requirement
Negated (LED off)	$0\text{ V} \leq V_{OH} \leq 3,6\text{ V}$	$-100\text{ }\mu\text{A} < I_{OH} < 100\text{ }\mu\text{A}$
Asserted (LED on)	$I_{OL} = 15\text{ mA}$	$0 \leq V_{OL} \leq 0,225\text{ V}$

The READY LED signal behavior is defined in 10.4.1.

NOTE 23 - SATA devices use the pin used by the READY LED signal (i.e., P11) for activity indication and staggered spin-up disable (see SATA-2). The output characteristics differ from those in table 72.

6 Phy layer

6.1 Phy layer overview

The phy layer defines 8b10b coding and OOB signals. Phy layer state machines interface between the link layer and the physical layer to perform the phy reset sequence and keep track of dword synchronization.

6.2 8b10b coding

6.2.1 8b10b coding overview

All information transferred in SAS is encoded into 10-bit characters using 8b10b encoding. Information includes data bytes (e.g., representing data in a frame) and control bytes (e.g., used for frame delimiters).

Running disparity (RD) shall be maintained separately on each physical link in each direction. During a connection (see 4.1.12), expander devices shall convert incoming 10-bit characters to 8-bit bytes and generate the 10-bit character with correct disparity for the output physical link. Phys within a device may or may not begin operation with the same disparity after the reset sequence.

6.2.2 8b10b coding introduction

Information to be transmitted across a physical link shall be encoded eight bits at a time into a 10-bit character and then transmitted serially bit-by-bit across the physical link. Information received over the physical link shall be collected ten bits at a time, and those characters that are used for data, called data characters, shall be decoded into the correct 8-bit data bytes. The 10-bit characters support all 256 8-bit combinations. Some of the remaining 10-bit characters, referred to as control characters, are used for functions that are to be distinguishable from the contents of a frame. The rest of the 10-bit characters are invalid characters.

8b10b coding ensures that sufficient transitions are present in the serial bit stream to make clock recovery possible at the receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some of the control characters of the transmission code contain a distinct and easily recognizable bit pattern called a comma pattern which assists a receiver in achieving character and dword alignment on the incoming bit stream.

6.2.3 8b10b coding notation conventions

This subclause uses letter notation for describing information bits and control variables. Such notation differs from the bit notation specified by the remainder of this standard. The following text describes the translation process between these notations and provides a translation example. It also describes the conventions used to name valid characters. This text is provided for the purposes of terminology clarification only.

An unencoded information byte is composed of eight information bits A, B, C, D, E, F, G, H and the control variable Z. This information is encoded into the bits a, b, c, d, e, f, g, h, i, j of a 10-bit character.

An information bit contains either a binary zero or a binary one. A control variable has either the value D or the value K. When the control variable associated with an unencoded information byte contains the value D, that byte is referred to as a data byte. When the control variable associated with an unencoded information byte contains the value K, that byte is referred to as a control byte.

The information bit labeled A corresponds to bit 0 in the numbering scheme of this standard, B corresponds to bit 1, and so on, as shown in table 73. Bit H is the most significant bit of the byte and bit A is the least significant bit of the byte.

Table 73 — Bit designations

Bit notation:	7	6	5	4	3	2	1	0	Control variable
Unencoded bit notation:	H	G	F	E	D	C	B	A	Z

Each valid character has been given a name using the following convention:

Zxx.y

where:

- Z is the control variable of the unencoded information byte. The value of Z is used to indicate whether the character is a data character (Z = D) or a control character (Z = K).
- xx is the decimal value of the binary number composed of the bits E, D, C, B, and A of the unencoded information byte in that order.
- y is the decimal value of the binary number composed of the bits H, G, and F of the unencoded information byte in that order.

Table 74 shows the conversion from byte notation to the character naming convention.

Table 74 — Conversion from byte notation to character name example

Byte notation	BCh						
Bit notation	7	6	5	4	3	2	1 0 Control
	1	0	1	1	1	1	0 0 K
Unencoded bit notation	H	G	F	E	D	C	B A Z
	1	0	1	1	1	1	0 0 K
Unencoded bit notation reordered to conform with Zxx.y naming convention	Z	E	D	C	B	A	H G F
	K	1	1	1	0	0	1 0 1
Character name	K	28		.	5		

Most Kxx.y combinations do not result in valid characters within the 8b10b coding scheme. Only those combinations that result in control characters as specified by table 76 are considered valid.

6.3 Character encoding and decoding

6.3.1 Introduction

This subclause describes how to select valid characters (i.e., 8b10b encoding) and check the validity of received characters (i.e., 10b8b decoding). It also specifies the ordering rules to be followed when transmitting the bits within a character.

6.3.2 Transmission order

Within the definition of the 8b10b code, the bit positions of the characters are labeled a, b, c, d, e, i, f, g, h, and j. Bit a shall be transmitted first, followed by bits b, c, d, e, i, f, g, h, and j, in that order.

NOTE 24 - Bit i is transmitted between bit e and bit f, rather than in the order that would be indicated by the letters of the alphabet.

Characters within primitives shall be transmitted sequentially beginning with the control character used to distinguish the primitive (e.g., K28.3 or K28.5) and proceeding character by character from left to right within the definition of the primitive until all characters of the primitive are transmitted.

The contents of a frame shall be transmitted sequentially beginning with the primitive used to denote the start of frame (e.g., SOAF, SOF, or SATA_SOF) and proceeding character-by-character from left to right within the definition of the frame until the primitive used to denote the end of frame (e.g., EOAF, EOF, or SATA_EOF) is transmitted.

6.3.3 Data and control characters

Table 75 and table 76 define the data characters (i.e., Dxx.y characters) and control characters (i.e., Kxx.y characters), respectively, and shall be used for both generating characters (i.e., encoding) and checking the validity of received characters (i.e., decoding). Each data character and control character entry has two columns that represent two (not necessarily different) characters, corresponding to the current value of the running disparity (current RD - or current RD +). RD is a binary parameter with a negative (-) or positive (+) value.

After power on, the transmitter may initialize the current RD to either positive or negative. Upon transmission of any character, the transmitter shall calculate a new value for its RD based on the contents of the transmitted character.

After power on, the receiver shall assume either the positive or negative value for its initial RD. Upon reception of any character, the receiver shall determine whether the character is valid or invalid and shall calculate a new value for its RD based on the contents of the received character.

The following rules for RD shall be used to calculate the new RD value for characters that have been transmitted (i.e., the transmitter's RD) and that have been received (i.e., the receiver's RD).

RD for a character shall be calculated on the basis of sub-blocks, where the first six bits (i.e., bits a, b, c, d, e, and i) form one sub-block (i.e., the six-bit sub-block) and the second four bits (i.e., bits f, g, h, and j) form the other sub-block (i.e., the four-bit sub-block):

- a) RD at the beginning of the six-bit sub-block is the RD at the end of the preceding character;
- b) RD at the beginning of the four-bit sub-block is the RD at the end of the preceding six-bit sub-block; and
- c) RD at the end of the character is the RD at the end of the four-bit sub-block.

RD for the sub-blocks shall be calculated as follows:

- a) If the sub-block contains more ones than zeros, then RD at the end of a sub-block is positive;
- b) If the sub-block contains more zeros than ones, then RD at the end of a sub-block is negative;
- c) If the sub-block contains equal numbers of zeros and ones, then:
 - A) if it is a six-bit sub-block containing 000111b, then RD at the end of the sub-block is positive;
 - B) if it is a six-bit sub-block containing 111000b, then RD at the end of the sub-block is negative;
 - C) if it is a four-bit sub-block containing 0011b, then RD at the end of the sub-block is positive;
 - D) if it is a four-bit sub-block containing 1100b, then RD at the end of the sub-block is negative; and
 - E) otherwise, RD at the end of the sub-block is the same as at the beginning of the sub-block.

All sub-blocks with equal numbers of zeros and ones have neutral disparity (i.e., the ending disparity is the same as the beginning disparity). In order to limit the run length of zeros or ones across adjacent sub-blocks, the 8b10b code rules specify that sub-blocks encoded as 000111b or 0011b are generated only when the RD at the beginning of the sub-block is positive, ensuring that RD at the end of these sub-blocks is also positive. Likewise, sub-blocks containing 111000b or 1100b are generated only when the RD at the beginning of the sub-block is negative, ensuring that RD at the end of these sub-blocks is also negative.

Table 75 defines the data characters.

Table 75 — Data characters (part 1 of 5)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D00.0	000 00000	00h	100111 0100	011000 1011
D01.0	000 00001	01h	011101 0100	100010 1011
D02.0	000 00010	02h	101101 0100	010010 1011
D03.0	000 00011	03h	110001 1011	110001 0100
D04.0	000 00100	04h	110101 0100	001010 1011
D05.0	000 00101	05h	101001 1011	101001 0100
D06.0	000 00110	06h	011001 1011	011001 0100
D07.0	000 00111	07h	111000 1011	000111 0100
D08.0	000 01000	08h	111001 0100	000110 1011
D09.0	000 01001	09h	100101 1011	100101 0100
D10.0	000 01010	0Ah	010101 1011	010101 0100
D11.0	000 01011	0Bh	110100 1011	110100 0100
D12.0	000 01100	0Ch	001101 1011	001101 0100
D13.0	000 01101	0Dh	101100 1011	101100 0100
D14.0	000 01110	0Eh	011100 1011	011100 0100
D15.0	000 01111	0Fh	010111 0100	101000 1011
D16.0	000 10000	10h	011011 0100	100100 1011
D17.0	000 10001	11h	100011 1011	100011 0100
D18.0	000 10010	12h	010011 1011	010011 0100
D19.0	000 10011	13h	110010 1011	110010 0100
D20.0	000 10100	14h	001011 1011	001011 0100
D21.0	000 10101	15h	101010 1011	101010 0100
D22.0	000 10110	16h	011010 1011	011010 0100
D23.0	000 10111	17h	111010 0100	000101 1011
D24.0	000 11000	18h	110011 0100	001100 1011
D25.0	000 11001	19h	100110 1011	100110 0100
D26.0	000 11010	1Ah	010110 1011	010110 0100
D27.0	000 11011	1Bh	110110 0100	001001 1011
D28.0	000 11100	1Ch	001110 1011	001110 0100
D29.0	000 11101	1Dh	101110 0100	010001 1011
D30.0	000 11110	1Eh	011110 0100	100001 1011
D31.0	000 11111	1Fh	101011 0100	010100 1011
D00.1	001 00000	20h	100111 1001	011000 1001
D01.1	001 00001	21h	011101 1001	100010 1001
D02.1	001 00010	22h	101101 1001	010010 1001
D03.1	001 00011	23h	110001 1001	110001 1001
D04.1	001 00100	24h	110101 1001	001010 1001
D05.1	001 00101	25h	101001 1001	101001 1001
D06.1	001 00110	26h	011001 1001	011001 1001
D07.1	001 00111	27h	111000 1001	000111 1001
D08.1	001 01000	28h	111001 1001	000110 1001
D09.1	001 01001	29h	100101 1001	100101 1001
D10.1	001 01010	2Ah	010101 1001	010101 1001
D11.1	001 01011	2Bh	110100 1001	110100 1001
D12.1	001 01100	2Ch	001101 1001	001101 1001
D13.1	001 01101	2Dh	101100 1001	101100 1001
D14.1	001 01110	2Eh	011100 1001	011100 1001
D15.1	001 01111	2Fh	010111 1001	101000 1001
D16.1	001 10000	30h	011011 1001	100100 1001
D17.1	001 10001	31h	100011 1001	100011 1001
D18.1	001 10010	32h	010011 1001	010011 1001
D19.1	001 10011	33h	110010 1001	110010 1001

Table 75 — Data characters (part 2 of 5)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D20.1	001 10100	34h	001011 1001	001011 1001
D21.1	001 10101	35h	101010 1001	101010 1001
D22.1	001 10110	36h	011010 1001	011010 1001
D23.1	001 10111	37h	111010 1001	000101 1001
D24.1	001 11000	38h	110011 1001	001100 1001
D25.1	001 11001	39h	100110 1001	100110 1001
D26.1	001 11010	3Ah	010110 1001	010110 1001
D27.1	001 11011	3Bh	110110 1001	001001 1001
D28.1	001 11100	3Ch	001110 1001	001110 1001
D29.1	001 11101	3Dh	101110 1001	010001 1001
D30.1	001 11110	3Eh	011110 1001	100001 1001
D31.1	001 11111	3Fh	101011 1001	010100 1001
D00.2	010 00000	40h	100111 0101	011000 0101
D01.2	010 00001	41h	011101 0101	100010 0101
D02.2	010 00010	42h	101101 0101	010010 0101
D03.2	010 00011	43h	110001 0101	110001 0101
D04.2	010 00100	44h	110101 0101	001010 0101
D05.2	010 00101	45h	101001 0101	101001 0101
D06.2	010 00110	46h	011001 0101	011001 0101
D07.2	010 00111	47h	111000 0101	000111 0101
D08.2	010 01000	48h	111001 0101	000110 0101
D09.2	010 01001	49h	100101 0101	100101 0101
D10.2	010 01010	4Ah	010101 0101	010101 0101
D11.2	010 01011	4Bh	110100 0101	110100 0101
D12.2	010 01100	4Ch	001101 0101	001101 0101
D13.2	010 01101	4Dh	101100 0101	101100 0101
D14.2	010 01110	4Eh	011100 0101	011100 0101
D15.2	010 01111	4Fh	010111 0101	101000 0101
D16.2	010 10000	50h	011011 0101	100100 0101
D17.2	010 10001	51h	100011 0101	100011 0101
D18.2	010 10010	52h	010011 0101	010011 0101
D19.2	010 10011	53h	110010 0101	110010 0101
D20.2	010 10100	54h	001011 0101	001011 0101
D21.2	010 10101	55h	101010 0101	101010 0101
D22.2	010 10110	56h	011010 0101	011010 0101
D23.2	010 10111	57h	111010 0101	000101 0101
D24.2	010 11000	58h	110011 0101	001100 0101
D25.2	010 11001	59h	100110 0101	100110 0101
D26.2	010 11010	5Ah	010110 0101	010110 0101
D27.2	010 11011	5Bh	110110 0101	001001 0101
D28.2	010 11100	5Ch	001110 0101	001110 0101
D29.2	010 11101	5Dh	101110 0101	010001 0101
D30.2	010 11110	5Eh	011110 0101	100001 0101
D31.2	010 11111	5Fh	101011 0101	010100 0101
D00.3	011 00000	60h	100111 0011	011000 1100
D01.3	011 00001	61h	011101 0011	100010 1100
D02.3	011 00010	62h	101101 0011	010010 1100
D03.3	011 00011	63h	110001 1100	110001 0011
D04.3	011 00100	64h	110101 0011	001010 1100
D05.3	011 00101	65h	101001 1100	101001 0011
D06.3	011 00110	66h	011001 1100	011001 0011
D07.3	011 00111	67h	111000 1100	000111 0011
D08.3	011 01000	68h	111001 0011	000110 1100
D09.3	011 01001	69h	100101 1100	100101 0011

Table 75 — Data characters (part 3 of 5)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D10.3	011 01010	6Ah	010101 1100	010101 0011
D11.3	011 01011	6Bh	110100 1100	110100 0011
D12.3	011 01100	6Ch	001101 1100	001101 0011
D13.3	011 01101	6Dh	101100 1100	101100 0011
D14.3	011 01110	6Eh	011100 1100	011100 0011
D15.3	011 01111	6Fh	010111 0011	101000 1100
D16.3	011 10000	70h	011011 0011	100100 1100
D17.3	011 10001	71h	100011 1100	100011 0011
D18.3	011 10010	72h	010011 1100	010011 0011
D19.3	011 10011	73h	110010 1100	110010 0011
D20.3	011 10100	74h	001011 1100	001011 0011
D21.3	011 10101	75h	101010 1100	101010 0011
D22.3	011 10110	76h	011010 1100	011010 0011
D23.3	011 10111	77h	111010 0011	000101 1100
D24.3	011 11000	78h	110011 0011	001100 1100
D25.3	011 11001	79h	100110 1100	100110 0011
D26.3	011 11010	7Ah	010110 1100	010110 0011
D27.3	011 11011	7Bh	110110 0011	001001 1100
D28.3	011 11100	7Ch	001110 1100	001110 0011
D29.3	011 11101	7Dh	101110 0011	010001 1100
D30.3	011 11110	7Eh	011110 0011	100001 1100
D31.3	011 11111	7Fh	101011 0011	010100 1100
D00.4	100 00000	80h	100111 0010	011000 1101
D01.4	100 00001	81h	011101 0010	100010 1101
D02.4	100 00010	82h	101101 0010	010010 1101
D03.4	100 00011	83h	110001 1101	110001 0010
D04.4	100 00100	84h	110101 0010	001010 1101
D05.4	100 00101	85h	101001 1101	101001 0010
D06.4	100 00110	86h	011001 1101	011001 0010
D07.4	100 00111	87h	111000 1101	000111 0010
D08.4	100 01000	88h	111001 0010	000110 1101
D09.4	100 01001	89h	100101 1101	100101 0010
D10.4	100 01010	8Ah	010101 1101	010101 0010
D11.4	100 01011	8Bh	110100 1101	110100 0010
D12.4	100 01100	8Ch	001101 1101	001101 0010
D13.4	100 01101	8Dh	101100 1101	101100 0010
D14.4	100 01110	8Eh	011100 1101	011100 0010
D15.4	100 01111	8Fh	010111 0010	101000 1101
D16.4	100 10000	90h	011011 0010	100100 1101
D17.4	100 10001	91h	100011 1101	100011 0010
D18.4	100 10010	92h	010011 1101	010011 0010
D19.4	100 10011	93h	110010 1101	110010 0010
D20.4	100 10100	94h	001011 1101	001011 0010
D21.4	100 10101	95h	101010 1101	101010 0010
D22.4	100 10110	96h	011010 1101	011010 0010
D23.4	100 10111	97h	111010 0010	000101 1101
D24.4	100 11000	98h	110011 0010	001100 1101
D25.4	100 11001	99h	100110 1101	100110 0010
D26.4	100 11010	9Ah	010110 1101	010110 0010
D27.4	100 11011	9Bh	110110 0010	001001 1101
D28.4	100 11100	9Ch	001110 1101	001110 0010
D29.4	100 11101	9Dh	101110 0010	010001 1101
D30.4	100 11110	9Eh	011110 0010	100001 1101
D31.4	100 11111	9Fh	101011 0010	010100 1101

Table 75 — Data characters (part 4 of 5)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D00.5	101 00000	A0h	100111 1010	011000 1010
D01.5	101 00001	A1h	011101 1010	100010 1010
D02.5	101 00010	A2h	101101 1010	010010 1010
D03.5	101 00011	A3h	110001 1010	110001 1010
D04.5	101 00100	A4h	110101 1010	001010 1010
D05.5	101 00101	A5h	101001 1010	101001 1010
D06.5	101 00110	A6h	011001 1010	011001 1010
D07.5	101 00111	A7h	111000 1010	000111 1010
D08.5	101 01000	A8h	111001 1010	000110 1010
D09.5	101 01001	A9h	100101 1010	100101 1010
D10.5	101 01010	AAh	010101 1010	010101 1010
D11.5	101 01011	ABh	110100 1010	110100 1010
D12.5	101 01100	ACh	001101 1010	001101 1010
D13.5	101 01101	ADh	101100 1010	101100 1010
D14.5	101 01110	AEnh	011100 1010	011100 1010
D15.5	101 01111	AFh	010111 1010	101000 1010
D16.5	101 10000	B0h	011011 1010	100100 1010
D17.5	101 10001	B1h	100011 1010	100011 1010
D18.5	101 10010	B2h	010011 1010	010011 1010
D19.5	101 10011	B3h	110010 1010	110010 1010
D20.5	101 10100	B4h	001011 1010	001011 1010
D21.5	101 10101	B5h	101010 1010	101010 1010
D22.5	101 10110	B6h	011010 1010	011010 1010
D23.5	101 10111	B7h	111010 1010	000101 1010
D24.5	101 11000	B8h	110011 1010	001100 1010
D25.5	101 11001	B9h	100110 1010	100110 1010
D26.5	101 11010	BAh	010110 1010	010110 1010
D27.5	101 11011	BBh	110110 1010	001001 1010
D28.5	101 11100	BCh	001110 1010	001110 1010
D29.5	101 11101	BDh	101110 1010	010001 1010
D30.5	101 11110	BEh	011110 1010	100001 1010
D31.5	101 11111	BFh	101011 1010	010100 1010
D00.6	110 00000	C0h	100111 0110	011000 0110
D01.6	110 00001	C1h	011101 0110	100010 0110
D02.6	110 00010	C2h	101101 0110	010010 0110
D03.6	110 00011	C3h	110001 0110	110001 0110
D04.6	110 00100	C4h	110101 0110	001010 0110
D05.6	110 00101	C5h	101001 0110	101001 0110
D06.6	110 00110	C6h	011001 0110	011001 0110
D07.6	110 00111	C7h	111000 0110	000111 0110
D08.6	110 01000	C8h	111001 0110	000110 0110
D09.6	110 01001	C9h	100101 0110	100101 0110
D10.6	110 01010	CAh	010101 0110	010101 0110
D11.6	110 01011	CBh	110100 0110	110100 0110
D12.6	110 01100	CCh	001101 0110	001101 0110
D13.6	110 01101	CDh	101100 0110	101100 0110
D14.6	110 01110	CEh	011100 0110	011100 0110
D15.6	110 01111	CFh	010111 0110	101000 0110
D16.6	110 10000	D0h	011011 0110	100100 0110
D17.6	110 10001	D1h	100011 0110	100011 0110
D18.6	110 10010	D2h	010011 0110	010011 0110
D19.6	110 10011	D3h	110010 0110	110010 0110
D20.6	110 10100	D4h	001011 0110	001011 0110
D21.6	110 10101	D5h	101010 0110	101010 0110

Table 75 — Data characters (part 5 of 5)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D22.6	110 10110	D6h	011010 0110	011010 0110
D23.6	110 10111	D7h	111010 0110	000101 0110
D24.6	110 11000	D8h	110011 0110	001100 0110
D25.6	110 11001	D9h	100110 0110	100110 0110
D26.6	110 11010	DAh	010110 0110	010110 0110
D27.6	110 11011	DBh	110110 0110	001001 0110
D28.6	110 11100	DCh	001110 0110	001110 0110
D29.6	110 11101	DDh	101110 0110	010001 0110
D30.6	110 11110	DEh	011110 0110	100001 0110
D31.6	110 11111	DFh	101011 0110	010100 0110
D00.7	111 00000	E0h	100111 0001	011000 1110
D01.7	111 00001	E1h	011101 0001	100010 1110
D02.7	111 00010	E2h	101101 0001	010010 1110
D03.7	111 00011	E3h	110001 1110	110001 0001
D04.7	111 00100	E4h	110101 0001	001010 1110
D05.7	111 00101	E5h	101001 1110	101001 0001
D06.7	111 00110	E6h	011001 1110	011001 0001
D07.7	111 00111	E7h	111000 1110	000111 0001
D08.7	111 01000	E8h	111001 0001	000110 1110
D09.7	111 01001	E9h	100101 1110	100101 0001
D10.7	111 01010	EAh	010101 1110	010101 0001
D11.7	111 01011	EBh	110100 1110	110100 1000
D12.7	111 01100	ECh	001101 1110	001101 0001
D13.7	111 01101	EDh	101100 1110	101100 1000
D14.7	111 01110	EEh	011100 1110	011100 1000
D15.7	111 01111	EFh	010111 0001	101000 1110
D16.7	111 10000	F0h	011011 0001	100100 1110
D17.7	111 10001	F1h	100011 0111	100011 0001
D18.7	111 10010	F2h	010011 0111	010011 0001
D19.7	111 10011	F3h	110010 1110	110010 0001
D20.7	111 10100	F4h	001011 0111	001011 0001
D21.7	111 10101	F5h	101010 1110	101010 0001
D22.7	111 10110	F6h	011010 1110	011010 0001
D23.7	111 10111	F7h	111010 0001	000101 1110
D24.7	111 11000	F8h	110011 0001	001100 1110
D25.7	111 11001	F9h	100110 1110	100110 0001
D26.7	111 11010	FAh	010110 1110	010110 0001
D27.7	111 11011	FBh	110110 0001	001001 1110
D28.7	111 11100	FCh	001110 1110	001110 0001
D29.7	111 11101	FDh	101110 0001	010001 1110
D30.7	111 11110	FEh	011110 0001	100001 1110
D31.7	111 11111	FFh	101011 0001	010100 1110

Table 76 defines the control characters. Comma patterns, which are two bits of one polarity followed by five bits of the opposite polarity (i.e., 0011111b or 1100000b), are underlined.

Table 76 — Control characters

Name	Control byte		Control character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	1Ch	001111 0100	110000 1011
K28.1	001 11100	3Ch	<u>001111</u> 1001	<u>110000</u> 0110
K28.2	010 11100	5Ch	001111 0101	110000 1010
K28.3	011 11100	7Ch	001111 0011	110000 1100
K28.4	100 11100	9Ch	001111 0010	110000 1101
K28.5	101 11100	BCh	<u>001111</u> 1010	<u>110000</u> 0101
K28.6	110 11100	DCh	001111 0110	110000 1001
K28.7	111 11100	FCh	<u>001111</u> 1000	<u>110000</u> 0111
K23.7	111 10111	F7h	111010 1000	000101 0111
K27.7	111 11011	FBh	110110 1000	001001 0111
K29.7	111 11101	FDh	101110 1000	010001 0111
K30.7	111 11110	FEh	011110 1000	100001 0111

NOTE 25 - K28.1, K28.5, and K28.7 are the only characters which contain comma patterns. Comma patterns do not appear in any data characters and do not appear across any adjacent data characters. The K28.7 control character introduces a comma pattern when followed by any of the following characters: K28.y, D3.y, D11.y, D12.y, D19.y, D20.y, or D28.y, where y is a value in the range 0 to 7, inclusive. None of the other control characters introduce a comma pattern when adjacent to any other character. Therefore, K28.7 is not used, ensuring that comma patterns do not appear in any sequence of characters except the first 7 bits of K28.1 or K28.5.

The only control characters used in this standard are K28.3, K28.5, and K28.6, as defined in table 77.

Table 77 — Control character usage

First character of a dword	Usage in SAS physical links	Usage in SATA physical links
K28.3	Primitives used only inside STP connections	All primitives except ALIGN
K28.5	ALIGN and most primitives defined in this standard	ALIGN
K28.6	Not used	SATA_ERROR

See 7.2 for details on primitives, which use those control characters.

6.3.4 Encoding characters in the transmitter

To transmit a data byte, the transmitter shall select the appropriate character from table 75 based on the current value of the transmitter's RD. To transmit a control byte, the transmitter shall select the appropriate character from table 76 based on the current value of the transmitter's RD. After the transmitting the character,

the transmitter shall calculate a new value for its RD based on that character. This new value shall be used as the transmitter's current RD for the next character transmitted. This process is called 8b10b encoding.

6.3.5 Decoding characters in the receiver

After receiving a character, the receiver shall search the character column in table 75 and table 76 corresponding to its current RD to determine the data byte or control byte to which the character corresponds. This process is called 10b8b decoding. If the received character is not found in the proper column, then the character shall be considered invalid and the dword containing the character shall be considered an invalid dword.

Regardless of the received character's validity, the received character shall be used to calculate a new value of RD in the receiver. This new value shall be used as the receiver's current RD for the next received character.

Detection of a code violation does not necessarily indicate that the character in which the code violation was detected is in error. Code violations may result from a prior error that altered the RD of the bit stream but did not result in a detectable error at the character in which the error occurred. The example shown in table 78 exhibits this behavior. These errors may span dword boundaries. Expanders forwarding such a dword forward it as an ERROR (see 7.2.5.8).

Table 78 — Delayed code violation example

	RD	First character	RD	Second character	RD	Third character	RD
Transmitted character stream	-	D21.1	-	D10.2	-	D23.5	+
Transmitted bit stream	-	101010 1001	-	010101 0101	-	111010 1010	+
Bit stream after error	-	101010 10 <u>1</u> 1 (error in second to last bit)	+	010101 0101	+	111010 1010	+
Decoded character stream	-	D21.0 (rather than D21.1) (not detected as an error)	+	D10.2 (no error)	+	Code violation (although D23.5 was properly received)	+

6.4 Dwords, primitives, data dwords, and invalid dwords

All characters transferred in SAS are grouped into four-character sequences called dwords.

A primitive is a dword whose first character is K28.3 or K28.5 and whose remaining three characters are data characters with correct disparity.

Primitives are defined with both negative and positive starting RD (see 6.3.3). SAS defines primitives starting with K28.5 (see 7.2.5 and 7.2.6). SATA defines primitives starting with K28.3 and K28.5, which are used in SAS during STP connections (see 7.2.7).

A data dword is a dword that contains four data characters with correct disparity.

A dword containing an invalid character shall be considered an invalid dword.

6.5 Bit order

Dwords transmitted in a STP connection shall be transmitted in the bit order specified by SATA.

Dwords for other types of connections and outside of connections shall be transmitted in the bit order in figure 122.

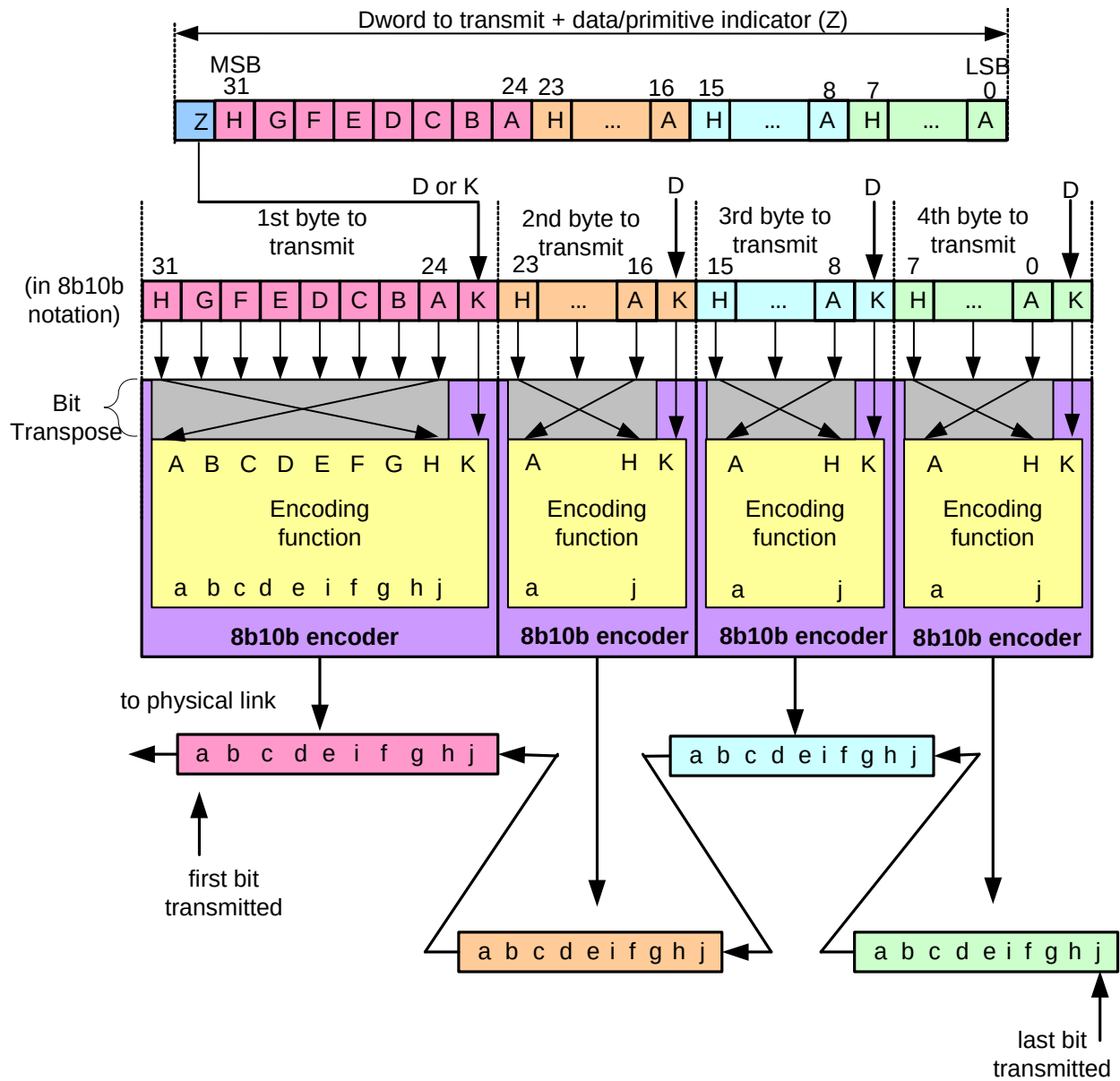


Figure 122 — SAS bit transmission logic

Figure 123 shows the SAS bit reception order.

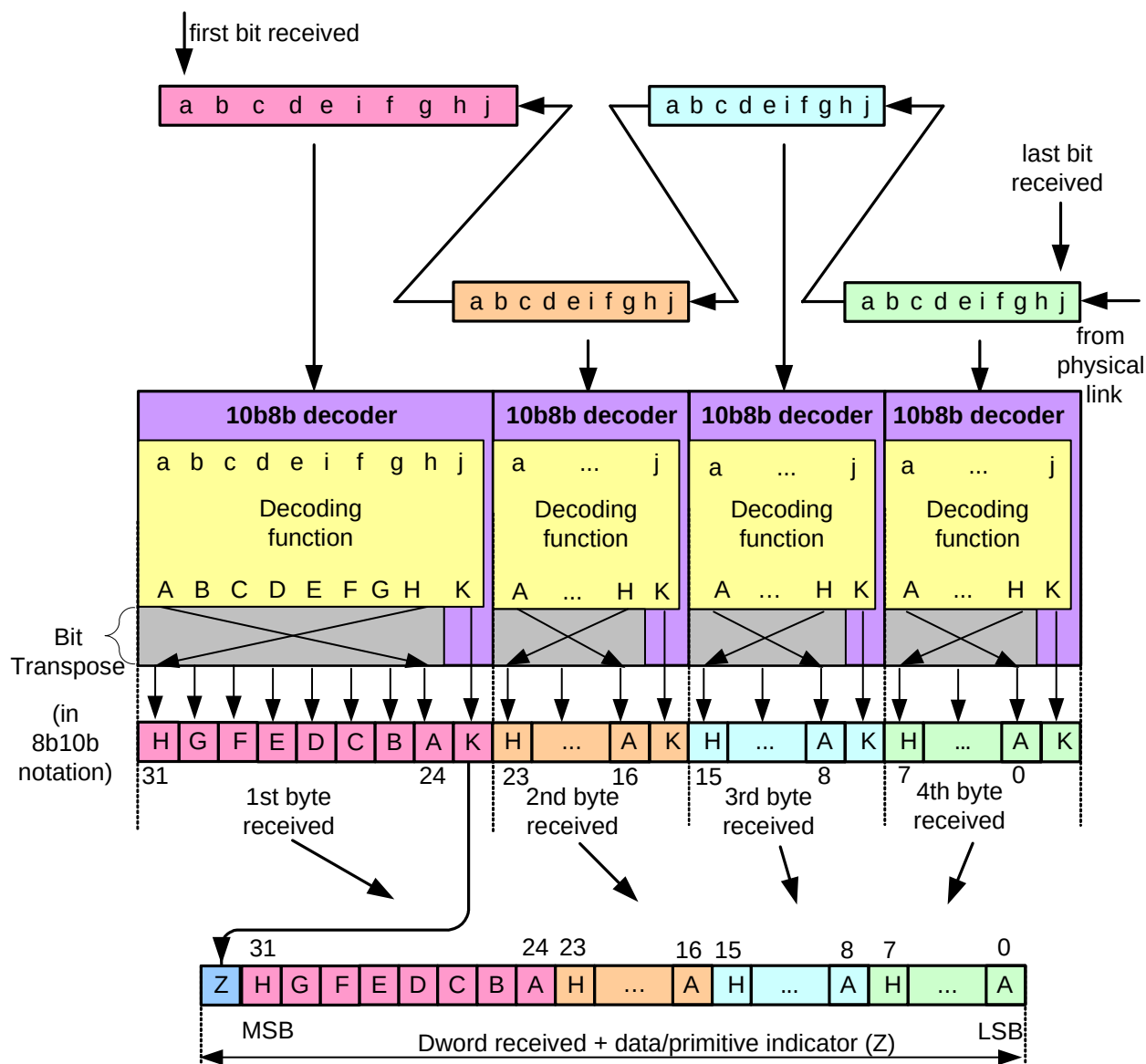


Figure 123 — SAS bit reception logic

6.6 Out of band (OOB) signals

6.6.1 OOB signals overview

Out of band (OOB) signals are low-speed signal patterns that do not appear in normal data streams. OOB signals consist of defined amounts of idle time followed by defined amounts of burst time. During the idle time, the physical link carries D.C. idle (see 3.1.45). During the burst time, the physical link carries signal transitions. The signals are differentiated by the length of idle time between the burst times.

SATA defines two OOB signals: COMINIT/COMRESET and COMWAKE. COMINIT and COMRESET are used in this standard interchangeably. Phys compliant with this standard identify themselves with an additional SAS-specific OOB signal called COMSAS.

Table 79 defines the timing specifications for OOB signals.

Table 79 — OOB signal timing specifications

Parameter	Minimum	Nominal	Maximum	Comments
OOB Interval (OOBI) ^a	665,067 ps	666,6 ps	668,267 ps	The time basis for burst times and idle times used to create OOB signals. Based on 1,5 Gbps clock tolerance with center-spreading SSC (see table 58 in 5.3.3 and table 67 in 5.3.8.1).
COMSAS detect timeout	13,65 μs			The minimum time a receiver device shall allow to detect COMSAS after transmitting COMSAS. Derived from: $OOBI \times 512 \times 40$
^a OOBI is different than UI(OOB) defined in SATA (e.g., SAS has tighter clock tolerance). This is a fixed value equal to the UI for 1,5 Gbps with center-spreading SSC, regardless of the actual transfer rate being used to create the burst time.				

NOTE 26 - Previous versions of this standard defined OOBI based on the nominal UI for 1,5 Gbps with physical link rate long term stability tolerance (see table 58 in 5.3.3) but not with SSC modulation (see table 67 in 5.3.8.1). Interconnects compliant with previous versions of this standard may have assumed phys had that characteristic. To interoperate with interconnects compliant with previous versions of this standard, phys should create OOB burst times and idle times based on the UI for 1,5 Gbps without SSC modulation (e.g., if SSC is disabled, reflecting physical link rate long term stability but not SSC modulation).

Editor's Note 23: The COMSAS detect timeout calculation is based on the nominal OOBI. If based on the minimum OOBI, it would be 13.62 microseconds. If based on the maximum OOBI, it would be 13.686 microseconds. Was the proper OOBI value used for the calculation?

6.6.2 Transmitting OOB signals

Table 80 describes the OOB signal transmitter requirements for the burst time, idle time, negation times, and signal times that are used to form each OOB signal.

Table 80 — OOB signal transmitter device requirements

Signal	Burst time	Idle time	Negation time	Signal time ^e
COMWAKE	160 OOB ^a	160 OOB ^a	280 OOB ^b	2 200 OOB ^g
COMINIT/COMRESET	160 OOB ^a	480 OOB ^c	800 OOB ^d	4 640 OOB ⁱ
COMSAS	160 OOB ^a	1 440 OOB ^f	2 400 OOB ^h	12 000 OOB ^j
^a 160 OOB ⁱ is nominally 106,6 ns (see table 79). ^b 280 OOB ⁱ is nominally 186,6 ns. ^c 480 OOB ⁱ is nominally 320 ns. ^d 800 OOB ⁱ is nominally 533,3 ns. ^e A signal time is six burst times plus six idle times plus one negation time. ^f 1 440 OOB ⁱ is nominally 960 ns. ^g 2 200 OOB ⁱ (e.g., COMWAKE) is nominally 1 466,6 ns. ^h 2 400 OOB ⁱ is nominally 1 600 ns. ⁱ 4 640 OOB ⁱ (e.g., COMINIT/COMRESET) is nominally 3 093,3 ns. ^j 12 000 OOB ⁱ (e.g., COMSAS) is nominally 8 000 ns.				

To transmit an OOB signal, the transmitter device shall repeat these steps six times:

- 1) transmit D.C. idle for an idle time; and
- 2) transmit an OOB burst with either starting disparity consisting of D24.3 characters or ALIGN (0) primitives for a burst time. The OOB burst should consist of D24.3 characters.

NOTE 27 - Transmitter devices compliant with future versions of this standard may not transmit OOB bursts consisting of ALIGN [0] primitives.

The transmitter device shall then transmit D.C. idle for an OOB signal negation time.

The transmitter device shall use signal output levels during burst time and idle time as described in 5.3.6.5.

The D24.3 characters or ALIGN (0) primitives used in OOB signals shall be transmitted at 1,5 Gbps. The OOB burst is only required to generate an envelope for the detection circuitry, as required for any signaling that may be A.C. coupled. A burst of D24.3 characters at 1,5 Gbps is equivalent to a square wave pattern that has a one (see 6.3.3) for 2 OOBⁱ and a zero (see 6.3.3) for 2 OOBⁱ. A transmitter may use this square wave pattern for the OOB signal. The start of the pattern may be one or zero. The signal rise and fall times:

- a) shall be greater than (i.e., slower) or equal to the minimum (i.e., fastest) rise and fall times allowed by the fastest supported physical link rate of the transmitter device (see table 61 in 5.3.6.2); and
- b) shall be less than (i.e., faster) or equal to the maximum (i.e., slowest) rise and fall times allowed at 1,5 Gbps.

Figure 124 describes OOB signal transmission by the SP transmitter (see 6.8). The COMWAKE Transmitted, COMINIT Transmitted, and COMSAS Transmitted messages are sent to the SP state machine (see 6.8).

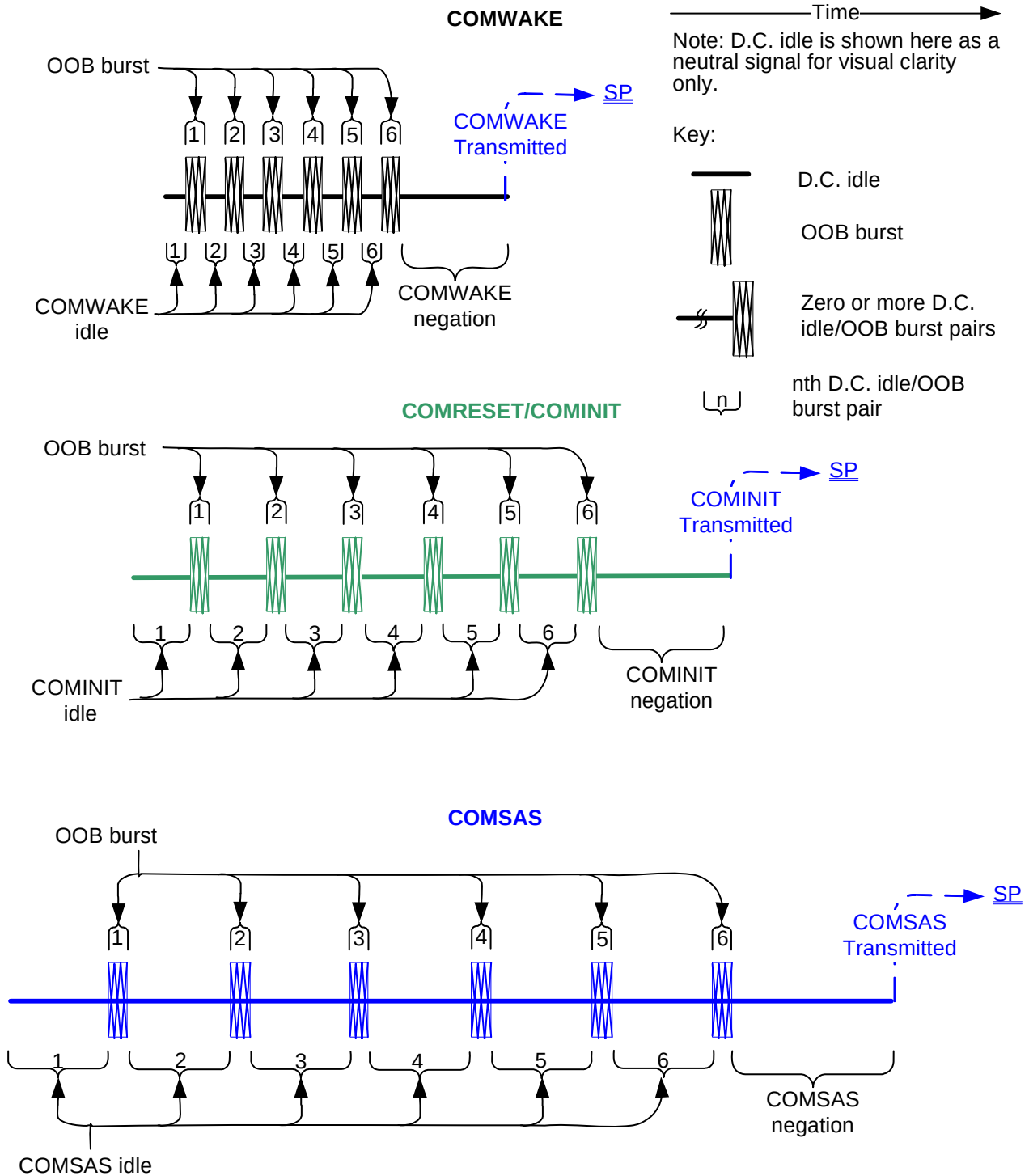


Figure 124 — OOB signal transmission

6.6.3 Receiving OOB signals

Table 81 describes the OOB signal receiver device requirements for detecting burst times, assuming T_{burst} is the length of the detected burst time. The burst time is not used to distinguish between signals.

Table 81 — OOB signal receiver device burst time detection requirements

Signal	may detect	shall detect
COMWAKE	$T_{burst} \leq 100 \text{ ns}$	$T_{burst} > 100 \text{ ns}$
COMINIT/COMRESET	$T_{burst} \leq 100 \text{ ns}$	$T_{burst} > 100 \text{ ns}$
COMSAS	$T_{burst} \leq 100 \text{ ns}$	$T_{burst} > 100 \text{ ns}$

Table 82 describes the OOB signal receiver device requirements for detecting idle times, assuming T_{idle} is the length of the detected idle time.

Table 82 — OOB signal receiver device idle time detection requirements

Signal	may detect	shall detect	shall not detect
COMWAKE	$35 \text{ ns} \leq T_{idle} < 175 \text{ ns}$	$101,3 \text{ ns} \leq T_{idle} \leq 112 \text{ ns}$	$T_{idle} < 35 \text{ ns}$ or $T_{idle} \geq 175 \text{ ns}$
COMINIT/ COMRESET	$175 \text{ ns} \leq T_{idle} < 525 \text{ ns}$	$304 \text{ ns} \leq T_{idle} \leq 336 \text{ ns}$	$T_{idle} < 175 \text{ ns}$ or $T_{idle} \geq 525 \text{ ns}$
COMSAS	$525 \text{ ns} \leq T_{idle} < 1\,575 \text{ ns}$	$911,7 \text{ ns} \leq T_{idle} \leq 1\,008 \text{ ns}$	$T_{idle} < 525 \text{ ns}$ or $T_{idle} \geq 1\,575 \text{ ns}$

Table 83 describes the OOB signal receiver device requirements for detecting negation times, assuming T_{idle} is the length of the detected idle time.

Table 83 — OOB signal receiver device negation time detection requirements

Signal	shall detect
COMWAKE	$T_{idle} > 175 \text{ ns}$
COMINIT/COMRESET	$T_{idle} > 525 \text{ ns}$
COMSAS	$T_{idle} > 1\,575 \text{ ns}$

A receiver device shall detect an OOB signal after receiving four consecutive idle time/burst time pairs (see figure 125) while the SP_DWS state machine (see 6.9) has not achieved dword synchronization (see 6.8.4.9 and 6.8.5.8), and may but should not detect an OOB signal after receiving four consecutive idle time/burst time pairs while the SP_DWS state machine has achieved dword synchronization. It is not an error to receive more than four idle time/burst time pairs. A receiver device shall not detect the same OOB signal again until it has detected the corresponding negation time (e.g., a COMINIT negation time for a COMINIT) or has detected a different OOB signal (e.g., if a receiver device previously detected COMINIT, then receives four sets of COMWAKE idle times followed by burst times, the receiver device detects COMWAKE. The receiver device may then detect COMINIT again).

A SAS receiver device shall detect OOB bursts formed from any of the following:

- D24.3 characters at 1,5 Gbps;
- ALIGN (0) primitives at 1,5 Gbps; or
- ALIGN (0) primitives at 3 Gbps.

NOTE 28 - ALIGN (0) primitives at 3 Gbps provide interoperability with transmitter devices compliant with previous versions of this standard and SATA.

A SAS receiver device shall not qualify the OOB burst based on the characters received.

Figure 125 describes SAS OOB signal detection by the SP receiver (see 6.8). The COMWAKE Detected, COMWAKE Completed, COMINIT Detected, COMSAS Detected, and COMSAS Completed messages are sent to the SP state machine (see 6.8) to indicate that an OOB signal has been partially or fully detected.

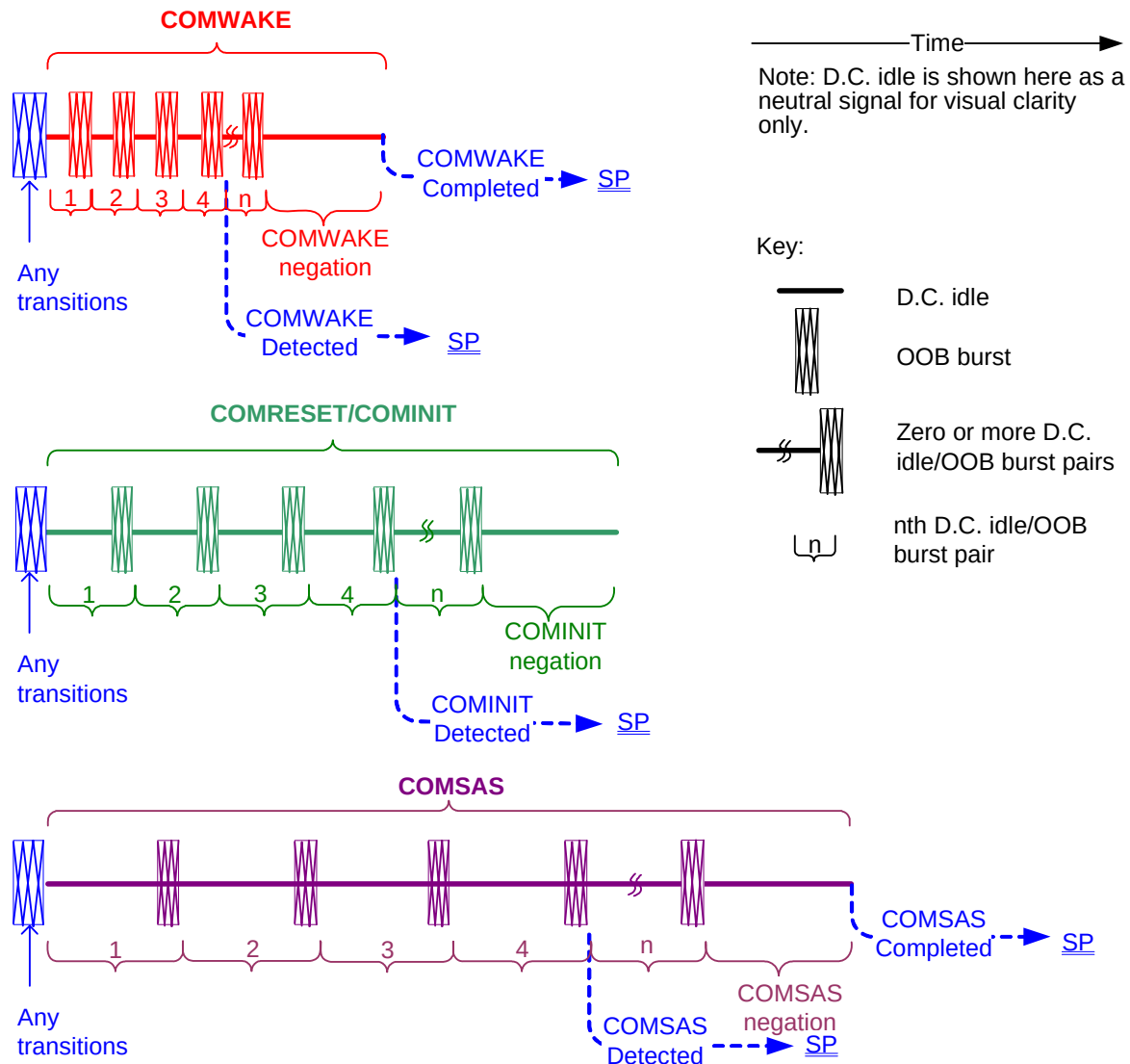


Figure 125 — OOB signal detection

Expander devices shall not forward OOB signals. An expander device shall run the link reset sequence independently on each physical link.

6.6.4 Transmitting the SATA port selection signal

The SATA port selection signal shown in figure 126 causes the attached SATA port selector to select the attached phy (one of the port selector's host phys) as the active phy (see SATA-2).

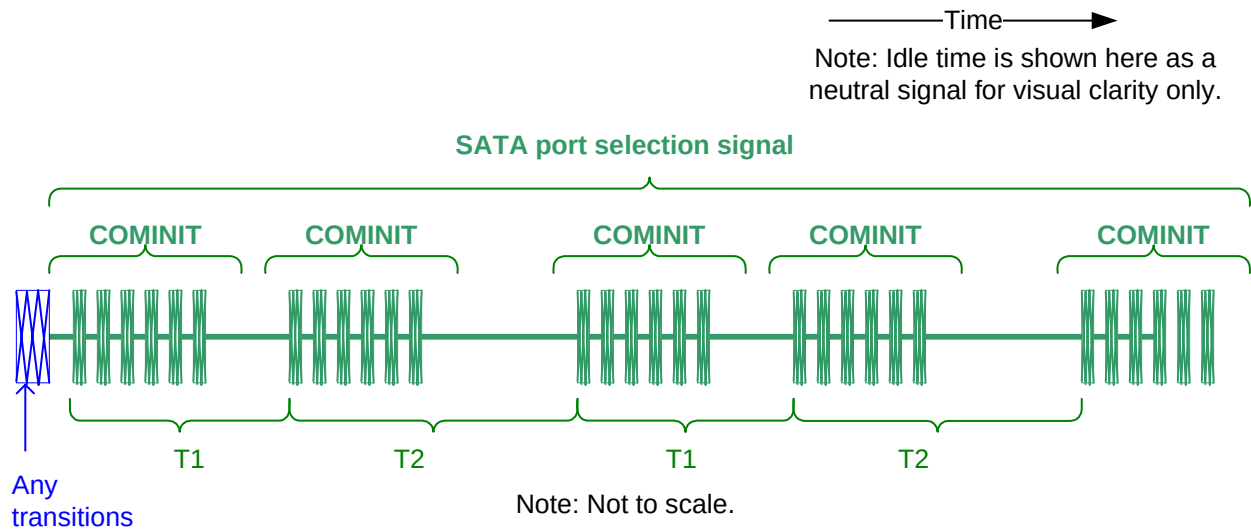


Figure 126 — SATA port selection signal

The SATA port selection signal shall be composed of 5 COMINIT signals, each starting a specified time interval, T1 or T2, as shown in figure 126, after the start of the OOB burst portion of the previous COMINIT signal. The values of T1 and T2 shall be as shown in table 84.

Table 84 — SATA port selection signal transmitter device requirements

Parameter	Time
T1	3×10^6 OOB ^a
T2	12×10^6 OOB ^b
^a 3×10^6 OOB ^a is nominally 2 ms (see table 79). ^b 12×10^6 OOB ^b is nominally 8 ms.	

See 6.8.6 and 10.4.3.27 for information on usage of the SATA port selection signal.

6.7 Phy reset sequences

6.7.1 Phy reset sequences overview

The phy reset sequence consists of:

- 1) an OOB sequence (see 6.7.2.1 and 6.7.4.1);
- 2) a speed negotiation sequence (see 6.7.2.2 and 6.7.4.2); and
- 3) if the physical link is a SAS physical link and multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), a multiplexing sequence (see 6.7.4.3).

The phy reset sequence shall only affect the phy, not the port or device containing the phy or other phys in the same port or device.

A phy shall originate a phy reset sequence after:

- a) power on;
- b) hard reset (i.e., receiving a HARD_RESET primitive sequence before an IDENTIFY address frame) (see 4.4.2);

- c) management application layer request (see 6.8.1);
- d) losing dword synchronization and not attempting to re-acquire dword synchronization (see 6.8.4.9 and 6.8.5.8);
- e) Receive Identify Timeout timer expires (see 7.9.4); and
- f) for expander phys, after a hot-plug timeout (see 6.7.5).

A SAS phy may originate a phy reset sequence after a hot-plug timeout (see 6.7.5).

After receiving a HARD_RESET primitive sequence before an IDENTIFY address frame, a phy should start the phy reset sequence within 250 ms.

Table 85 defines phy reset sequence timing parameters used by the SP state machine (see 6.8).

Table 85 — Phy reset sequence timing specifications

Parameter	Minimum	Maximum	Comments
Hot-plug timeout	10 ms	500 ms	The time after which an expander phy shall retry an unsuccessful phy reset sequence, and after which a SAS initiator phy should retry an unsuccessful phy reset sequence (see 6.7.5).

6.7.2 SATA phy reset sequence

6.7.2.1 SATA OOB sequence

Figure 127 shows the SATA OOB sequence between a SATA host and SATA device. The SATA OOB sequence is defined by SATA (see SATA-2 for detailed requirements).

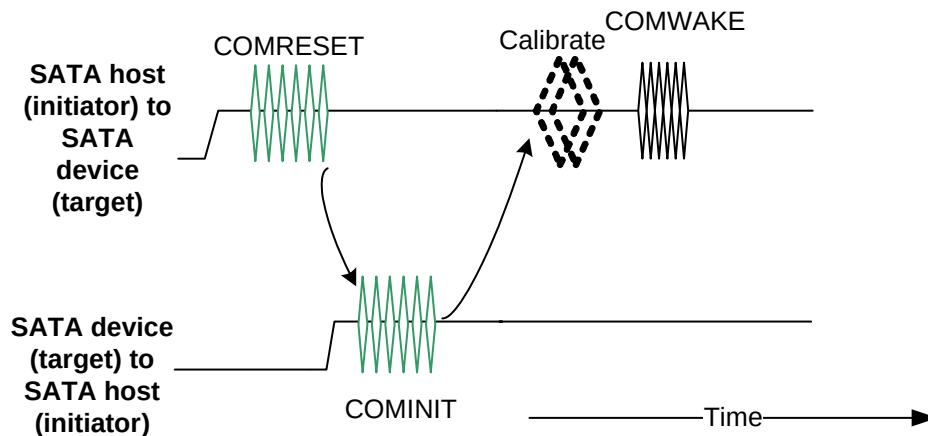


Figure 127 — SATA OOB sequence

6.7.2.2 SATA speed negotiation sequence

Figure 128 shows the speed negotiation sequence between a SATA host and SATA device. The SATA speed negotiation sequence is defined by SATA; see SATA-2 for detailed requirements.

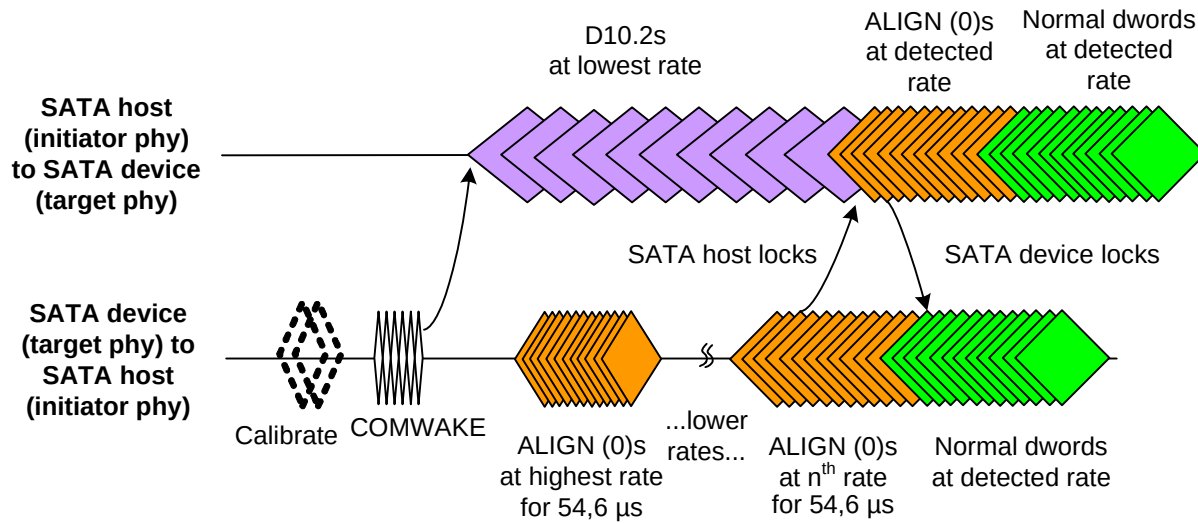


Figure 128 — SATA speed negotiation sequence

Table 86 defines SATA speed negotiation sequence timing parameters used by the SP state machine (see 6.8).

Table 86 — SATA speed negotiation sequence timing specifications

Parameter	Time	Comments
Await ALIGN timeout	1 310 720 OOB ^a	The minimum time during SATA speed negotiation that a phy shall allow for an ALIGN (0) to be received after detecting COMWAKE Completed. Derived from: $32\,768 \times 40$ (see SATA-2).
COMWAKE response time	533 ns	The maximum time during SATA speed negotiation after detecting COMWAKE Completed before which a phy shall start transmitting D10.2 characters.
^a 1 310 720 OOB ^a is nominally 873,813 μs (see table 79).		

Editor's Note 24: Should Await ALIGN Timeout be based on OOB^a use the minimum, nominal, or maximum OOB^a value? Await ALIGN Timeout would range from min=871.717 microseconds, nom=873.813 microseconds, max=875.911 microseconds

The transmitter device shall use SATA signal output levels during the SATA speed negotiation sequence as described in 5.3.6.5.

6.7.3 SAS to SATA phy reset sequence

SAS initiator phys and expander phys may support SATA (e.g., support being directly attached to a SATA device or a SATA port selector).

To initiate a phy reset sequence a phy shall:

- 1) transmit a COMINIT; and

- 2) in response to receiving a COMINIT, transmit a COMSAS.

The COMSAS identifies the phy as a SAS phy or expander phy instead of a SATA phy.

If a SATA phy is attached to the physical link, it either:

- misinterprets the COMSAS to be a COMRESET and responds with a COMINIT; or
- ignores the COMSAS and provides no response within a COMSAS detect timeout.

Either response indicates to the phy that a SATA phy is attached. As a result, the phy shall transmit COMWAKE and enter the SATA speed negotiation sequence (see 6.7.2.2).

Figure 129 shows a reset sequence between a SAS phy or expander phy (i.e., a phy compliant with this standard) and a SATA phy (i.e., a phy in a SATA device, defined by SATA). The two possible cases are presented. The first case is that the SATA phy ignores the COMSAS and provides no response within a COMSAS detect timeout. The second case is that the SATA phy misinterprets the COMSAS to be a COMRESET and responds with a COMINIT. The SP state machine treats these two cases the same, and determines that a SATA phy is attached after a COMSAS detect timeout. The SATA speed negotiation sequence shall be entered after COMWAKE.

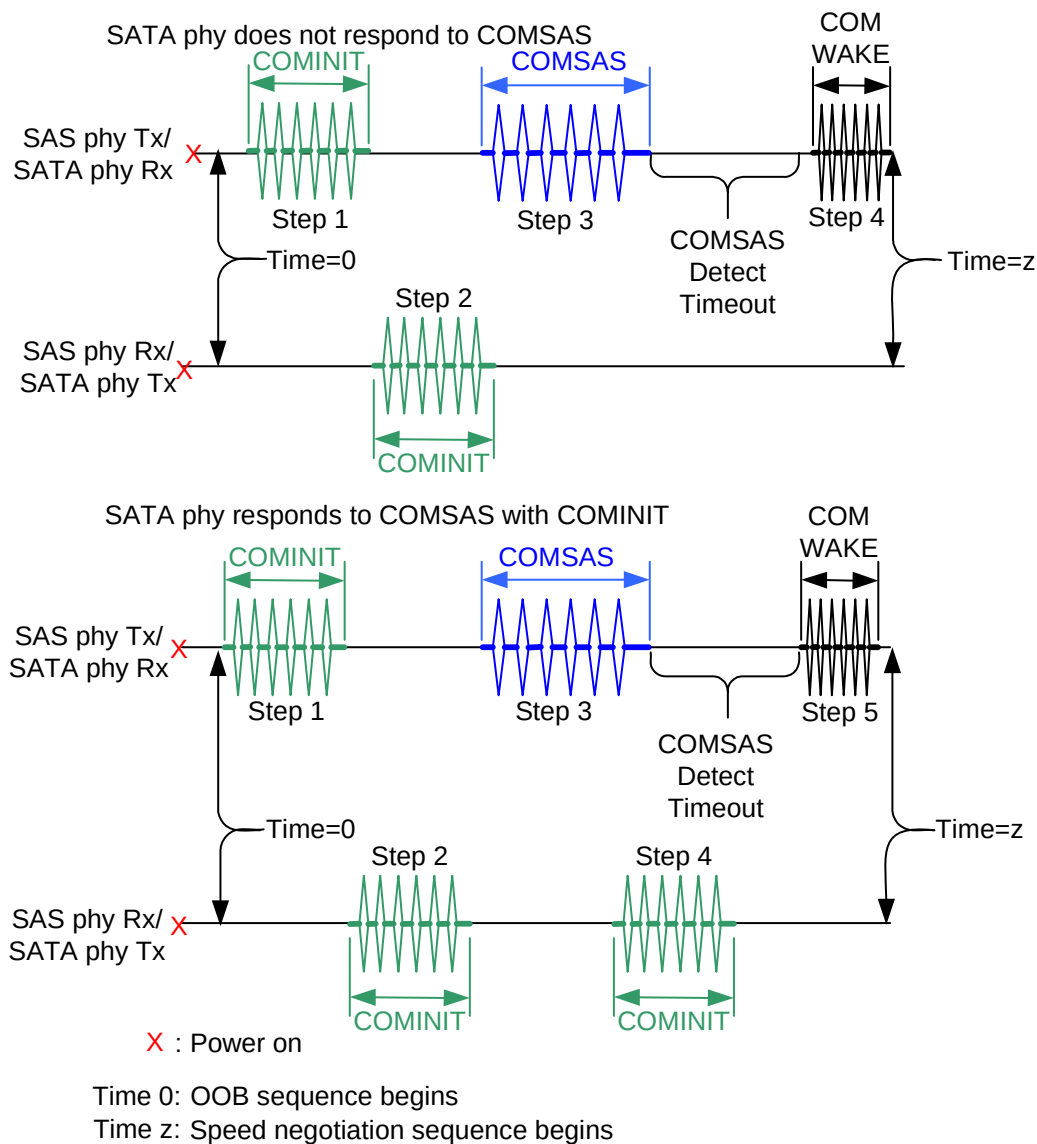


Figure 129 — SAS to SATA OOB sequence

6.7.4 SAS to SAS phy reset sequence

6.7.4.1 SAS OOB sequence

To initiate a SAS OOB sequence a phy shall transmit a COMINIT.

On receipt of a COMINIT a phy shall either:

- a) if the receiving phy has not yet transmitted a COMINIT, transmit a COMINIT followed by a COMSAS;
or
- b) if the receiving phy has transmitted a COMINIT, transmit a COMSAS.

NOTE 29 - If the receiving phy does not respond to a COMINIT within the minimum hot-plug timeout (see 6.7.5), the other phy may transmit another COMINIT. If repeated, this results in a livelock.

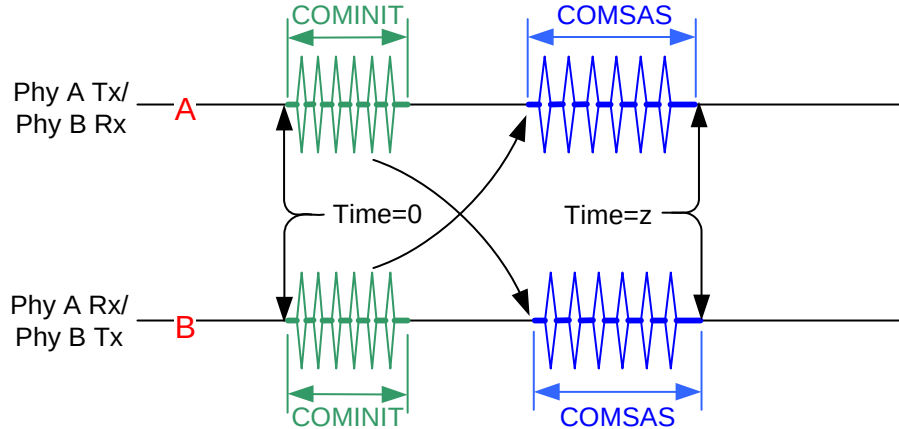
On receipt of a COMSAS, if the receiving phy has not yet transmitted a COMSAS, the phy shall transmit a COMSAS.

After completing the transmission of a COMSAS and the successful receipt of a COMSAS the SAS OOB sequence is complete and the SAS speed negotiation sequence begins.

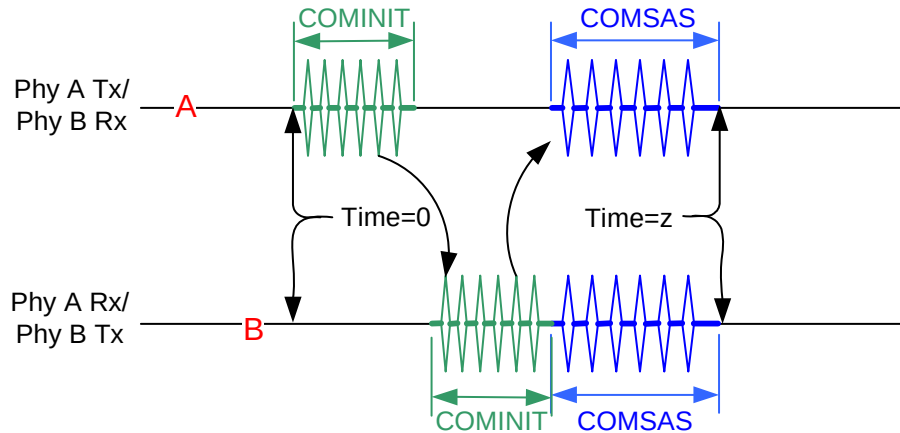
A phy shall distinguish between COMINIT and COMSAS and continue with a SAS speed negotiation sequence (see 6.7.4.2) after completing the SAS OOB sequence.

Figure 130 shows several different SAS OOB sequences between phy A and phy B, with phy A starting the SAS OOB sequence at the same time as phy B, before phy B, and before phy B powers on.

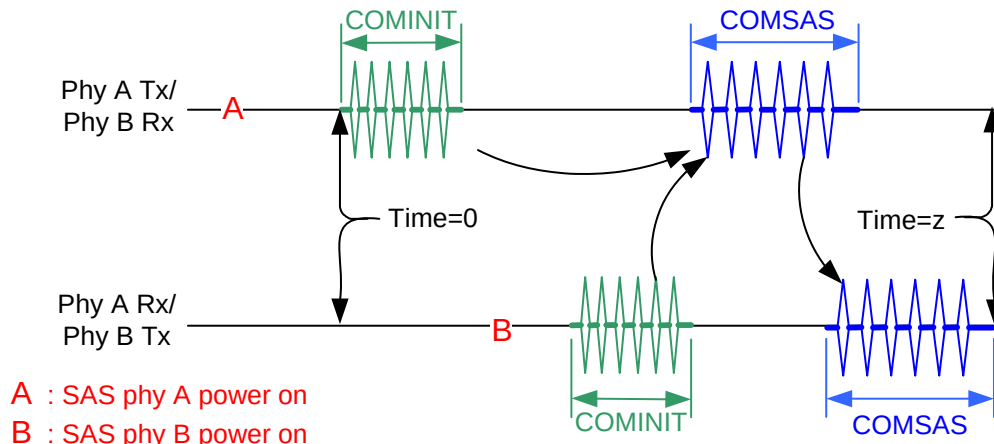
Scenario 1: Both SAS phys start SAS OOB sequence at same time



Scenario 2: SAS phy A starts SAS OOB sequence



Scenario 3: SAS phy B misses SAS phy A's COMINIT



Time 0: SAS phy reset sequence begins

Time z: SAS speed negotiation sequence begins

Figure 130 — SAS to SAS OOB sequence

6.7.4.2 SAS speed negotiation sequence

6.7.4.2.1 SAS speed negotiation sequence overview

The SAS speed negotiation sequence establishes communications between the two phys of a physical link at the highest possible transmission rate.

The SAS speed negotiation sequence is a peer-to-peer negotiation technique that does not assume initiator and target (i.e., host and device) roles. The rules for speed negotiation are the same for both participating phys.

The SAS speed negotiation sequence consists of a set of speed negotiation windows (SNWs). Each SNW is identified by a name (e.g., Speed Negotiation Window-1 or SNW-1).

SNWs conform to one of three defined types:

- a) speed negotiation without training: SNW-1, SNW-2 and Final-SNW (see 6.7.4.2.3.2);
- b) phy capabilities exchange: SNW-3 (see 6.7.4.2.3.3); and
- c) speed negotiation with training: Train-SNW (see 6.7.4.2.3.4).

Many of the timing parameters used for defining the SNWs are common to multiple SNW types. All of the timing specifications for all SNW types are defined in 6.7.4.2.2.

A SAS speed negotiation sequence may or may not include all three types of SNWs. Phys may implement a subset of SNWs provided that the subset implements a valid speed negotiation sequence. SAS speed negotiation sequences are defined in 6.7.4.2.4.

The transmitter device shall use SAS signal output levels during the SAS speed negotiation sequence as described in 5.3.6.5.

Editor's Note 25: when 6 Gbps is added, make sure to cross reference the appropriate section

6.7.4.2.2 SAS speed negotiation sequence timing specifications

Table 87 defines the timing specifications for the SAS speed negotiation sequence.

Table 87 — SAS speed negotiation sequence timing specifications

Parameter	Acronym	Time ^a	Comments
Rate change delay time	RCDT	750 000 OOB _I ^b	The time the transmitter device shall transmit D.C. idle at the beginning of SNW-1, SNW-2, SNW-3, Final-SNW, and Train-SNW.
Speed negotiation transmit time	SNTT	163 840 OOB _I ^c	During SNW-1, SNW-2, and Final-SNW, the time after RCDT during which ALIGN (0) or ALIGN (1) is transmitted. During SNW-3, the time after RCDT in which bit cells and D.C. idle are transmitted.
Speed negotiation lock time	SNLT	153 600 OOB _I ^d	The maximum time for a phy to reply with ALIGN (1) during SNW-1, SNW-2, and Final-SNW.
Actual lock time	ALT		The time during SNW-1, SNW-2, and Final-SNW at which actual dword synchronization occurs to the received ALIGN (0) or ALIGN (1) and the phy begins transmitting ALIGN (1) rather than ALIGN (0).
SNW time	SNWT	913 840 OOB _I ^e	The duration of SNW-1, SNW-2, SNW-3, or Final-SNW.
Bit cell time	BCT	2 200 OOB _I ⁱ	The time to transmit a COMWAKE or D.C. idle during SNW-3.
Maximum training time	MTT	29 998 080 OOB _I ^f	The maximum time for training to complete during Train-SNW.
Training lock time	TLT	28 497 920 OOB _I ^g	The maximum time for a phy to reply with TRAIN_DONE during Train-SNW.
Actual training time	ATT		The time at which training of the receiver is complete.
Train-SNW time	TWT		The actual duration of Train-SNW.
Maximum Train-SNW time	MTWT	30 748 080 OOB _I ^h	The maximum duration of Train-SNW.

^a OOB_I is defined in table 79 (see 6.6.1).
^b 750 000 OOB_I (e.g., RCDT) is nominally 500 μ s. Equal to: $18\,750 \times 40$.
^c 163 840 OOB_I (e.g., SNTT) is nominally 109,226 μ s. Equal to: $4\,096 \times 40$.
^d 153 600 OOB_I (e.g., SNLT) is nominally 102,4 μ s. Equal to: $(4\,096 - 256) \times 40$.
^e 913 840 OOB_I (e.g., SNWT) is nominally 609,226 μ s. Equal to: RCDT + SNTT.
^f 29 998 080 OOB_I (e.g., MTT) is nominally 19,998 719 ms. Equal to: $11\,718 \times 64 \times 40$. This is the time of the maximum number of complete training patterns that fit into 20 ms.
^g 28 497 920 OOB_I (e.g., TLT) is nominally 18,998 613 ms. Equal to: $11\,132 \times 64 \times 40$. This is the time of the maximum number of complete training patterns that fit into 19 ms.
^h 30 748 080 OOB_I (e.g., MTWT) is nominally 20,498 719 ms. Equal to: RCDT + MTT.
ⁱ 2 200 OOB_I is nominally 1,466.6 μ s. Equal to the COMWAKE signal time (see table 80 in 6.6.2).

Editor's Note 26: When these times are used by a receiver, should it use the minimum, nominal, or

maximum value of OOB? Since SNW-1, SNW-2, and Final-SNW are required to not use SSC, should the values used by those SNWs use UI(G1) instead?

6.7.4.2.3 Speed negotiation window (SNW) definitions

6.7.4.2.3.1 SNW definitions overview

During each SNW, a phy shall either:

- if it supports the SNW, transmit and receive as defined for the SNW; or
- if it does not support the SNW, transmit D.C. idle and not receive.

If a phy supports the SNW and receives the expected transmission, then the SNW is valid. If a phy does not receive the expected transmission from the attached phy, then the SNW is invalid.

NOTE 30 - If a phy transmits D.C. idle during a SNW, then the attached phy does not receive the expected transmission and the SNW is invalid.

6.7.4.2.3.2 SNW-1, SNW-2, and Final-SNW

Figure 131 defines SNW-1, SNW-2, and Final-SNW, including:

- SNW time (SNWT);
- rate change delay time (RCDT);
- speed negotiation transmit time (SNTT);
- speed negotiation lock time (SNLT); and
- actual lock time (ALT).

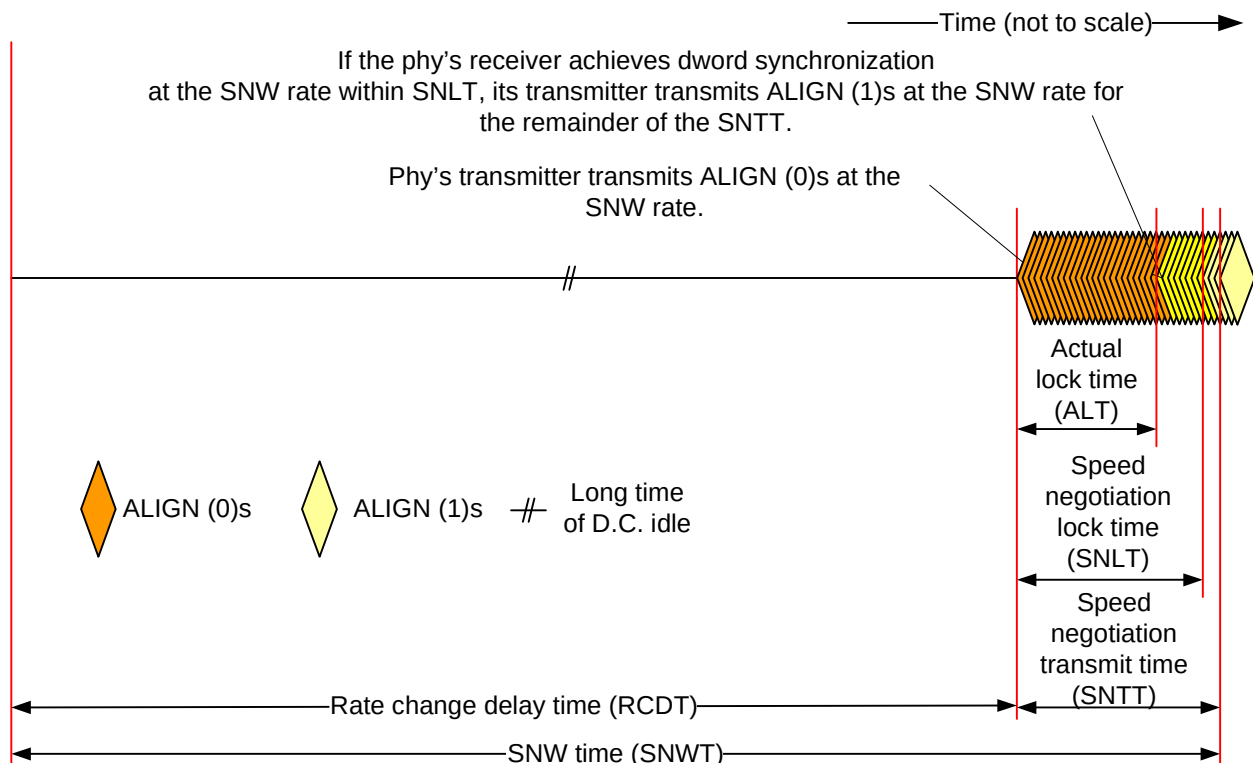


Figure 131 — SNW-1, SNW-2, and Final-SNW

If the phy supports the SNW, it shall transmit:

- D.C. idle for an RCDT; and
- ALIGNs at the SNW rate for the remainder of the SNWT (i.e., for SNTT).

If the phy does not support the SNW, it shall transmit D.C. idle for the entire SNWT.

Table 88 defines the SNW rate used in SNW-1, SNW-2, and Final-SNW.

Table 88 — SNW rates used in SNW-1, SNW-2, and Final-SNW

SNW	SNW rate
SNW-1	1,5 Gbps
SNW-2	3 Gbps
Final-SNW	Based on SNW-1, SNW-2, and SNW-3 validity: a) 1,5 Gbps if SNW-1 is valid and SNW-2 is invalid; and b) 3 Gbps if SNW-2 is valid and SNW-3 is invalid.

If the phy supports the SNW, it shall attempt to synchronize on an incoming series of dwords at that rate for the SNLT after RCDT. The received dwords may be ALIGN (0) or ALIGN (1) primitives. If the phy achieves dword synchronization within the SNLT, it shall change from transmitting ALIGN (0) primitives to transmitting ALIGN (1) primitives for the remainder of the SNTT (i.e., the remainder of the SNW time). The point at which the phy achieves dword synchronization is called the actual lock time (ALT). If the phy does not achieve dword synchronization within the SNLT, it shall continue transmitting ALIGN (0) primitives for the remainder of the SNTT (i.e., the remainder of the SNW time).

At the end of the SNTT, if a phy is both transmitting and receiving ALIGN (1) primitives, it shall consider the SNW to be valid. If the phy is not both transmitting and receiving ALIGN (1) primitives, it shall consider the SNW to be invalid.

The phy shall disable SSC (see 5.3.8) during SNW-1, SNW-2, and Final-SNW.

6.7.4.2.3.3 SNW-3

SNW-3 allows the phys to exchange phy capabilities to establish phy parameters used in Train-SNW.

Figure 132 defines SNW-3, including:

- a) SNW time (SNWT);
- b) rate change delay time (RCDT); and

c) speed negotiation transmit time (SNTT).

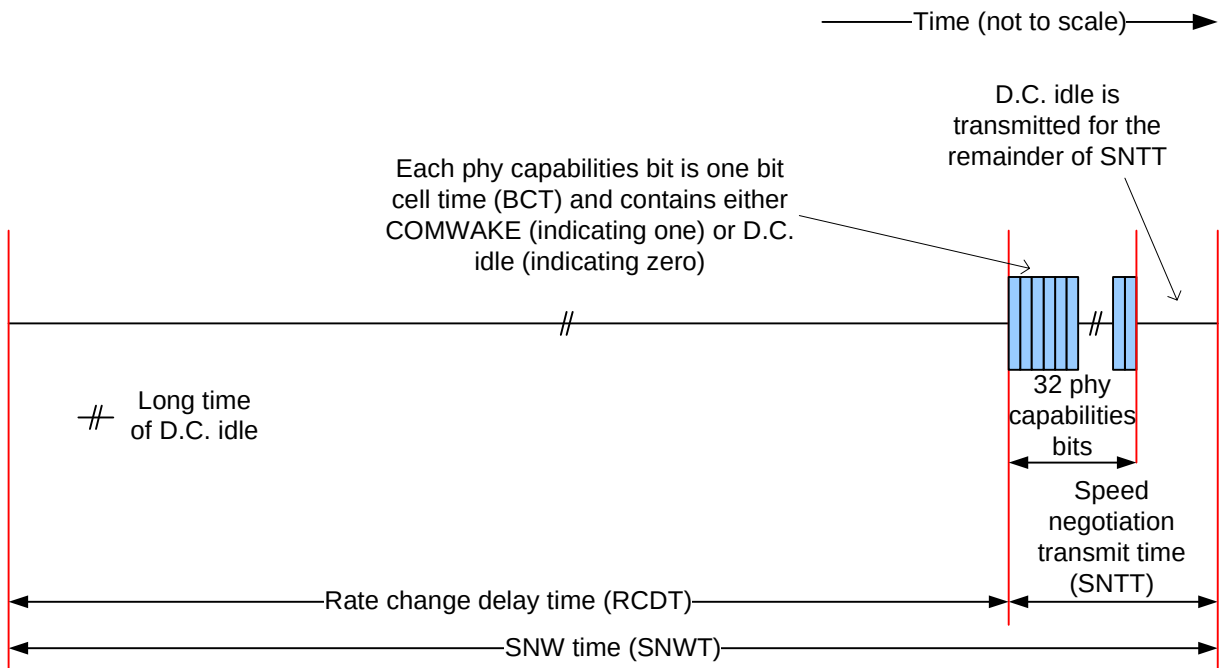


Figure 132 — SNW-3

Table 89 defines the content of each phy capabilities bit.

Table 89 — SNW-3 phy capabilities bit

Value	Transmitted
One	COMWAKE (see 6.6)
Zero	D.C. idle

If the phy supports SNW-3, it shall:

- 1) transmit D.C. idle for an RCDT;
- 2) transmit 32 phy capabilities bits; and
- 3) transmit D.C. idle for the remainder of SNTT,

and shall receive a 32-bit phy capabilities value from the attached phy. If the attached phy does not support SNW-3, the phy capabilities bits are all set to zero (i.e., D.C. idle).

If the phy does not support SNW-3, the phy shall transmit D.C. idle for the entire SNWT and ignore any SNW-3 phy capabilities bits received.

The first phy capabilities bit is the START bit and is set to one. Each of the remaining 31 phy capabilities bits is set to one or zero. The receiver shall use the START bit to detect the beginning of the phy capabilities bits.

The phy shall consider SNW-3 to be valid if it supports SNW-3 and receive at least one phy capabilities bit set to one. If the phy does not support SNW-3 or does not receive at least one phy capabilities bit set to one, it shall consider SNW-3 to be invalid.

The phy may transmit with SSC enabled or disabled (see 5.3.8) during SNW-3.

Table 90 defines the SNW-3 phy capabilities. For each bit defined as reserved, the phy shall transmit a zero (i.e., D.C. idle) and shall ignore the received value. Byte 0 shall be transmitted first and byte 3 shall be

transmitted last. Within each byte, bit 7 shall be transmitted first and bit 0 shall be transmitted last (e.g., overall, the START bit is transmitted first and the PARITY bit is transmitted last).

Table 90 — SNW-3 phy capabilities

Byte\Bit	7	6	5	4	3	2	1	0
0	START (1b)	TX SSC TYPE	Reserved		REQUESTED LOGICAL LINK RATE			
1	Supported settings						Reserved	
	G1 WITHOUT SSC	G1 WITH SSC	G2 WITHOUT SSC	G2 WITH SSC	G3 WITHOUT SSC	G3 WITH SSC		
2	Reserved							
3	Reserved							PARITY

The START bit shall be set to one. The phy's receiver shall use this bit to establish the timing for the subsequent bits.

A TX SSC TYPE bit set to one indicates that the phy's transmitter uses center-spreading SSC when SSC is enabled (e.g., the phy is an expander phy)(see 5.3.8). A TX SSC TYPE bit set to zero indicates that the phy's transmitter uses down-spreading SSC when SSC is enabled (e.g., the phy is a SAS phy), or that the phy does not support SSC.

NOTE 31 - The phy receiver may use the TX SSC TYPE bit to optimize its CDR circuitry.

The REQUESTED LOGICAL LINK RATE field indicates if the phy supports multiplexing (see 6.10) and, if so, the logical link rate that the phy is requesting. If the phy is managed by an SMP target port, the field is based on the REQUESTED LOGICAL LINK RATE field in the SMP PHY CONTROL function (see 10.4.3.27).

Table 91 defines the requested logical link rate based on the transmitted and received REQUESTED LOGICAL LINK RATE fields.

Table 91 — Requested logical link rate

Transmitted REQUESTED LOGICAL LINK RATE field	Received REQUESTED LOGICAL LINK RATE field	Requested logical link rate
0h (i.e., no multiplexing)	Any	Negotiated physical link rate
8h (i.e., 1,5 Gbps)	8h (i.e., 1,5 Gbps)	1,5 Gbps
	9h (i.e., 3 Gbps)	
	Ah (i.e., 6 Gbps)	
	Bh - Fh (i.e., future rates)	
9h (i.e., 3 Gbps)	8h (i.e., 1,5 Gbps)	1,5 Gbps
	9h (i.e., 3 Gbps)	3 Gbps
	Ah (i.e., 6 Gbps)	
	Bh - Fh (i.e., future rates)	
Ah (i.e., 6 Gbps)	8h (i.e., 1,5 Gbps)	1,5 Gbps
	9h (i.e., 3 Gbps)	3 Gbps
	Ah (i.e., 6 Gbps)	6 Gbps
	Bh - Fh (i.e., future rates)	

Table 92 defines whether or not multiplexing is enabled and defines the negotiated logical link rate based on the requested logical link rate (see table 91) and the negotiated physical link rate (see 6.7.4.2.4).

Table 92 — Multiplexing negotiation

Requested logical link rate (see table 91)	Negotiated physical link rate	Multiplexing	Negotiated logical link rate
1,5 Gbps	1,5 Gbps	Disabled	1,5 Gbps
	3 Gbps	Enabled	1,5 Gbps
	6 Gbps		3 Gbps
3 Gbps	1,5 Gbps	Disabled	1,5 Gbps
	3 Gbps		3 Gbps
	6 Gbps	Enabled	3 Gbps
6 Gbps	1,5 Gbps	Disabled	1,5 Gbps
	3 Gbps		3 Gbps
	6 Gbps		6 Gbps
Negotiated physical link rate	1,5 Gbps	Disabled	1,5 Gbps
	3 Gbps		3 Gbps
	6 Gbps		6 Gbps

The supported settings bits include the G1 WITHOUT SSC bit, the G1 WITH SSC bit, the G2 WITHOUT SSC bit, the G2 WITH SSC bit, the G3 WITHOUT SSC bit, and the G3 WITH SSC bit.

A G1 WITHOUT SSC bit set to one indicates that the phy supports G1 (i.e., 1,5 Gbps) without SSC. A G1 WITHOUT SSC bit set to zero indicates that the phy does not support G1 without SSC. If the phy supports SNW-1 and supports SNW-3, then the G1 WITHOUT SSC bit shall be set to one.

A G1 WITH SSC bit set to one indicates that the phy supports G1 (i.e., 1,5 Gbps) with SSC. A G1 WITH SSC bit set to zero indicates that the phy does not support G1 with SSC.

A G2 WITHOUT SSC bit set to one indicates that the phy supports G2 (i.e., 3 Gbps) without SSC. A G2 WITHOUT SSC bit set to zero indicates that the phy does not support G2 without SSC. If the phy supports SNW-2 and supports SNW-3, then the G2 WITHOUT SSC bit shall be set to one.

A G2 WITH SSC bit set to one indicates that the phy supports G2 (i.e., 3 Gbps) with SSC. A G2 WITH SSC bit set to zero indicates that the phy does not support G2 with SSC.

A G3 WITHOUT SSC bit set to one indicates that the phy supports G3 (i.e., 6 Gbps) without SSC. A G3 WITHOUT SSC bit set to zero indicates that the phy does not support G3 without SSC.

A G3 WITH SSC bit set to one indicates that the phy supports G3 (i.e., 6 Gbps) with SSC. A G3 WITH SSC bit set to zero indicates that the phy does not support G3 with SSC.

Table 93 defines the priority of the supported settings bits.

Table 93 — Supported settings bit priorities

Priority	Bit
Highest	G3 WITH SSC bit
...	G3 WITHOUT SSC bit
...	G2 WITH SSC bit
...	G2 WITHOUT SSC bit
...	G1 WITH SSC bit
Lowest	G1 WITHOUT SSC bit

If none of the transmitted supported settings bits are equal to their corresponding received supported settings bits, then the phy shall consider a phy reset problem (see 6.7.4.2.4) to have occurred.

The PARITY bit provides for error detection of all the SNW-3 phy capabilities bits. The PARITY bit shall be set to one or zero such that the total number of SNW-3 phy capabilities bits that are set to one is even, including the START bit and the PARITY bit. If the PARITY bit received is incorrect based upon the received SNW phy capabilities bits, then the parity is bad and the phy shall consider a phy reset problem (see 6.7.4.2.4) to have occurred.

Table 94 lists some example SNW-3 phy capabilities values.

Table 94 — Example SNW-3 phy capabilities values

Code ^a	Description
80540000h	Down-spreading SSC G1, G2, and G3 with SSC supported
80FC0001h	Down-spreading SSC G1, G2, and G3 with and without SSC supported
80A80000h	G1, G2, and G3 without SSC supported
C0FC0000h	Center-spreading SSC G1, G2, and G3 with and without SSC supported
C9FC0000h	Center-spreading SSC Requested 3 Gbps logical link rate G1, G2, and G3 with and without SSC supported
C8F00001h	Center-spreading SSC Requested 1,5 Gbps logical link rate G1 and G2 with and without SSC supported
^a Expressed as a 32-bit value with bit 0 (i.e., the START bit) as the MSB and bit 31 as the LSB (i.e., the PARITY bit).	

Editor's Note 27: There is interest in getting rid of G1, G2, and G3 and using 1,5 Gbps, 3 Gbps, and 6 Gbps throughout. Field/bit names should not start with numbers, though, so that would be awkward for the phy configuration bit names.

6.7.4.2.3.4 Train-SNW

Figure 133 defines the Train-SNW, including:

- a) maximum Train-SNW window time (MTWT);
- b) rate change delay time (RCDT);
- c) maximum train time (MTT);
- d) train lock time (TLT); and

e) actual training time (ATT).

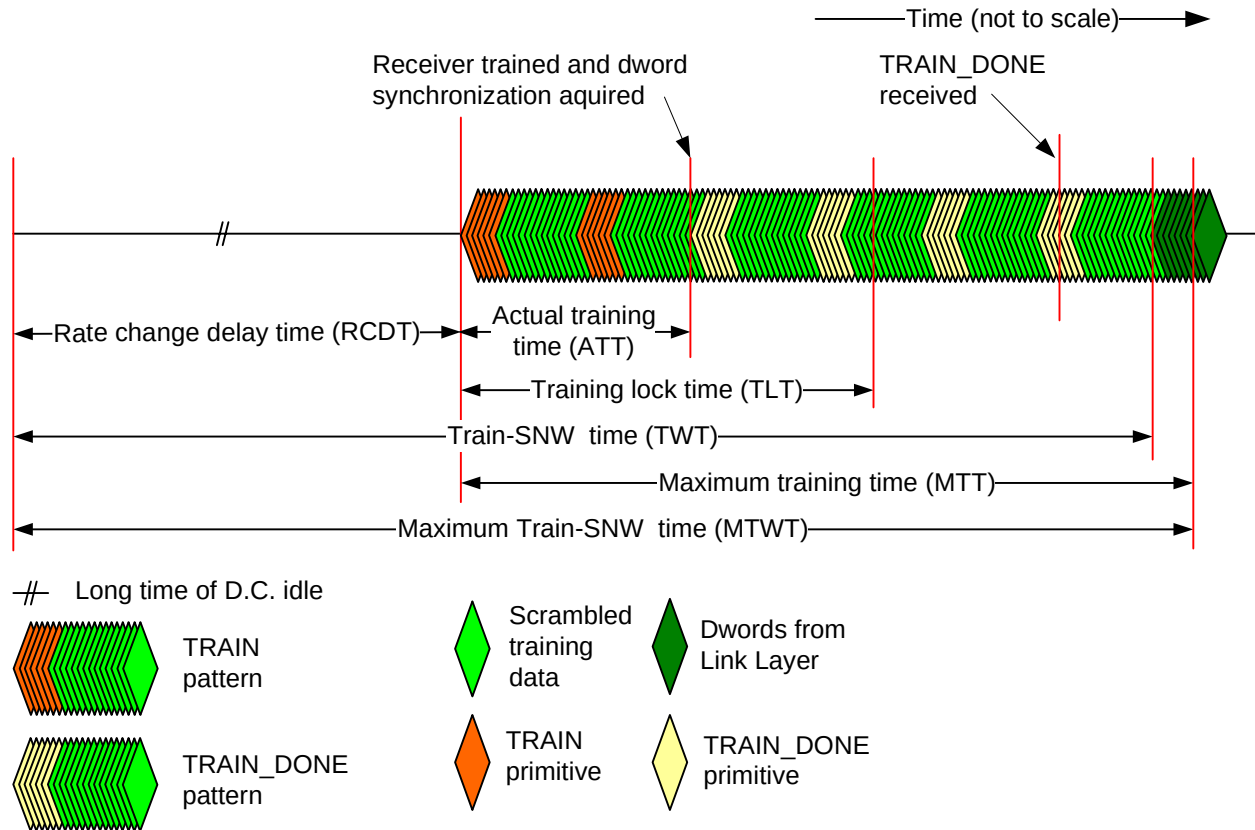


Figure 133 — Train-SNW

The Train-SNW utilizes TRAIN and TRAIN_DONE (see 7.2) to create training patterns, defined in table 95.

Table 95 — Training patterns

Training pattern	Description
TRAIN pattern	Sequence of: 1) TRAIN primitive sequence; and 2) 58 dwords set to 00000000h that are transmitted scrambled and 8b10b encoded.
TRAIN_DONE pattern	Sequence of: 1) TRAIN_DONE primitive sequence; and 2) 58 dwords set to 00000000h that are transmitted scrambled and 8b10b encoded.

The scrambler is the same as that defined for the link layer (see 7.6) and shall be initialized at the end of RCDT. The scrambler shall not be re-initialized for the remainder of the Train-SNW.

The phy shall start transmitting TRAIN patterns at the end of RCDT. The first TRAIN pattern may have either starting disparity. The number of TRAIN patterns transmitted is determined by the time required for the phy's receiver to complete training and acquire dword synchronization. The phy shall transmit at least one TRAIN pattern.

Editor's Note 28: A statement is needed here about what the receiver uses to base its TRAIN_DONE decision. Pattern comparison is a bad idea; maintaining dword synchronization for a length of time and knowledge about how long the DFE should take to reach the target BER are the intent.

If the phy's receiver is trained and acquires dword synchronization before TLT, then the phy shall stop transmitting TRAIN patterns and start transmitting TRAIN_DONE patterns. The phy shall transmit a minimum of four TRAIN_DONE patterns.

If the phy:

- a) transmits four or more TRAIN_DONE patterns; and
- b) receives a minimum of one TRAIN_DONE before MTT,

then the phy shall:

- a) stop transmitting TRAIN_DONE patterns;
- b) start transmitting dwords from the link layer; and
- c) consider the Train-SNW to be valid.

If the phy does not receive TRAIN_DONE before MTT and transmit four or more TRAIN_DONE patterns, then it shall consider the Train-SNW to be invalid.

6.7.4.2.4 SAS speed negotiation sequence

The SAS speed negotiation sequence consists of a set of SNWs (see 6.7.4.2.3) in the order shown in figure 134.

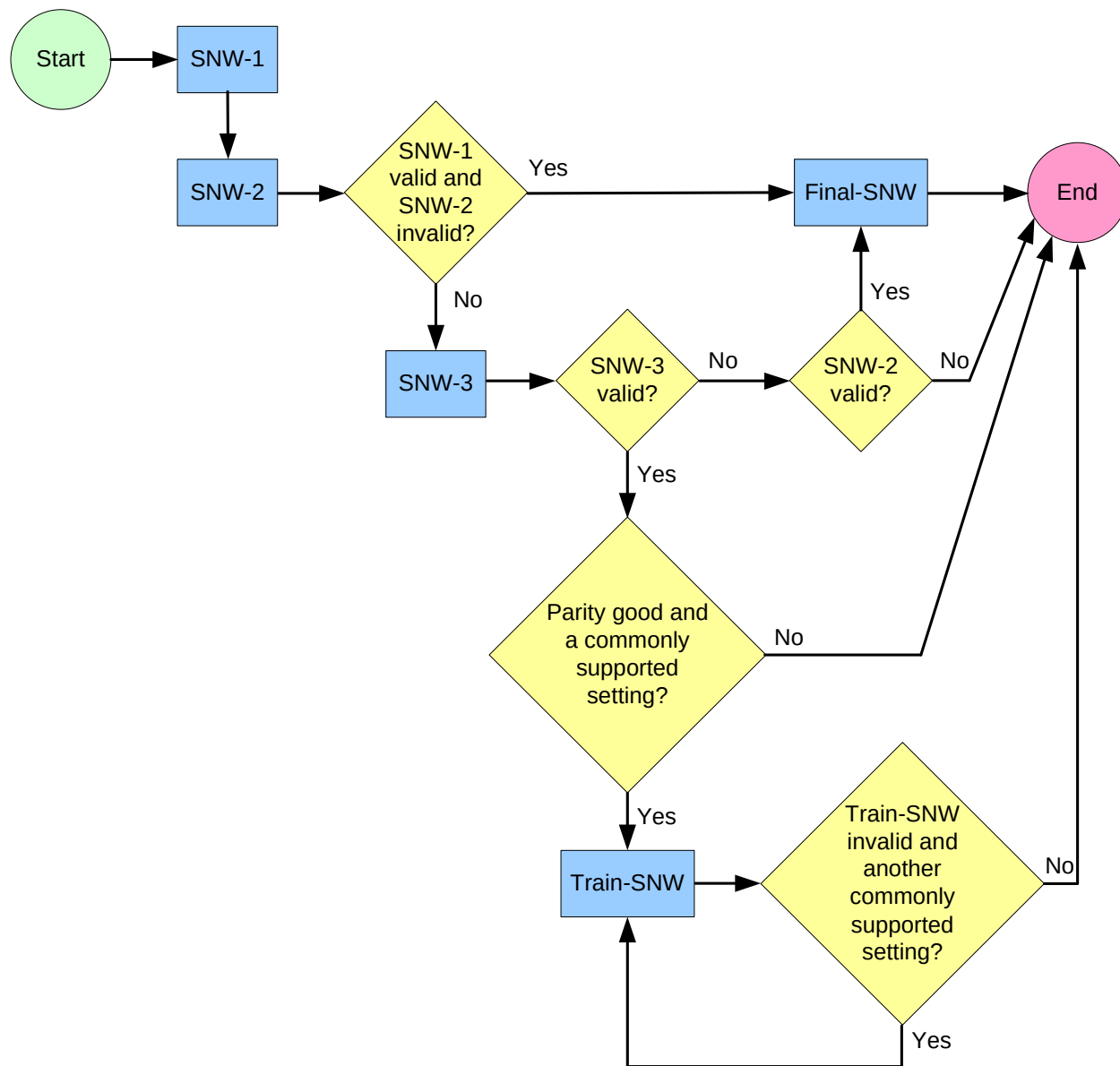


Figure 134 — SAS speed negotiation sequence SNW flowchart

Editor's Note 29: If a phy A that only supports SNW-1 is attached to a phy B that only supports SNW-3, phy A runs: OOB, SNW-1 (yes) SNW-2 (no), and back to OOB, while phy B runs: OOB SNW-1 (no), SNW-2 (no), SNW-3 (yes), and back to OOB. Since the SNW-3 involves COMWAKE, phy A interprets this as receiving a COMWAKE in response to its COMINIT, which falsely identifies a SATA port selector. Phy A's ATTACHED SATA PORT SELECTOR bit will be set incorrectly in the SMP DISCOVER response. When phy B runs SNW-1 again and sends COMINIT, that should cause phy A to realize that a SAS device is attached, not a SATA device. However, they may keep repeating this forever. In 7.11, rule d) causes a Broadcast (Change) when that bit toggles from 0 to 1, so infinite Broadcast (Changes) will result at the hot-plug timeout intervals.

Train-SNW is based on the highest untried commonly supported settings based on the outgoing and incoming SNW-3 supported settings bits (see 6.7.4.2.3).

If a Train-SNW is invalid and there are additional, untried, commonly supported settings exchanged during SNW-3, then a new Train-SNW shall be performed based on the next highest, untried, commonly supported settings.

A phy reset problem occurs if:

- a) after SNW-3, if SNW-1, SNW-2, and SNW-3 are all invalid;

Editor's Note 30: Failure to find any supported speed is not supposed to be a phy reset problem; that just means two incompatible devices are attached, which is not the same as a broken device being attached or the link having reliability problems (the original meaning). That could be counted with a new counter.

- b) after Final-SNW, if Final-SNW is invalid;
- c) after SNW-3, if SNW-3 is valid and:
 - A) the parity is bad; or
 - B) there are no commonly supported settings;
- or

Editor's Note 31: Bad parity definitely qualifies; "no commonly supported settings" should not qualify since that just means incompatible devices were attached. That could be counted with a new counter.

- d) after a Train-SNW, if the Train-SNW is invalid and there are no additional, untried, commonly supported settings.

Phy reset problems terminate the SAS speed negotiation sequence and are counted and reported in the PHY RESET PROBLEM COUNT field in the SMP REPORT PHY ERROR LOG page (see 10.4.3.10) and the Protocol-Specific Port log page (see 10.2.8.1).

Figure 135 shows speed negotiation between a phy A and phy B where phys participate in:

- 1) SNW-1, supported by phy A but not by phy B;
- 2) SNW-2, supported by both phys;
- 3) SNW-3, supported by phy A but not by phy B; and
- 4) Final-SNW negotiating 3 Gbps.

Phy A and phy B detect:

- a) SNW-1 invalid;
- b) SNW-2 valid; and
- c) SNW-3 invalid,

and proceed to Final-SNW negotiating 3 Gbps.

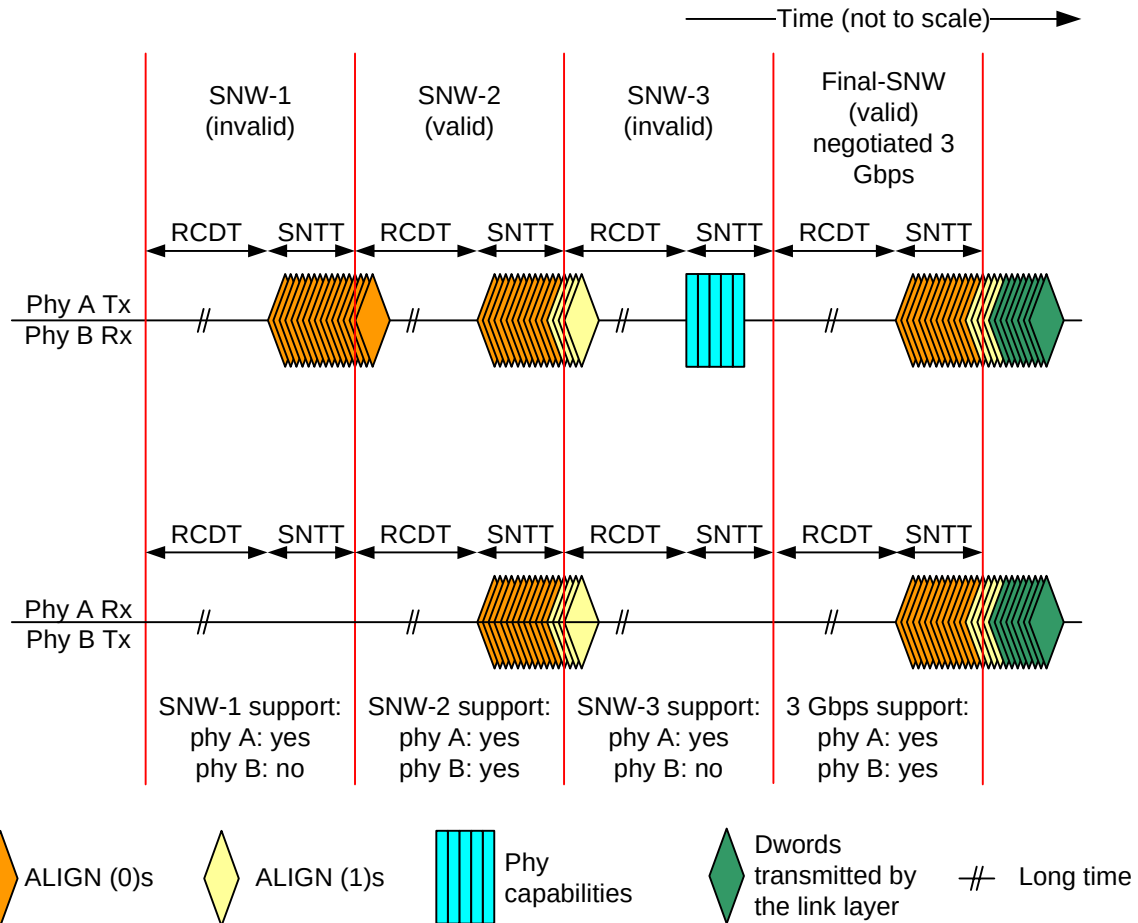


Figure 135 — SAS speed negotiation sequence (phy A: SNW-1 through SNW-3, phy B: SNW-2 only)

Figure 136 shows speed negotiation between a phy A and a phy B where both phys participate in:

- 1) SNW-1, supported by both phys;
- 2) SNW-2, supported by both phys;
- 3) SNW-3, supported by both phys; and
- 4) Train-SNW.

Phy A and phy B detect:

- a) SNW-1 valid;
- b) SNW-2 valid; and
- c) SNW-3 valid,

and proceed to Train-SNW negotiating based on SNW-3 phy capabilities bits.

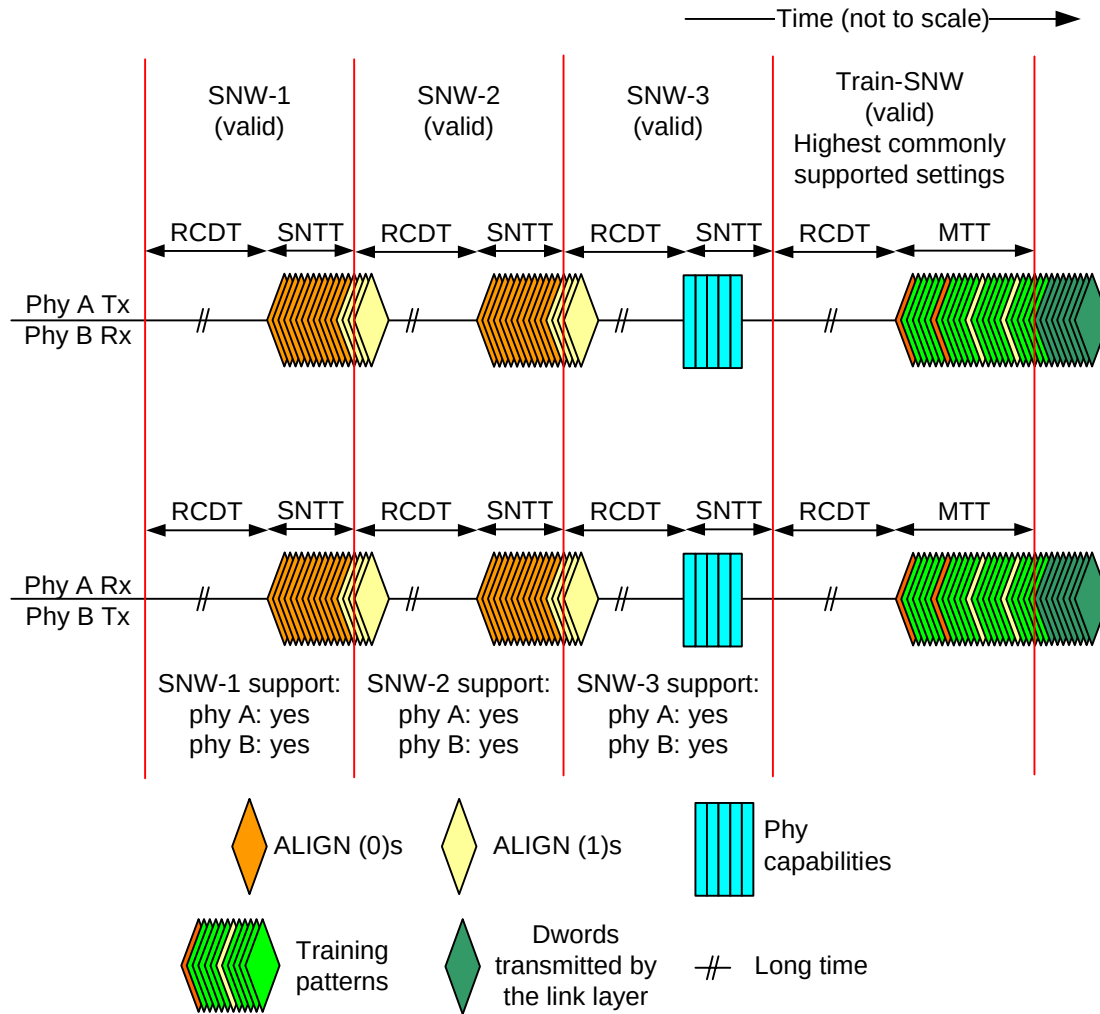


Figure 136 — SAS speed negotiation sequence (both phys SNW-1 through SNW-3)

Figure 137 shows a speed negotiation sequence where phy B does not achieve dword synchronization during Final-SNW, creating a phy reset problem. If this occurs, the handshake is not complete and the phy reset sequence is retried.

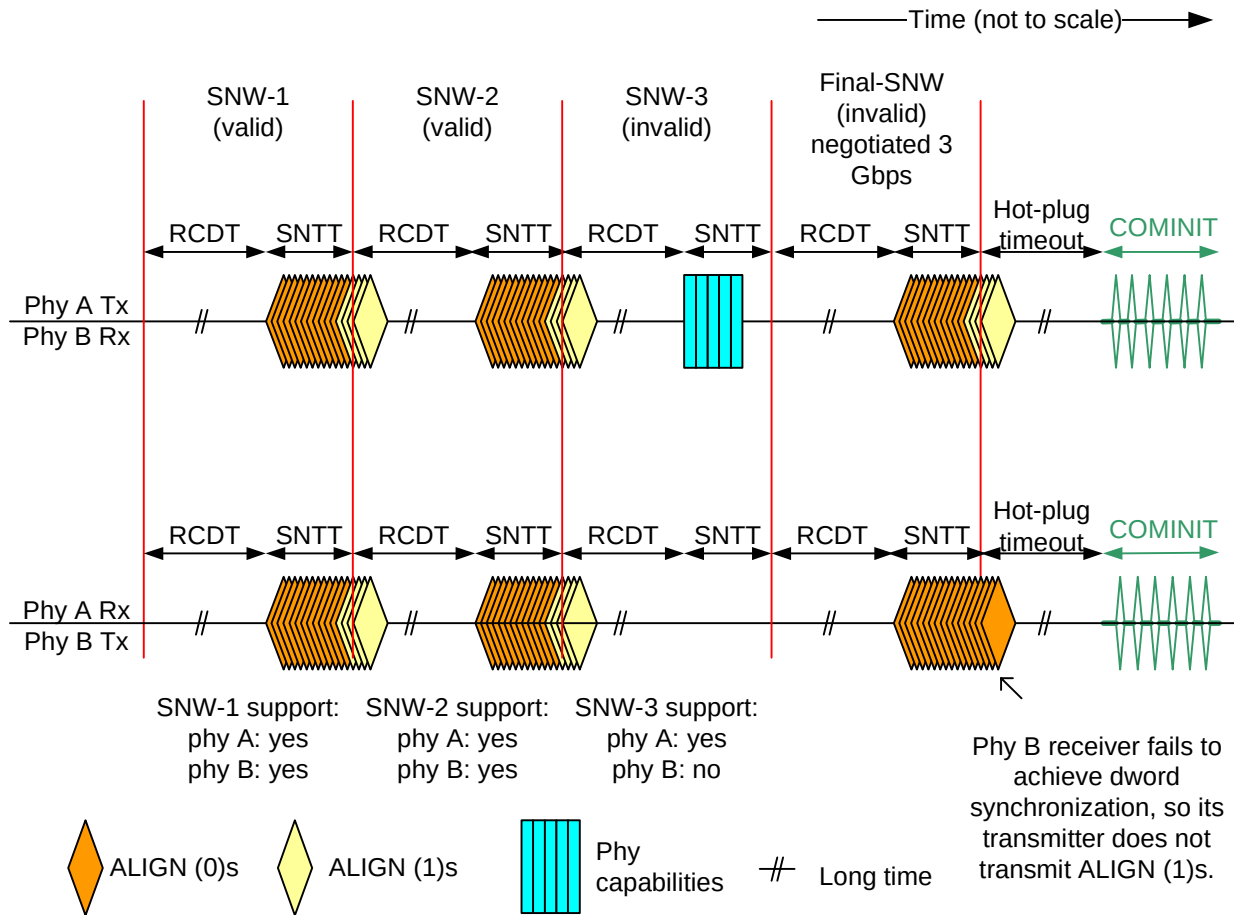


Figure 137 — SAS speed negotiation sequence - phy reset problem in Final-SNW

Figure 138 shows a speed negotiation sequence in which a phy reset problem is encountered in SNW-3 because the phys do not exchange the phy capabilities bits properly (e.g., due to a parity error).

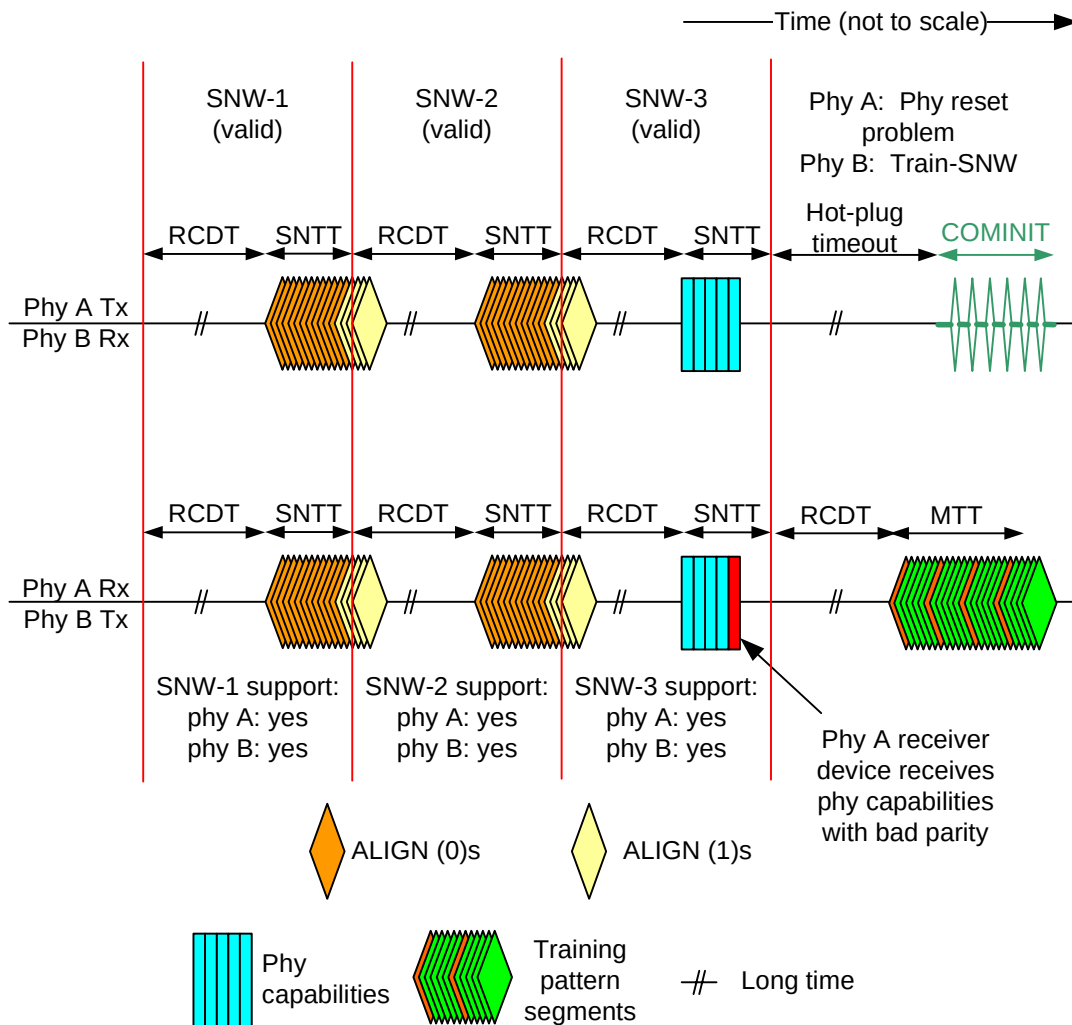


Figure 138 — SAS speed negotiation sequence - phy reset problem in SNW-3

Figure 139 shows a speed negotiation sequence in which a phy reset problem is encountered in Train-SNW because either phy does not complete training within the MTT interval of several training windows exhausting all commonly supported settings exchanged in SNW-3.

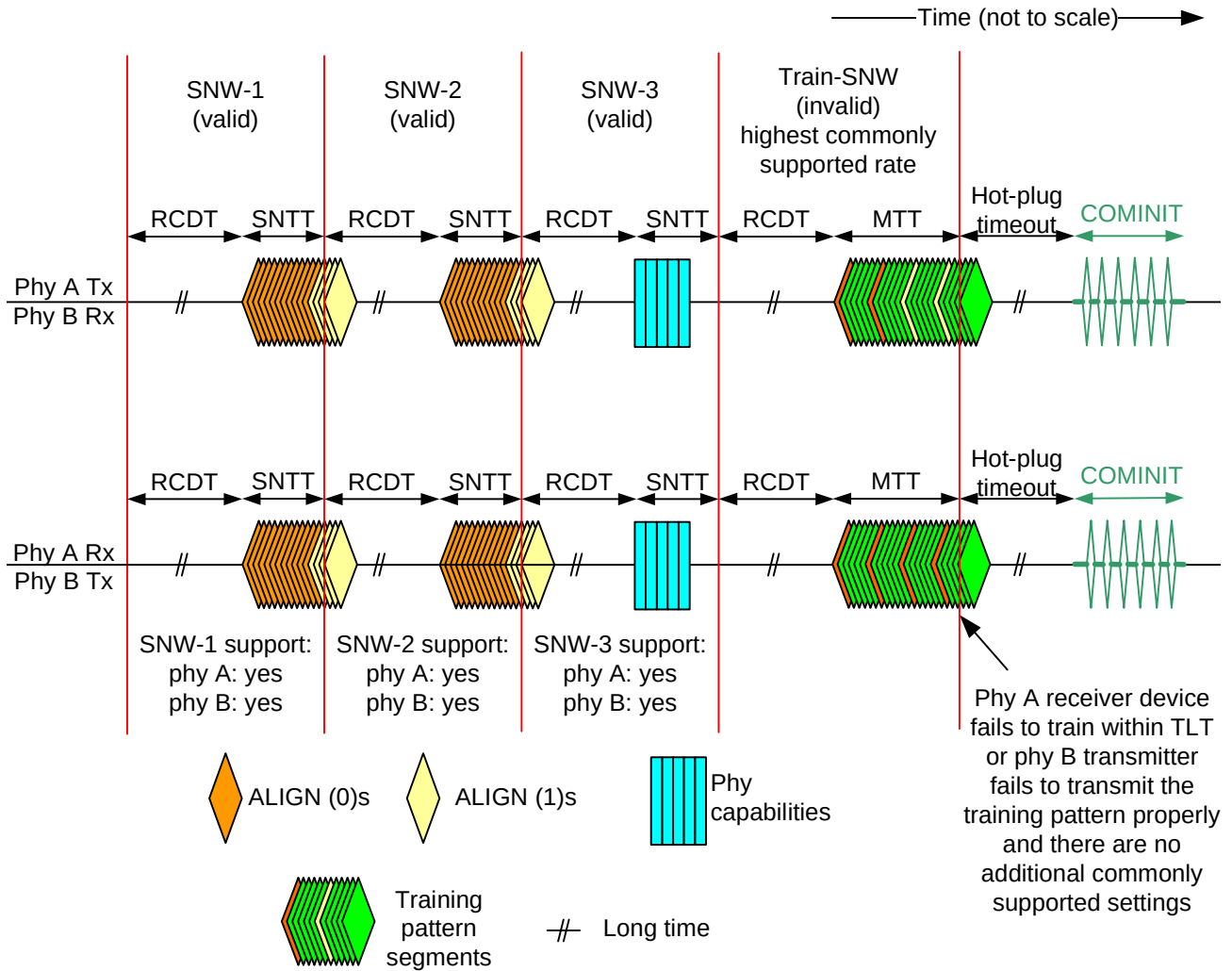


Figure 139 — SAS speed negotiation sequence - phy reset problem in Train-SNW

Figure 140 shows two Train-SNWs, if supported settings bits are exchanged that contain more than one commonly supported setting and the Train-SNW using the highest commonly supported setting is invalid, a second Train-SNW is performed using the next highest commonly supported setting.

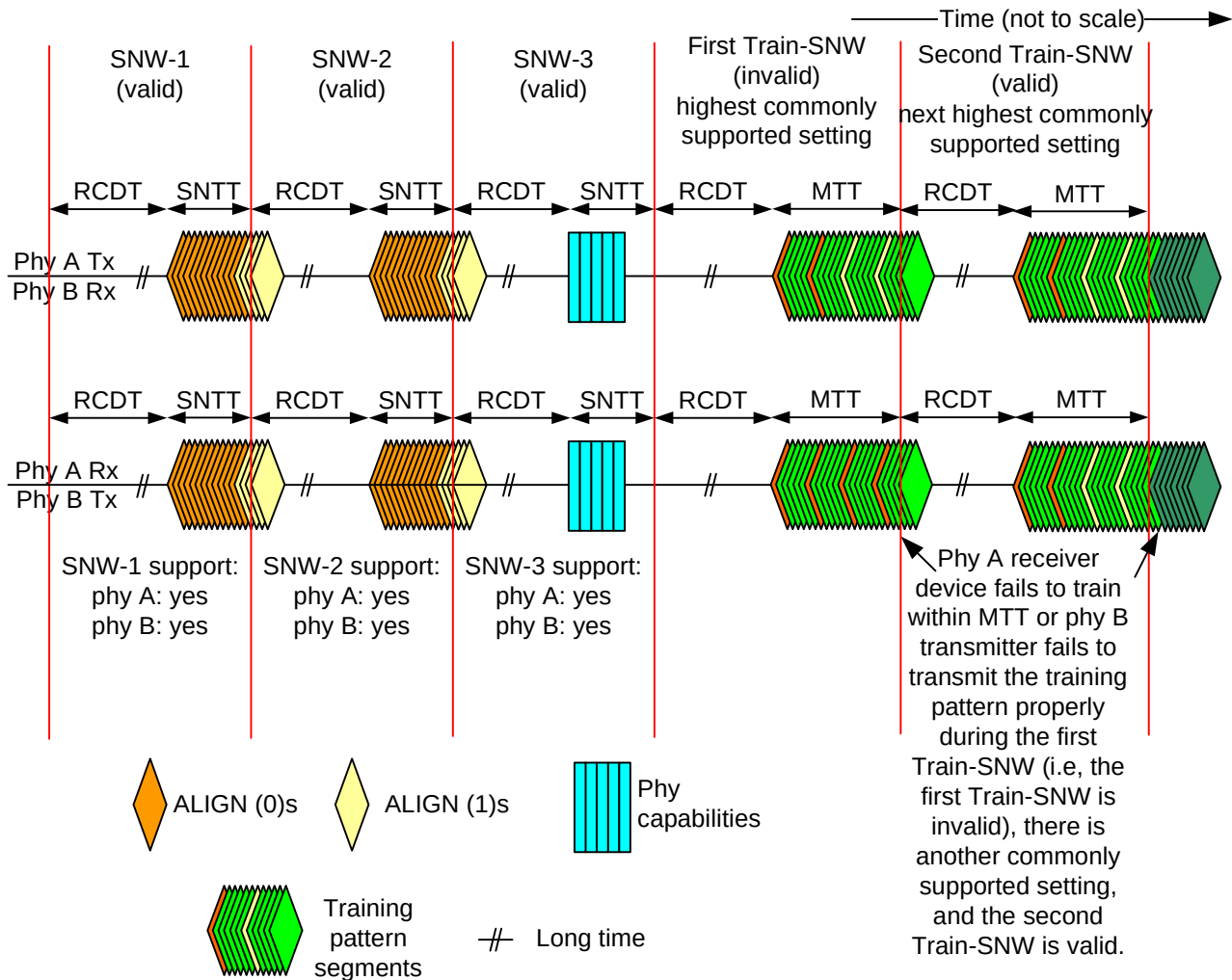


Figure 140 — SAS speed negotiation sequence - multiple Train-SNWs

For more examples of speed negotiations, see Annex C.

6.7.4.3 Multiplexing sequence

If SNW-3 indicates multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), the phy shall transmit the multiplexing sequence immediately after the speed negotiation sequence.

The multiplexing sequence is:

- 1) MUX (LOGICAL LINK 0);
- 2) MUX (LOGICAL LINK 1);
- 3) MUX (LOGICAL LINK 0);
- 4) MUX (LOGICAL LINK 1);
- 5) MUX (LOGICAL LINK 0); and
- 6) MUX (LOGICAL LINK 1).

The phy shall not transmit deletable primitives for physical link rate tolerance management (see 7.3) during the multiplexing sequence.

If SNW-3 indicates multiplexing is not enabled, the phy shall not transmit the multiplexing sequence.

The phy shall assign the incoming logical links to its logical phys based on the first MUX primitive it receives:

- a) MUX (LOGICAL LINK 0) indicates the position of logical link 0 and indicates the next dword is in logical link 1; or
- b) MUX (LOGICAL LINK 1) indicates the position of logical link 1 and indicates the next dword is in logical link 0.

The phy shall process MUX primitives in logic running off the received clock without using an elasticity buffer, because they are not accompanied by additional deletable primitives (e.g., ALIGNs and/or NOTIFYs).

Editor's Note 32: Simplify "in logic running off the received clock without using an elasticity buffer" by enhancing the figure in 7.3 showing such logic and referencing that here

The phy shall handle errors during the multiplexing sequence (i.e., after receiving the first MUX primitive) as follows:

- a) If the phy loses dword synchronization, it shall restart the link reset sequence rather than attempt to reestablish dword synchronization;
- b) If the phy receives a dword that is not a MUX primitive before receiving the MUX primitive expected in that position, it shall discard the dword;
- c) If the phy receives an invalid dword, it shall discard the dword; and
- d) If the phy receives a MUX primitive that does not match the MUX primitive expected in that position (i.e., it receives MUX (LOGICAL LINK 1) on logical link 0 or receives MUX (LOGICAL LINK 0) on logical link 1), it shall restart the link reset sequence.

6.7.5 Phy reset sequence after devices are attached

Since SATA and SAS signal cable connectors do not include power lines, it is not possible to detect the physical insertion of the signal cable connector onto a plug. Non-cabled environments may similarly not have a way to detect physical insertion of a device. As a result, every time a phy reset sequence is originated:

- a) expander phys that are enabled but not active shall originate a new phy reset sequence repeatedly, with no more than a hot-plug timeout (see table 85 in 6.7.1) between each attempt, until a speed negotiation sequence completes successfully;
- b) SAS initiator phys should originate a new phy reset sequence after every hot-plug timeout; and
- c) SAS target phys should not originate a new phy reset sequence after their first attempt.

Figure 141 shows how two phys complete the phy reset sequence if the phys are not attached at power on. In this example, phy A and phy B are attached some time before phy B's second hot-plug timeout occurs. Phy B's OOB detection circuitry detects a COMINIT after the attachment, and therefore phy B transmits COMSAS, since it has both transmitted and received a COMINIT. Upon receiving COMSAS, phy A transmits its own COMSAS. The SAS speed negotiation sequence follows.

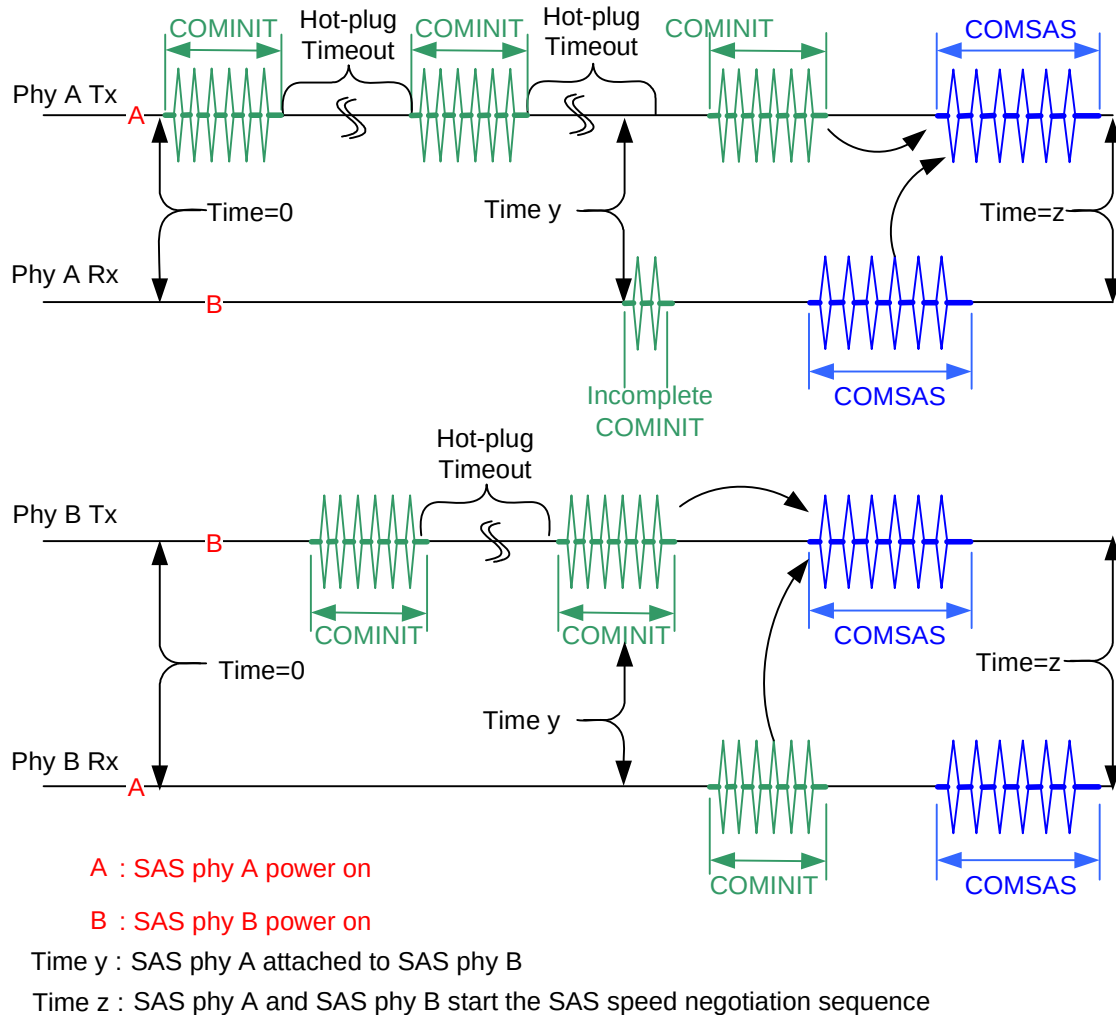


Figure 141 — Hot-plug and the phy reset sequence

6.8 SP (phy layer) state machine

6.8.1 SP state machine overview

The SP state machine controls the phy reset sequence. This state machine consists of three sets of states:

- OOB sequence (OOB) states;
- SAS speed negotiation (SAS) states; and
- SATA host emulation (SATA) states.

This state machine consists of the following states:

- SP0:OOB_COMINIT (see 6.8.3.2)(initial state);
- SP1:OOB_AwaitCOMX (see 6.8.3.3);
- SP2:OOB_NoCOMSASTimeout (see 6.8.3.4);
- SP3:OOB_AwaitCOMINIT_Sent (see 6.8.3.5);
- SP4:OOB_COMSAS (see 6.8.3.6);
- SP5:OOB_AwaitCOMSAS_Sent (see 6.8.3.7);

- g) SP6:OOB_AwaitNoCOMSAS (see 6.8.3.8);
- h) SP7:OOB_AwaitCOMSAS (see 6.8.3.9);
- i) SP8:SAS_Start (see 6.8.4.2);
- j) SP9:SAS_WindowNotSupported (see 6.8.4.3);
- k) SP10:SAS_AwaitALIGN (see 6.8.4.4);
- l) SP11:SAS_AwaitALIGN1 (see 6.8.4.5);
- m) SP12:SAS_AwaitSNW (see 6.8.4.6);
- n) SP13:SAS_Pass (see 6.8.4.7);
- o) SP14 SAS_Fail (see 6.8.4.8);
- p) SP15:SAS_PHY_Ready (see 6.8.4.9);
- q) SP16:SATA_COMWAKE (see 6.8.5.2);
- r) SP17:SATA_AwaitCOMWAKE (see 6.8.5.3);
- s) SP18:SATA_AwaitNoCOMWAKE (see 6.8.5.4);
- t) SP19:SATA_AwaitALIGN (see 6.8.5.5);
- u) SP20:SATA_AdjustSpeed (see 6.8.5.6);
- v) SP21:SATA_Transmit_ALIGN (see 6.8.5.7);
- w) SP22:SATA_PHY_Ready (see 6.8.5.8);
- x) SP23:SATA_PM_Partial (see 6.8.5.9);
- y) SP24:SATA_PM_Slumber (see 6.8.5.10);
- z) SP25:SATA_PortSel (see 6.8.6.2);
- aa) SP26:SATA_SpinupHold (see 6.8.7.2);
- ab) SP27:SAS_Settings (see 6.8.4.10);
- ac) SP28:SAS_TrainSetup (see 6.8.4.11);
- ad) SP29:SAS_Train (see 6.8.4.12); and
- ae) SP30:SAS_TrainingDone (see 6.8.4.13).

The SP state machine shall start in the SP0:OOB_COMINIT state after:

- a) a power on;
- b) a hard reset;
- c) receiving a Management Reset request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET in an expander device); or
- d) receiving a Disable Phy request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of DISABLE in an expander device).

The SP state machine shall maintain a MgmtReset state machine variable to determine whether a Management Reset request has been received. Any SP state that receives a Management Reset request shall set the MgmtReset state machine variable to one before making a transition to the SP0:OOB_COMINIT state (see 6.8.3.2). Any SP state that receives a power on, or a hard reset shall set the MgmtReset state machine variable to zero before making a transition to the SP0:OOB_COMINIT state.

The SP state machine shall maintain a Current SNW state machine variable to determine the current SNW (e.g., SNW-1, SNW-2, SNW-3, Final-SNW, Train-SNW, or Unsupported Phy Attached).

If the phy status is available through any of the following:

- a) the SMP DISCOVER response (see 10.4.3.9);
- b) the SMP DISCOVER LIST response (see 10.4.3.14);
- c) the Phy Control And Discover mode page (see 10.2.7.5); or
- d) the Protocol-Specific Port log page (see 10.2.8.1),

then the SP state machine shall maintain a ResetStatus state machine variable to determine the NEGOTIATED PHYSICAL LINK RATE field and/or the NEGOTIATED LOGICAL LINK RATE field.

If the phy supports SNW-3, then the SP state machine shall maintain a Commonly Supported Settings state machine variable that contains supported settings common between the phy and the attached phy.

If the phy supports SATA port selectors, the SP state machine shall maintain a COMWAKE_Received state machine variable to indicate whether a COMWAKE detected message was received in the SP0:OOB_COMINIT state or the SP1:OOB_AwaitCOMX state since the last time the SP0:OOB_COMINIT

state was entered, and the SP state machine shall transition to the SP25:SATA_PortSel state whenever it receives a Transmit SATA Port Selection Signal request.

The SP state machine sends the following messages to the SP_DWS state machine (see 6.9):

- a) Start DWS; and
- b) Stop DWS.

The SP state machine receives the following messages from the SP_DWS state machine:

- a) DWS Lost; and
- b) DWS Reset.

The SP state machine shall maintain the timers listed in table 96.

Table 96 — SP state machine timers

Timer	Initial value
COMSAS Detect Timeout timer	COMSAS detect timeout (see table 79 in 6.6.1)
Await ALIGN Timeout timer	Await ALIGN timeout (see table 86 in 6.7.2.2)
Hot-Plug Timeout timer	Hot plug timeout (see table 85 in 6.7.1)
RCDT timer	RCDT (see table 87 in 6.7.4.2)
SNLT timer	SNLT (see table 87 in 6.7.4.2)
SNTT timer	SNTT (see table 87 in 6.7.4.2)
TLT timer	TLT (see table 87 in 6.7.4.2)
MTT timer	MTT (see table 87 in 6.7.4.2)

6.8.2 SP transmitter and receiver

The SP transmitter transmits OOB signals and dwords on the physical link based on messages from the SP state machine (see 6.8).

The SP transmitter receives the following messages from the SP state machine:

- a) Transmit COMINIT;
- b) Transmit COMSAS;
- c) Transmit COMWAKE;
- d) Transmit SATA Port Selection Signal;
- e) Transmit D10.2;
- f) Set Rate with a Physical Link Rate argument, and an SSC On or an SSC Off argument;
- g) Transmit ALIGN with an argument indicating the specific type (e.g., Transmit ALIGN (0));
- h) Transmit Phy Capabilities Bits;
- i) Transmit TRAIN Pattern;
- j) Transmit TRAIN_DONE Pattern; and
- k) Transmit MUX Sequence.

When not otherwise instructed, the SP transmitter transmits D.C. idle.

Upon receiving a Transmit MUX Sequence message, the SP transmitter transmits:

- 1) MUX (LOGICAL LINK 0);
- 2) MUX (LOGICAL LINK 1);
- 3) MUX (LOGICAL LINK 0);
- 4) MUX (LOGICAL LINK 1);
- 5) MUX (LOGICAL LINK 0); and
- 6) MUX (LOGICAL LINK 1).

The SP transmitter shall complete any physical link rate change requested with the Set Rate message within RCDT (see table 87 in 6.7.4.2).

The SP transmitter sends the following messages to the SP state machine:

- a) COMINIT Transmitted;
- b) COMSAS Transmitted;
- c) COMWAKE Transmitted;
- d) SATA Port Selection Signal Transmitted;
- e) TRAIN_DONE Pattern Transmitted; and
- f) Phy Capabilities Bits Transmitted.

The SP receiver receives OOB signals and dwords from the physical link and sends messages to the SP state machine indicating what it has received.

The SP receiver receives the following messages from the SP state machine:

- a) Set Rate with a Physical Link Rate argument, and an SSC On argument or an SSC Off argument;
- b) Start Training; and
- c) Abort Training.

The SP receiver sends the following messages to the SP state machine:

- a) COMINIT Detected;
- b) COMSAS Detected;
- c) COMWAKE Detected;
- d) COMSAS Completed;
- e) COMWAKE Completed;
- f) ALIGN Received with an argument indicating the specific type (e.g., ALIGN Received (0));
- g) Phy Capabilities Bits Received with arguments indicating the supported settings bits received;
- h) Training Completed;
- i) TRAIN_DONE Received; and
- j) Dword Received.

The ALIGN Received, Dword Received, and TRAIN_DONE Received messages are only sent when the SP_DWS state machine has achieved dword synchronization.

For SATA speed negotiation, the ALIGN Received (0) message includes an argument containing the physical link rate at which the ALIGN (0) primitives were detected. For SAS speed negotiation, only ALIGNs at the physical link rate specified by the last Set Rate message received by the SP transmitter cause ALIGN Received messages.

The SP transmitter relationship to other transmitters is defined in 4.3.2. The SP receiver relationship to other receivers is defined in 4.3.3.

6.8.3.2 SP0:OOB_COMINIT state

6.8.3.2.1 State description

This state is the initial state for this state machine.

Upon entry into this state, the phy shall:

- a) set the COMWAKE_Received state machine variable to zero;
- b) send a Stop DWS message to the SP_DWS state machine;
- c) send a Phy Layer Not Ready confirmation to the link layer;
- d) set the ATTACHED SATA DEVICE bit to zero in the SMP DISCOVER response (see 10.4.3.9);
- e) if this state was entered due to power on, set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response (see 10.4.3.9); and
- f) if this state was not entered because of a Disable Phy request, send a Transmit COMINIT message to the SP transmitter.

If this state was entered because of a Disable Phy request, then upon entry into this state, this state shall:

- a) ignore COMINIT Detected messages until this state is re-entered due to a power on, hard reset, or Management Reset request; and
- b) set the ResetStatus state machine variable to DISABLED.

If this state was entered due to power on or hard reset, then upon entry into this state, this state shall set the ResetStatus state machine variable to UNKNOWN.

If this state was entered because of a Management Reset request, then upon entry into this state, this state shall:

- a) if the ResetStatus state machine variable is not set to RESET_IN_PROGRESS, SPINUP_HOLD, G1, G2, or G3, set the ResetStatus state machine variable to UNKNOWN; or
- b) if the ResetStatus state machine variable is set to RESET_IN_PROGRESS, SPINUP_HOLD, G1, G2, or G3, set the ResetStatus state machine variable to RESET_IN_PROGRESS.

If this state was not entered due to a power on, hard reset, Disable Phy, or Management Reset request, then upon entry into this state, this state shall:

- a) if the ResetStatus state machine variable is not set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, set the ResetStatus state machine variable to UNKNOWN; or
- b) if the ResetStatus state machine variable is set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, not change the ResetStatus state machine variable.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

- a) if the ResetStatus state machine variable is not set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, set the ResetStatus state machine variable to PORT_SELECTOR;
- b) set the COMWAKE_Received state machine variable to one; and
- c) if the ATTACHED SATA PORT SELECTOR bit is set to zero in the SMP DISCOVER response (see 10.4.3.9):
 - A) set the ATTACHED SATA PORT SELECTOR bit to one in the SMP DISCOVER response; and
 - B) send a SATA Port Selector Change confirmation to the link layer.

This state machine waits for receipt of a COMINIT Transmitted message and/or a COMINIT Detected message.

6.8.3.2.2 Transition SP0:OOB_COMINIT to SP1:OOB_AwaitCOMX

This transition shall occur if this state receives a COMINIT Transmitted message and has not received a COMINIT Detected message.

6.8.3.2.3 Transition SP0:OOB_COMINIT to SP3:OOB_AwaitCOMINIT_Sent

This transition shall occur if this state receives a COMINIT Detected message and has not received a COMINIT Transmitted message.

6.8.3.2.4 Transition SP0:OOB_COMINIT to SP4:OOB_COMSAS

This transition shall occur if this state receives both a COMINIT Transmitted message and a COMINIT Detected message.

6.8.3.3 SP1:OOB_AwaitCOMX state**6.8.3.3.1 State description**

Upon entry into this state, the phy shall initialize and start the Hot-Plug Timeout timer if this phy is:

- a) an expander phy; or
- b) an initiator phy or target phy implementing the Hot-Plug Timeout timer.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

- a) if the ResetStatus state machine variable is not set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, set the ResetStatus state machine variable to PORT_SELECTOR;
- b) set the COMWAKE_Received state machine variable to one; and
- c) if the ATTACHED SATA PORT SELECTOR bit is set to zero in the SMP DISCOVER response (see 10.4.3.9):
 - A) set the ATTACHED SATA PORT SELECTOR bit to one in the SMP DISCOVER response; and
 - B) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.3.2 Transition SP1:OOB_AwaitCOMX to SP0:OOB_COMINIT

This transition shall occur if the Hot-Plug Timeout timer expires.

If the COMWAKE_Received state machine variable is set to zero and the ATTACHED SATA PORT SELECTOR bit is set to one in the SMP DISCOVER response (see 10.4.3.9), the state machine shall, before the transition:

- a) set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response; and
- b) send a SATA Port Selector Change confirmation to the link layer.

Before the transition, if this state was entered from SP0:OOB_COMINIT, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.3.3.3 Transition SP1:OOB_AwaitCOMX to SP4:OOB_COMSAS

This transition shall occur after receiving either a COMINIT Detected message or a COMSAS Detected message. If COMSAS Detected was received, this transition shall include a COMSAS Detected argument.

If the ATTACHED SATA PORT SELECTOR bit is set to one in the SMP DISCOVER response (see 10.4.3.9), the state machine shall, before the transition:

- a) set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response; and
- b) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.4 SP2:OOB_NoCOMSASTimeout state**6.8.3.4.1 State description**

Upon entry into this state, the phy shall initialize and start the Hot-Plug Timeout timer if this phy is:

- a) this phy is an expander phy; or
- b) this phy is an initiator phy or target phy implementing the Hot-Plug Timeout timer.

6.8.3.4.2 Transition SP2:OOB_NoCOMSASTimeout to SP0:OOB_COMINIT

This transition shall occur if the Hot-Plug Timeout timer expires.

6.8.3.4.3 Transition SP2:OOB_NoCOMSASTimeout to SP4:OOB_COMSAS

This transition shall occur after receiving a COMINIT Detected message.

6.8.3.5 SP3:OOB_AwaitCOMINIT_Sent state**6.8.3.5.1 State description**

This state waits for a COMINIT Transmitted message.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

- a) if the ResetStatus state machine variable is not set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, set the ResetStatus state machine variable to PORT_SELECTOR; and
- b) if the ATTACHED SATA PORT SELECTOR bit is set to zero in the SMP DISCOVER response (see 10.4.3.9):
 - A) set the ATTACHED SATA PORT SELECTOR bit to one in the SMP DISCOVER response; and
 - B) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.5.2 Transition SP3:OOB_AwaitCOMINIT_Sent to SP4:OOB_COMSAS

This transition shall occur after receiving a COMINIT Transmitted message.

6.8.3.6 SP4:OOB_COMSAS state**6.8.3.6.1 State description**

Upon entry into this state, the phy shall send a Transmit COMSAS message to the SP transmitter.

This state waits for receipt of a COMSAS Transmitted message and/or a COMSAS Detected message.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

- a) if the ResetStatus state machine variable is not set to PHY_RESET_PROBLEM, SPINUP_HOLD, or UNSUPPORTED_PHY_ATTACHED, set the ResetStatus state machine variable to PORT_SELECTOR; and
- b) if the ATTACHED SATA PORT SELECTOR bit is set to zero in the SMP DISCOVER response (see 10.4.3.9):
 - A) set the ATTACHED SATA PORT SELECTOR bit to one in the SMP DISCOVER response; and
 - B) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.6.2 Transition SP4:OOB_COMSAS to SP5:OOB_AwaitCOMSAS_Sent

This transition shall occur if this state receives a COMSAS Detected message or this state was entered with a COMSAS Detected argument, and this state has not received a COMSAS Transmitted message.

If the ATTACHED SATA PORT SELECTOR bit is set to one in the SMP DISCOVER response (see 10.4.3.9), the state machine shall set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response and send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.6.3 Transition SP4:OOB_COMSAS to SP6:OOB_AwaitNoCOMSAS

This transition shall occur if this state receives both a COMSAS Transmitted message and a COMSAS Detected message.

If the ATTACHED SATA PORT SELECTOR bit is set to one in the SMP DISCOVER response (see 10.4.3.9), the state machine shall set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response and send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.6.4 Transition SP4:OOB_COMSAS to SP7:OOB_AwaitCOMSAS

This transition shall occur if this state receives a COMSAS Transmitted message and has not received a COMSAS Detected message.

6.8.3.7 SP5:OOB_AwaitCOMSAS_Sent state

6.8.3.7.1 State description

This state waits for receipt of a COMSAS Transmitted message.

6.8.3.7.2 Transition SP5:OOB_AwaitCOMSAS_Sent to SP6:OOB_AwaitNoCOMSAS

This transition shall occur after receiving a COMSAS Transmitted message.

If this state received a COMSAS Completed message, it shall include a COMSAS Completed argument with the transition.

6.8.3.8 SP6:OOB_AwaitNoCOMSAS state

6.8.3.8.1 State description

This state machine waits for a COMSAS Completed message, which indicates that COMSAS has been completely received.

6.8.3.8.2 Transition SP6:OOB_AwaitNoCOMSAS to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.3.8.3 Transition SP6:OOB_AwaitNoCOMSAS to SP8:SAS_Start

This transition shall occur after receiving a COMSAS Completed message, or shall occur if a COMSAS Completed argument was received in the transition.

6.8.3.9 SP7:OOB_AwaitCOMSAS state

6.8.3.9.1 State description

Upon entry into this state, the phy shall initialize and start the COMSAS Detect Timeout timer.

6.8.3.9.2 Transition SP7:OOB_AwaitCOMSAS to SP2:OOB_NoCOMSASTimeout

This transition shall occur if the phy does not support SATA and the COMSAS Detect Timeout timer expires. The state machine shall set the MgmtReset state machine variable to zero before the transition.

6.8.3.9.3 Transition SP7:OOB_AwaitCOMSAS to SP6:OOB_AwaitNoCOMSAS

This transition shall occur after receiving a COMSAS Detected message.

The state machine shall set the MgmtReset state machine variable to zero before the transition.

The state machine shall set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response (see 10.4.3.9). If the ATTACHED SATA PORT SELECTOR bit in the SMP DISCOVER response was set to one prior to this transition, the state machine shall send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.9.4 Transition SP7:OOB_AwaitCOMSAS to SP16:SATA_COMWAKE

This transition shall occur if:

- a) the phy supports SATA;
- b) the COMSAS Detect Timeout timer expires; and
 - A) the MgmtReset state machine variable is set to one; or
 - B) the phy does not implement SATA spinup hold.

The state machine shall set the MgmtReset state machine variable to zero before the transition.

The state machine shall set the ATTACHED SATA DEVICE bit to one in the SMP DISCOVER response (see 10.4.3.9) before the transition.

6.8.3.9.5 Transition SP7:OOB_AwaitCOMSAS to SP26:SATA_SpinupHold

This transition shall occur if:

- a) the phy supports SATA;
- b) the COMSAS Detect Timeout timer expires;
- c) the phy implements SATA spinup hold; and
- d) the MgmtReset state machine variable is set to zero.

The state machine shall set the ATTACHED SATA DEVICE bit to one in the SMP DISCOVER response (see 10.4.3.9) before the transition.

6.8.4 SAS speed negotiation states**6.8.4.1 SAS speed negotiation states overview**

Figure 143 shows the SAS speed negotiation states, in which the phy has detected that it is attached to a SAS phy or expander phy rather than a SATA phy, and performs the SAS speed negotiation sequence. These states are indicated by state names with a prefix of SAS.

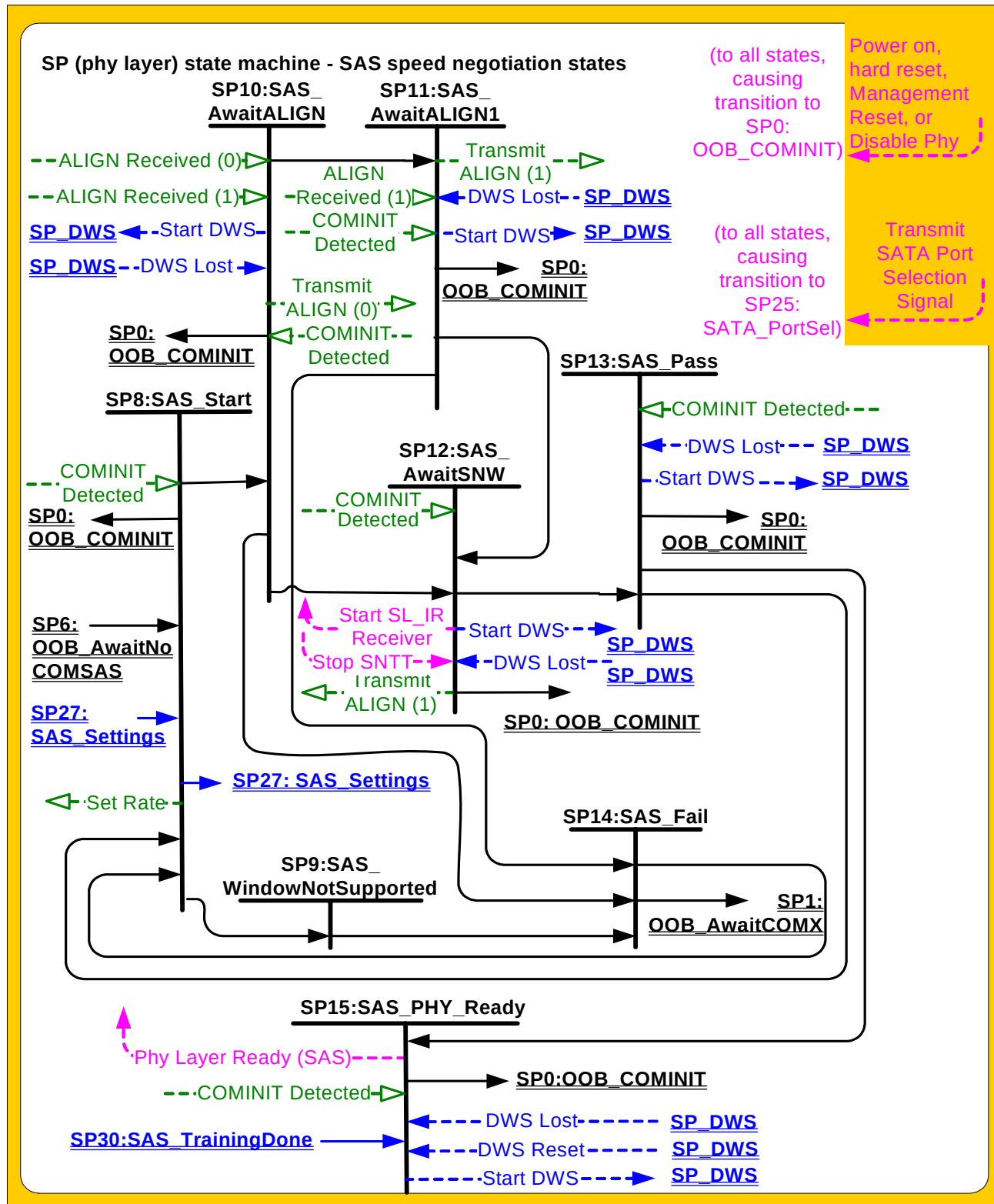


Figure 143 — SP (phy layer) state machine - SAS speed negotiation states

Figure 144 shows the SAS speed negotiation states related to SNW-3 and Train-SNW.

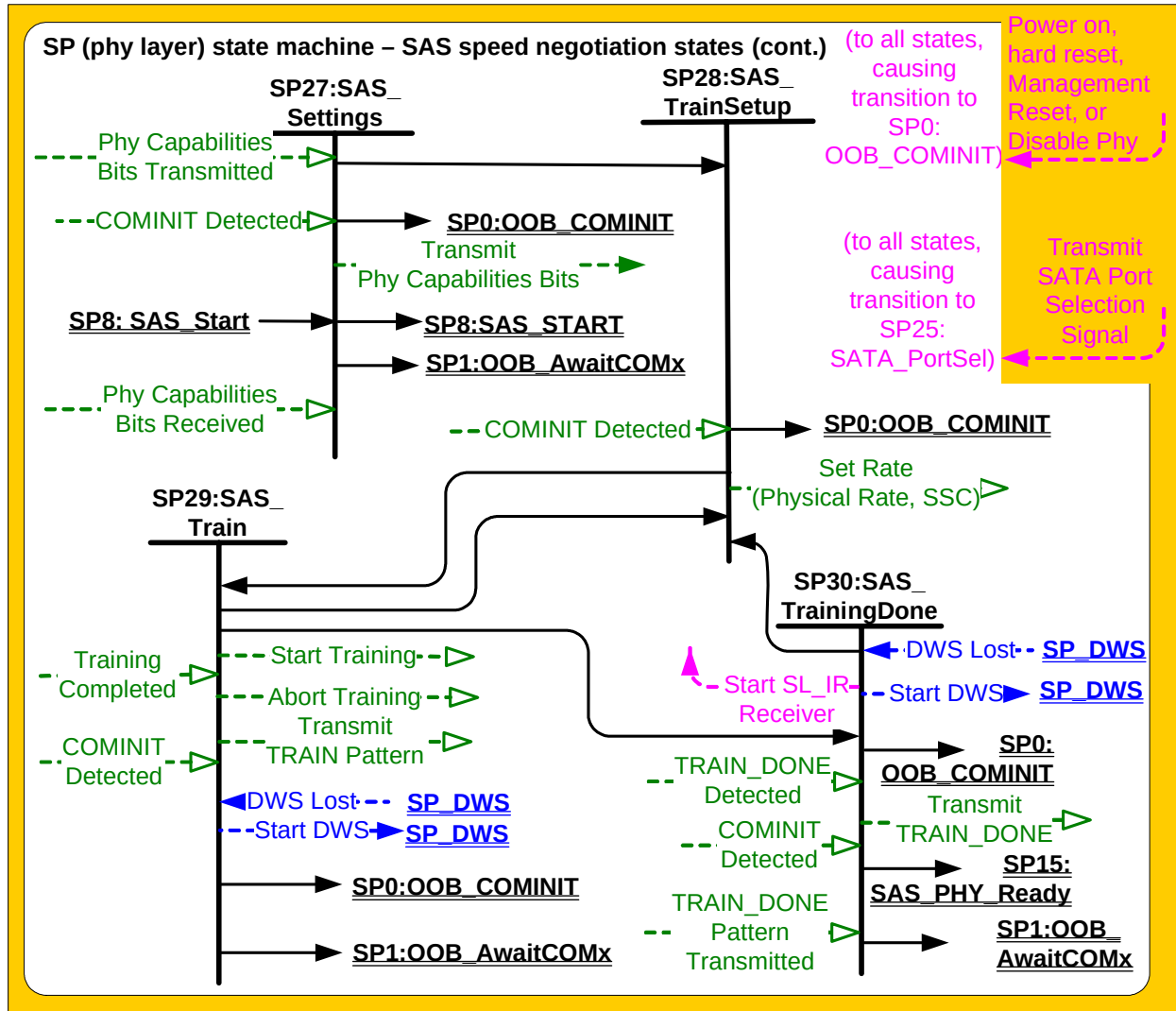


Figure 144 — SP (phy layer) state machine - SAS speed negotiation states for SNW-3 and Train-SNW

6.8.4.2 SP8:SAS_Start state

6.8.4.2.1 State description

This is the initial state for the SAS speed negotiation sequence.

Upon entry into this state, the phy shall:

- initialize and start the RCDT timer;
- send a Set Rate message to the SP transmitter with the argument set to:
 - 1.5 Gbps, if the transition into this state was from the SP6:OOB_AwaitNoCOMSAS state (i.e., if this is the first SNW); or
 - the value of the SAS SNW Rate argument.

Editor's Note 33: No "SAS SNW Rate argument" is defined.

If this state is entered from SP6:OOB_AwaitNoCOMSAS, then the Current SNW state machine variable shall be set to SNW-1. If this state is not entered from SP6:OOB_AwaitNoCOMSAS, then the Current SNW state machine variable shall be set to:

- a) SNW-2 if the Current SNW state machine variable is set to SNW-1;
- a) SNW-3 if the Current SNW state machine variable is set to SNW-2, and either SNW-1 is invalid or SNW-2 is valid;
- b) Final-SNW if the Current SNW state machine variable is set to SNW-2, SNW-1 is valid, and SNW-2 is invalid;
- c) Final-SNW if the Current SNW state machine variable is set to SNW-3, SNW-3 is invalid, and SNW-2 is valid; or
- d) Unsupported Phy Attached if the Current SNW state machine variable is set to SNW-3, SNW-3 is invalid, and SNW-2 is invalid.

After the Current SNW state machine variable is updated, if the Current SNW state machine variable is not set to Unsupported Phy Attached, then this state shall:

- a) initialize and start the RCDT timer; and
- b) if the SNW specified by the Current SNW state machine variable is supported, send a Set Rate message to the SP transmitter and SP receiver with:
 - A) the Physical Link Rate argument set to 1,5 Gbps and an SSC Off argument, if the Current SNW state machine variable is set to SNW-1;
 - B) the Physical Link Rate argument set to 3,0 Gbps and an SSC Off argument, if the Current SNW state machine variable is set to SNW-2;
 - C) the Physical Link Rate arguments set to 1,5 Gbps and an SSC Off argument, if the Current SNW state machine variable is set to Final-SNW, SNW-1 was valid and SNW-2 was invalid;
 - D) the Physical Link Rate arguments set to 3,0 Gbps and an SSC Off argument, if the Current SNW state machine variable is set to Final-SNW and SNW-2 was valid; or
 - E) the Physical Link Rate argument set to 1,5 Gbps and either an SSC On argument or an SSC Off argument, if the Current SNW state machine variable is set to SNW-3.

During this state D.C. idle shall be transmitted.

6.8.4.2.2 Transition SP8:SAS_Start to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.2.3 Transition SP8:SAS_Start to SP1:OOB_AwaitCOMX

This transition shall occur if the Current SNW state machine variable is set to Unsupported Phy Attached.

Before the transition, this state shall set the ResetStatus state machine variable to UNSUPPORTED_PHY_ATTACHED.

6.8.4.2.4 Transition SP8:SAS_Start to SP9:SAS_WindowNotSupported

This transition shall occur after the RCDT timer expires if the SNW indicated by the Current SNW state machine variable is not supported.

6.8.4.2.5 Transition SP8:SAS_Start to SP10:SAS_AwaitALIGN

This transition shall occur after the RCDT timer expires if:

- a) the Current SNW state machine variable is not set to SNW-3; and
- b) the SNW indicated by the Current SNW state machine variable is supported.

6.8.4.2.6 Transition SP8:SAS_Start to SP27:SAS_Settings

This transition shall occur after the RCDT timer expires if:

- a) the Current SNW state machine variable is set to SNW-3; and

- b) SNW-3 is supported.

6.8.4.3 SP9:SAS_WindowNotSupported state

6.8.4.3.1 State description

Upon entry into this state, the phy shall initialize and start the SNTT timer.

During this state D.C. idle shall be transmitted.

6.8.4.3.2 Transition SP9:SAS_WindowNotSupported to SP14:SAS_Fail

This transition shall occur after the SNTT timer expires.

6.8.4.4 SP10:SAS_AwaitALIGN state

6.8.4.4.1 State description

Upon entry into this state, the phy shall:

- a) initialize and start the SNTT timer and SNLT timer;
- b) send a Start DWS message to the SP_DWS state machine; and
- c) repeatedly send Transmit ALIGN (0) messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.4.2 Transition SP10:SAS_AwaitALIGN to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.4.3 Transition SP10:SAS_AwaitALIGN to SP11:SAS_AwaitALIGN1

This transition shall occur if this state receives an ALIGN Received (0) message before the SNLT timer expires.

6.8.4.4.4 Transition SP10:SAS_AwaitALIGN to SP12:SAS_AwaitSNW

This transition shall occur if this state receives an ALIGN Received (1) message before the SNLT timer expires.

6.8.4.4.5 Transition SP10:SAS_AwaitALIGN to SP14:SAS_Fail

This transition shall occur if the SNTT timer expires.

6.8.4.5 SP11:SAS_AwaitALIGN1 state

6.8.4.5.1 State description

This state shall repeatedly send Transmit ALIGN (1) messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.5.2 Transition SP11:SAS_AwaitALIGN1 to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.5.3 Transition SP11:SAS_AwaitALIGN1 to SP12:SAS_AwaitSNW

This transition shall occur if this state receives an ALIGN Received (1) message before the SNLT timer expires. This indicates that the other phy has been able to achieve dword synchronization in the current SNW.

6.8.4.5.4 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_Fail

This transition shall occur if the SNTT timer expires. This indicates that the other phy has not been able to achieve dword synchronization in the current SNW.

6.8.4.6 SP12:SAS_AwaitSNW state**6.8.4.6.1 State description**

This state shall repeatedly send Transmit ALIGN (1) messages to the SP transmitter.

If the Current SNW state machine variable is set to Final-SNW, this state shall send a Start SL_IR Receiver confirmation to the link layer.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

This state waits for the SNTT timer to expire or for a Stop SNTT request.

6.8.4.6.2 Transition SP12:SAS_AwaitSNW to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.6.3 Transition SP12:SAS_AwaitSNW to SP13:SAS_Pass

This transition shall occur after the SNTT timer expires or after receiving a Stop SNTT request.

6.8.4.7 SP13:SAS_Pass state**6.8.4.7.1 State description**

This state determines if:

- a) another SAS SNW is required; or
- b) the SAS speed negotiation sequence is complete.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.7.2 Transition SP13:SAS_Pass to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.7.3 Transition SP13:SAS_Pass to SP8:SAS_Start

This transition shall occur if the Current SNW state machine variable is not set to Final-SNW.

6.8.4.7.4 Transition SP13:SAS_Pass to SP15:SAS_PHY_Ready

This transition shall occur if the Current SNW state machine variable is set to Final-SNW.

6.8.4.8 SP14:SAS_Fail state**6.8.4.8.1 State description**

This state determines if:

- a) another SAS SNW is required; or
- b) the SAS speed negotiation sequence is complete.

6.8.4.8.2 Transition SP14:SAS_Fail to SP1:OOB_AwaitCOMX

This transition shall occur if the Current SNW state machine variable is set to Final-SNW.

Before the transition, this state shall set the ResetStatus state machine variable to PHY_RESET_PROBLEM.

6.8.4.8.3 Transition SP14:SAS_Fail to SP8:SAS_Start

This transition shall occur if the Current SNW state machine variable is not set to Final-SNW.

6.8.4.9 SP15:SAS_PHY_Ready state**6.8.4.9.1 State description**

This state waits for a COMINIT Detected message, a DWS Lost message, or a DWS Reset message.

Upon entry into this state, the phy shall:

- a) if multiplexing is enabled (see table 92 in 6.7.4.2.3.3), send a Transmit MUX Sequence message to the SP transmitter;
- b) send a Phy Layer Ready (SAS) confirmation to the link layer to indicate that the physical link has been brought up successfully in SAS mode;
- c) if the SP transmitter is transmitting at 1.5 Gbps, set the ResetStatus state machine variable to G1;
- d) if the SP transmitter is transmitting at 3 Gbps, set the ResetStatus state machine variable to G2; and
- e) if the SP transmitter is transmitting at 6 Gbps, set the ResetStatus state machine variable to G3.

While in this state dwords from the link layer are transmitted at the negotiated physical link rate at the rate established in the previous SNW.

If multiplexing is disabled, each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

NOTE 32 - If multiplexing is enabled and this state receives a DWS Lost message, this state does not send a Start DWS message and the state machine transitions to SP0:OOB_COMINIT.

6.8.4.9.2 Transition SP15:SAS_PHY_Ready to SP0:OOB_COMINIT

This transition shall occur after:

- a) receiving a DWS Lost message, if this state does not send a Start DWS message;
- b) receiving a DWS Lost message followed by a COMINIT Detected message, if this state does not send a Start DWS message; or
- c) receiving a DWS Reset message.

This transition may but should not occur after receiving a COMINIT Detected message before receiving a DWS Lost message, or after receiving a COMINIT Detected message after sending a Start DWS message (i.e., the SP state machine should ignore COMINIT Detected messages unless the SP_DWS state machine has indicated loss of dword synchronization).

6.8.4.10 SP27:SAS_Settings state**6.8.4.10.1 State description**

This state transmits and receives phy capabilities bits.

Upon entry to this state, the phy shall:

- a) initialize and start the SNTT timer;
- b) set the Commonly Supported Settings state machine variable to indicate that there are no commonly supported settings; and
- c) send a Transmit Phy Capabilities Bits message to the SP transmitter.

If a Phy Capabilities Bits Received message is received with the argument of parity good, then this state shall set the Commonly Supported Settings state machine variable to the supported settings that are set to one in both the transmitted and received phy capabilities bits.

This state waits for the SNTT timer to expire.

6.8.4.10.2 Transition SP27:SAS_Settings to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.10.3 Transition SP27:SAS_Settings to SP1:OOB_AwaitCOMX

This transition shall occur after the SNTT timer expires if:

- a) a Phy Capabilities Bits Received message is received with an argument of parity bad; or
- b) no commonly supported settings exist after the Commonly Supported Settings state machine variable is set as a result of receiving a Phy Capabilities Bits Received message.

Before the transition, this state shall:

- a) if a Phy Capabilities Bits Received message is received with an argument of parity bad, set the ResetStatus state machine variable to PHY_RESET_PROBLEM; or
- b) if no commonly supported settings exist after the Commonly Supported Settings state machine variable is set, set the ResetStatus state machine variable to UNSUPPORTED_PHY_ATTACHED.

6.8.4.10.4 Transition SP27:SAS_Settings to SP8:SAS_Start

This transition shall occur if:

- a) the SNTT timer expires; and
- b) a Phy Capabilities Bits Received message is not received during this state.

6.8.4.10.5 Transition SP27:SAS_Settings to SP28:SAS_TrainSetup

This transition shall occur:

- a) after the SNTT timer expires; and
- b) if the Commonly Supported Settings state machine variable indicates there is at least one commonly supported setting.

6.8.4.11 SP28:SAS_TrainSetup

6.8.4.11.1 State description

Upon entry into this state, the phy shall:

- a) initialize and start the RCDT timer; and
- b) send a Set Rate message to the SP transmitter and receiver with the arguments set to reflect the highest priority commonly supported setting contained in the Commonly Supported Settings state machine variable.

After the Set Rate message is sent, the Commonly Supported Settings state machine variable shall be set to indicate that the supported settings bit used for the Set Rate message is no longer in common.

During this state D.C. idle shall be transmitted.

6.8.4.11.2 Transition SP28:SAS_TrainSetup to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.11.3 Transition SP28:SAS_TrainSetup to SP29:SAS_Train

This transition shall occur after the RCDT timer expires.

6.8.4.12 SP29:SAS_Train state**6.8.4.12.1 State description**

Upon entry into this state, the phy shall:

- a) initialize and start the MTT timer;
- b) initialize and start the TLT timer;
- c) send a Start Training message to the SP receiver; and
- d) send a Start DWS message to the SP_DWS state machine.

This state shall repeatedly send Transmit TRAIN Pattern messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state shall send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization.

If the MTT timer expires, then this state shall send an Abort Training message to the SP receiver.

6.8.4.12.2 Transition SP29:SAS_Train to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.12.3 Transition SP29:SAS_Train to SP1:OOB_AwaitCOMX

This transition shall occur if

- a) the MTT timer expires; and
- b) the Commonly Supported Settings state machine variable does not contain additional commonly supported settings.

This is a phy reset problem.

Before the transition, this state shall set the ResetStatus state machine variable to PHY_RESET_PROBLEM.

6.8.4.12.4 Transition SP29:SAS_Train to SP30:SAS_TrainingDone

This transition shall occur if:

- a) this state receives a Training Completed message before the TLT timer expires; and
- b) dword synchronization is acquired.

6.8.4.12.5 Transition SP29:SAS_Train to SP28:SAS_TrainSetup

This transition shall occur if:

- a) the MTT timer expires; and
- b) the Commonly Supported Settings state machine variable contains additional commonly supported settings.

6.8.4.13 SP30:SAS_TrainingDone state**6.8.4.13.1 State description**

This state shall repeatedly send Transmit TRAIN_DONE Pattern messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

This state waits for the MTT timer to expire or a TRAIN_DONE Received message from the receiver.

This state shall send a Start SL_IR Receiver confirmation to the link layer when a TRAIN_DONE Received message is received.

6.8.4.13.2 Transition SP30:SAS_TrainingDone to SP0:OOB_COMINIT

This transition shall occur after:

- a) receiving a DWS Lost message if this state does not send a Start DWS message; or
- b) receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.4.13.3 Transition SP30:SAS_TrainingDone to SP1:OOB_AwaitCOMX

This transition shall occur if:

- a) TRAIN_DONE Received message is not received before the MTT timer expires; and
- b) the Commonly Supported Settings state machine variable does not contain additional commonly supported settings.

This is a phy reset problem.

Before the transition, this state shall set the ResetStatus state machine variable to PHY_RESET_PROBLEM.

6.8.4.13.4 Transition SP30:SAS_TrainingDone to SP28:SAS_TrainSetup

This transition shall occur if:

- a) the MTT timer expires; and
- b) the Commonly Supported Settings state machine variable contains additional commonly supported settings.

6.8.4.13.5 Transition SP30:SAS_TrainingDone to SP15:SAS_PHY_Ready

This transition shall occur if this state receives:

- a) at least four TRAIN_DONE Pattern Transmitted messages; and
- b) a TRAIN_DONE Received message before the MTT timer expires.

6.8.5 SATA host emulation states

6.8.5.1 SATA host emulation states overview

Figure 145 shows the SATA host emulation states, in which the phy has detected that it is attached to a SATA phy and behaves as if it were a SATA host phy, initiating the SATA speed negotiation sequence. These states are indicated by state names with a prefix of SATA.

The power management states defined in this standard are for SAS initiator phys that support SATA; expander devices that support SATA do not support power management in this standard.

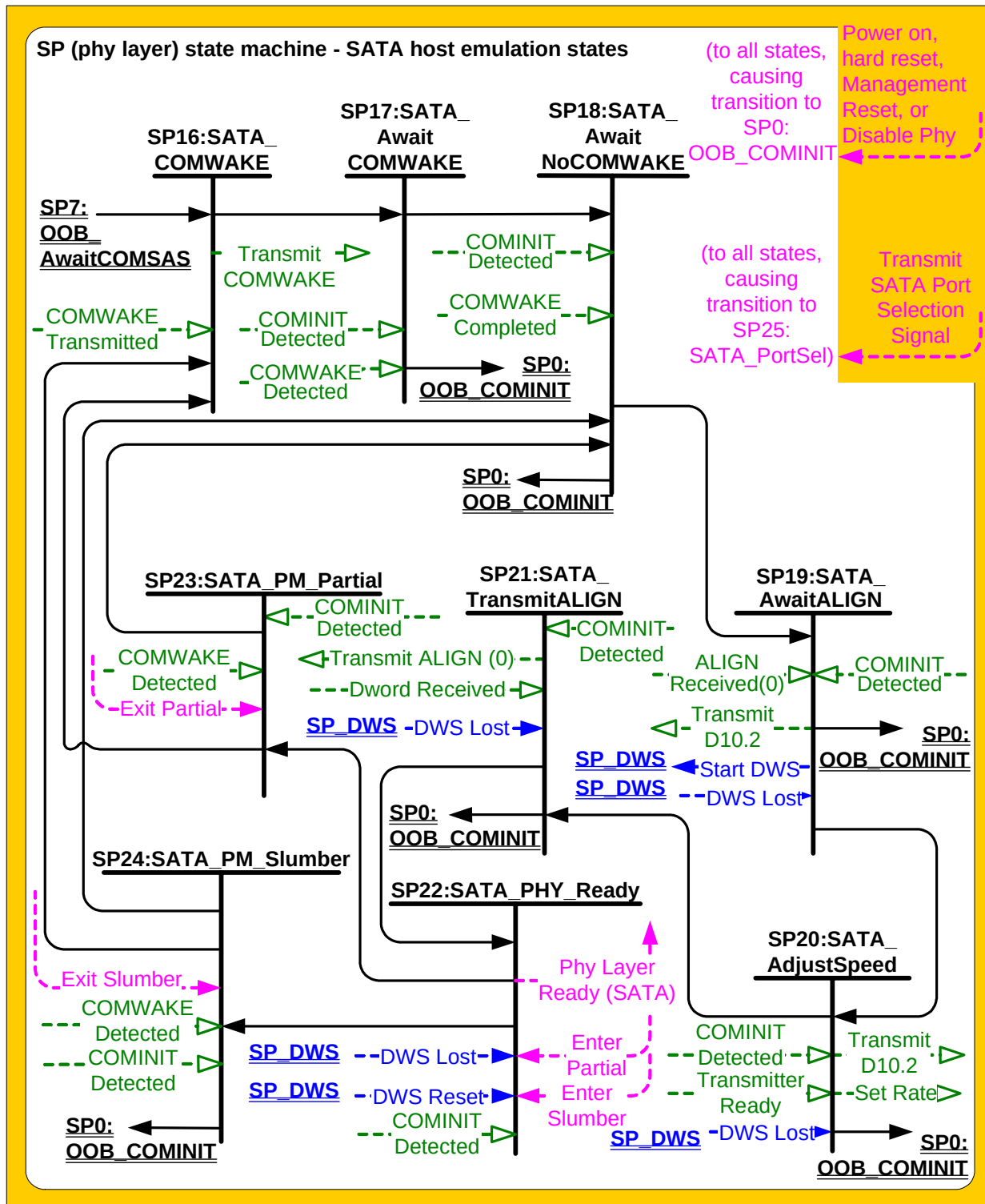


Figure 145 — SP (phy layer) state machine - SATA host emulation states

6.8.5.2 SP16:SATA_COMWAKE state**6.8.5.2.1 State description**

This state shall send a Transmit COMWAKE message to the SP transmitter and wait for a COMWAKE Transmitted message.

6.8.5.2.2 Transition SP16:SATA_COMWAKE to SP17:SATA_AwaitCOMWAKE

This transition shall occur after receiving a COMWAKE Transmitted message.

6.8.5.3 SP17:SATA_AwaitCOMWAKE state**6.8.5.3.1 State description**

This state waits for COMWAKE to be received.

6.8.5.3.2 Transition SP17:SATA_AwaitCOMWAKE to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.3.3 Transition SP17:SATA_AwaitCOMWAKE to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur after receiving a COMWAKE Detected message.

6.8.5.4 SP18:SATA_AwaitNoCOMWAKE state**6.8.5.4.1 State description**

This state waits for a COMWAKE Completed message.

6.8.5.4.2 Transition SP18:SATA_AwaitNoCOMWAKE to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.5.4.3 Transition SP18:SATA_AwaitNoCOMWAKE to SP19:SATA_AwaitALIGN

This transition shall occur after receiving a COMWAKE Completed message.

6.8.5.5 SP19:SATA_AwaitALIGN state**6.8.5.5.1 State description**

Upon entry into this state, the phy shall send a Start DWS message to the SP_DWS state machine.

This state shall:

- a) repeatedly send Transmit D10.2 messages to the SP transmitter;
- b) initialize and start the Await ALIGN Timeout timer; and
- c) wait for an ALIGN Received (0) message to be received or for the Await ALIGN Timeout timer to expire.

The phy shall start transmitting D10.2 characters no later than a COMWAKE response time (see 6.7.2.2) after entry into this state.

6.8.5.5.2 Transition SP19:SATA_AwaitALIGN to SP0:OOB_COMINIT

This transition shall occur if the Await ALIGN Timeout timer expires or after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

Before the transistion, this state shall:

- a) if the Await ALIGN Timeout timer expires, set the ResetStatus state machine variable to UNSUPPORTED_PHY_ATTACHED;
- b) after receiving a DWS Lost message, set the ResetStatus state machine variable to UNKNOWN; or
- c) after receiving a COMINIT Detected message, set the ResetStatus state machine variable to UNKNOWN.

6.8.5.5.3 Transition SP19:SATA_AwaitALIGN to SP20:SATA_AdjustSpeed

This transition shall occur if this state receives an ALIGN Received (0) message before the Await ALIGN Timeout timer expires. The ALIGN Received (0) message indicates an ALIGN (0) was received at any of the physical link rates supported by this phy.

6.8.5.6 SP20:SATA_AdjustSpeed state

6.8.5.6.1 State description

This state waits for the SP transmitter to adjust to the same physical link rate of the ALIGNs that were detected by the receiver circuitry.

This state shall:

- 1) send a Set Rate message to the SP transmitter; and
- 2) repeatedly send Transmit D10.2 messages to the SP transmitter.

6.8.5.6.2 Transition SP20:SATA_AdjustSpeed to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.5.6.3 Transition SP20:SATA_AdjustSpeed to SP21:SATA_TransmitALIGN

This transition shall occur after receiving a Transmitter Ready message.

6.8.5.7 SP21:SATA_TransmitALIGN state

6.8.5.7.1 State description

This state shall repeatedly send Transmit ALIGN (0) messages to the SP transmitter.

6.8.5.7.2 Transition SP21:SATA_TransmitALIGN to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.5.7.3 Transition SP21:SATA_TransmitALIGN to SP22:SATA_PHY_Ready

This transition shall occur after receiving three consecutive Dword Received messages containing primitives other than ALIGN (0).

6.8.5.8 SP22:SATA_PHY_Ready state

6.8.5.8.1 State description

While in this state dwords from the link layer are transmitted at the negotiated physical link rate at the rate established in the previous state.

Upon entry into this state, this state shall:

- a) if the SP transmitter is transmitting at 1.5 Gbps, set the ResetStatus state machine variable to G1;
- b) if the SP transmitter is transmitting at 3 Gbps, set the ResetStatus state machine variable to G2; or
- c) if the SP transmitter is transmitting at 6 Gbps, set the ResetStatus state machine variable to G3.

This state shall send a Phy Layer Ready (SATA) confirmation to the link layer to indicate that the physical link has been brought up successfully in SATA mode.

This state waits for a COMINIT Detected message, a DWS Lost message, or a DWS Reset message.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.5.8.2 Transition SP22:SATA_PHY_Ready to SP0:OOB_COMINIT

This transition shall occur after:

- a) receiving a DWS Lost message, if this state does not send a Start DWS message;
- a) receiving a DWS Lost message followed by a COMINIT Detected message, if this state does not send a Start DWS message; or
- b) receiving a DWS Reset message.

This transition may but should not occur after receiving a COMINIT Detected message before receiving a DWS Lost message, or after receiving a COMINIT Detected message after sending a Start DWS message (i.e., the SP state machine should ignore COMINIT Detected messages unless the SP_DWS state machine has indicated loss of dword synchronization).

6.8.5.8.3 Transition SP22:SATA_PHY_Ready to SP23:SATA_PM_Partial

This transition shall occur after receiving an Enter Partial request.

6.8.5.8.4 Transition SP22:SATA_PHY_Ready to SP24:SATA_PM_Slumber

This transition shall occur after receiving an Enter Slumber request.

6.8.5.9 SP23:SATA_PM_Partial state

6.8.5.9.1 State description

This state waits for a COMWAKE Detected message or an Exit Partial request.

6.8.5.9.2 Transition SP23:SATA_PM_Partial to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

Before the transition, this state shall set the ResetStatus state machine variable to UNKNOWN.

6.8.5.9.3 Transition SP23:SATA_PM_Partial to SP16:SATA_COMWAKE

This transition shall occur after receiving a Exit Partial request.

6.8.5.9.4 Transition SP23:SATA_PM_Partial to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur after receiving a COMWAKE Detected message.

6.8.5.10 SP24:SATA_PM_Slumber state

6.8.5.10.1 State description

This state waits for a COMWAKE Detected message or an Exit Slumber request.

6.8.5.10.2 Transition SP24:SATA_PM_Slumber to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.10.3 Transition SP24:SATA_PM_Slumber to SP16:SATA_COMWAKE

6.8.5.10.4 Transition SP24:SATA_PM_Slumber to SP18:SATA_AwaitNoCOMWAKE

6.8.6 SATA port selector state SP25:SATA_PortSel

Figure 146 shows the SP25:SATA_PortSel state. This state controls transmission of the SATA port selection signal when a specified phy processes a Transmit SATA Port Selection Signal request.

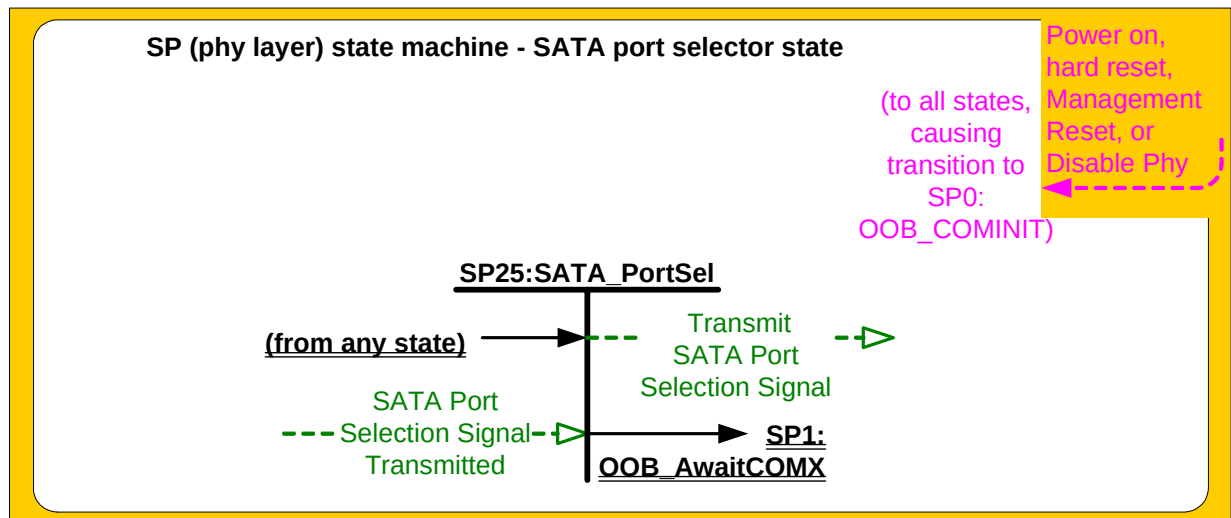


Figure 146 — SP (phy layer) state machine – SATA port selector state

- a) set the ResetStatus state machine variable to UNKNOWN;
- b) send a Transmit SATA Port Selection Signal message to the SP transmitter;
- c) set the ATTACHED SATA PORT SELECTOR bit to zero in the SMP DISCOVER response (see 10.4.3.9); and
- d) set the ATTACHED SATA DEVICE bit to zero in the SMP DISCOVER response.

This transition shall occur after receiving a SATA Port Selection Signal Transmitted message.

6.8.7 SATA spinup hold state SP26:SATA_SpinupHold

6.8.7.1 State description

Figure 147 shows the SP26:SATA_SpinupHold state.

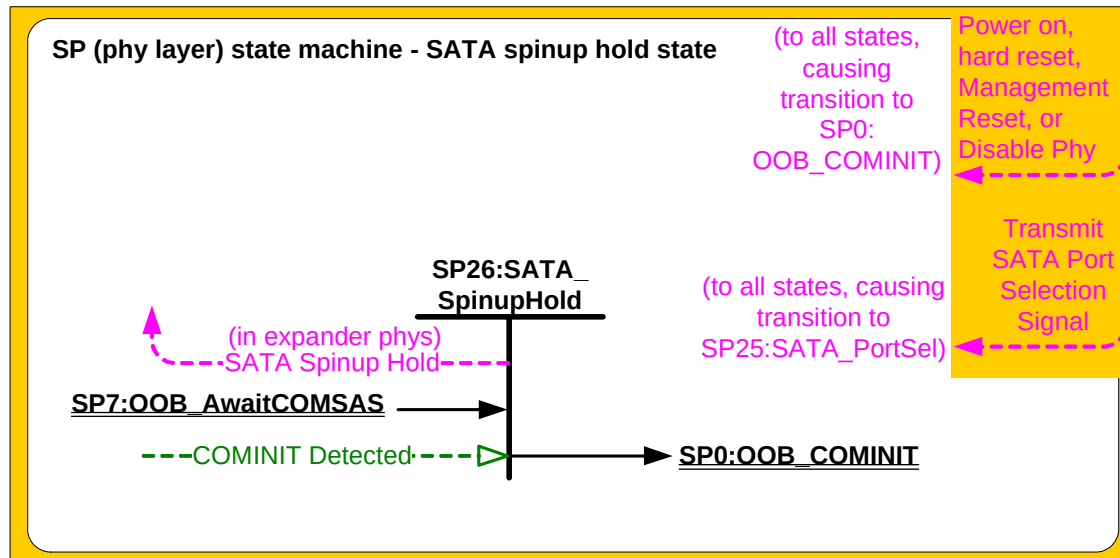


Figure 147 — SP (phy layer) state machine - SATA spinup hold state

Upon entry into this state, this state shall:

- a) if the ResetStatus state machine variable is set to SPINUP_HOLD, not change the ResetStatus state machine variable; or
- b) if the ResetStatus state machine variable is not set to SPINUP_HOLD:
 - A) set the ResetStatus state machine variable to SPINUP_HOLD; and
 - B) if this state machine is in an expander phy, send a SATA Spinup Hold confirmation to the link layer.

6.8.7.2 Transition SP26:SATA_SpinupHold to SP0:OOB_COMINIT

This transition shall occur if this state receives a COMINIT Detected message.

6.9 SP_DWS (phy layer dword synchronization) state machine

6.9.1 SP_DWS state machine overview

Each phy includes an SP_DWS state machine and an SP_DWS receiver.

The SP_DWS state machine establishes the same dword boundaries at the receiver as at the attached transmitter by searching for control characters. The SP_DWS receiver monitors and decodes the incoming data stream and forces K28.5 characters into the first character position to effectively perform dword alignment when requested by the SP_DWS state machine. K28.5 characters with either positive or negative disparity shall be accepted. The SP_DWS receiver continues to reestablish dword alignment by forcing received K28.5 characters into the first character position until a K28.5-based primitive (i.e., K28.5, Dxx.y, Dxx.y, Dxx.y) with correct disparity on each data character is detected. The resultant primitives, dwords and valid dword indicators (e.g., encoding error indicators) are sent to this state machine to enable it to determine the dword synchronization policy.

After dword synchronization has been achieved, this state machine monitors invalid dwords that are received. When an invalid dword is detected, it requires two valid dwords to nullify its effect. When four invalid dwords are detected without nullification, dword synchronization is considered lost.

While dword synchronization is lost, the data stream received is invalid and dwords shall not be passed to the link layer.

This state machine consists of the following states:

- a) SP_DWS0:AcquireSync (see 6.9.3)(initial state);
- b) SP_DWS1:Valid1 (see 6.9.4);
- c) SP_DWS2:Valid2 (see 6.9.5);
- d) SP_DWS3:SyncAcquired (see 6.9.6);
- e) SP_DWS4:Lost1 (see 6.9.7);
- f) SP_DWS5:Lost1Recovered (see 6.9.8);
- g) SP_DWS6:Lost2 (see 6.9.9);
- h) SP_DWS7:Lost2Recovered (see 6.9.10);
- i) SP_DWS8:Lost3 (see 6.9.11); and
- j) SP_DWS9:Lost3Recovered (see 6.9.12).

This state machine shall start in the SP_DWS0:AcquireSync state after:

- a) power on;
- b) hard reset;
- c) receiving a Management Reset request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET in an expander device);
- d) receiving a Disable Phy request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of DISABLE in an expander device); or
- e) receiving a Stop DWS message from the SP state machine.

This state machine receives the following messages from the SP state machine (see 6.8):

- a) Start DWS; and
- b) Stop DWS.

This state machine sends the following messages to the SP state machine:

- a) DWS Lost; and
- b) DWS Reset.

The SP_DWS state machine shall maintain the timers listed in table 97.

Table 97 — SP_DWS timers

Timer	Initial value
DWS Reset Timeout timer	1 ms

The SP_DWS receiver also sends Dword Received confirmations to the link layer state machine receivers (e.g., SL_IR, SL, SSP, SMP, and XL). If multiplexing is enabled (see table 92 in 6.7.4.2.3.3), the SP_DWS receiver shall use the first incoming MUX primitive to determine the logical phy to which it sends each Dword Received confirmation and shall not send an Dword Received confirmations until it receives the first incoming MUX primitive.

Editor's Note 34: SP_DWS doesn't send Dword Received confirmations upstream until the phy reset sequence is completed. Some feedback from SP is needed to know when this happens.

Upon receiving a Find Dword message, the SP_DWS receiver shall monitor the input data stream and force each K28.5 character detected into the first character position of a possible dword. If the next three characters are data characters with correct disparity, it shall send the dword as a Dword Received (Primitive) message to the SP_DWS state machine. Until it receives another Find Dword message, for every four characters it receives it shall:

- a) send a Dword Received (Invalid) message to the SP_DWS state machine if the dword is an invalid dword (see 3.1.112);
- b) send a Dword Received (Primitive) message to the SP_DWS state machine if the dword is a primitive (i.e., the dword contains a K28.5 character in the first character position followed by three data characters); or
- c) send a Dword Received (Data Dword) message to the SP_DWS state machine if the dword is a data dword (i.e., it is not an invalid dword or a primitive).

The SP_DWS receiver relationship to other receivers is defined in 4.3.3.

6.9.3 SP_DWS0:AcquireSync state

6.9.3.1 State description

This is the initial state of this state machine.

After receiving a Start DWS message, this state shall:

- a) send a Find Dword message to the SP_DWS receiver; and
- b) initialize and start the DWS Reset Timeout timer;

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2, this state shall send a Find Dword message to the SP_DWS receiver. and the DWS Reset Timeout timer shall continue running.

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2 and the DWS Reset Timeout timer has expired, this state may send a DWS Reset message to the SP state machine (e.g., if the phy chooses to initiate a new link reset sequence because dword synchronization has been lost for too long).

This state shall not send a DWS Reset message to the SP until the DWS Reset Timeout timer expires.

If the DWS Reset Timeout timer expires, this state may send a DWS Reset message to the SP state machine.

6.9.3.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1

This transition shall occur after sending a Find Dword message and receiving a Dword Received (Primitive) message.

6.9.4 SP_DWS1:Valid1 state

6.9.4.1 State description

This state is reached after one valid primitive has been received. This state waits for a second valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.9.4.2 Transition SP_DWS1:Valid1 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.9.4.3 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2

This transition shall occur after receiving a Dword Received (Primitive) message.

6.9.5 SP_DWS2:Valid2 state**6.9.5.1 State description**

This state is reached after two valid primitives have been received without adjusting the dword synchronization. This state waits for a third valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.9.5.2 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.9.5.3 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (Primitive) message.

6.9.6 SP_DWS3:SyncAcquired state**6.9.6.1 State description**

This state is reached after three valid primitives have been received without adjusting the dword synchronization.

The most recently received primitive and all subsequent dwords shall be forwarded for processing by the link layer.

This state waits for a Dword Received (Invalid) message, which indicates that dword synchronization might be lost, or an Incorrect Mux Received message, which indicates that dword synchronization is lost.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.6.2 Transition SP_DWS3:SyncAcquired to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.6.3 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.7 SP_DWS4:Lost1 state**6.9.7.1 State description**

This state is reached when one invalid dword has been received and not nullified. This state waits for a Dword Received message or an Incorrect Mux Received message.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.7.2 Transition SP_DWS4:Lost1 to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.7.3 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.7.4 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.8 SP_DWS5:Lost1Recovered state**6.9.8.1 State description**

This state is reached when a valid dword has been received after one invalid dword had been received. This state waits for a Dword Received message or an Incorrect Mux Received message.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.8.2 Transition SP_DWS5:Lost1Recovered to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.8.3 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.8.4 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.9 SP_DWS6:Lost2 state**6.9.9.1 State description**

This state is reached when two invalid dwords have been received and not nullified. This state waits for a Dword Received message or an Incorrect Mux Received message.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.9.2 Transition SP_DWS6:Lost2 to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.9.3 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.9.4 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.10 SP_DWS7:Lost2Recovered state**6.9.10.1 State description**

This state is reached when a valid dword has been received after two invalid dwords had been received. This state waits for a Dword Received message or an Incorrect Mux Received message.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.10.2 Transition SP_DWS7:Lost2Recovered to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.10.3 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.10.4 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.11 SP_DWS8:Lost3 state**6.9.11.1 State description**

This state is reached when three invalid dwords have been received and not nullified. This state waits for a Dword Received message or an Incorrect Mux Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.11.2 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.11.3 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.12 SP_DWS9:Lost3Recovered state**6.9.12.1 State description**

This state is reached when a valid dword has been received after three invalid dwords had been received. This state waits for a Dword Received message or an Incorrect Mux Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

If an Incorrect Mux Received message is received, this state shall send a DWS Lost message to the SP state machine.

6.9.12.2 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.12.3 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.10 Multiplexing

If SNW-3 indicates multiplexing is enabled (see table 92 in 6.7.4.2.3.3), the phy shall begin multiplexing immediately after the multiplexing sequence (see 6.7.4.3).

Figure 149 shows multiplexing disabled (i.e., one logical link).

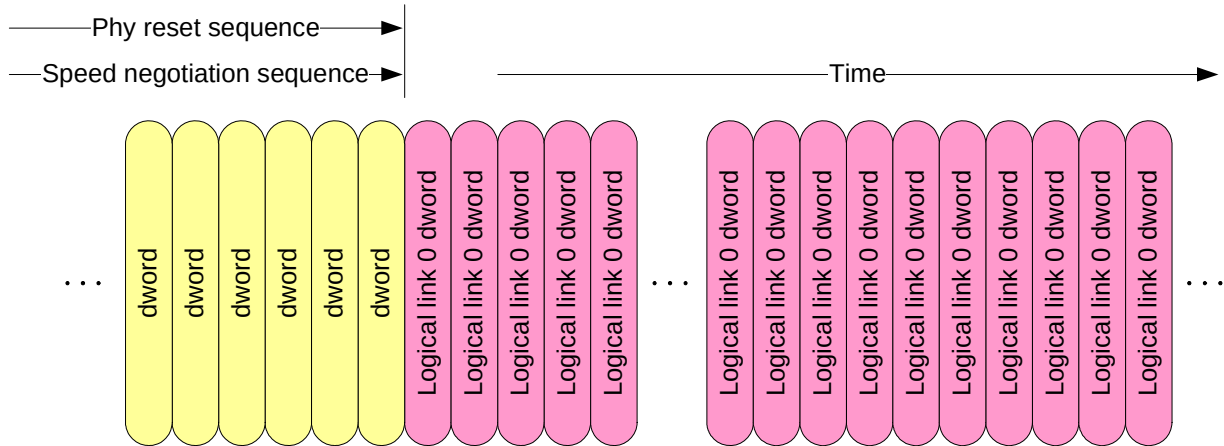


Figure 149 — Multiplexing disabled

Figure 150 shows multiplexing enabled (i.e., two logical links).

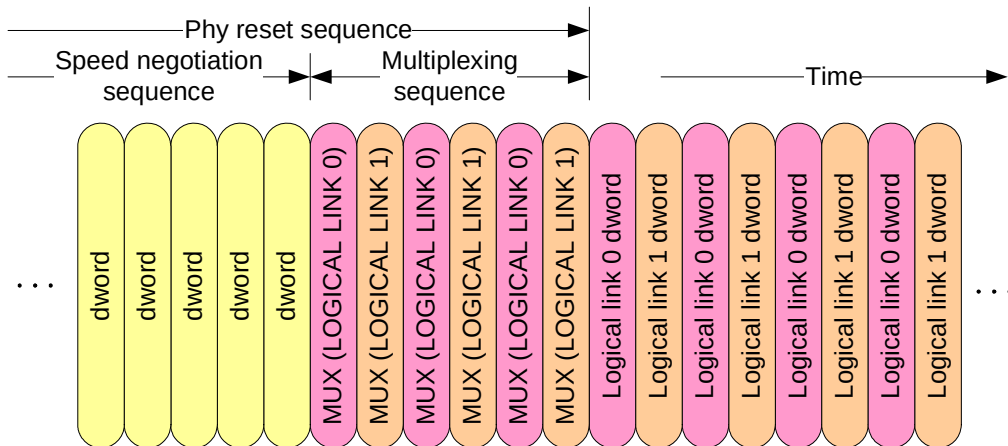


Figure 150 — Multiplexing enabled

After the multiplexing sequence completes, each logical phy shall honor the deletable primitive insertion requirements for physical link rate tolerance management defined in 7.3. The logical phys shall ignore MUX primitives.

If a phy with multiplexing enabled ever loses dword synchronization, it shall restart the link reset sequence rather than attempt to reestablish dword synchronization.

Once the multiplexing sequence is complete, the phy shall not perform another multiplexing sequence until a new link reset sequence.

Once the multiplexing sequence is complete:

- a logical phy originating dwords shall transmit MUX as a deletable primitive (e.g., in place on an ALIGN) at least once every millisecond; and
- a logical phy forwarding dwords should transmit MUX as a deletable primitive at least once every millisecond to confirm the logical link numbers.

Transmitting NOTIFY has higher priority than transmitting MUX.

NOTE 33 - Periodic MUX transmission is for the convenience of logic analyzers and to provide additional assurance that the receiving phy is in agreement with the transmitting phy.

If a phy ever receives a MUX primitive that does not match the MUX primitive expected in that position (i.e., it receives MUX (LOGICAL LINK 1) on logical link 0 or receives MUX (LOGICAL LINK 0) on logical link 1), it shall perform a link reset sequence.

6.11 Spin-up

If a SAS target device receives COMSAS during the reset sequence, it shall not spin-up until allowed by the SA_PC state machine (see 10.2.10).

Expander devices that detect an attached SATA phy may halt the automatic phy reset sequence after the COMSAS Detect Timeout (see 6.8) to delay spin-up; this is called SATA spinup hold. This is reported in the NEGOTIATED PHYSICAL LINK RATE field in the SMP DISCOVER response (see 10.4.3.9) and is released with the SMP PHY CONTROL function (see 10.4.3.27).

NOTE 34 - Enclosures supporting both SATA devices and SAS target devices may need to sequence power to each attached device to avoid excessive power consumption during power on, since the SATA devices may spin-up automatically after power on if staggered spin-up is not implemented (see SATA-2).

7 Link layer

7.1 Link layer overview

The link layer defines primitives, address frames, and connections. Link layer state machines interface to the port layer and the phy layer and perform the identification and hard reset sequences, connection management, and SSP, STP, and SMP specific frame transmission and reception.

7.2 Primitives

7.2.1 Primitives overview

Primitives are dwords whose first character is a K28.3 or K28.5. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last characters. Table 98 defines the primitive format.

Table 98 — Primitive format

Character	Description
First	K28.5 control character (for primitives defined in this standard) or K28.3 control character (for primitives defined by SATA).
Second	Constant data character.
Third	Constant data character.
Last	Constant data character.

7.2.2 Primitive summary

Table 99 defines the deletable primitives.

Table 99 — Deletable primitives

Primitive	Use ^a	From ^b			To ^b			Primitive sequence type ^c
		I	E	T	I	E	T	
ALIGN (0)	All, SpNeg	I	E	T	I	E	T	Single
ALIGN (1)								
ALIGN (2)	All							
ALIGN (3)								
MUX (LOGICAL LINK 0)	All	I	E	T	I	E	T	Single
MUX (LOGICAL LINK 1)								
NOTIFY (ENABLE SPINUP)	All	I	E				T	Single
NOTIFY (POWER LOSS EXPECTED)		I	E				T	
NOTIFY (RESERVED 1)					I	E	T	
NOTIFY (RESERVED 2)					I	E	T	

^a The Use column indicates when the primitive is used:

- a) NoConn: SAS logical links, outside connections;
- b) Conn: SAS logical links, inside connections;
- c) All: SAS logical links, both outside connections or inside any type of connection;
- d) STP: SAS logical links, inside STP connections; or
- e) SpNeg: SAS physical links, during speed negotiation.

^b The From and To columns indicate the type of ports that originate each primitive or are the intended destinations of each primitive:

- a) I for SAS initiator ports;
- b) E for expander ports; and
- c) T for SAS target ports.

Expander ports are not considered originators of primitives that are passing through from expander port to expander port.

^c The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).

Table 100 defines the primitives not specific to the type of connection.

Table 100 — Primitives not specific to type of connection (part 1 of 2)

Primitive	Use ^a	From ^b			To ^b			Primitive sequence type ^c
		I	E	T	I	E	T	
AIP (NORMAL)	NoConn		E					Extended
AIP (RESERVED 0)								
AIP (RESERVED 1)								
AIP (RESERVED 2)								
AIP (RESERVED WAITING ON PARTIAL)					I	E	T	
AIP (WAITING ON CONNECTION)			E					
AIP (WAITING ON DEVICE)			E					
AIP (WAITING ON PARTIAL)			E					
BREAK	All	I	E	T	I	E	T	Redundant
BREAK_REPLY	All	I	E	T	I	E	T	Redundant
BROADCAST (CHANGE)	NoConn	I	E		I			Redundant
BROADCAST (SES)				T	I			
BROADCAST (EXPANDER)			E		I			
BROADCAST (ASYNCHRONOUS EVENT)				T	I			
BROADCAST (RESERVED 3)								
BROADCAST (RESERVED 4)								
BROADCAST (RESERVED CHANGE 0)					I			
BROADCAST (RESERVED CHANGE 1)					I			
CLOSE (CLEAR AFFILIATION)	STP	I					T	Triple
CLOSE (NORMAL)	Conn	I		T				
CLOSE (RESERVED 0)					I		T	
CLOSE (RESERVED 1)								
EOAF	NoConn	I	E	T	I	E	T	Single
ERROR	All		E		I	E	T	Single
HARD_RESET	NoConn	I	E		I	E	T	Redundant
OPEN_ACCEPT	NoConn	I		T	I		T	Single

Table 100 — Primitives not specific to type of connection (part 2 of 2)

Primitive	Use ^a	From ^b			To ^b			Primitive sequence type ^c		
		I	E	T	I	E	T			
OPEN_REJECT (BAD DESTINATION)	NoConn		E		I		T	Single		
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)		I	E	T						
OPEN_REJECT (NO DESTINATION)			E							
OPEN_REJECT (PATHWAY BLOCKED)			E							
OPEN_REJECT (PROTOCOL NOT SUPPORTED)		I		T						
OPEN_REJECT (RESERVED ABANDON 1)										
OPEN_REJECT (RESERVED ABANDON 2)										
OPEN_REJECT (RESERVED ABANDON 3)										
OPEN_REJECT (RESERVED CONTINUE 0)										
OPEN_REJECT (RESERVED CONTINUE 1)										
OPEN_REJECT (RESERVED INITIALIZE 0)										
OPEN_REJECT (RESERVED INITIALIZE 1)										
OPEN_REJECT (RESERVED STOP 0)										
OPEN_REJECT (RESERVED STOP 1)										
OPEN_REJECT (RETRY)		I	E	T						
OPEN_REJECT (STP RESOURCES BUSY)			E	T					I	
OPEN_REJECT (WRONG DESTINATION)		I		T						T
OPEN_REJECT (ZONE VIOLATION)			E						I	
SOAF	NoConn	I	E	T	I	E	T	Single		
TRAIN	SpNeg	I	E	T	I	E	T	Redundant		
TRAIN_DONE										
^a The Use column indicates when the primitive is used: a) NoConn: SAS logical links, outside connections; b) Conn: SAS logical links, inside connections; c) All: SAS logical links, both outside connections or inside any type of connection; d) STP: SAS logical links, inside STP connections; or e) SpNeg: SAS physical links, during speed negotiation. ^b The From and To columns indicate the type of ports that originate each primitive or are the intended destinations of each primitive: a) I for SAS initiator ports; b) E for expander ports; and c) T for SAS target ports. Expander ports are not considered originators of primitives that are passing through from expander port to expander port. ^c The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, an extended primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).										

Table 101 defines the primitives used only inside SSP and SMP connections.

Table 101 — Primitives used only inside SSP and SMP connections

Primitive	Use ^a	From ^b			To ^b			Primitive sequence type ^c
		I	E	T	I	E	T	
ACK	SSP	I		T	I		T	Single
CREDIT_BLOCKED	SSP	I		T	I		T	Single
DONE (ACK/NAK TIMEOUT)	SSP	I		T	I		T	Single
DONE (CREDIT TIMEOUT)		I		T				
DONE (NORMAL)		I		T				
DONE (RESERVED 0)								
DONE (RESERVED 1)								
DONE (RESERVED TIMEOUT 0)								
DONE (RESERVED TIMEOUT 1)								
EOF	SSP, SMP	I		T	I		T	Single
NAK (CRC ERROR)	SSP	I		T	I		T	Single
NAK (RESERVED 0)								
NAK (RESERVED 1)								
NAK (RESERVED 2)								
RRDY (NORMAL)	SSP	I		T	I		T	Single
RRDY (RESERVED 0)								
RRDY (RESERVED 1)								
SOF	SSP, SMP	I		T	I		T	Single
^a The Use column indicates when the primitive is used: a) SSP: SAS logical links, inside SSP connections; or b) SMP: SAS logical links, inside SMP connections. ^b The From and To columns indicate the type of ports that originate each primitive or are the intended destinations of each primitive: a) I for SSP initiator ports and SMP initiator ports; b) E for expander ports; and c) T for SSP target ports and SMP target ports. Expander ports are not considered originators of primitives that are passing through from expander port to expander port. ^c The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, an extended primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).								

Table 102 lists the primitives used only inside STP connections and on SATA physical links.

Table 102 — Primitives used only inside STP connections and on SATA physical links

Primitive	Use ^a	From ^b			To ^b			Primitive sequence type ^c
		I	E	T	I	E	T	
SATA_CONT	STP, SATA	I		T	I		T	Single
SATA_DMAT	STP, SATA	I		T	I		T	Single
SATA_EOF	STP, SATA	I		T	I		T	Single
SATA_ERROR ^d	SATA		E				T	Single
SATA_HOLD	STP, SATA	I		T	I		T	Continued
SATA_HOLDA	STP, SATA	I		T	I		T	Continued
SATA_PMACK	STP, SATA							Repeated
SATA_PMNAK	STP, SATA	I	E				T	Repeated
SATA_PMREQ_P	STP, SATA							Continued
SATA_PMREQ_S	STP, SATA							Continued
SATA_R_ERR	STP, SATA	I		T	I		T	Continued
SATA_R_IP	STP, SATA	I		T	I		T	Continued
SATA_R_OK	STP, SATA	I		T	I		T	Continued
SATA_R_RDY	STP, SATA	I		T	I		T	Continued
SATA_SOF	STP, SATA	I		T	I		T	Single
SATA_SYNC	STP, SATA	I		T	I		T	Continued
SATA_WTRM	STP, SATA	I		T	I		T	Continued
SATA_X_RDY	STP, SATA	I		T	I		T	Continued

^a The Use column indicates when the primitive is used:

a) STP: SAS logical links, inside STP connections; or

b) SATA: SATA physical links.

^b The From and To columns indicate the type of ports that originate each primitive or are the intended destinations of each primitive:

a) I for STP initiator ports and SATA host ports;

b) E for expander ports; and

c) T for STP target ports and SATA device ports.

Expander ports are not considered originators of primitives that are passing through from expander port to expander port.

^c The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, an extended primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).

^d Although included in this table, SATA_ERROR is not a primitive (see 3.1.169) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword, used by expander devices forwarding an error onto a SATA physical link (see 7.2.7.1).

7.2.3 Primitive encodings

Table 103 defines the primitive encoding for deletable primitives.

Table 103 — Primitive encoding for deletable primitives

Primitive	Character				Hexadecimal
	1 st	2 nd	3 rd	4 th (last)	
ALIGN (0)	K28.5	D10.2	D10.2	D27.3	BC4A4A7Bh
ALIGN (1)	K28.5	D07.0	D07.0	D07.0	BC070707h
ALIGN (2)	K28.5	D01.3	D01.3	D01.3	BC616161h
ALIGN (3)	K28.5	D27.3	D27.3	D27.3	BC7B7B7Bh
MUX (LOGICAL LINK 0)	K28.5	D02.0	D16.7	D31.4	BC02F09Fh
MUX (LOGICAL LINK 1)	K28.5	D04.7	D31.4	D27.4	BCE49F9Bh
NOTIFY (ENABLE SPINUP)	K28.5	D31.3	D31.3	D31.3	BC7F7F7Fh
NOTIFY (POWER LOSS EXPECTED)	K28.5	D31.3	D07.0	D01.3	BC7F0761h
NOTIFY (RESERVED 1)	K28.5	D31.3	D01.3	D07.0	BC7F6107h
NOTIFY (RESERVED 2)	K28.5	D31.3	D10.2	D10.2	BC7F4A4Ah

Table 104 defines the primitive encoding for primitives not specific to type of connection.

Table 104 — Primitive encoding for primitives not specific to type of connection (part 1 of 2)

Primitive	Character				Hexadecimal
	1 st	2 nd	3 rd	4 th (last)	
AIP (NORMAL)	K28.5	D27.4	D27.4	D27.4	BC9B9B9Bh
AIP (RESERVED 0)	K28.5	D27.4	D31.4	D16.7	BC9B9FF0h
AIP (RESERVED 1)	K28.5	D27.4	D16.7	D30.0	BC9BF01Eh
AIP (RESERVED 2)	K28.5	D27.4	D29.7	D01.4	BC9BFD81h
AIP (RESERVED WAITING ON PARTIAL)	K28.5	D27.4	D01.4	D07.3	BC9B8167h
AIP (WAITING ON CONNECTION)	K28.5	D27.4	D07.3	D24.0	BC9B6718h
AIP (WAITING ON DEVICE)	K28.5	D27.4	D30.0	D29.7	BC9B1EFDh
AIP (WAITING ON PARTIAL)	K28.5	D27.4	D24.0	D04.7	BC9B18E4h
BREAK	K28.5	D02.0	D24.0	D07.3	BC021867h
BREAK_REPLY	K28.5	D02.0	D29.7	D16.7	BC02FDF0h
BROADCAST (CHANGE)	K28.5	D04.7	D02.0	D01.4	BCE40281h
BROADCAST (SES)	K28.5	D04.7	D07.3	D29.7	BCE467FDh
BROADCAST (EXPANDER)	K28.5	D04.7	D01.4	D24.0	BCE48118h
BROADCAST (ASYNCHRONOUS EVENT)	K28.5	D04.7	D04.7	D04.7	BCE4E4E4h
BROADCAST (RESERVED 3)	K28.5	D04.7	D16.7	D02.0	BCE4F002h
BROADCAST (RESERVED 4)	K28.5	D04.7	D29.7	D30.0	BCE4FD1Eh
BROADCAST (RESERVED CHANGE 0)	K28.5	D04.7	D24.0	D31.4	BCE4189Fh
BROADCAST (RESERVED CHANGE 1)	K28.5	D04.7	D27.4	D07.3	BCE49B67h
CLOSE (CLEAR AFFILIATION)	K28.5	D02.0	D07.3	D04.7	BC0267E4h
CLOSE (NORMAL)	K28.5	D02.0	D30.0	D27.4	BC021E9Bh
CLOSE (RESERVED 0)	K28.5	D02.0	D31.4	D30.0	BC029F1Eh
CLOSE (RESERVED 1)	K28.5	D02.0	D04.7	D01.4	BC02E481h
EOAF	K28.5	D24.0	D07.3	D31.4	BC18679Fh

Table 104 — Primitive encoding for primitives not specific to type of connection (part 2 of 2)

Primitive	Character				Hexadecimal
	1 st	2 nd	3 rd	4 th (last)	
ERROR	K28.5	D02.0	D01.4	D29.7	BC0281FDh
HARD_RESET	K28.5	D02.0	D02.0	D02.0	BC020202h
OPEN_ACCEPT	K28.5	D16.7	D16.7	D16.7	BCF0F0F0h
OPEN_REJECT (BAD DESTINATION)	K28.5	D31.4	D31.4	D31.4	BC9F9F9Fh
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	K28.5	D31.4	D04.7	D29.7	BC9FE4FDh
OPEN_REJECT (NO DESTINATION)	K28.5	D29.7	D29.7	D29.7	BCFD FDFDh
OPEN_REJECT (PATHWAY BLOCKED)	K28.5	D29.7	D16.7	D04.7	BCFD F0E4h
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	K28.5	D31.4	D29.7	D07.3	BC9FFD67h
OPEN_REJECT (RESERVED ABANDON 1)	K28.5	D31.4	D30.0	D16.7	BC9F1EF0h
OPEN_REJECT (RESERVED ABANDON 2)	K28.5	D31.4	D07.3	D02.0	BC9F6702h
OPEN_REJECT (RESERVED ABANDON 3)	K28.5	D31.4	D01.4	D30.0	BC9F811Eh
OPEN_REJECT (RESERVED CONTINUE 0)	K28.5	D29.7	D02.0	D30.0	BCFD021Eh
OPEN_REJECT (RESERVED CONTINUE 1)	K28.5	D29.7	D24.0	D01.4	BCFD1881h
OPEN_REJECT (RESERVED INITIALIZE 0)	K28.5	D29.7	D30.0	D31.4	BCFD1E9Fh
OPEN_REJECT (RESERVED INITIALIZE 1)	K28.5	D29.7	D07.3	D16.7	BCFD67F0h
OPEN_REJECT (RESERVED STOP 0)	K28.5	D29.7	D31.4	D07.3	BCFD9F67h
OPEN_REJECT (RESERVED STOP 1)	K28.5	D29.7	D04.7	D27.4	BCFDE49Bh
OPEN_REJECT (RETRY)	K28.5	D29.7	D27.4	D24.0	BCFD9B18h
OPEN_REJECT (STP RESOURCES BUSY)	K28.5	D31.4	D27.4	D01.4	BC9F9B81h
OPEN_REJECT (WRONG DESTINATION)	K28.5	D31.4	D16.7	D24.0	BC9FF018h
OPEN_REJECT (ZONE VIOLATION)	K28.5	D31.4	D02.0	D27.4	BC9F029Bh
SOAF	K28.5	D24.0	D30.0	D01.4	BC181E81h
TRAIN	K28.5	D30.3	D30.3	D30.3	BC7E7E7Eh
TRAIN_DONE	K28.5	D30.3	D30.3	D10.2	BC7E7E4Ah

Table 105 defines the primitive encodings for primitives used only inside SSP and SMP connections.

Table 105 — Primitive encoding for primitives used only inside SSP and SMP connections

Primitive	Character				Hexadecimal
	1 st	2 nd	3 rd	4 th (last)	
ACK	K28.5	D01.4	D01.4	D01.4	BC818181h
CREDIT_BLOCKED	K28.5	D01.4	D07.3	D30.0	BC81671Eh
DONE (ACK/NAK TIMEOUT)	K28.5	D30.0	D01.4	D04.7	BC1E81E4h
DONE (CREDIT TIMEOUT)	K28.5	D30.0	D07.3	D27.4	BC1E679Bh
DONE (NORMAL)	K28.5	D30.0	D30.0	D30.0	BC1E1E1Eh
DONE (RESERVED 0)	K28.5	D30.0	D16.7	D01.4	BC1EF081h
DONE (RESERVED 1)	K28.5	D30.0	D29.7	D31.4	BC1EFD9Fh
DONE (RESERVED TIMEOUT 0)	K28.5	D30.0	D27.4	D29.7	BC1E9BFDh
DONE (RESERVED TIMEOUT 1)	K28.5	D30.0	D31.4	D24.0	BC1E9F18h
EOF	K28.5	D24.0	D16.7	D27.4	BC18F09Bh
NAK (CRC ERROR)	K28.5	D01.4	D27.4	D04.7	BC819BE4h
NAK (RESERVED 0)	K28.5	D01.4	D31.4	D29.7	BC819FFDh
NAK (RESERVED 1)	K28.5	D01.4	D04.7	D24.0	BC81E418h
NAK (RESERVED 2)	K28.5	D01.4	D16.7	D07.3	BC81F067h
RRDY (NORMAL)	K28.5	D01.4	D24.0	D16.7	BC8118F0h
RRDY (RESERVED 0)	K28.5	D01.4	D02.0	D31.4	BC81029Fh
RRDY (RESERVED 1)	K28.5	D01.4	D30.0	D02.0	BC811E02h
SOF	K28.5	D24.0	D04.7	D07.3	BC18E467h

Table 106 lists the primitive encodings for primitives used only inside STP connections and on SATA physical links.

Table 106 — Primitive encoding for primitives used only inside STP connections and on SATA physical links

Primitive	Character				Hexadecimal
	1 st	2 nd	3 rd	4 th (last)	
SATA_CONT	K28.3	D10.5	D25.4	D25.4	7CAA9999h
SATA_DMAT	K28.3	D21.5	D22.1	D22.1	7CB53636h
SATA_EOF	K28.3	D21.5	D21.6	D21.6	7CB5D5D5h
SATA_ERROR ^{a b}	K28.6	D02.0	D01.4	D29.7	DC0281FDh
SATA_HOLD	K28.3	D10.5	D21.6	D21.6	7CAAD5D5h
SATA_HOLDA	K28.3	D10.5	D21.4	D21.4	7CAA9595h
SATA_PMACK	K28.3	D21.4	D21.4	D21.4	7C959595h
SATA_PMNAK	K28.3	D21.4	D21.7	D21.7	7C95F5F5h
SATA_PMREQ_P	K28.3	D21.5	D23.0	D23.0	7CB51717h
SATA_PMREQ_S	K28.3	D21.4	D21.3	D21.3	7C957575h
SATA_R_ERR	K28.3	D21.5	D22.2	D22.2	7CB55656h
SATA_R_IP	K28.3	D21.5	D21.2	D21.2	7CB55555h
SATA_R_OK	K28.3	D21.5	D21.1	D21.1	7CB53535h
SATA_R_RDY	K28.3	D21.4	D10.2	D10.2	7C954A4Ah
SATA_SOF	K28.3	D21.5	D23.1	D23.1	7CB53737h
SATA_SYNC	K28.3	D21.4	D21.5	D21.5	7C95B5B5h
SATA_WTRM	K28.3	D21.5	D24.2	D24.2	7CB55858h
SATA_X_RDY	K28.3	D21.5	D23.2	D23.2	7CB55757h
^a Except for SATA_ERROR, all values are defined by SATA (see SATA-2). ^b Although included in this table, SATA_ERROR is not a primitive (see 3.1.169) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword, used by expander devices forwarding an error onto a SATA physical link (see 7.2.7.1).					

7.2.4 Primitive sequences

7.2.4.1 Primitive sequences overview

Table 107 summarizes the types of primitive sequences.

Table 107 — Primitive sequences

Primitive sequence type	Number of times the transmitter transmits the primitive to transmit the primitive sequence	Number of times the receiver receives the primitive to detect the primitive sequence	Reference
Single	1	1	7.2.4.2
Repeated	1 or more	1	7.2.4.3
Continued	2 followed by SATA_CONT	1	7.2.4.4
Extended	3	1	7.2.4.5
Triple	3	3	7.2.4.6
Redundant	6	3	7.2.4.7

Any number of deletable primitives may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements. Rate matching deletable primitives shall be sent inside primitive sequences inside of connections if rate matching is enabled (see 7.13).

7.2.4.2 Single primitive sequence

Primitives labeled as single primitive sequences (e.g., RRDY, SATA_SOF) shall be transmitted one time to form a single primitive sequence.

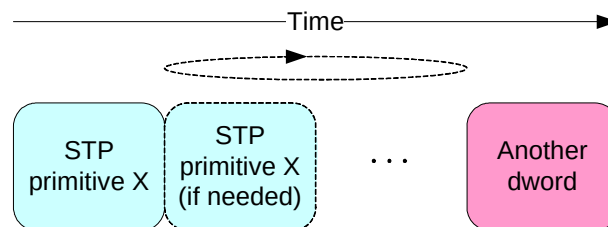
Receivers count each primitive received that is labeled as a single primitive sequence as a distinct single primitive sequence.

ALIGNs, NOTIFYs, and MUXs are called deletable primitives (see 7.3).

7.2.4.3 Repeated primitive sequence

Primitives that form repeated primitive sequences (e.g., SATA_PMACK) shall be transmitted one or more times. Only STP primitives form repeated primitive sequences. Any number of deletable primitives may be sent inside repeated primitive sequences as described in 7.2.4.1.

Figure 151 shows an example of transmitting a repeated primitive sequence.



NOTE: Another dword is a dword other than a deletable primitive or STP primitive X

Figure 151 — Transmitting a repeated primitive sequence

Receivers do not count the number of times a repeated primitive is received (i.e., receivers are simply in the state of receiving the primitive). An expander device forwarding a repeated primitive sequence may transmit more repeated primitives than it receives (i.e., expand) or transmit fewer repeated primitives than it receives (i.e., contract).

Figure 152 shows an example of receiving a repeated primitive sequence.

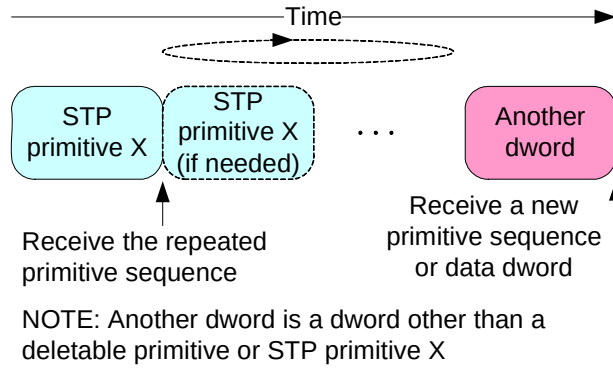


Figure 152 — Receiving a repeated primitive sequence

7.2.4.4 Continued primitive sequence

Primitives that form continued primitive sequences (e.g., SATA_HOLD) shall be transmitted as specified in 7.17.3. Any number of deletable primitives may be sent inside continued primitive sequences as described in 7.2.4.1.

7.2.4.5 Extended primitive sequence

Primitives that form extended primitive sequences (e.g., AIP) shall be transmitted three times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

A receiver shall detect an extended primitive sequence after the primitive is received one time. The receiver shall process an extended primitive sequence the same as a single primitive sequence (see 7.2.4.2).

Figure 153 shows examples of extended primitive sequences.

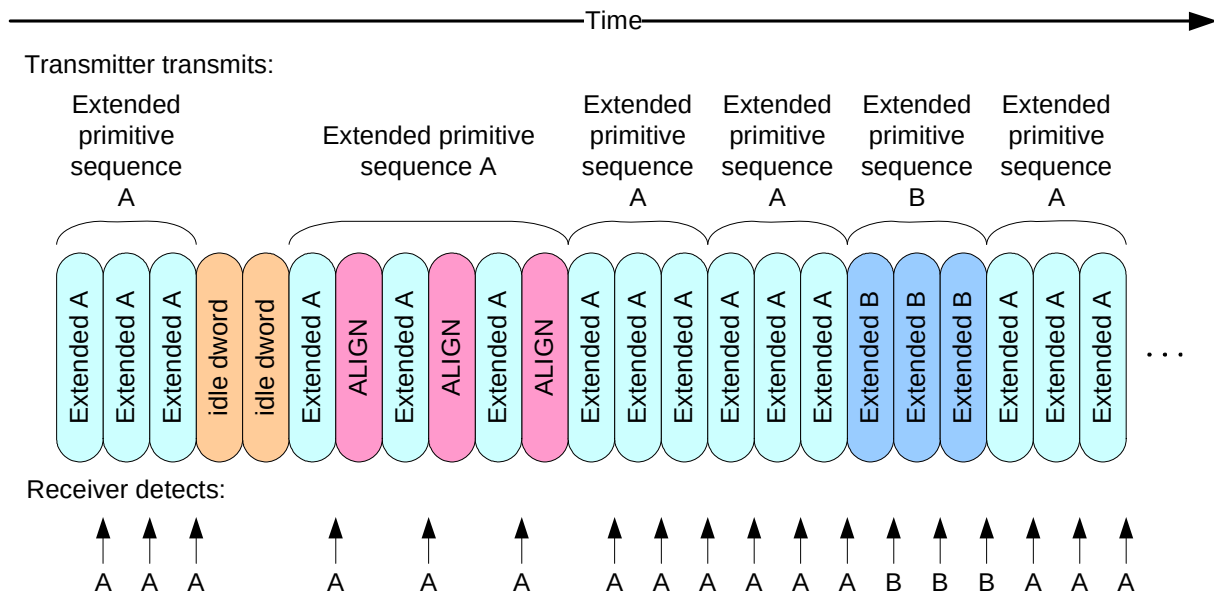


Figure 153 — Extended primitive sequences

7.2.4.6 Triple primitive sequence

Primitives that form triple primitive sequences (e.g., CLOSE (NORMAL)) shall be sent three times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

Receivers shall detect a triple primitive sequence after the identical primitive is received in three consecutive dwords. After detecting a triple primitive sequence, a receiver shall not detect a second instance of the same triple primitive sequence until it has received three consecutive dwords that are not any of the following:

- a) the original primitive; or
- b) a deletable primitive.

Figure 154 shows examples of triple primitive sequences.

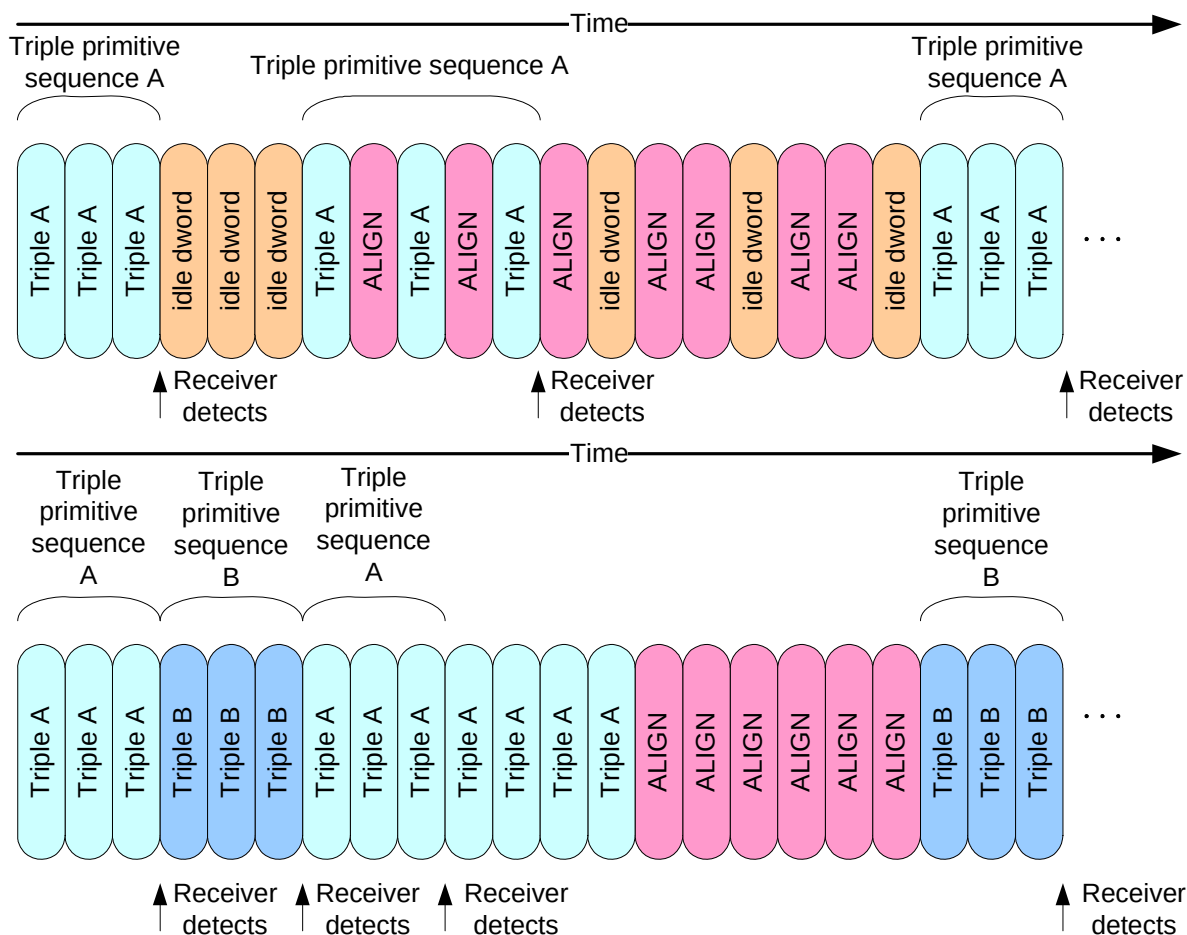


Figure 154 — Triple primitive sequences

7.2.4.7 Redundant primitive sequence

Primitives that form redundant primitive sequences (e.g., BROADCAST (CHANGE)) shall be sent six times consecutively. Any number of deletable primitives may be sent inside primitive sequences as described in 7.2.4.1.

A receiver shall detect a redundant primitive sequence after the identical primitive is received in any three of six consecutive dwords. After detecting a redundant primitive sequence, a receiver shall not detect a second instance of the same redundant primitive sequence until it has received six consecutive dwords that are not any of the following:

- a) the original primitive; or
- b) a deletable primitive.

Editor's Note 35: 07-089 attempts to clarify what happens if the receiver sees interleaved redundant primitives like A B A B A B. It should not be required to detect both nor required to ignore both, since the transmitter should never send such a pattern. Other patterns of interest: A B

C B C B C or A B C D A A. The latter implies there are up to 4 possible primitive sequences at any time that might be underway. It is probably simplest to define that after detecting redundant primitive A, ignore the next 5 dwords before starting to match any another primitive.

Figure 155 shows examples of redundant primitive sequences.

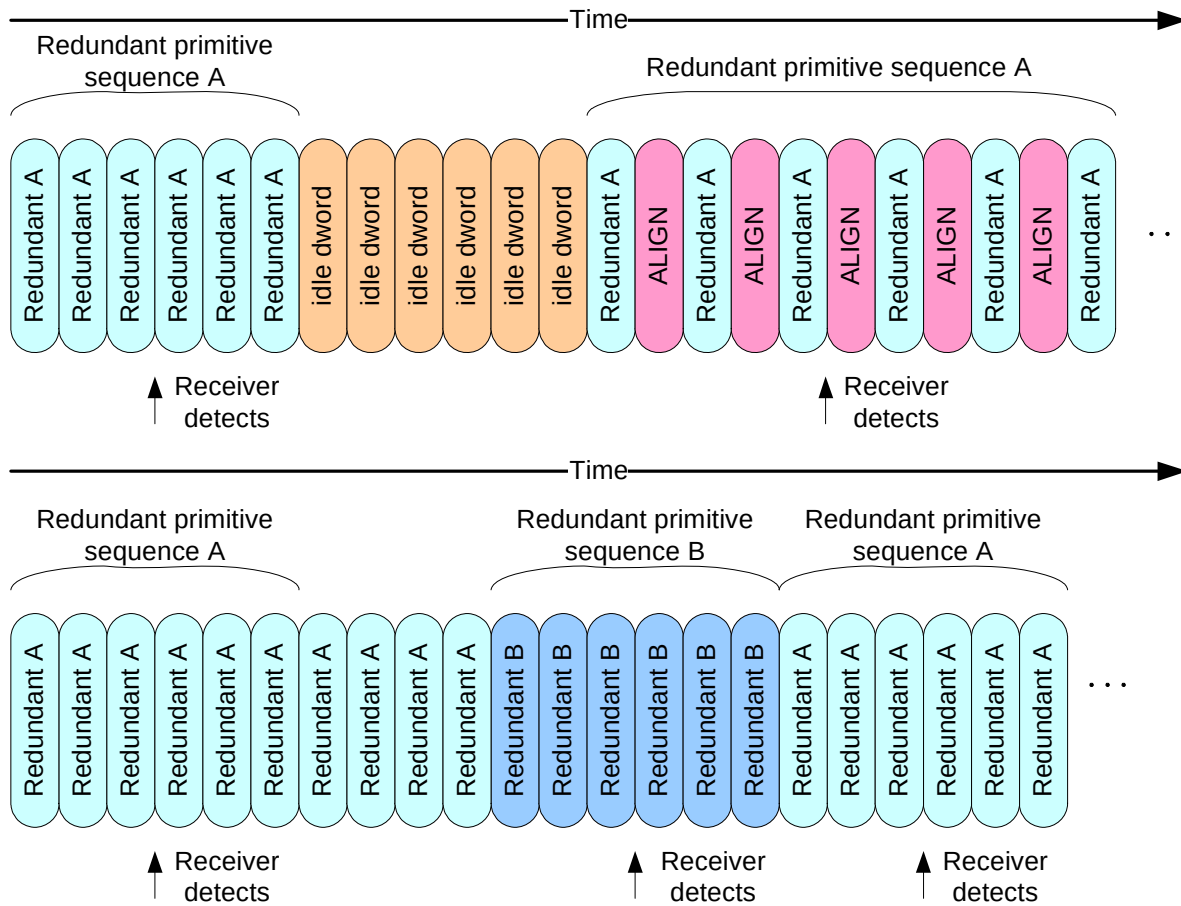


Figure 155 — Redundant primitive sequences

7.2.5 Primitives not specific to type of connections

7.2.5.1 AIP (Arbitration in progress)

AIP is sent by an expander device after a connection request to specify that the connection request is being processed and specify the status of the connection request.

The versions of AIP representing different statuses are defined in table 108.

Table 108 — AIP primitives

Primitive	Description
AIP (NORMAL)	Expander device has accepted the connection request. This may be sent multiple times (see 7.12.4.3).
AIP (RESERVED 0)	Reserved. Processed the same as AIP (NORMAL).
AIP (RESERVED 1)	
AIP (RESERVED 2)	
AIP (WAITING ON CONNECTION)	Expander device has determined the routing for the connection request, but either the destination phys are all being used for connections or there are insufficient routing resources to complete the connection request. This may be sent multiple times (see 7.12.4.3).
AIP (WAITING ON DEVICE)	Expander device has determined the routing for the connection request and forwarded it to the output physical link. This is sent one time (see 7.12.4.3).
AIP (WAITING ON PARTIAL)	Expander device has determined the routing for the connection request, but the destination phys are all busy with other partial pathways. This may be sent multiple times (see 7.12.4.3).
AIP (RESERVED WAITING ON PARTIAL)	Reserved. Processed the same as AIP (WAITING ON PARTIAL).

See 7.12 for details on connections.

7.2.5.2 ALIGN

ALIGNs are used for:

- a) OOB signals (see 6.6);
- b) character and dword alignment during the speed negotiation sequence (see 6.7.4.2);
- c) physical link rate tolerance management after the phy reset sequence (see 7.3); and
- d) rate matching during connections (see 7.13).

ALIGNs are deletable primitives (see 7.3).

Table 109 defines the different versions of ALIGN primitives.

Table 109 — ALIGN primitives

Primitive	Description
ALIGN (0)	Used for OOB signals, the speed negotiation sequence, physical link rate tolerance management, and rate matching.
ALIGN (1)	Used for the speed negotiation sequence, physical link rate tolerance management, and rate matching.
ALIGN (2)	Used for physical link rate tolerance management and rate matching.
ALIGN (3)	

Phys may use ALIGN (0) to construct OOB signals as described in 6.6. Phys use ALIGN (0) and ALIGN (1) during the speed negotiation sequence as described in 6.7.4.2. Phys shall rotate through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3) for all ALIGNs sent after the phy reset sequence.

Phys receiving ALIGNs after the phy reset sequence shall not verify the rotation and shall accept any of the ALIGNs at any time.

Phys shall only detect an ALIGN after decoding all four characters in the primitive.

NOTE 35 - SATA devices are allowed to decode every dword starting with a K28.5 as an ALIGN, since ALIGN is the only primitive defined starting with K28.5.

For physical link rate tolerance management and rate matching, ALIGNs may be replaced by NOTIFYs (see 7.2.5.11) or MUXs (see 7.2.5.10). ALIGNs shall not be replaced by NOTIFYs or MUXs during OOB signals or speed negotiation.

7.2.5.3 BREAK

BREAK is used to abort a connection request or break a connection.

See 7.12.6 and 7.12.8 for details on breaking connections.

7.2.5.4 BREAK_REPLY

BREAK_REPLY is used to confirm the receipt of a BREAK.

See 7.12.6 and 7.12.8 for details on breaking connections.

7.2.5.5 BROADCAST

BROADCASTs are used to notify all SAS ports in a domain of an event.

The versions of BROADCAST representing different Broadcast types are defined in table 110.

Table 110 — BROADCAST primitives

Primitive	Description
BROADCAST (CHANGE)	Broadcast (Change)
BROADCAST (RESERVED CHANGE 0)	Broadcast (Reserved Change 0)
BROADCAST (RESERVED CHANGE 1)	Broadcast (Reserved Change 1)
BROADCAST (SES)	Broadcast (SES)
BROADCAST (EXPANDER)	Broadcast (Expander)
BROADCAST (ASYNCHRONOUS EVENT)	Broadcast (Asynchronous Event)
BROADCAST (RESERVED 3)	Broadcast (Reserved 3)
BROADCAST (RESERVED 4)	Broadcast (Reserved 4)

A phy that has not completed the link reset sequence shall not transmit a BROADCAST. A phy shall not transmit a BROADCAST inside a connection.

A BROADCAST received by a phy that has not completed the link reset sequence shall be ignored.

See 4.1.13 for definitions of Broadcasts.

7.2.5.6 CLOSE

CLOSE is used to close a connection. This primitive may be originated by a SAS initiator port or a SAS target port.

The versions of CLOSE representing different reasons are defined in table 111.

Table 111 — CLOSE primitives

Primitive	Description
CLOSE (CLEAR AFFILIATION)	Close an open STP connection and clear the affiliation (see 7.17.4). Processed the same as CLOSE (NORMAL) if: a) the connection is not an STP connection; b) the connection is an STP connection, but affiliations are not implemented by the STP target port; or c) the connection is an STP connection, but an affiliation is not present.
CLOSE (NORMAL)	Close a connection.
CLOSE (RESERVED 0)	Reserved. Processed the same as CLOSE (NORMAL).
CLOSE (RESERVED 1)	

See 7.12.7 for details on closing connections.

7.2.5.7 EOAF (End of address frame)

EOAF specifies the end of an address frame.

See 7.8 for details on address frames.

7.2.5.8 ERROR

ERROR should be sent by an expander device when it is forwarding dwords from a SAS physical link or SATA physical link to a SAS physical link and it receives an invalid dword or an ERROR.

NOTE 36 - Since an 8b10b coding error in one dword is sometimes not detected until the next dword (see table 78 in 6.3.5), expander devices should avoid deleting invalid dwords or ERRORS unless necessary (e.g., if the elasticity buffer is full) to avoid hiding evidence that an error has occurred.

See 7.15 for details on error handling by expander devices.

7.2.5.9 HARD_RESET

HARD_RESET is used to force a phy to generate a hard reset to its port. This primitive is only valid after the phy reset sequence without an intervening identification sequence (see 4.4) and shall be ignored at other times.

7.2.5.10 MUX (Multiplex)

MUX is used if multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3) as follows:

- a) transmitted during the multiplexing sequence (see 6.7.4.3); and
- b) substituted for an ALIGN (see 7.2.5.2) being transmitted for physical link rate tolerance management (see 7.3) or rate matching (see 7.13) to confirm the logical link number.

Substitution of a MUX for an ALIGN may or may not affect the ALIGN rotation (i.e., the MUX may take the place of one of the ALIGNs in the rotation through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3), or it may delay the rotation).

MUXs are deletable primitives (see 7.3).

The versions of MUX are defined in table 112.

Table 112 — MUX primitives

Primitive	Description
MUX (LOGICAL LINK 0)	Establishes the position of dwords in logical link 0.
MUX (LOGICAL LINK 1)	Establishes the position of dwords in logical link 1.

See 6.10 for details on multiplexing.

7.2.5.11 NOTIFY

7.2.5.11.1 NOTIFY overview

NOTIFY may be transmitted in place of any ALIGN (see 7.2.5.2) being transmitted for physical link rate tolerance management (see 7.3) and rate matching (see 7.13). Substitution of a NOTIFY for an ALIGN may or may not affect the ALIGN rotation (i.e., the NOTIFY may take the place of one of the ALIGNs in the rotation through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3), or it may delay the rotation). A specific NOTIFY shall not be transmitted in more than three consecutive dwords until at least three other dwords have been transmitted.

NOTIFYs are deletable primitives (see 7.3).

NOTIFY shall not be forwarded through expander devices. Expander devices shall substitute an ALIGN for a NOTIFY if necessary.

SAS target devices are not required to detect every transmitted NOTIFY.

The versions of NOTIFY representing different reasons are defined in table 113.

Table 113 — NOTIFY primitives

Primitive	Description	Reference
NOTIFY (ENABLE SPINUP)	Specify to a SAS target device that it may temporarily consume additional power while transitioning into the active or idle power condition state.	7.2.5.11.2
NOTIFY (POWER LOSS EXPECTED)	Specify to a SAS target device that power loss may occur within the time specified by the POWER LOSS TIMEOUT field in the Shared Port Control mode page (see 10.2.7.6).	7.2.5.11.3
NOTIFY (RESERVED 1)	Reserved.	
NOTIFY (RESERVED 2)		

NOTIFY (RESERVED 1) and NOTIFY (RESERVED 2) shall be ignored by all devices.

7.2.5.11.2 NOTIFY (ENABLE SPINUP)

NOTIFY (ENABLE SPINUP) is transmitted by a SAS initiator port or expander port and is used to specify to an SAS target device that it may temporarily consume additional power (e.g., while spinning-up rotating media) while transitioning into the active or idle power condition state. The length of time the SAS target device consumes additional power and the amount of additional power is vendor specific. NOTIFY (ENABLE SPINUP) shall interact with the device's power condition state transitions, controlled by the Power Conditions mode page (see SPC-4) and/or the START STOP UNIT command (see SBC-3), as described in 10.2.10.

SAS initiator devices and expander devices shall use NOTIFY (ENABLE SPINUP) while attached to SSP target devices (i.e., devices that report SSP target port support in their IDENTIFY address frames). They shall transmit one NOTIFY (ENABLE SPINUP) after power on when the enclosure is ready for initial spin-up. After the initial NOTIFY (ENABLE SPINUP), they shall transmit NOTIFY (ENABLE SPINUP) periodically. Otherwise, the selection of when and how often to transmit NOTIFY (ENABLE SPINUP) is outside the scope of this standard.

NOTE 37 - The SAS initiator device or expander device uses NOTIFY (ENABLE SPINUP) to avoid exceeding enclosure power supply capabilities during spin-up of multiple SAS target devices. It may choose to rotate transmitting NOTIFY (ENABLE SPINUP) across all of its ports, distributing it to N ports at a time if the enclosure power supply is capable of powering N SAS target devices spinning up at a time. An expander device may allow this timing to be configured by an NVRAM programmed with enclosure-specific sequencing patterns, or may employ more complex, dynamic interaction with the enclosure power supply.

NOTE 38 - NOTIFY (ENABLE SPINUP) should be transmitted as frequently as possible to avoid incurring application layer timeouts.

I_T nexus loss, logical unit reset, and hard reset shall not cause a SAS target device to spin-up automatically on receipt of NOTIFY (ENABLE SPINUP).

A SAS target device with multiple SAS target ports shall honor NOTIFY (ENABLE SPINUP) from all its SAS target ports equivalently (e.g., if a SAS target device contains two SSP target ports A and B, powers on in the Stopped state, and receives a START STOP UNIT command with the START bit set to one through SSP target port A, then a NOTIFY (ENABLE SPINUP) received on SSP target port B causes the SAS target device to spin up its rotating media (see 10.2.10)).

7.2.5.11.3 NOTIFY (POWER LOSS EXPECTED)

NOTIFY (POWER LOSS EXPECTED) is transmitted by a SAS initiator port or expander port and is used to specify to a SAS target device that power loss may occur within the time specified in the POWER LOSS TIMEOUT field in the Shared Port Control mode page (see 10.2.7.6).

NOTIFY (POWER LOSS EXPECTED) shall be transmitted at least three times by the SAS initiator port or expander port.

If a SAS target device supports NOTIFY (POWER LOSS EXPECTED) and receives NOTIFY (POWER LOSS EXPECTED) on an SSP target port, then the device server for each logical unit to which the SSP target port has access shall:

- 1) stop writing data to the media on a block boundary (e.g., all write activity shall continue until a block boundary is reached then all writing shall stop);
- 2) clear all task sets as defined in SAM-4; and
- 3) establish a unit attention condition for the initiator port associated with every I_T nexus as defined in SAM-4 (e.g., with the additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION).

If a SAS target device supports NOTIFY (POWER LOSS EXPECTED) and receives NOTIFY (POWER LOSS EXPECTED) on an SSP target port, then the SAS target device shall:

- a) on each phy that receives NOTIFY (POWER LOSS EXPECTED), if there is an SSP connection, then transmit a BREAK on that connection; and
- b) on each phy that does not receive NOTIFY (POWER LOSS EXPECTED), if there is an SSP connection, then transmit a BREAK or a CLOSE on that connection.

The SCSI application layer that receives a Power Loss Expected event shall:

- a) start the power loss timer;
- b) send an Accept_Reject OPENs (Reject SSP) request to all ST_T state machines (i.e., all SSP connection requests result in OPEN_REJECT (RETRY));
- c) if a SCSI Command Received transport protocol service indication is received, then the device server shall abort that command and send an Accept_Reject OPENs (Reject SSP) request to the ST_T state

machine on which the SCSI Command Received transport protocol service indication was received;
and

- d) if the power loss timeout timer expires, then the SCSI application layer shall send an Accept_Reject OPENs (Accept SSP) request to all ST_T state machines.

If any frames are received by the SAS target device after receiving NOTIFY (POWER LOSS EXPECTED) before a connection is closed, then the SAS target device shall discard the received frames.

After power on, the power loss timeout timer shall be initialized and stopped until a NOTIFY (POWER LOSS EXPECTED) is received.

7.2.5.12 OPEN_ACCEPT

OPEN_ACCEPT specifies the acceptance of a connection request.

See 7.12 for details on connection requests.

7.2.5.13 OPEN_REJECT

OPEN_REJECT specifies that a connection request has been rejected and specifies the reason for the rejection. The result of some OPEN_REJECTs is to abandon (i.e., not retry) the connection request and the result of other OPEN_REJECTs is to retry the connection request.

All of the OPEN_REJECT versions defined in table 114 shall result in the originating port abandoning the connection request.

Table 114 — Abandon-class OPEN_REJECT primitives

Primitive	Originator	Description
OPEN_REJECT (BAD DESTINATION)	Expander phy	A connection request routes to a destination expander phy in the same expander port as the source expander phy and the expander port is using the direct routing method (see 4.6.7.1).
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	Any phy	The requested connection rate is not supported on some physical link on the pathway between the source phy and destination phy. When a SAS initiator phy is directly attached to a SAS target phy, the requested connection rate is not supported by the destination phy. If the connection rate is 1,5 Gbps, this shall be considered an abandon-class OPEN_REJECT. If the connection rate is greater than 1,5 Gbps, the connection request shall be modified and reattempted as described in 7.8.3.
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	Destination phy	Phy with destination SAS address exists but the destination phy does not support the requested initiator/target role, protocol, initiator connection tag, or features (i.e., the values in the INITIATOR PORT bit, the PROTOCOL field, the INITIATOR CONNECTION TAG field, and/or the FEATURES field in the OPEN address frame are not supported).
OPEN_REJECT (RESERVED ABANDON 1)	Unknown	Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 2)		
OPEN_REJECT (RESERVED ABANDON 3)		
OPEN_REJECT (STP RESOURCES BUSY)	Destination phy	STP target port with destination SAS address exists but the STP target port supports affiliations and is not able to establish an affiliation with this STP initiator port (e.g., because it has reached its maximum number of affiliations), or the STP target port does not support affiliations and all of the available ATA task file registers have been allocated to other STP initiator ports (see 7.17.4). Process the same as OPEN_REJECT (WRONG DESTINATION) for non-STP connection requests.
OPEN_REJECT (WRONG DESTINATION)	Destination phy	The destination SAS address does not match the SAS address of the SAS port to which the connection request was delivered.
OPEN_REJECT (ZONE VIOLATION)	Zoning expander phy	The connection request is from a zone group that does not have permission to access the zone group that contains the destination phy according to the zone permission table of an unlocked zoning expander device.

All of the OPEN_REJECT versions defined in table 115 shall result in the originating port retrying the connection request.

Table 115 — Retry-class OPEN_REJECT primitives

Primitive	Originator	Description
OPEN_REJECT (NO DESTINATION) ^a	Expander phy	An expander device in the pathway is not configuring and determines that: a) there is no such destination phy; b) the connection request routes to a destination expander phy in the same expander port as the source expander phy and the expander port is using the subtractive routing method; or c) the SAS address is valid for an STP target port in an STP/SATA bridge, but the initial Register - Device to Host FIS has not been successfully received (see 10.4.3.11).
OPEN_REJECT (PATHWAY BLOCKED) ^b	Expander phy	An expander device determined the pathway was blocked by higher priority connection requests.
OPEN_REJECT (RESERVED CONTINUE 0) ^c	Unknown	Reserved. Process the same as OPEN_REJECT (RETRY).
OPEN_REJECT (RESERVED CONTINUE 1) ^c		
OPEN_REJECT (RESERVED INITIALIZE 0) ^a	Unknown	Reserved. Process the same as OPEN_REJECT (NO DESTINATION).
OPEN_REJECT (RESERVED INITIALIZE 1) ^a		
OPEN_REJECT (RESERVED STOP 0) ^b	Unknown	Reserved. Process the same as OPEN_REJECT (PATHWAY BLOCKED).
OPEN_REJECT (RESERVED STOP 1) ^b		
OPEN_REJECT (RETRY) ^c	Destination phy or zoning expander phy	Either: a) a phy with destination SAS address exists but is temporarily not able to accept connections (see 7.16.1, 7.17.5, and 7.18.3); b) an expander device in the pathway is configuring and would otherwise have returned OPEN_REJECT (NO DESTINATION)(see 4.7.2 and 7.12.4.2.5); c) an expander device in the pathway is locked and would otherwise have returned OPEN_REJECT (ZONE VIOLATION)(see 4.9.3.5 and 7.12.4.2.5); or d) an expander device in the pathway has reduced functionality (see 4.6.8 and 7.12.4.2.5).
^a If the I_T Nexus Loss timer is already running, it continues running; if it is not already running, it is initialized and started. Stop retrying the connection request if the I_T Nexus Loss timer expires. ^b If the I_T Nexus Loss timer is already running, it continues running. Stop retrying the connection request if the I_T Nexus Loss timer expires. ^c If the I_T Nexus Loss timer (see 8.2.2) is already running, it is stopped.		

NOTE 39 - Some SAS logical phys compliant with earlier versions of this standard also transmit OPEN_REJECT (RETRY) if they receive an OPEN address frame while their SL_CC state machines are in the SL_CC5:BreakWait state (see 7.14.4.7).

When a SAS logical phy detects more than one reason to transmit an OPEN_REJECT, the SL_CC state machine determines the priority in the SL_CC2:Selected state (see 7.14.4.4).

When an expander logical phy detects more than one reason to transmit an OPEN_REJECT, the ECM determines the priority (see 7.12.4).

See 7.12 for details on connection requests.

7.2.5.14 SOAF (Start of address frame)

SOAF specifies the start of an address frame.

See 7.8 for details on address frames.

7.2.5.15 TRAIN

TRAIN is used during Train-SNW during speed negotiation.

See 6.7.4.2.3.4 for details on Train-SNW.

7.2.5.16 TRAIN_DONE

TRAIN_DONE is used during Train-SNW during speed negotiation.

See 6.7.4.2.3.4 for details on Train-SNW.

7.2.6 Primitives used only inside SSP and SMP connections

7.2.6.1 ACK (Acknowledge)

ACK specifies the positive acknowledgement of an SSP frame.

See 7.16.3 for details on SSP frame transmission.

7.2.6.2 CREDIT_BLOCKED

CREDIT_BLOCKED specifies that no more credit is going to be sent during this connection.

See 7.16.4 for details on SSP flow control.

7.2.6.3 DONE

DONE is used to start closing an SSP connection and specify a reason for doing so. This primitive may be originated by an SSP initiator port or an SSP target port. DONE is not used to close an SMP or STP connection.

The versions of DONE representing different reasons are defined in table 116. The SSP state machine describes when these are used (see 7.16.8).

Table 116 — DONE primitives

Primitive	Description
DONE (ACK/NAK TIMEOUT)	The SSP state machine (see 7.16.8) timed out waiting for an ACK or NAK, and the phy is going to transmit BREAK in 1 ms unless DONE is received within 1 ms of transmitting the DONE (ACK/NAK TIMEOUT).
DONE (RESERVED TIMEOUT 0)	Reserved. Processed the same as DONE (ACK/NAK TIMEOUT).
DONE (RESERVED TIMEOUT 1)	
DONE (NORMAL)	Finished transmitting all frames.
DONE (RESERVED 0)	Reserved. Processed the same as DONE (NORMAL).
DONE (RESERVED 1)	
DONE (CREDIT TIMEOUT)	The SSP state machine (see 7.16.8) timed out waiting for an RRDY or received a CREDIT BLOCKED, and the phy is going to transmit BREAK if it provides transmit frame credit for 1 ms without receiving a frame or a DONE.

See 7.16.7 for details on closing SSP connections.

7.2.6.4 EOF (End of frame)

EOF specifies the end of an SSP or SMP frame.

See 7.16.3 for details on SSP frame transmission and 7.18.1 for details on SMP frame transmission.

7.2.6.5 NAK (Negative acknowledgement)

NAK specifies the negative acknowledgement of an SSP frame and the reason for doing so.

The versions of NAK representing different reasons are defined in table 117.

Table 117 — NAK primitives

Primitive	Description
NAK (CRC ERROR)	The frame had a bad CRC, or an invalid dword or an ERROR was received during frame reception.
NAK (RESERVED 0)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 1)	
NAK (RESERVED 2)	

See 7.16.3 for details on SSP frame transmission.

7.2.6.6 RRDY (Receiver ready)

RRDY is used to increase SSP frame credit.

The versions of RRDY representing different reasons are defined in table 118.

Table 118 — RRDY primitives

Primitive	Description
RRDY (NORMAL)	Increase transmit frame credit by one.
RRDY (RESERVED 0)	Reserved. Processed the same as RRDY (NORMAL).
RRDY (RESERVED 1)	

A phy shall not transmit RRDY after transmitting CREDIT_BLOCKED in a connection. See 7.16.4 for details on SSP flow control.

7.2.6.7 SOF (Start of frame)

SOF specifies the start of an SSP or SMP frame.

See 7.16.3 for details on SSP frame transmission and 7.18.1 for details on SMP frame transmission.

7.2.7 Primitives used only inside STP connections and on SATA physical links

7.2.7.1 SATA_ERROR

SATA_ERROR should be sent by an expander device when it is forwarding dwords from a SAS logical link to a SATA physical link and it receives an invalid dword or an ERROR.

NOTE 40 - Since an 8b10b coding error in one dword is sometimes not detected until the next dword (see table 78 in 6.3.5), expander devices should avoid deleting invalid dwords or ERRORS unless necessary (e.g., if the elasticity buffer is full) to avoid hiding evidence that an error has occurred.

See 6.9 for details on error handling by expander devices.

Although included in this subclause, SATA_ERROR is not a primitive (see 3.1.169) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword.

7.2.7.2 SATA_PMACK, SATA_PMNAK, SATA_PMREQ_P, and SATA_PMREQ_S (Power management acknowledgements and requests)

SATA_PMREQ_P and SATA_PMREQ_S request entry into the interface power management partial and slumber states. SATA_PMACK is used to accept a power management request. SATA_PMNAK is used to reject a power management request.

See 7.10 for rules on handling the power management primitives.

7.2.7.3 SATA_HOLD and SATA_HOLD_A (Hold and hold acknowledge)

See 7.17.2 for rules on STP flow control, which uses SATA_HOLD and SATA_HOLD_A.

7.2.7.4 SATA_R_RDY and SATA_X_RDY (Receiver ready and transmitter ready)

When a SATA port has a frame to transmit, it transmits SATA_X_RDY and waits for SATA_R_RDY before transmitting the frame.

7.2.7.5 Other primitives used inside STP connections and on SATA physical links

Other primitives used in STP connections and on SATA physical links are defined in SATA.

7.3 Physical link rate tolerance management

7.3.1 Physical link rate tolerance management overview

The internal clock for a device is typically based on a PLL with its own clock generator and is used when transmitting dwords on the logical link. When receiving, however, dwords need to be latched based on a clock derived from the input bit stream itself. Although the input clock is nominally a fixed frequency, it may differ slightly from the internal clock frequency up to the physical link rate tolerance defined in table 58 (see 5.3.3). Over time, if the input clock is faster than the internal clock, the phy receiver may receive a dword and not be able to forward it to an internal buffer; this is called an overrun. If the input clock is slower than the internal clock, the phy receiver may not have a dword when needed in an internal buffer; this is called an underrun.

To solve this problem, phy transmitters insert deletable primitives in the dword stream. Phy receivers may pass deletable primitives through to their internal buffers, or may strip them out when an overrun occurs. Phy receivers add deletable primitives when an underrun occurs. The internal logic shall ignore all deletable primitives that arrive in the internal buffers.

Elasticity buffer circuitry, as shown in figure 156, is required to absorb the slight differences in frequencies between the phys. The frequency tolerance for a phy is specified in 5.3.3. The depth of the elasticity buffer is vendor-specific but shall accommodate the physical link rate tolerance management deletable primitive insertion requirements in table 119.

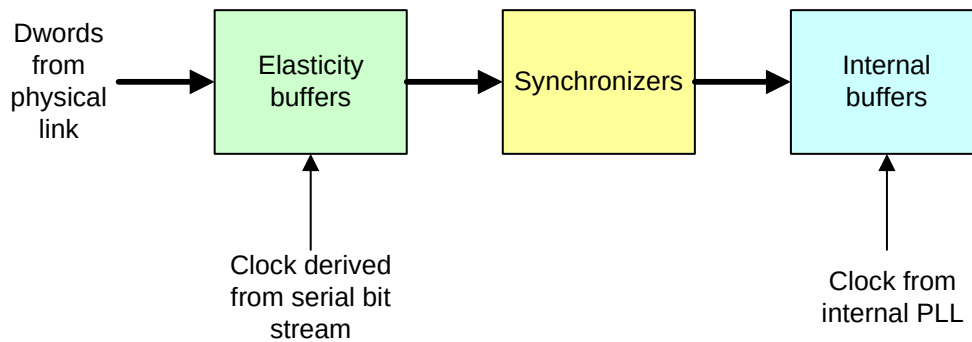


Figure 156 — Elasticity buffers

Editor's Note 36: expand this figure to have another box that does not go through the elasticity buffer. Point to here from 6.10's discussion of MUX primitives and 7.2.x discussion of NOTIFY primitives

7.3.2 Phys originating dwords

A logical phy that is the original source for the dword stream (i.e., a logical phy that is not an expander logical phy forwarding dwords from another expander logical phy) shall insert deletable primitives for physical link rate tolerance management after the phy reset sequence completes as described in table 119.

Table 119 — Physical link rate tolerance management deletable primitive insertion requirement

Physical link rate	Requirement
1,5 Gbps	One deletable primitive within every 128 dwords ^a
3 Gbps	Two deletable primitives within every 256 dwords ^b
6 Gbps	Four deletable primitives within every 512 dwords
^a Phys compliant with previous versions of this standard were required to insert one deletable primitive within every 2 048 dwords at 1,5 Gbps. ^b Phys compliant with previous versions of this standard were required to insert two deletable primitives within every 4 096 dwords at 3 Gbps.	

NOTE 41 - These numbers account for the worst case clock frequency differences between the fastest phy transmitter and the slowest phy receiver (e.g., a center-spreading expander phy originating dwords in an STP connection at +2 400 ppm that are forwarded to a down-spreading SATA device with an internal clock at -5 350 ppm). The difference of 7 750 ppm (i.e., 0,775 % or 1/129) is less than the ALIGN insertion rate of 1/128 (i.e., 7 813 ppm or 0,78125 %), ensuring there are enough deletable primitives for the phy receiver to delete without having to buffer dwords.

Deletable primitives inserted for physical link rate tolerance management are in addition to deletable primitives inserted for rate matching (see 7.13). See Annex H for a summary of their combined requirements.

See 7.2.5.2 for details on rotating through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3). NOTIFYs may also be transmitted in place of ALIGNs (see 7.2.5.11) on SAS logical links. MUXs may also be transmitted in place of ALIGNs on multiplexed SAS physical links.

7.3.3 Expander phys forwarding dwords

An expander device that is forwarding dwords (i.e., is not the original source) is allowed to insert or delete as many deletable primitives as required to match the transmit and receive connection rates. It is not required to transmit the number of deletable primitives for physical link rate tolerance management described in table 119 when forwarding to a SAS logical link. It shall increase or reduce that number based on clock frequency differences between the phy transmitting the dwords to the expander device and the expander device's receiving phy (e.g., if receiving at -100 ppm and transmitting at +100 ppm, it transmits fewer deletable primitives that it receives).

The expander device is also required to insert deletable primitives for rate matching (see 7.13). During an STP connection, the expander device shall:

- preserve the incoming rate of any additional deletable primitives that it receives that are not discarded because of physical link rate tolerance management or rate matching (e.g. the 1/128 deletable primitives received from an originating STP initiator phy compliant with previous versions of this standard for STP initiator phy throttling); or
- transmit one deletable primitive within every 128 dwords,

without discarding any data dwords or primitives. It may reduce the length of repeated primitive sequences (i.e., primitive, SATA_CONT, and data dword sequences).

NOTE 42 - One possible implementation for expander devices forwarding dwords is for the expander device to delete all deletable primitives received and to insert deletable primitives at the transmit port whenever its elasticity buffer is empty.

The STP target port of an STP/SATA bridge is allowed to insert or delete as many deletable primitives as required to match the transmit and receive connection rates. It is not required to transmit any particular

number of deletable primitives for physical link rate tolerance management when forwarding to a SAS logical link and is not required to ensure that any deletable primitives it transmits are in pairs.

NOTE 43 - Due to physical link rate tolerance management deletable primitive removal, the STP target port may not receive a pair of deletable primitives every 256 dwords, even if the STP initiator port transmitted them in pairs. However, the rate of the dword stream allows for deletable primitive insertion by the STP/SATA bridge. One possible implementation is for the STP/SATA bridge to delete all deletable primitives received by the STP target port and to insert two consecutive ALIGNs at the SATA host port when its elasticity buffer is empty or when 254 non-ALIGN dwords have been transmitted. It may need to buffer up to 2 dwords concurrently being received by the STP target port while it does so.

7.4 Idle physical links

Idle dwords are vendor-specific data dwords which are scrambled (see 7.6).

Phys shall transmit idle dwords if there are no other dwords to transmit and:

- a) no connection is open; or
- b) an SSP or SMP connection is open.

SATA_SYNC is a continued primitive sequence which may contain vendor-specific data dwords (see 7.2.4.4) which are scrambled (see 7.6) during an STP connection.

7.5 CRC

7.5.1 CRC overview

All frames include cyclic redundancy check (CRC) values to help detect transmission errors.

Frames transmitted in an STP connection shall include a CRC as defined by SATA (see SATA-2). Address frames, SSP frames, and SMP frames shall include a CRC as defined by this standard.

Annex D contains information on CRC generation/checker implementation.

Table 120 defines notation used in the following text describing CRC calculation. Arithmetic is modulo 2.

Table 120 — CRC notation and definitions (part 1 of 2)

Notation	Definition
$T(i)$	A transformation over the non-negative integers (i.e., Z^+): $T(i) = i + 7 - 2(i \bmod 8), i \geq 0, i \in Z^+$
$F(x)$	A polynomial representing the bits covered by the CRC: $F(x) = b_0x^{(k-1)} + b_1x^{(k-2)} + \dots + b_{(k-2)}x + b_{(k-1)}$ where: k is the number of bits b_i describes a bit, where the bit index i denotes that bit b_i is more significant than bit $b_{(i+1)}$ For example, if the frame, except for the CRC field, contains one data dword set to 516F3019h (i.e., $F(x) = 516F3019h$), then: $F(x) = x^{30} + x^{28} + x^{24} + x^{22} + x^{21} + x^{19} + x^{18} + x^{17} + x^{16} + x^{13} + x^{12} + x^4 + x^3 + x$ (i.e., $F(x) = 8AF60C98h$)
$F_t(x)$	$F(x)$ with the bit positions of each byte transposed (i.e., bit 7 is bit 0, bit 6 is bit 1, etc.): $F_t(x) = b_{T(0)}x^{(k-1)} + b_{T(1)}x^{(k-2)} + \dots + b_{T(k-2)}x + b_{T(k-1)}$ For example, if the frame, except for the CRC field, contains one data dword set to 516F3019h (i.e., $F(x) = 516F3019h$), then: $F_t(x) = x^{31} + x^{27} + x^{25} + x^{23} + x^{22} + x^{21} + x^{20} + x^{18} + x^{17} + x^{11} + x^{10} + x^7 + x^4 + x^3$ (i.e., $F_t(x) = 8AF60C98h$)

Table 120 — CRC notation and definitions (part 2 of 2)

Notation	Definition
$L(x)$	The identity polynomial of degree 31 (i.e., a polynomial with all of the coefficients set to one): $L(x) = x^{31} + x^{30} + \dots + x + 1$ (i.e., $L(x) = \text{FFFFFFFFh}$)
$G(x)$	The CRC generator polynomial (i.e., the divisor polynomial): $G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (i.e., $G(x) = \text{1\_04C11DB7h}$)
$R(x)$	The remainder polynomial, which is of degree less than 32.
$R_t(x)$	$R(x)$ with the bit positions of each byte transposed.
$Q(x)$	A quotient polynomial resulting from CRC calculation by the transmitter. This value is discarded.
$Q'(x)$	A quotient polynomial resulting from CRC calculation by the receiver. This value is discarded.
$M(x)$	A polynomial representing the transmitted frame including the CRC field, which is of degree $k+31$.
$M'(x)$	A polynomial representing the received frame including the received CRC field. If the received frame has no errors, then $M'(x) = M(x)$ and $M'(x)$ is of degree $k+31$.
$M'_t(x)$	$M'(x)$ with the bit positions of each byte transposed.
$R'(x)$	The result of finding the remainder of an error-free reception of $M(x)$, and also the remainder of $x^{32}L(x) / G(x)$, which is a unique constant polynomial: $R'(x) = x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$ (i.e., $R'(x) = \text{C704DD7Bh}$)
$R'_t(x)$	$R'(x)$ with the bit positions of each byte transposed: $R'_t(x) = x^{28} + x^{27} + x^{26} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{17} + x^{16} + x^{14} + x^{10} + x^5 + 1$ (i.e., $R'_t(x) = \text{1CDF4421h}$)

7.5.2 CRC generation

The CRC is calculated from $F(x)$ as follows:

$$x^k L(x) + x^{32} F_t(x) = Q(x) G(x) + R(x)$$

That is:

- 1) the frame $F(x)$, not including the CRC field, is transposed into $F_t(x)$;
- 2) the first 32 bits of the transposed frame are inverted (i.e., $x^k L(x)$ is added);
- 3) 32 bits of zero are appended to the end (i.e., $F_t(x)$ is multiplied by x^{32}); and
- 4) this result is divided by the generator polynomial $G(x)$ to find the remainder $R(x)$.

The transmitter shall present $M(x)$ to the 8b10b encoder:

$$M(x) = x^{32} F(x) + L(x) + R_t(x)$$

That is, the inverted transposed remainder is appended to the end of the frame, then this result (i.e., $M(x)$) is presented to the 8b10b encoder for transmission.

For the purposes of CRC computation, inverting the first 32 bits of a frame may be performed by one of the following methods:

- a) inverting the first 32 bits of the frame $F(x)$ and seeding the CRC remainder register with 00000000h;
- b) seeding the CRC remainder register with FFFFFFFFh; or
- c) prepending the constant 62F52692h to $F(x)$ and seeding the CRC remainder register with 00000000h.

NOTE 44 - The bit order of $F(x)$ used to calculate the CRC is the same order as the bit transmission order (i.e., the bits within each byte encoded into a data dword are transposed to match the implicit transposition in the 8b10b encoding process).

Figure 157 shows the CRC process for an address frame, an SSP frame and SMP frame.

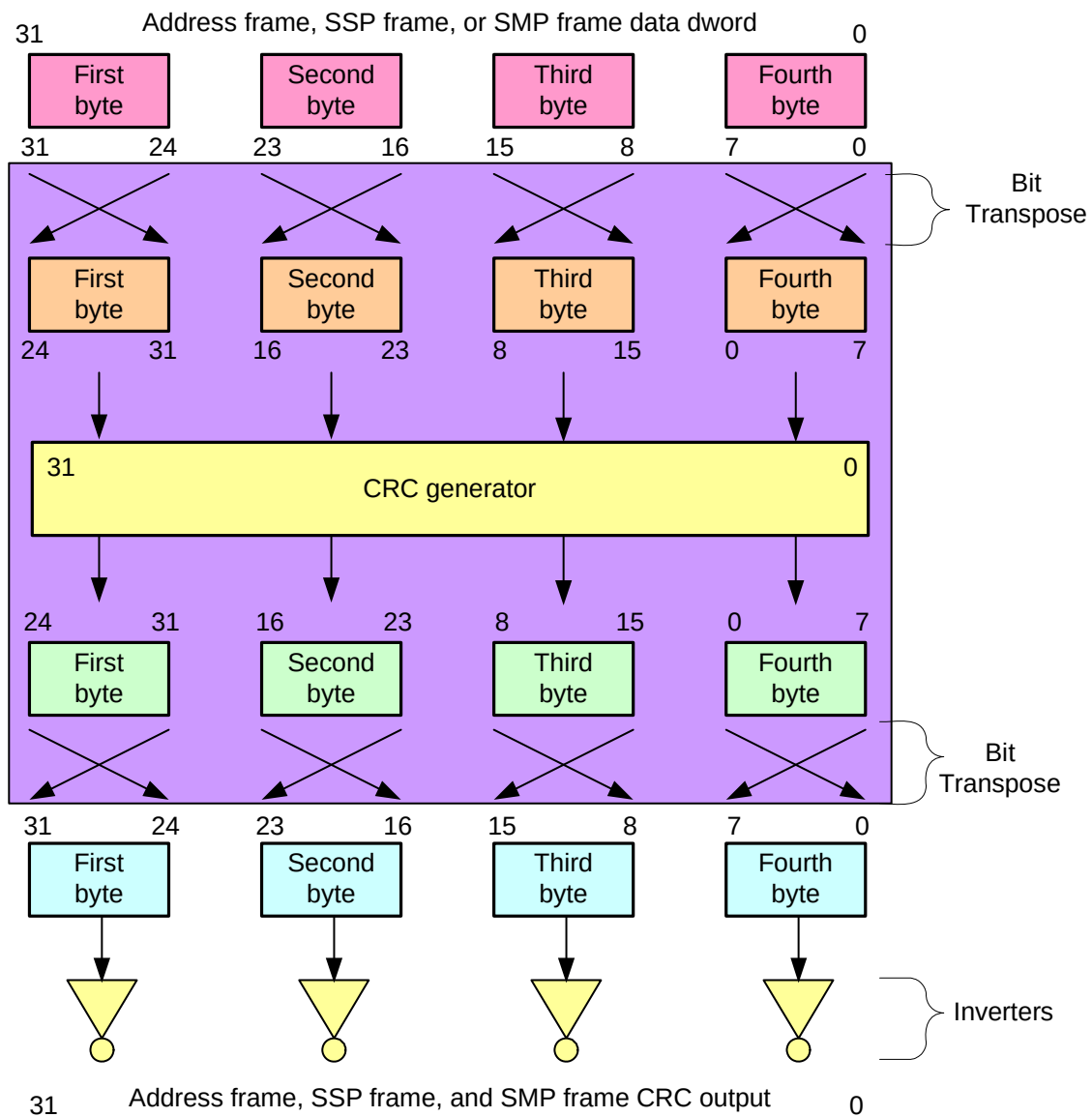


Figure 157 — Address frame, SSP frame, and SMP frame CRC bit ordering

Dwords in STP frames are little-endian and are fed into the STP CRC generator without swapping bits within each byte and without inverting the output like the SAS CRC generator. Figure 158 shows the STP CRC bit ordering.

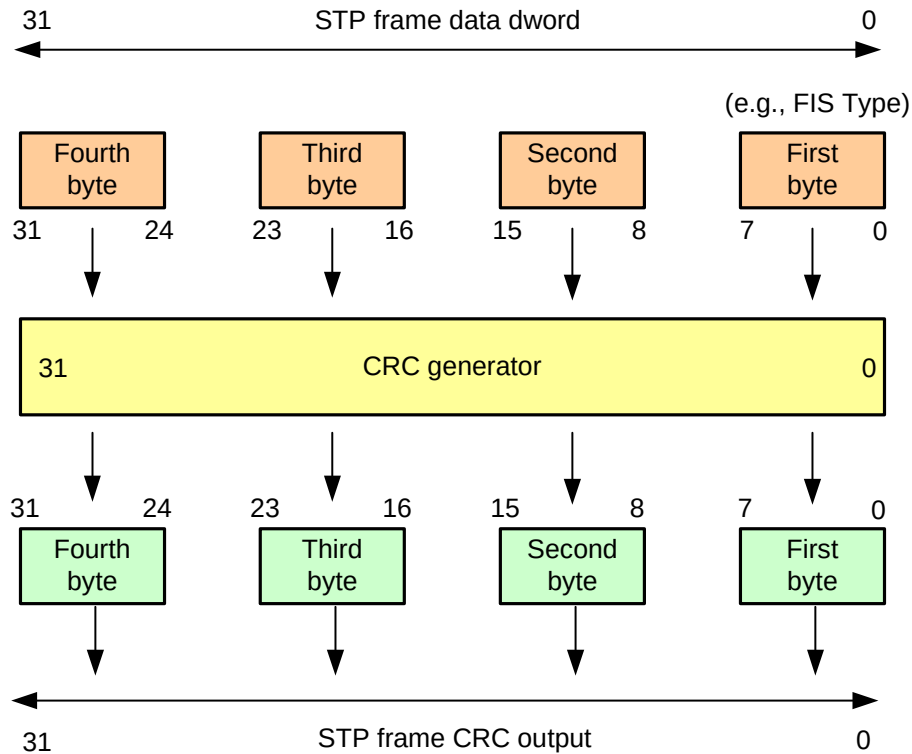


Figure 158 — STP frame CRC bit ordering

Since STP is little-endian, the first byte of a dword is in bits 7:0 rather than 31:24 as in SSP and SMP. As a result, the first byte contains the least-significant bit. In SSP and SMP, the first byte contains the most-significant bit.

See 7.7 for details on how the CRC generator fits into the dword flow along with the scrambler.

7.5.3 CRC checking

The CRC of received frame is calculated by the receiver in the same manner that it is generated by the transmitter.

That is:

- 1) the received frame $M'(x)$, including the CRC field, is transposed into $M_t'(x)$;
- 2) the first 32 bits of the received transposed frame are inverted;
- 3) 32 bits of zero are appended to the end; and
- 4) this result is divided by the generator polynomial $G(x)$ to find the remainder.

A received frame that has not incurred any CRC detectable errors during transmission generates a remainder equal to $R'(x)$.

If there were no transmission errors, then the received frame $M'(x)$ equals $M(x)$:

$$\begin{aligned} M'(x) &= M(x) \\ &= x^{32}F(x) + L(x) + R_t(x) \end{aligned}$$

The CRC $R'(x)$ is derived as follows:

$$\begin{aligned} x^{(k+32)}L(x) + x^{32}M_t'(x) &= x^{(k+32)}L(x) + x^{32}(x^{32}F_t(x) + L(x) + R(x)) \\ &= x^{32}Q(x)G(x) + x^{32}L(x) \end{aligned}$$

However, $G(x)$ divides $x^{32}L(x)$:

$$x^{32}L(x) = Q'(x)G(x) + R'(x)$$

Since $L(x)$ and $G(x)$ are known and constant, $R'(x)$ is known and constant and is expected by the receiver after calculating the CRC of a received frame.

From the previous two results:

$$x^{(k+32)}L(x) + x^{32}(x^{32}F_t(x) + L(x) + R(x)) = (x^{32}Q(x) + Q'(x))G(x) + R'(x)$$

$R'(x)$ is then transposed and inverted, in the same manner as is done by the transmitter, to obtain:

$$R_t'(x) + L(x) = 1CDF4421h$$

Alternatively to this process, the receiver may check the CRC validity of the frame by stripping off the last 32 bits, leaving $F(x)$, and calculating the CRC as defined in 7.5.2. The frame has a valid CRC if the result $L(x) + R_t(x)$ equals the last 32 bits of the frame which were stripped.

See 7.7 for details on where the CRC checker fits into the dword flow along with the descrambler.

7.6 Scrambling

Scrambling is used to reduce the probability of long strings of repeated patterns appearing on the physical link.

All data dwords are scrambled. Table 121 lists the scrambling for different types of data dwords.

Table 121 — Scrambling for different data dword types

Connection state	Data dword type	Description of scrambling
Outside connections	SAS idle dword	When a connection is not open and there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
	Address frame	After an SOAF, all data dwords shall be scrambled until the EOAF.
Inside SSP connection	SSP frame	After an SOF, all data dwords shall be scrambled until the EOF.
	SSP idle dword	When there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
Inside SMP connection	SMP frame	After an SOF, all data dwords shall be scrambled until the EOF.
	SMP idle dword	When there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
Inside STP connection	STP frame	After a SATA_SOF, all data dwords shall be scrambled until the SATA_EOF.
	Continued primitive	After a SATA_CONT, vendor-specific scrambled data dwords shall be sent until a primitive other than a deletable primitive is transmitted.

Data dwords being transmitted shall be XORed with a defined pattern to produce a scrambled value encoded and transmitted on the physical link. Received data dwords shall be XORed with the same pattern after decoding to produce the original data dword value, provided there are no transmission errors.

The pattern that is XORed with the data dwords is defined by the output of a linear feedback shift register implemented with the following polynomial:

$$G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1.$$

The output of the pattern generator is 16 bits wide. For each data dword the output of the generator is applied to the lower 16 bits (i.e., bits 15 through 0) of the 32-bit data dword being transmitted or received; the next output of the generator is applied to the upper 16 bits (i.e., bits 31 through 16).

NOTE 45 - Scrambling is not based on data feedback, so the sequence of values XORed with the data being transmitted is constant.

The value of the linear feedback shift register shall be initialized at each SOF and SOAF to FFFFh.

For detailed requirements about scrambling of data dwords following SATA_SOF and SOF_CONT, see SATA-2.

NOTE 46 - STP scrambling uses two linear feedback shift registers, since continued primitive sequences may occur inside STP frames and the STP frame and the continued primitive sequence have independent scrambling patterns.

Annex F contains information on scrambling implementations.

7.7 Bit order of CRC and scrambler

Figure 159 shows how data dwords and primitives are routed to the bit transmission logic in figure 122 (see 6.5). Data dwords go through the CRC generator and scrambler.

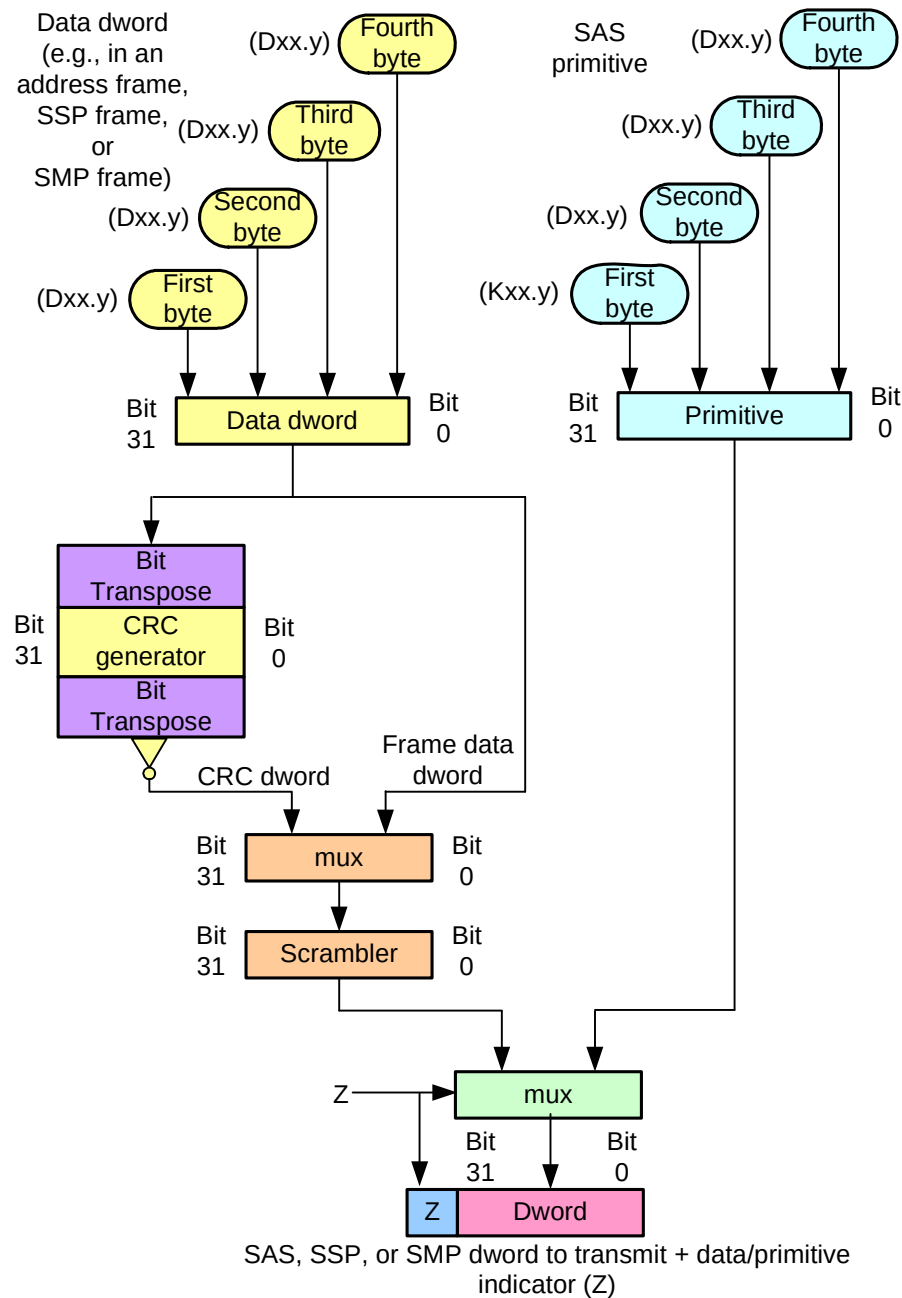


Figure 159 — Transmit path bit ordering

Figure 160 shows the routing of dwords received from the bit reception logic in figure 123 (see 6.5).

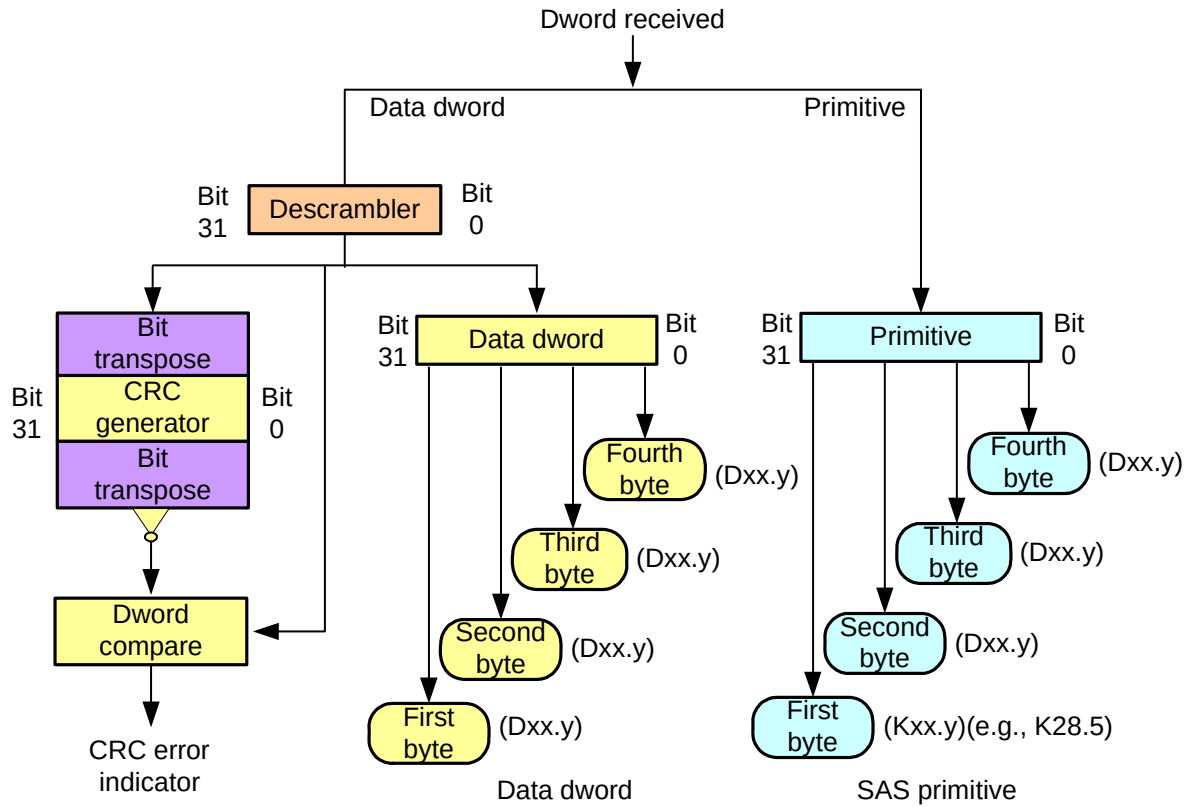


Figure 160 — Receive path bit ordering

Figure 161 shows the STP transmit path bit ordering.

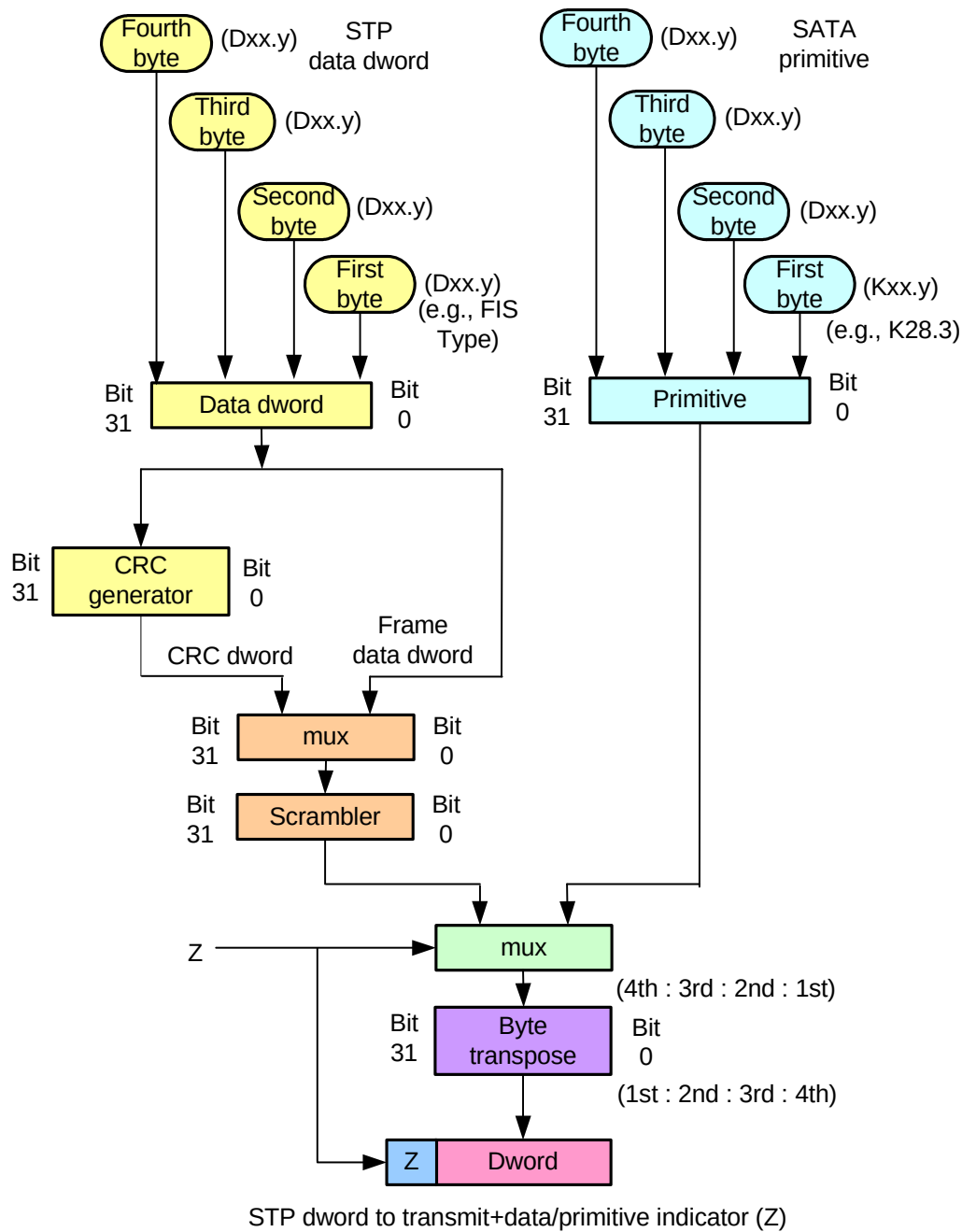


Figure 161 — STP transmit path bit ordering

Figure 162 shows the STP receive path bit ordering.

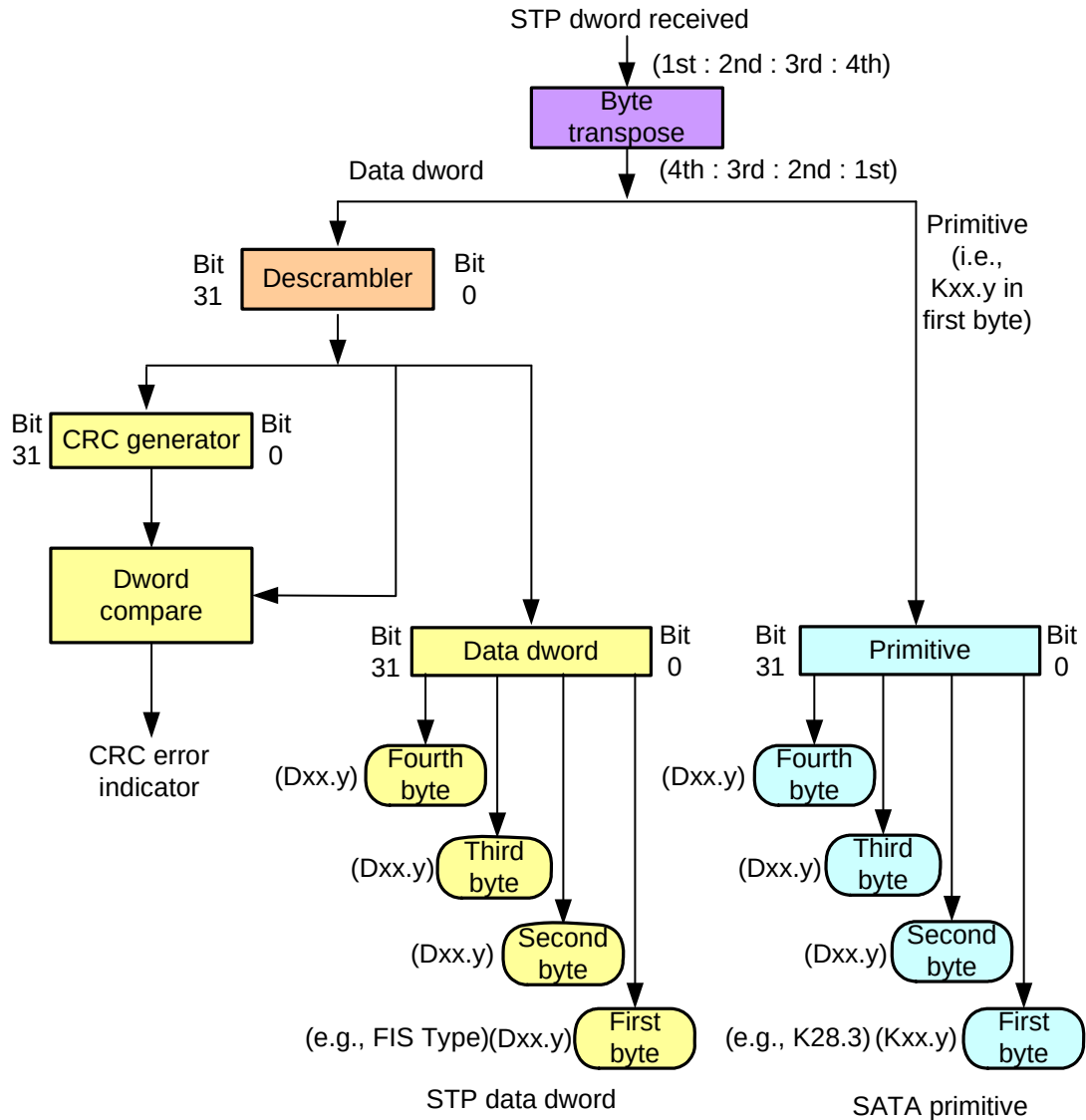


Figure 162 — STP receive path bit ordering

7.8 Address frames

7.8.1 Address frames overview

Address frames are used for the identification sequence and for connection requests.

Address frames are preceded by SOAF and followed by EOAF as shows in figure 163.

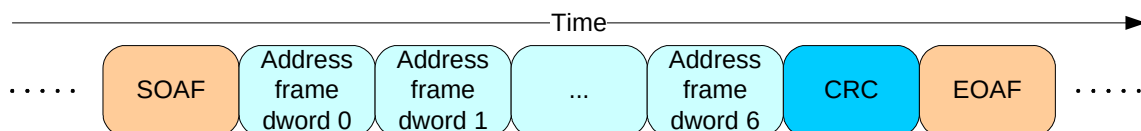


Figure 163 — Address frame transmission

Address frames shall only be sent outside connections. Address frames shall not be terminated early. All data dwords in an address frame shall be scrambled.

Table 122 defines the address frame format.

Table 122 — Address frame format

Byte\Bit	7	6	5	4	3	2	1	0
0					ADDRESS FRAME TYPE			
1	Frame type dependent bytes							
27								
28								
31	CRC						(LSB)	

The ADDRESS FRAME TYPE field indicates the type of address frame and is defined in table 123. This field determines the definition of the frame type dependent bytes.

Table 123 — ADDRESS FRAME TYPE field

Code	Address frame type	Description
0h	IDENTIFY	Identification sequence
1h	OPEN	Connection request
All others	Reserved	

The CRC field contains a CRC value (see 7.5) that is computed over the entire address frame prior to the CRC field.

Address frames with unknown address frame types, incorrect lengths, or CRC errors shall be ignored by the recipient.

7.8.2 IDENTIFY address frame

Table 124 defines the IDENTIFY address frame format used for the identification sequence. The IDENTIFY address frame is sent by each logical phy after the phy reset sequence completes if the physical link is a SAS physical link. The IDENTIFY address frame sent by each logical phy in a physical phy shall be identical.

Table 124 — IDENTIFY address frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved	DEVICE TYPE			ADDRESS FRAME TYPE (0h)			
1	Reserved				REASON			
2	Reserved				SSP INITIATOR PORT	STP INITIATOR PORT	SMP INITIATOR PORT	Restricted (for OPEN address frame)
3	Reserved				SSP TARGET PORT	STP TARGET PORT	SMP TARGET PORT	Restricted (for OPEN address frame)
4	DEVICE NAME							
11								
12	SAS ADDRESS							
19								
20	PHY IDENTIFIER							
21	Reserved					INSIDE ZPSDS PERSISTENT	REQUESTED INSIDE ZPSDS	BREAK_REPLY CAPABLE
22	Reserved							
27								
28	(MSB)	CRC						(LSB)
31								

The DEVICE TYPE field indicates the type of device containing the phy, and is defined in table 125.

Table 125 — DEVICE TYPE field

Code	Description
001b	End device
010b	Expander device
011b	Expander device compliant with a previous version of this standard
All others	Reserved

The ADDRESS FRAME TYPE field shall be set to 0h.

The REASON field indicates the reason for the link reset sequence and is defined in table 126.

Table 126 — REASON field

Code	Description
0h	Unknown reason
1h	Power on
2h	Hard reset (e.g., phy received a HARD_RESET primitive during the hard reset sequence)(see 4.4.2), or SMP PHY CONTROL function HARD RESET phy operation (see 10.4.3.27)
3h	SMP PHY CONTROL function LINK RESET phy operation, or TRANSMIT SATA PORT SELECTION SIGNAL phy operation (see 10.4.3.27)
4h	Loss of dword synchronization (see 6.9)
5h	After the multiplexing sequence completes, MUX (LOGICAL LINK 0) received in logical link 1 or MUX (LOGICAL LINK 1) received in logical link 0 (see 6.10).
6h	I_T nexus loss timer expired in the STP target port of an STP/SATA bridge when the phy was attached to a SATA device (see 4.5)
7h	Break Timeout Timer expired (see 7.12.8)
8h	Phy test function stopped (see 10.4.3.28)
9h	Expander device reduced functionality (see 4.6.8)
Ah - Fh	Reserved

An SSP INITIATOR PORT bit set to one indicates that an SSP initiator port is present. An SSP INITIATOR PORT bit set to zero indicates that an SSP initiator port is not present. Expander devices shall set the SSP INITIATOR PORT bit to zero.

An STP INITIATOR PORT bit set to one indicates that an STP initiator port is present. An STP INITIATOR PORT bit set to zero indicates that an STP initiator port is not present. Expander devices shall set the STP INITIATOR PORT bit to zero.

An SMP INITIATOR PORT bit set to one indicates that an SMP initiator port is present. An SMP INITIATOR PORT bit set to zero indicates that an SMP initiator port is not present. Expander devices may set the SMP INITIATOR PORT bit to one.

An SSP TARGET PORT bit set to one indicates that an SSP target port is present. An SSP TARGET PORT bit set to zero indicates that an SSP target port is not present. Expander devices shall set the SSP TARGET PORT bit to zero.

An STP TARGET PORT bit set to one indicates that an STP target port is present. An STP TARGET PORT bit set to zero indicates that an STP target port is not present. Expander devices shall set the STP TARGET PORT bit to zero.

An SMP TARGET PORT bit set to one indicates that an SMP target port is present. An SMP TARGET PORT bit set to zero indicates that an SMP target port is not present. Expander devices shall set the SMP TARGET PORT bit to one.

The DEVICE NAME field indicates the device name (see 4.2.4) of the SAS device or expander device transmitting the IDENTIFY address frame. A DEVICE NAME field set to 00000000 00000000h indicates the device name is not provided in this field.

NOTE 47 - In expander devices, the DEVICE NAME field, if not set to 00000000 00000000h, contains the same value as the SAS ADDRESS field.

For SAS ports, the SAS ADDRESS field indicates the port identifier (see 4.2.7) of the SAS port transmitting the IDENTIFY address frame. For expander ports, the SAS ADDRESS field indicates the device name (see 4.2.4) of the expander device transmitting the IDENTIFY address frame.

The PHY IDENTIFIER field indicates the phy identifier of the phy transmitting the IDENTIFY address frame.

The REQUESTED INSIDE ZPSDS bit indicates the value of the REQUESTED INSIDE ZPSDS bit in the zone phy information (see 4.9.3.1) at the time the IDENTIFY address frame is transmitted. If the phy transmitting the IDENTIFY address frame is contained in an end device, a non-zoning expander device, or a zoning expander device with zoning disabled, then the REQUESTED INSIDE ZPSDS bit shall be set to zero.

The INSIDE ZPSDS PERSISTENT bit indicates the value of the INSIDE ZPSDS PERSISTENT bit in the zone phy information (see 4.9.3.1) at the time the IDENTIFY address frame is transmitted. If the phy transmitting the IDENTIFY address frame is contained in an end device, a non-zoning expander device, or a zoning expander device with zoning disabled, then the INSIDE ZPSDS PERSISTENT bit shall be set to zero.

The BREAK_REPLY CAPABLE field indicates that the phy is capable of responding to received BREAK primitive sequences with a BREAK_REPLY primitive sequence (see 7.12.5).

See 4.1.4 for additional requirements concerning the DEVICE TYPE field, SSP INITIATOR PORT bit, STP INITIATOR PORT bit, SMP INITIATOR PORT bit, SSP TARGET PORT bit, STP TARGET PORT bit, SMP TARGET PORT bit, and SAS ADDRESS field.

The CRC field is defined in 7.8.1.

7.8.3 OPEN address frame

Table 127 defines the OPEN address frame format used for connection requests.

Table 127 — OPEN address frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	INITIATOR PORT	PROTOCOL			ADDRESS FRAME TYPE (1h)			
1	FEATURES				CONNECTION RATE			
2	(MSB)							
3	INITIATOR CONNECTION TAG							(LSB)
4	DESTINATION SAS ADDRESS							
11								
12	SOURCE SAS ADDRESS							
19								
20	SOURCE ZONE GROUP							
21	PATHWAY BLOCKED COUNT							
22	(MSB)							
23	ARBITRATION WAIT TIME							(LSB)
24	MORE COMPATIBLE FEATURES							
27								
28	(MSB)							
31	CRC							(LSB)

An INITIATOR PORT bit set to one specifies that the source port is acting as a SAS initiator port. An INITIATOR PORT bit set to zero specifies that the source port is acting as a SAS target port. If a SAS target/initiator port sets the INITIATOR PORT bit to one, it shall operate only in its initiator role during the connection. If a SAS target/initiator port sets the INITIATOR PORT bit to zero, it shall operate only in its target role during the connection.

If a SAS target/initiator port accepts an OPEN address frame with the INITIATOR PORT bit set to one, it shall operate only in its target role during the connection. If a SAS target/initiator port accepts an OPEN address frame with the INITIATOR PORT bit set to zero, it shall operate only in its initiator role during the connection.

The **PROTOCOL** field specifies the protocol for the connection being requested and is defined in table 128.

Table 128 — PROTOCOL field

Code	Description
000b	SMP
001b	SSP
010b	STP
All others	Reserved

The **ADDRESS FRAME TYPE** field shall be set to 1h.

The **FEATURES** field specifies any additional features that are incompatible with previous versions of this standard and is defined in table 129.

Table 129 — FEATURES field

Code	Description
0h	No additional features
All others	Reserved

The **CONNECTION RATE** field specifies the connection rate (see 4.1.12) being requested between the source and destination, and is defined in table 130.

Table 130 — CONNECTION RATE field

Code	Description
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future connection rates
All others	Reserved

A SAS initiator port shall set the initial **CONNECTION RATE** field to:

- the highest supported connection rate supported by a potential pathway as determined during the discover process (e.g., based on the logical link rates of each logical link reported in the SMP DISCOVER responses); or
- the logical link rate of the logical phy used to transmit the OPEN address frame.

If a SAS initiator port selected a connection rate based on discover process information but the connection request results in **OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)**, the discover process information is no longer current and should be run again.

A SAS target port shall set the initial **CONNECTION RATE** field to:

- the last known good connection rate established with the SAS initiator port; or
- for the first frame it intends to transmit in the connection, the connection rate that was used by the SAS initiator port to deliver the command or task management function for that frame.

Each time a connection request with a connection rate greater than 1,5 Gbps results in **OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)**, the SAS port shall reattempt the connection request with a lower

connection rate (e.g., drop from 6 Gbps to 3 Gbps or 1.5 Gbps) and send the same frames in the resulting connection that it intended to send at the initial connection rate.

The INITIATOR CONNECTION TAG field is used for SSP and STP connection requests to provide a SAS initiator port an alternative to using the SAS target port's SAS address for context lookup when the SAS target port originates a connection request. SSP or STP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh if they do not require the field be provided by the SAS target port. If they do require the field to be provided, an SSP or STP initiator port should set the INITIATOR CONNECTION TAG field to a unique value per SAS target port. When requesting a connection to a SAS initiator port, a SAS target port shall set the INITIATOR CONNECTION TAG field to the most recent value received or the value received in one of the connection requests for one of the outstanding commands or task management functions from the SAS initiator port. A SAS initiator port shall use the same INITIATOR CONNECTION TAG field value for all connection requests to the same SAS target port, and shall only change the INITIATOR CONNECTION TAG field value when it has no commands or task management functions outstanding to that SAS target port. SAS target ports are not required to check consistency of the INITIATOR CONNECTION TAG field in different connection requests from the same SAS initiator port. SMP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh for SMP connection requests.

The DESTINATION SAS ADDRESS field specifies the port identifier (see 4.2.7) of the SAS port to which a connection is being requested.

The SOURCE SAS ADDRESS field specifies the port identifier (see 4.2.7) of the SAS port that originated the OPEN address frame.

The SOURCE ZONE GROUP field identifies the zone group of the phy making the connection request. Zone group values between 128 and 255, inclusive, are reserved. The SOURCE ZONE GROUP field shall be:

- a) set to zero when transmitted by an end device;
- b) set to zero when transmitted by an expander device on a phy with the INSIDE ZPSDS bit set to zero;
- c) set to the source zone group for the outgoing connection request as described in table 33 (see 4.9.3.5) when transmitted by an expander device on a phy with the INSIDE ZPSDS bit set to one;
- d) ignored when received by an end device;
- e) ignored when received by an expander device on a phy with the INSIDE ZPSDS bit set to zero; and
- f) used to determine the source zone group for the incoming connection request as described in table 33 (see 4.9.3.5) when received by an expander device on a phy with the INSIDE ZPSDS bit set to one.

The PATHWAY BLOCKED COUNT field specifies the number of times the port has retried this connection request due to receiving OPEN_REJECT (PATHWAY BLOCKED), OPEN_REJECT (RESERVED STOP 0), or OPEN_REJECT (RESERVED STOP 1). The port shall not increment the PATHWAY BLOCKED COUNT value past FFh. If the port changes connection requests, it shall set the PATHWAY BLOCKED COUNT field to 00h.

The ARBITRATION WAIT TIME field specifies how long the port transmitting the OPEN address frame has been waiting for a connection request to be accepted or rejected. This time is maintained by the port layer in an Arbitration Wait Time timer (see 8.2.2). For values from 0000h to 7FFFh, the Arbitration Wait Time timer increments in one microsecond steps. For values from 8000h to FFFFh, the Arbitration Wait Time timer

increments in one millisecond steps. The maximum value represents 32 767 ms + 32 768 μ s. Table 131 describes several values of the ARBITRATION WAIT TIME field. See 7.12.3 for details on arbitration fairness.

Table 131 — ARBITRATION WAIT TIME field

Code	Description
0000h	0 μ s
0001h	1 μ s
...	...
7FFFh	32 767 μ s
8000h	0 ms + 32 768 μ s
8001h	1 ms + 32 768 μ s
...	...
FFFFh	32 767 ms + 32 768 μ s

The MORE COMPATIBLE FEATURES field shall be set to zero. The destination phy shall ignore the MORE COMPATIBLE FEATURES field.

The CRC field is defined in 7.8.1.

7.9 Identification and hard reset sequence

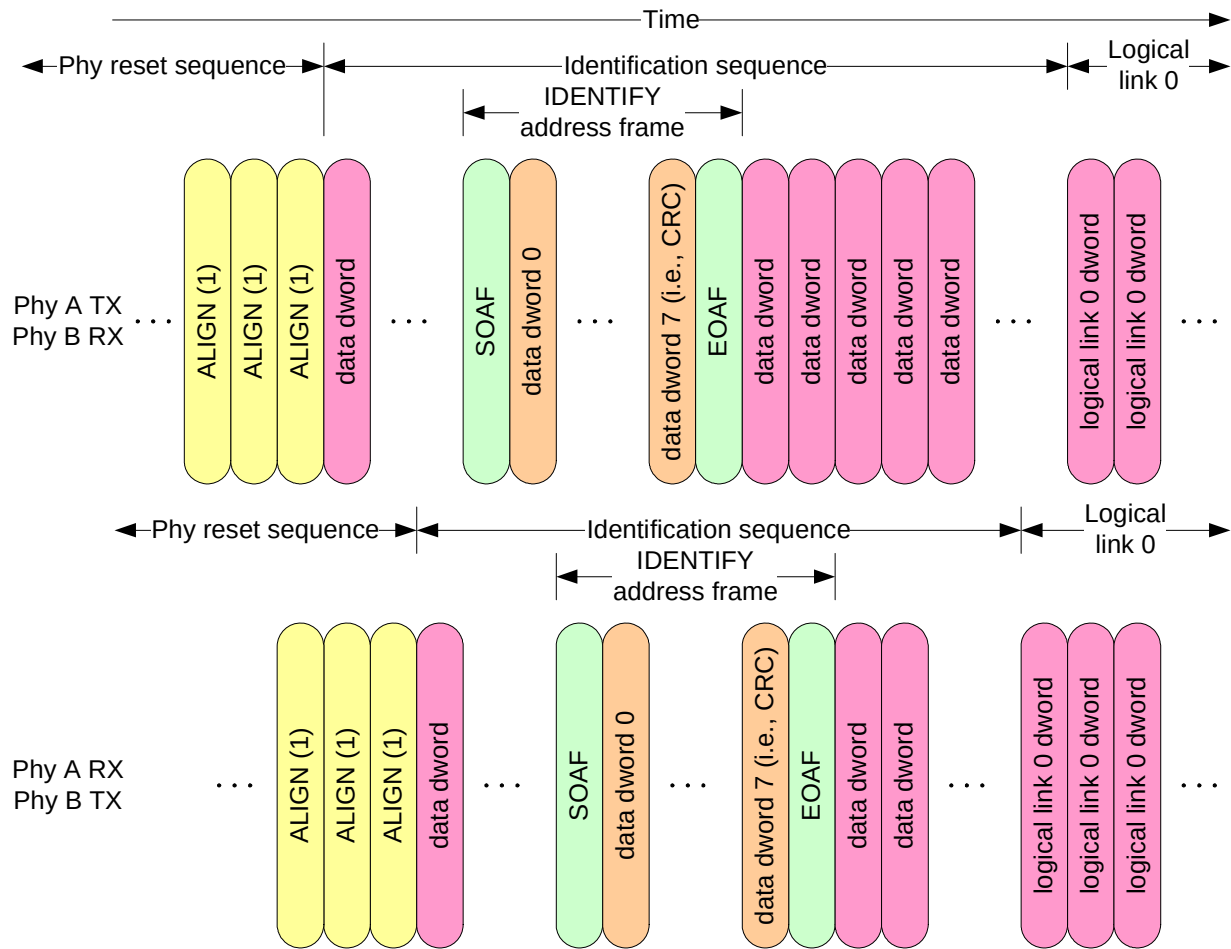
7.9.1 Identification and hard reset sequence overview

After the phy reset sequence (see 6.7) has been completed indicating the physical link is using SAS rather than SATA, each logical phy either:

- a) performs an identification sequence by transmitting one or three IDENTIFY address frames (see 7.8.2); or
- b) performs a hard reset sequence by transmitting a HARD_RESET primitive sequence (see 7.2.5.9).

If multiplexing (see 6.10) is enabled (see table 92 in 6.7.4.2.3.3), the identification sequence and hard reset sequence begin with a multiplexing sequence (see 6.7.4.3).

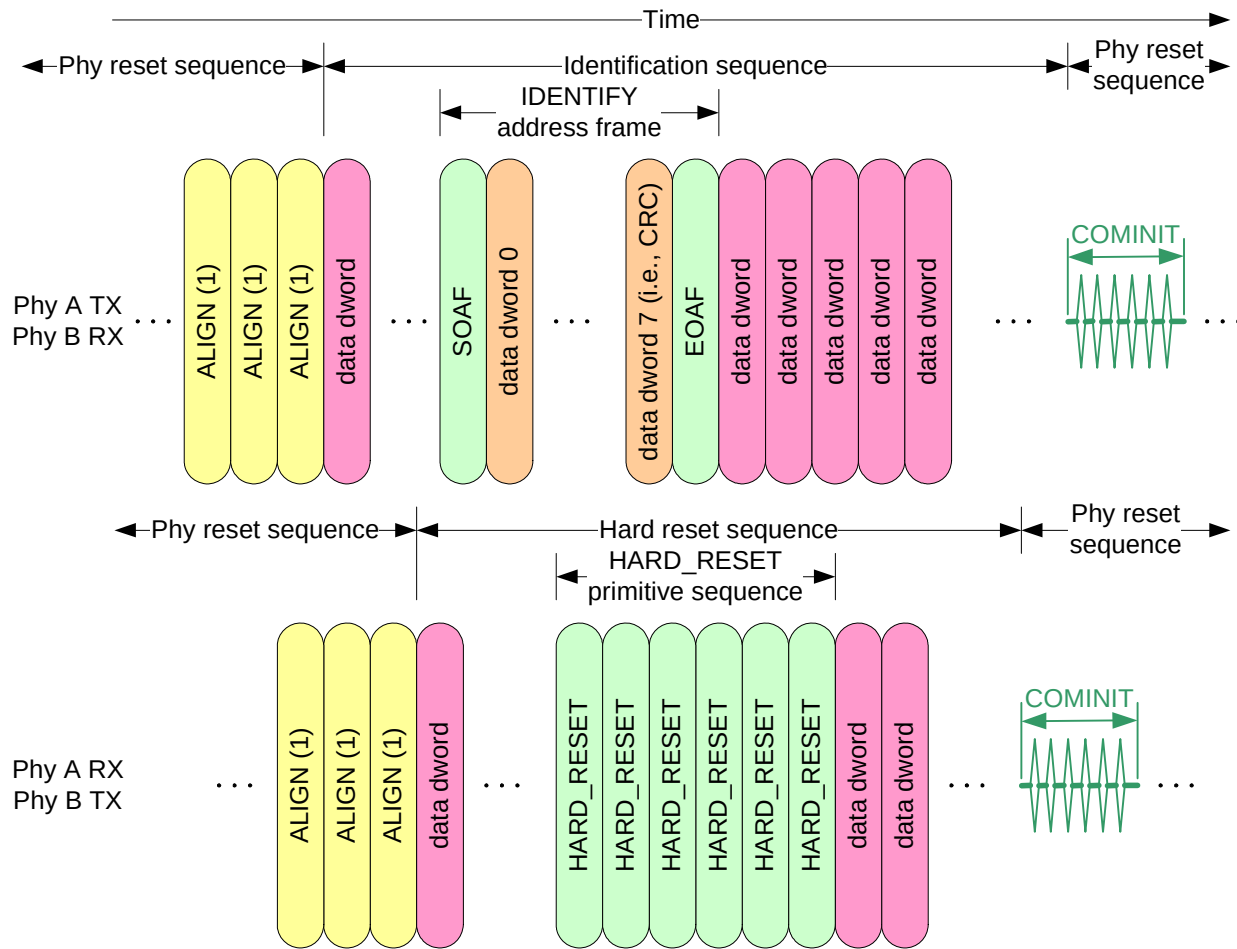
Figure 164 shows two phys with multiplexing disabled performing the identification sequence. Only one IDENTIFY address frame is shown in this example.



NOTE: Phys transmit deletable primitives for clock skew management after the phy reset sequence.

Figure 164 — Identification sequence

Figure 165 shows phy A performing the identification sequence and phy B performing the hard reset sequence. Multiplexing is disabled and only one IDENTIFY address frame is shown in this example.



NOTE: Phys transmit deletable primitives for clock skew management after the phy reset sequence.

Figure 165 — Hard reset sequence

Each logical phy receives an IDENTIFY address frame or a HARD_RESET primitive sequence from the logical phy to which it is attached. The combination of a phy reset sequence, an optional hard reset sequence followed by another phy reset sequence, and an identification sequence is called a link reset sequence (see 4.4.1).

If a phy receives a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, the SAS address in the outgoing IDENTIFY address frame(s) and the SAS address in the incoming IDENTIFY address frame determine the port to which a phy belongs (see 4.1.4). The phy ignores subsequent IDENTIFY address frames and HARD_RESET primitives until another phy reset sequence occurs.

If a phy receives a HARD_RESET primitive sequence within 1 ms of phy reset sequence completion, it shall be considered a reset event and cause a hard reset (see 4.4.2) of the port containing that phy.

If a phy does not receive a HARD_RESET primitive sequence or a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, it shall restart the phy reset sequence.

7.9.2 SAS initiator device rules

After a link reset sequence, or after receiving a Broadcast (Change), a management application client behind an SMP initiator port should perform a discover process (see 4.7).

When a discover process is performed after a link reset sequence, the management application client discovers all the devices in the SAS domain. When a discover process is performed after a Broadcast (Change), the management application client determines which devices have been added to or removed from the SAS domain.

The discover information may be used to select connection rates for connection requests (see 7.8.3).

7.9.3 Expander device rules

After completing the link reset sequence on a phy and completing internal initialization, the ECM within an expander device shall be capable of routing connection requests through that phy. The expander device may return OPEN_REJECT (NO DESTINATION) until it is ready to process connection requests.

After a link reset sequence, or after receiving a Broadcast (Change), the management application client behind an SMP initiator port in a self-configuring expander device shall follow the SAS initiator device rules (see 7.9.2) to perform a discover process (see 4.7).

The ECM of an externally configurable expander device is dependent on the completion of the discover process (see 4.7) for routing connection requests using the table routing method.

Editor's Note 37: Mentioning Broadcast (Change) in 7.9.2 and 7.9.3 is not ideal, since 7.9 is the identification and hard reset sequence section. The general application client rules for handling direct attached resets as well as Broadcast (Changes) belongs elsewhere.

7.9.4 SL_IR (link layer identification and hard reset) state machines

7.9.4.1 SL_IR state machines overview

The SL_IR (link layer identification and hard reset) state machines control the flow of dwords on the physical link that are associated with the identification and hard reset sequences. The state machines are as follows:

- a) SL_IR_TIR (transmit IDENTIFY or HARD_RESET) state machine (see 7.9.4.3);
- b) SL_IR_RIF (receive IDENTIFY address frame) state machine (see 7.9.4.4); and
- c) SL_IR_IRC (identification and hard reset control) state machine (see 7.9.4.5).

The SL_IR state machines send the following messages to the SL state machines (see 7.14) in SAS devices or the XL (see 7.15) state machine in expander devices:

- a) Enable Disable SAS Link (Enable); and
- b) Enable Disable SAS Link (Disable).

The SL_IR_IRC state machine shall maintain the timers listed in table 132.

Table 132 — SL_IR_IRC timers

Timer	Initial value
Receive Identify Timeout timer	1 ms

Figure 166 shows the SL_IR state machines.

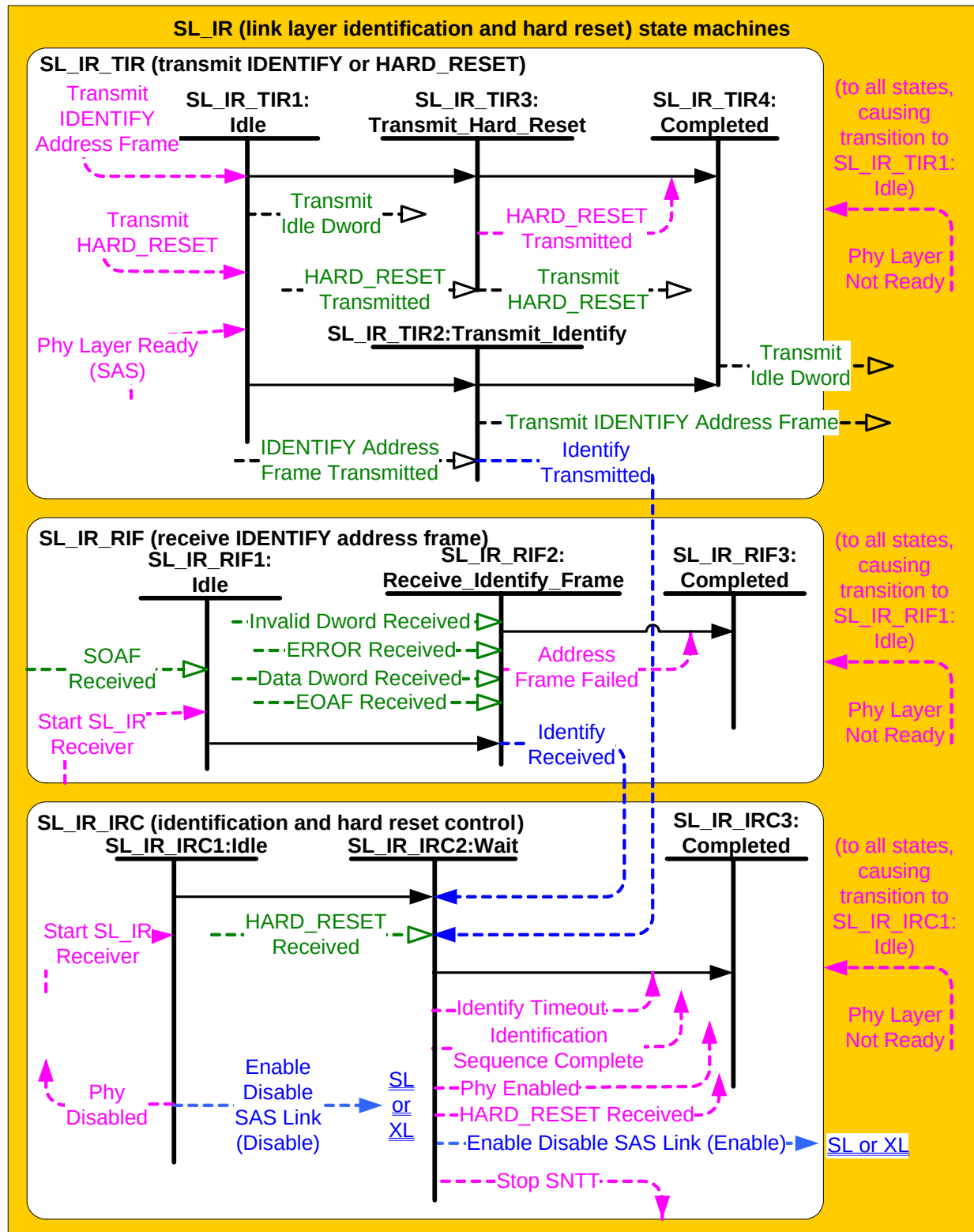


Figure 166 — SL_IR (link layer identification and hard reset) state machines

7.9.4.2 SL_IR transmitter and receiver

The SL_IR transmitter receives the following messages from the SL_IR state machines indicating primitive sequences, frames, and dwords to transmit:

- a) Transmit IDENTIFY Address Frame;
- b) Transmit HARD_RESET; and
- c) Transmit Idle Dword.

Upon receiving a Transmit IDENTIFY Address Frame message, the SL_IR transmitter shall transmit:

- 1) SOAF;
- 2) data dwords;
- 3) EOAF; and
- 4) at least 3 idle dwords.

NOTE 48 - Phys compliant with previous versions of this standard were not required to transmit idle dwords after EOAF.

The SL_IR transmitter sends the following messages to the SL_IR state machines:

- a) HARD_RESET Transmitted; and
- b) IDENTIFY Address Frame Transmitted.

The SL_IR receiver sends the following messages to the SL_IR state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

- a) SOAF Received;
- b) Data Dword Received;
- c) EOAF Received;
- d) ERROR Received;
- e) Invalid Dword Received; and
- f) HARD_RESET Received.

The SL_IR receiver shall not require reception of any idle dwords after an IDENTIFY address frame.

The SL_IR receiver shall ignore all other dwords.

The SL_IR transmitter relationship to other transmitters is defined in 4.3.2. The SL_IR receiver relationship to other receivers is defined in 4.3.3.

7.9.4.3 SL_IR_TIR (transmit IDENTIFY or HARD_RESET) state machine

7.9.4.3.1 SL_IR_TIR state machine overview

The SL_IR_TIR state machine's function is to transmit one or three IDENTIFY address frames or a HARD_RESET primitive after the phy layer enables the link layer. This state machine consists of the following states:

- a) SL_IR_TIR1:Idle (see 7.9.4.3.2)(initial state);
- b) SL_IR_TIR2:Transmit_Identify (see 7.9.4.3.3);
- c) SL_IR_TIR3:Transmit_Hard_Reset (see 7.9.4.3.4); and
- d) SL_IR_TIR4:Completed (see 7.9.4.3.5).

This state machine shall start in the SL_IR_TIR1:Idle state. This state machine shall transition to the SL_IR_TIR1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

Editor's Note 38: need to clarify how SP works with two SL_IRs in multiplexing. Does each one send Stop SNTT upon receiving its IDENTIFY address frame?

7.9.4.3.2 SL_IR_TIR1:Idle state**7.9.4.3.2.1 State description**

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL_IR transmitter.

7.9.4.3.2.2 Transition SL_IR_TIR1:Idle to SL_IR_TIR2:Transmit_Identify

This transition shall occur after both:

- a) a Phy Layer Ready (SAS) confirmation is received; and
- b) a Transmit IDENTIFY Address Frame request is received.

7.9.4.3.2.3 Transition SL_IR_TIR1:Idle to SL_IR_TIR3:Transmit_Hard_Reset

This transition shall occur after both:

- a) a Phy Layer Ready (SAS) confirmation is received; and
- b) a Transmit HARD_RESET request is received.

7.9.4.3.3 SL_IR_TIR2:Transmit_Identify state**7.9.4.3.3.1 State description**

Upon entry into this state, this state shall send either one or three Transmit IDENTIFY Address Frame messages to the SL_IR transmitter.

NOTE 49 - Phys compliant with previous versions of this standard only transmitted one Transmit IDENTIFY Address Frame message.

After this state receives an IDENTIFY Address Frame Transmitted message in response to its first Transmit IDENTIFY Address Frame message, this state shall send an Identify Transmitted message to the SL_IR_IRC state machine.

7.9.4.3.3.2 Transition SL_IR_TIR2:Transmit_Identify to SL_IR_TIR4:Completed

If this state sends one Transmit IDENTIFY Address Frame message, this transition shall occur after sending an Identify Transmitted message to the SL_IR_IRC state machine.

If this state sends three Transmit IDENTIFY Address Frame messages, this transition shall occur after receiving three Identify Transmitted messages.

7.9.4.3.4 SL_IR_TIR3:Transmit_Hard_Reset state**7.9.4.3.4.1 State description**

Upon entry into this state, this state shall send a Transmit HARD_RESET message to the SL_IR transmitter.

After this state receives a HARD_RESET Transmitted message, this state shall send a HARD_RESET Transmitted confirmation to the management application layer.

7.9.4.3.4.2 Transition SL_IR_TIR3:Transmit_Hard_Reset to SL_IR_TIR4:Completed

This transition shall occur after sending a HARD_RESET Transmitted confirmation to the management application layer.

7.9.4.3.5 SL_IR_TIR4:Completed state

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL_IR transmitter.

7.9.4.4 SL_IR_RIF (receive IDENTIFY address frame) state machine

7.9.4.4.1 SL_IR_RIF state machine overview

The SL_IR_RIF state machine receives an IDENTIFY address frame and checks the IDENTIFY address frame to determine if the frame should be accepted or discarded by the link layer.

This state machine consists of the following states:

- a) SL_IR_RIF1:Idle (see 7.9.4.4.2)(initial state);
- b) SL_IR_RIF2:Receive_Identify_Frame (see 7.9.4.4.3); and
- c) SL_IR_RIF3:Completed (see 7.9.4.4.4).

This state machine shall start in the SL_IR_RIF1:Idle state. This state machine shall transition to the SL_IR_RIF1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.4.4.2 SL_IR_RIF1:Idle state

7.9.4.4.2.1 State description

This state waits for an SOAF to be received from the physical link, indicating an address frame is arriving.

7.9.4.4.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame

This transition shall occur after both:

- a) a Start SL_IR Receiver confirmation is received; and
- b) an SOAF Received message is received.

7.9.4.4.3 SL_IR_RIF2:Receive_Identify_Frame state

7.9.4.4.3.1 State description

This state receives the dwords of an address frame and the EOAF.

If this state receives an SOAF Received message, then this state shall discard any previously received dwords for the address frame, send an Address Frame Failed confirmation to the management application layer to indicate that an invalid address frame was received, and start receiving the new address frame.

If this state receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state shall discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid address frame was received.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOAF Received message and before an EOAF Received message, then this state shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid address frame was received.

After receiving an EOAF Received message, this state shall check if it the received frame is a valid IDENTIFY address frame.

This state shall accept an IDENTIFY address frame and send an Identify Received message to the SL_IR_IRC state machine if:

- a) the ADDRESS FRAME TYPE field is set to 0h (i.e., IDENTIFY);
- b) the number of bytes between the SOAF and EOAF is 32; and
- c) the CRC field contains a good CRC.

Otherwise, this state shall discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid address frame was received.

7.9.4.4.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed

This transition shall occur after sending an Identify Received message confirmation.

7.9.4.4.4 SL_IR_RIF3:Completed state

This state waits for a Phy Layer Not Ready confirmation.

7.9.4.5 SL_IR_IRC (identification and hard reset control) state machine**7.9.4.5.1 SL_IR_IRC state machine overview**

The SL_IR_IRC state machine ensures that IDENTIFY address frames have been both received and transmitted before enabling the rest of the link layer, and notifies the link layer if a HARD_RESET primitive sequence is received before an IDENTIFY address frame has been received.

This state machine consists of the following states:

- a) SL_IR_IRC1:Idle (see 7.9.4.5.2)(initial state);
- b) SL_IR_IRC2:Wait (see 7.9.4.5.3); and
- c) SL_IR_IRC3:Completed (see 7.9.4.5.4).

This state machine shall start in the SL_IR_IRC1:Idle state. This state machine shall transition to the SL_IR_IRC1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.4.5.2 SL_IR_IRC1:Idle state**7.9.4.5.2.1 State description**

This state waits for the link layer to be enabled. Upon entry into this state, this state shall:

- a) send an Enable Disable SAS Link (Disable) message to SL state machines (see 7.14) or XL state machine (see 7.15) halting any link layer activity; and
- b) send a Phy Disabled confirmation to the port layer and the management application layer indicating that the phy is not ready for use.

7.9.4.5.2.2 Transition SL_IR_IRC1:Idle to SL_IR_IRC2:Wait

This transition shall occur after a Start SL_IR Receiver confirmation is received.

7.9.4.5.3 SL_IR_IRC2:Wait state**7.9.4.5.3.1 State description**

This state ensures that an IDENTIFY address frame has been received by the SL_IR_RIF state machine and that an IDENTIFY address frame has been transmitted by the SL_IR_TIR state machine before enabling the rest of the link layer. The IDENTIFY address frames may be transmitted and received on the physical link in any order.

After this state receives an Identify Received message, it shall send a Stop SNTT request to the phy layer.

After this state receives an Identify Transmitted message, it shall initialize and start the Receive Identify Timeout timer. If an Identify Received message is received before the Receive Identify Timeout timer expires, this state shall:

- a) send an Identification Sequence Complete confirmation to the management application layer, with arguments carrying the contents of the incoming IDENTIFY address frame;
- b) send an Enable Disable SAS Link (Enable) message to the SL state machines (see 7.14) in a SAS logical phy or the XL state machine (see 7.15) in an expander logical phy indicating that the rest of the link layer may start operation; and
- c) send a Phy Enabled confirmation to the port layer and the management application layer indicating that the phy is ready for use.

If the Receive Identify Timeout timer expires before an Identify Received message is received, this state shall send an Identify Timeout confirmation to the management application layer to indicate that an identify timeout occurred.

If this state receives a HARD_RESET Received message before an Identify Received message is received, this state shall send a HARD_RESET Received confirmation to the port layer and the management application layer and a Stop SNTT request to the phy layer.

If this state receives a HARD_RESET Received message after an Identify Received message is received, the HARD_RESET Received message shall be ignored.

7.9.4.5.3.2 Transition SL_IR_IRC2:Wait to SL_IR_IRC3:Completed

This transition shall occur after sending a HARD_RESET Received confirmation, Identify Timeout confirmation, or an Identification Sequence Complete and an Phy Enabled confirmation.

7.9.4.5.4 SL_IR_IRC3:Completed state

This state waits for a Phy Layer Not Ready confirmation.

7.10 Power management

SATA interface power management is not supported in STP.

STP initiator ports shall not generate SATA_PMREQ_P, SATA_PMREQ_S, or SATA_PMACK. If an STP initiator port receives SATA_PMREQ_P or SATA_PMREQ_S, it shall reply with SATA_PMNAK.

If an expander device receives SATA_PMREQ_P or SATA_PMREQ_S from a SATA device while an STP connection is not open, it shall not forward it to any STP initiator port and shall reply with SATA_PMNAK. If one of these primitives arrives while an STP connection is open, it may forward the primitive to the STP initiator port.

SCSI idle and standby power conditions, implemented with the START STOP UNIT command (see SBC-3) and the Power Condition mode page (see SPC-4), may be supported by SSP initiator ports and SSP target ports as described in 10.2.10.

ATA idle and standby power modes, implemented with the IDLE, IDLE IMMEDIATE, STANDBY, STANDBY IMMEDIATE, and CHECK POWER MODE commands (see ATA8-ACS), may be supported by STP initiator ports. The ATA sleep power mode, implemented with the SLEEP command, shall not be used.

7.11 SAS domain changes (Broadcast (Change) usage)

After power on or receiving Broadcast (Change) via an SMP initiator port, the management application client should scan the SAS domain using the discover process (see 4.7) to search for SAS initiator devices, SAS target devices, and expander devices.

The expander device shall originate Broadcast (Change) from at least one phy in each expander port other than the expander port that is the cause for originating Broadcast (Change).

Expander devices shall originate Broadcast (Change) for the following expander phy-related reasons:

- a) after an expander phy's SP state machine transitions from the SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready state to the SP0:OOB_COMINIT state (see 6.8);

NOTE 50 - This occurs when the expander phy is reset or disabled with the SMP PHY CONTROL function DISABLE, LINK RESET, HARD RESET, or TRANSMIT SATA PORT SELECTION SIGNAL phy operations (see 10.4.3.27) as well as when dword synchronization is unexpectedly lost;

- b) after a virtual phy has been disabled with the SMP PHY CONTROL function DISABLE phy operation or started processing a reset requested by the LINK RESET or HARD RESET phy operations (see 10.4.3.27);
- c) after an expander phy's SP state machine reaches the SP26:SATA_SpinupHold state and sends a SATA Spinup Hold confirmation as defined in 6.8.7 and 6.11;

- d) after an expander phy's SP state machine receives a COMWAKE Detected message in states SP0:OOB_COMINIT, SP1:OOB_AwaitCOMX, SP3:OOB_AwaitCOMINIT_Sent, or SP4:OOB_COMSAS if the ATTACHED SATA PORT SELECTOR bit is set to zero in the SMP DISCOVER response (see 10.4.3.9) prior to receiving the COMWAKE detected message (see 6.8.3 and table 342 in 10.4.3.9);
- e) after an expander phy's SP state machine transitions from the SP1:OOB_AwaitCOMX state to the SP0:OOB_COMINIT state if the ATTACHED SATA PORT SELECTOR bit was set to one in the SMP DISCOVER response upon entry to SP1:OOB_AwaitCOMX, and if no COMWAKE detected message was received while in SP1:OOB_AwaitCOMX before the transition to SP0:OOB_COMINIT (see 6.8.3.3.2);
- f) after an expander phy completes the link reset sequence (see 7.9);
- g) after a virtual phy has been enabled or completed processing a reset requested by the SMP PHY CONTROL function LINK RESET or HARD RESET phy operations (see 10.4.3.27); and
- h) after an STP/SATA bridge receives an initial Register - Device to host FIS (see 9.3.1).

In zoning expander devices with zoning enabled, forwarding Broadcasts is subject to restrictions defined in 4.9.5.

In zoning expander devices with zoning enabled, a Broadcast (Change) for an expander phy-related reason shall be originated from the source zone group of the expander phy causing the Broadcast (Change) or from zone group 1.

Expander devices shall originate Broadcast (Change) for the following expander device-related reasons:

- a) after a self-configuring expander device has changed its CONFIGURING bit from one to zero in the SMP REPORT GENERAL response (see 10.4.3.3) as described in 4.7.2. In zoning expander devices with zoning enabled, the source zone group shall be 01h;
- b) after a locked expander device is unlocked (see 4.9.6.5 and 10.4.3.22), with the source zone group as specified in 4.9.6.5 and 10.4.3.22.

Expander devices shall forward Broadcast (Change) for the following reasons:

- a) after an expander phy receives Broadcast (Change).

For a virtual phy, if there is any time after a reset is originated during which connection requests to the attached SAS address result in connection responses of OPEN_REJECT (NO DESTINATION), the expander device shall originate the Broadcast (Change) twice, once at the start of the reset (i.e., when the SAS address becomes unavailable) and once at its completion (i.e., when the SAS address becomes available). If there is no such time window, the expander device shall originate the Broadcast (Change) once.

SAS initiator ports may originate Broadcast (Change) to force other SAS initiator ports and expander ports to re-run the discover process, but should not be sent by SAS target ports.

A SAS initiator port that receives Broadcast (Change) shall follow the SAS initiator device rules (see 7.9.2) to discover and configure the topology.

An expander device that receives Broadcast (Change) shall follow the expander device rules (see 7.9.3) to discover and configure the topology.

See 10.4.3.3 for details on counting Broadcast (Change) origination in an expander device.

7.12 Connections

7.12.1 Connections overview

A connection is opened between a SAS initiator port and a SAS target port before communication begins. A connection is established between one SAS initiator phy in the SAS initiator port and one SAS target phy in the SAS target port.

SSP initiator ports open SSP connections to transmit SCSI commands, task management functions, or transfer write data. SSP target ports open SSP connections to transfer read data or transmit status.

SMP initiator ports open SMP connections to transmit SMP requests and receive SMP responses.

STP initiator ports and STP target ports open STP connections to transmit SATA frames. An STP target port in an expander device opens STP connections on behalf of SATA devices.

The OPEN address frame is used to request that a connection be opened (see 7.12.2.1). AIP, OPEN_ACCEPT and OPEN_REJECT are the responses to an OPEN address frame (see 7.12.2.2). BREAK is used to abort connection requests (see 7.12.6) and to unilaterally break a connection (see 7.12.8). CLOSE is used for orderly closing a connection (see 7.12.7).

Connections use a single pathway from the SAS initiator phy to the SAS target phy. While a connection is open, only one pathway shall be used for that connection.

For STP connections, connections may be between the STP initiator port and an STP target port of an STP/SATA bridge in an expander device. The SATA device behind the STP/SATA bridge is not aware of SAS connection management.

A wide port may have separate connections on each of its phys.

7.12.2 Opening a connection

7.12.2.1 Connection request

The OPEN address frame (see 7.8.3) is used to open a connection from a source port to a destination port using one source phy and one destination phy.

To make a connection request, the source port shall transmit an OPEN address frame through an available phy. The source phy shall transmit idle dwords after the OPEN address frame until it receives a response or aborts the connection request with BREAK.

After transmitting an OPEN address frame, the source phy shall initialize and start a 1 ms Open Timeout timer. Whenever an AIP is received, the source phy shall reinitialize and restart the Open Timeout timer. Source phys are not required to enforce a limit on the number of AIPs received before aborting the connection request. When any connection response is received, the source phy shall reinitialize the Open Timeout timer. If the Open Timeout timer expires before a connection response is received, the source phy shall transmit BREAK to abort the connection request (see 7.12.6).

The OPEN address frame flows through expander devices onto intermediate physical links. If an expander device on the pathway is unable to forward the connection request, it returns OPEN_REJECT (see 7.12.4). If the OPEN address frame reaches the destination, it returns either OPEN_ACCEPT or OPEN_REJECT unless the OPEN address frame passed an OPEN address frame from the destination with higher arbitration priority (see 7.12.3). Rate matching shall be used on any physical links in the pathway with negotiated physical link rates that are faster than the requested connection rate (see 7.13).

A wide port should not attempt to establish more connections to a destination port than the destination port width or the width of the narrowest physical link on the pathway to the destination port. A wide port should not attempt to establish more connections than the width of the narrowest common physical link on the pathways to the destination ports of those connections. Additional requirements for STP connection requests are defined in 7.17.5. Additional requirements for SMP connection requests are defined in 7.18.3.

Figure 167 shows an example of the simultaneous connection recommendations for wide ports.

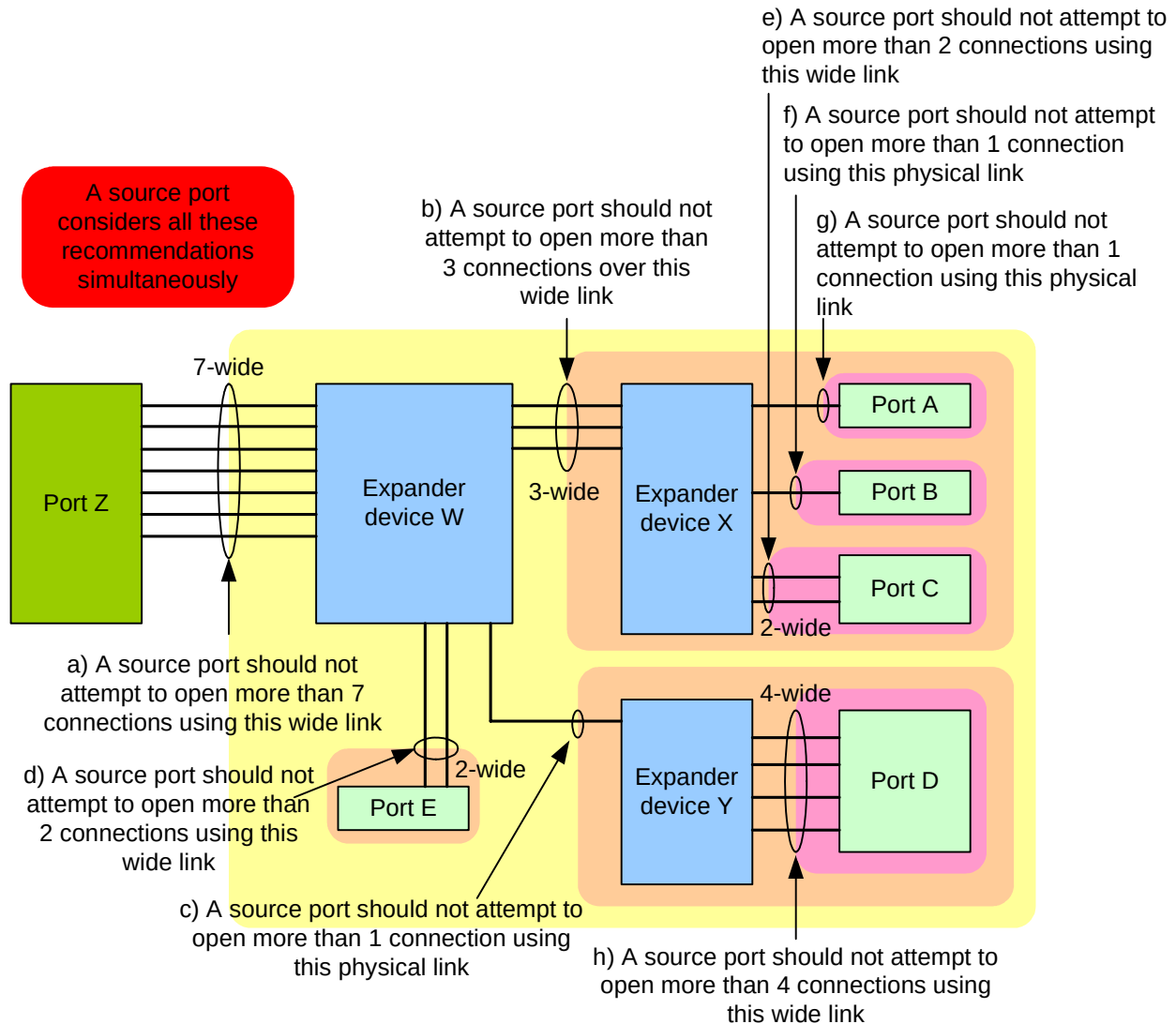


Figure 167 — Example simultaneous connection recommendations for wide ports

In figure 167, some of the recommendations are combined as follows:

- a) recommendations a), b), and e) together mean port Z should not attempt to open more than 2 connections to port C;
- b) recommendations a), b), e), f), and g) together mean that if port Z has 2 connections open to ports A, B, and X, it should not attempt to open more than 1 connection to port C. If it has 6 connections open to ports A, B, D, E, W, X, and Y, it should not attempt to open more than 1 connection to port C; and
- c) recommendations a), c), and h) together mean port Z should not attempt to open more than 1 connection to port D. If it has a connection open to port Y, it should not attempt to open another connection to port D until the first connection is closed.

7.12.2.2 Results of a connection request

After a phy transmits an OPEN address frame, it shall expect one or more of the results listed in table 133.

Table 133 — Connection Results of a connection request

Result	Description
Receive AIP	Arbitration in progress. When an expander device is trying to open a connection to the selected destination port, it returns an AIP to the source phy. The source phy shall reinitialize and restart its Open Timeout timer each time it receives an AIP. AIP is sent by an expander device while it is internally arbitrating for access to an expander port.
Receive OPEN_ACCEPT	Connection request accepted. OPEN_ACCEPT is transmitted by the destination phy.
Receive OPEN_REJECT	Connection request rejected. OPEN_REJECT is transmitted by the destination phy or by an expander device in the partial pathway. The different versions are described in 7.2.5.13. See 4.5 for I_T nexus loss handling. See 7.8.3 for handling of OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) for connection rates greater than 1,5 Gbps.
Receive OPEN address frame	If AIP has been previously received, this indicates an overriding connection request. If AIP has not yet been received, this indicates two connection requests crossing on the physical link. Arbitration fairness determines which one wins (see 7.12.3).
Receive BREAK	The destination phy or an expander device in the partial pathway may reply with BREAK indicating the connection is not being established.
Open Timeout timer expires	The source phy shall abort the connection request by transmitting BREAK (see 7.12.6). See 4.5 for I_T nexus loss handling.

7.12.3 Arbitration fairness

SAS supports least-recently used arbitration fairness for connection requests.

Each SAS port and expander port shall include an Arbitration Wait Time timer which counts the time from the moment when the port makes a connection request until the request is accepted or rejected. The Arbitration Wait Time timer is in the port layer state machine (see 8.2.2). The Arbitration Wait Time timer shall count in microseconds from 0 μ s to 32 767 μ s and in milliseconds from 32 768 μ s to 32 767 ms + 32 768 μ s. The Arbitration Wait Time timer shall stop incrementing when its value reaches 32 767 ms + 32 768 μ s.

A SAS port (i.e., SAS initiator ports and SAS target ports) shall start the Arbitration Wait Time timer when it transmits the first OPEN address frame (see 7.8.3) for the connection request. When the SAS port retransmits the OPEN address frame (e.g., after losing arbitration and handling an inbound OPEN address frame), it shall set the ARBITRATION WAIT TIME field to the current value of the Arbitration Wait Time timer.

A SAS port should set the Arbitration Wait Time timer to zero when it transmits the first OPEN address frame for the connection request. A SAS initiator port or SAS target port may be unfair by setting the ARBITRATION WAIT TIME field in the OPEN address frame (see 7.8.3) to a higher value than its Arbitration Wait Time timer indicates. However, an unfair SAS port shall not set the ARBITRATION WAIT TIME field to a value greater than or equal to 8000h; this limits the amount of unfairness and helps prevent livelocks.

The expander port that receives an OPEN address frame shall set the Arbitration Wait Time timer to the value of the incoming ARBITRATION WAIT TIME field and start the Arbitration Wait Time timer as it arbitrates for internal access to the outgoing expander port. When the expander device transmits the OPEN address frame out another expander port, it shall set the outgoing ARBITRATION WAIT TIME field to the current value of the Arbitration Wait Time timer maintained by the incoming expander port.

A SAS port shall stop the Arbitration Wait Time timer and set it to zero when it has no more frames to send.

A SAS port shall stop the Arbitration Wait Time timer and set it to zero when it receives one of the following connection responses:

- a) OPEN_ACCEPT;
- b) OPEN_REJECT (PROTOCOL NOT SUPPORTED);
- c) OPEN_REJECT (RESERVED ABANDON 1);
- d) OPEN_REJECT (RESERVED ABANDON 2);
- e) OPEN_REJECT (RESERVED ABANDON 3);
- f) OPEN_REJECT (STP RESOURCES BUSY); or
- g) OPEN_REJECT (WRONG DESTINATION).

NOTE 51 - Connection responses that are conclusively from the destination phy (see table 114 and table 115 in 7.2.5.13) are included in the list. Except for OPEN_REJECT (RETRY), connection responses that are only from or may be from expander phys are not included.

When an OPEN_REJECT (RETRY) is received:

- a) if the CONTINUE AWT bit is set to one in the Protocol-Specific Port mode page (see 10.2.7.4), then a connection response of OPEN_REJECT (RETRY) shall not stop the Arbitration Wait Time timer and shall not set it to zero; or
- b) If the CONTINUE AWT bit is set to zero, then a SAS port shall stop the Arbitration Wait Time timer and set it to zero.

A SAS port should not stop the Arbitration Wait Time timer and set it to zero when it receives an incoming OPEN address frame that has priority over the outgoing OPEN address frame according to table 134, regardless of whether it replies with an OPEN_ACCEPT or an OPEN_REJECT.

When arbitrating for access to an outgoing expander port, the expander device shall select the connection request based on the rules described in 7.12.4.

If two connection requests pass on a physical link, the phy shall determine the winner by comparing OPEN address frame field contents using the arbitration priority described in table 134.

Table 134 — Arbitration priority for OPEN address frames passing on a physical link

Bits 79-64 (79 is MSB)	Bits 63-0 (0 is LSB)
ARBITRATION WAIT TIME field value	SOURCE SAS ADDRESS field value

See 7.8.3 for details on the OPEN address frame and the ARBITRATION WAIT TIME field.

7.12.4 Arbitration inside an expander device

7.12.4.1 Expander phy arbitration requirements

An expander phy shall set its Request Path request Retry Priority Status argument to IGNORE AWT when it requests a path after:

- a) it has forwarded an OPEN address frame to the physical link;
- b) an OPEN address frame is received with higher arbitration priority (see 7.12.3); and
- c) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame (see 7.15.4 and 7.15.9).

Otherwise, the expander phy shall set the Retry Priority Status argument to NORMAL.

See the XL state machine (see 7.15) for detailed expander phy requirements.

7.12.4.2 ECM arbitration requirements

7.12.4.2.1 ECM arbitration requirements overview

The ECM shall arbitrate and assign or deny path resources for Request Path requests (see 4.6.6.3) from each expander phy.

Arbitration includes adherence to the SAS arbitration fairness algorithm and path recovery. Path recovery is used to avoid potential deadlock scenarios within the SAS topology by deterministically choosing which partial pathway(s) to tear down to allow at least one connection to complete.

Several of the Request Path arguments are used for arbitration. The Arbitration Wait Time, Source SAS Address, and Connection Rate arguments are filled in from the received OPEN address frame and are used to by the ECM to compare Request Path requests. The Retry Priority Status argument is used to prevent the Arbitration Wait Time argument from being considered during an arbitration which occurs after a Backoff Retry response is sent by an expander phy (see 7.15.4).

When the ECM in an expander device receives a connection request:

- 1) if the destination SAS address is that of the expander device itself, the ECM shall arbitrate for access to its SMP target port;
- 2) if the destination SAS address matches the SAS address attached to one or more of the expander phys, the ECM shall arbitrate for access to those expander phys;
- 3) if the destination SAS address matches an enabled SAS address in the expander route table for one or more expander phys that is using the table routing method, the ECM shall arbitrate for access to those expander phys; and
- 4) if at least one expander phy is using the subtractive routing method, and the request did not come from one of those expander phys, the ECM shall arbitrate for access to one of those expander phys.

The ECM shall respond to each Request Path request by returning the following confirmations to the requesting expander logical phy while processing the Request Path request:

- a) Arbitrating (Normal) (see 7.12.4.2.2);
- b) Arbitrating (Waiting On Partial) (see 7.12.4.2.2);
- c) Arbitrating (Blocked On Partial) (see 7.12.4.2.2);
- d) Arbitrating (Waiting On Connection) (see 7.12.4.2.2).

The ECM shall complete responding to each Request Path request by returning one of the following confirmations to the requesting expander logical phy:

- a) Arb Won (see 7.12.4.2.3);
- b) Arb Lost (see 7.12.4.2.4); or
- c) Arb Reject (see 7.12.4.2.5).

7.12.4.2.2 Arbitrating confirmations

The ECM shall send an Arbitrating (Normal) confirmation after it has received a Request Path request.

The ECM shall send an Arbitrating (Waiting On Partial) confirmation if it is waiting on a partial pathway (see 4.1.11). The ECM is waiting on a partial pathway if:

- a) there is a destination port capable of routing to the requested destination SAS address;
- b) at least one phy within the destination port supports the requested connection rate;
- c) each of the phys within the destination port is returning a Phy Status (Partial Pathway) or Phy Status (Blocked Partial Pathway) response; and
- d) at least one of the phys within the destination port is returning a Phy Status (Partial Pathway) response.

The ECM shall send an Arbitrating (Blocked On Partial) confirmation if it is waiting on a blocked partial pathway (see 4.1.11). The ECM is waiting on a blocked partial pathway if:

- a) there is a destination port capable of routing to the requested destination SAS address;
- b) at least one phy within the destination port supports the requested connection rate; and

- c) each of the phys within the destination port is returning a Phy Status (Blocked Partial Pathway) response.

The ECM shall send an Arbitrating (Waiting On Connection) confirmation if it is waiting on a connection to complete (see 4.1.12). The ECM is waiting on a connection to complete if:

- a) the connection request is blocked by an active connection; or
- b) there are insufficient routing resources within the expander to complete the connection request.

A connection request shall be considered blocked by an active connection when:

- a) there is a destination port capable of routing to the requested destination SAS address;
- b) at least one phy within the destination port supports the requested connection rate;
- c) each of the phys within the destination port is returning a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response; and
- d) at least one of the phys within the destination port is returning a Phy Status (Connection) response.

7.12.4.2.3 Arb Won confirmation

The ECM shall generate the Arb Won confirmation when all of the following conditions are met:

- a) the Request Path request maps to a destination expander phy that:
 - A) supports the connection rate; and
 - B) is not reporting a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response, unless that expander phy is arbitrating for the requesting expander phy;
- b) there are sufficient routing resources to complete the connection request;
- c) no higher priority Request Path requests are present with the requesting expander phy as the destination; and
- d) the Request Path request is the highest priority Request Path request (see table 135 and table 136) mapping to the destination expander phy (i.e., only send one Arb Won confirmation for Request Path requests to the same destination phy).

If two or more Request Path requests contend and all of the Request Path requests include a Retry Priority Status argument set to NORMAL, the ECM shall select the winner by comparing the OPEN address frame contents described in table 135.

Table 135 — Arbitration priority for contending Request Path requests in the ECM when all requests have Retry Priority Status arguments of NORMAL

Bits 83-68 (83 is MSB)	Bits 67-4	Bits 3-0 (0 is LSB)
ARBITRATION WAIT TIME field value	SOURCE SAS ADDRESS field value	CONNECTION RATE field value

If two or more Request Path requests contend and one or more of the Request Path requests include a Retry Priority Status argument set to IGNORE AWT, the ECM shall select the winner from the set of Request Path requests with Retry Priority Status arguments set to IGNORE AWT by comparing the OPEN address frame contents described in table 136.

Table 136 — Arbitration priority for contending Request Path requests in the ECM among requests with Retry Priority Status arguments of IGNORE AWT

Bits 67-4 (67 is MSB)	Bits 3-0 (0 is LSB)
SOURCE SAS ADDRESS field value	CONNECTION RATE field value

7.12.4.2.4 Arb Lost confirmation

The ECM shall generate the Arb Lost confirmation when all of the following conditions are met:

- a) the Request Path request maps to a destination expander logical phy that:

- A) supports the connection rate; and
- B) is not reporting a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response unless that expander logical phy is arbitrating for the requesting expander logical phy;
- b) there are sufficient routing resources to complete the connection request; and
- c) one of the following conditions are met:
 - A) the destination expander logical phy is making a Request Path request with the requesting expander logical phy as its destination (i.e., when two expander logical phys both receive an OPEN address frame destined for each other, the ECM provides the Arb Lost confirmation to the expander logical phy that received the lowest priority OPEN address frame); or
 - B) the ECM is sending an Arb Won confirmation to another expander logical phy that is using the requesting expander logical phy as the destination.

7.12.4.2.5 Arb Reject confirmation

The ECM shall generate the following Arb Reject confirmation when any of the following conditions are met and all the Arb Won conditions (see 7.12.4.2.3) are not met:

- 1) Arb Reject (Bad Destination) if the source expander logical phy and destination expander logical phy(s) are in the same expander port and are using the direct routing method;
- 2) Arb Reject (Retry) if the expander device is unable to process the connection request because it has reduced functionality (see 4.6.8);
- 3) if the source expander logical phy and destination expander logical phy(s) are in the same expander port and are using the table routing method or the subtractive routing method:
 - A) Arb Reject (No Destination) if the expander device is not configuring; and
 - B) Arb Reject (Retry) if the expander device is configuring;
- 4) if there are no destination expander logical phys (i.e., there is no direct routing or table routing match and there is no subtractive phy):
 - A) Arb Reject (No Destination) if the expander device is not configuring; and
 - B) Arb Reject (Retry) if the expander device is configuring;
- 5) if access to the destination expander logical phy(s) is prohibited by zoning (see 4.9.3):
 - A) Arb Reject (Zone Violation) if the zoning expander device is unlocked; and
 - B) Arb Reject (Retry) if the zoning expander device is locked;
- 6) Arb Reject (Connection Rate Not Supported) if none of the destination expander logical phys supports the connection rate; and
- 7) Arb Reject (Pathway Blocked) if all the destination expander logical phys that support the connection rate contain blocked partial pathways (i.e., are all returning Phy Status (Blocked Partial Pathway)) and pathway recovery rules require this Request Path request be rejected to release path resources (see 7.12.4.5).

7.12.4.3 Arbitration status

Arbitration status shall be conveyed between expander devices and by expander devices to SAS endpoints using AIP (see 7.2.5.1). This status is used to monitor the progress of connection attempts and to facilitate pathway recovery as part of deadlock recovery.

The arbitration status of an expander phy is set to the last type of AIP received.

Before an expander device transmits AIP, it may have transmitted an OPEN address frame on the same physical link. Arbitration fairness dictates which OPEN address frame wins (see 7.12.3).

After an expander device transmits an AIP, it shall not transmit an OPEN address frame unless it has higher arbitration priority than the incoming connection request.

After transmitting an AIP primitive sequence, an expander device shall transmit at least one other dword (e.g., an idle dword) before transmitting another AIP primitive sequence.

Expander devices shall transmit at least one AIP every 128 dwords while originating AIP (NORMAL), AIP (WAITING ON PARTIAL), or AIP (WAITING ON CONNECTION).

NOTE 52 - Expander devices compliant with previous versions of this standard were not required to transmit three consecutive AIP primitives, as AIP was defined as a single primitive sequence (see 7.2.4.2) rather than as an extended primitive sequence (see 7.2.4.5).

Expander devices shall transmit an AIP (e.g., an AIP (NORMAL)) within 128 dwords of receiving an OPEN address frame.

7.12.4.4 Partial Pathway Timeout timer

Each expander phy shall maintain a Partial Pathway Timeout timer. This timer is used to identify potential deadlock conditions and to request resolution by the ECM. An expander phy shall initialize the Partial Pathway Timeout timer to the time it reports in the PARTIAL PATHWAY TIMEOUT VALUE field in the SMP DISCOVER response (see 10.4.3.9) and run the Partial Pathway Timeout timer whenever the ECM provides confirmation to the expander phy that all expander phys within the requested destination port are blocked waiting on partial pathways.

NOTE 53 - The partial pathway timeout value allows flexibility in specifying how long an expander device waits before attempting pathway recovery. The recommended default value (see 10.4.3.9) was chosen to cover a wide range of topologies. Selecting small partial pathway timeout value values within a large topology may compromise performance because of the time a device waits after receiving OPEN_REJECT (PATHWAY BLOCKED) before it retries the connection request. Similarly, selecting large partial pathway timeout value values within a small topology may compromise performance due to waiting longer than necessary to detect pathway blockage.

When the Partial Pathway Timeout timer is not running, an expander phy shall initialize and start the Partial Pathway Timeout timer when all expander phys within the requested destination port contain a blocked partial pathway (i.e., are returning Phy Status (Blocked Partial Pathway)).

NOTE 54 - The Partial Pathway Timeout timer is not initialized and started if one or more of the expander phys within a requested destination port are being used for a connection.

When one of the conditions above is not met, the expander phy shall stop the Partial Pathway Timeout timer. If the timer expires, pathway recovery shall occur (see 7.12.4.5).

7.12.4.5 Pathway recovery

Pathway recovery provides a means to abort connection requests in order to prevent deadlock using pathway recovery priority comparisons. Pathway recovery priority comparisons compare the PATHWAY BLOCKED COUNT fields and SOURCE SAS ADDRESS fields of the OPEN address frames of the blocked connection requests as described in table 137.

Table 137 — Pathway recovery priority

Bits 71-64 (71 is MSB)	Bits 63-0 (0 is LSB)
PATHWAY BLOCKED COUNT field value	SOURCE SAS ADDRESS field value

When the Partial Pathway Timeout timer for an arbitrating expander phy expires (i.e., reaches a value of zero), the ECM shall determine whether to continue the connection request or to abort the connection request.

The ECM shall reply to a connection request with Arb Reject (Pathway Blocked) when:

- the Partial Pathway Timeout timer expires; and
- the pathway recovery priority of the arbitrating expander phy (i.e., the expander phy requesting the connection) is less than or equal to the pathway recovery priority of any of the expander phys within the destination port that are sending Phy Status (Blocked Partial Pathway) responses to the ECM.

The pathway blocked count and source SAS address values used to form the pathway recovery priority of a destination phy are those of the Request Path request if the phy sent a Request Path request to the ECM or those of the Forward Open indication if the phy received a Forward Open indication from the ECR.

7.12.5 BREAK handling

A phy aborts a connection request (see 7.12.6) and breaks a connection (see 7.12.8) by transmitting a BREAK primitive sequence.

Phys shall enable the BREAK_REPLY method of responding to received BREAK primitive sequences when:

- a) the BREAK_REPLY CAPABLE bit transmitted by the phy in the outgoing IDENTIFY address frame is set to one; and
- b) the BREAK_REPLY CAPABLE bit received by the phy in the incoming IDENTIFY address frame is set to one.

Phys shall disable the BREAK_REPLY method of responding to received BREAK primitive sequences if the BREAK_REPLY CAPABLE bit received by the phy in the incoming IDENTIFY address frame is set to zero.

Phys contained within SAS devices or expander devices that are compliant with this standard shall set the BREAK_REPLY CAPABLE bit to one in their outgoing IDENTIFY address frame.

If the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled, then the phy shall transmit a BREAK_REPLY primitive sequence in response to a received BREAK primitive sequence.

If the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled, then the phy shall transmit a BREAK primitive sequence in response to a received BREAK primitive sequence.

NOTE 55 - Phys compliant with earlier versions of this standard do not set the BREAK_REPLY CAPABLE bit in their outgoing IDENTIFY address frame.

7.12.6 Aborting a connection request

BREAK may be used to abort a connection request. The source phy shall transmit a BREAK after the Open Timeout timer expires or if it chooses to abort its request for any other reason before a connection is established.

After transmitting BREAK, the source phy shall initialize a Break Timeout timer to 1 ms and start the Break Timeout timer.

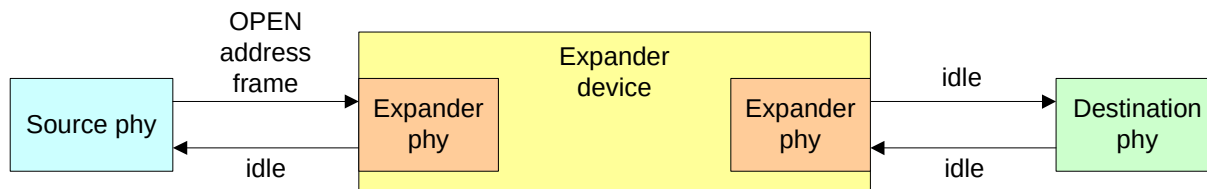
After a phy transmits a BREAK to abort a connection request, it shall expect one of the results listed in table 138.

Table 138 — Results of aborting a connection request

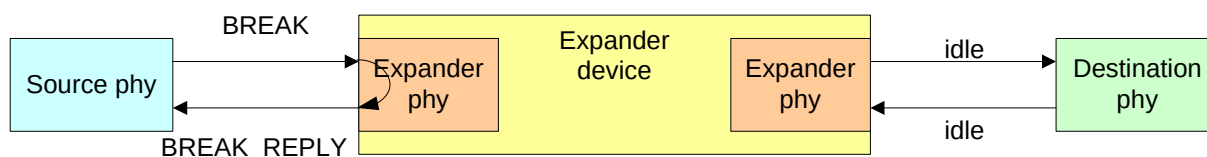
BREAK_REPLY method of responding to received BREAK primitive sequences	Result	Description
Disabled	Receive BREAK	This confirms that the connection request has been aborted.
	Receive BREAK_REPLY	Ignore.
Enabled	Receive BREAK	The originating phy shall transmit BREAK_REPLY and wait to receive BREAK_REPLY or for the BREAK Timeout timer to expire.
	Receive BREAK_REPLY	This confirms that the connection request has been aborted.
Enabled or disabled	Break Timeout timer expires	The originating phy shall assume the connection request has been aborted.

When a logical phy transmitting a BREAK is attached to an expander device, the BREAK or BREAK_REPLY response to the logical phy is generated by the expander logical phy to which the logical phy is attached, not the other SAS logical phy in the connection. If the expander device has transmitted a connection request to the destination, it shall also transmit BREAK to the destination. If the expander device has not transmitted a connection request to the destination, it shall not transmit BREAK to the destination. After transmitting BREAK or BREAK_REPLY back to the source phy, the expander device shall ensure that a connection response does not occur (i.e., the expander device shall not forward dwords from the destination any more). Figure 168 shows an example of BREAK usage. Multiplexing is disabled on all phys in the example.

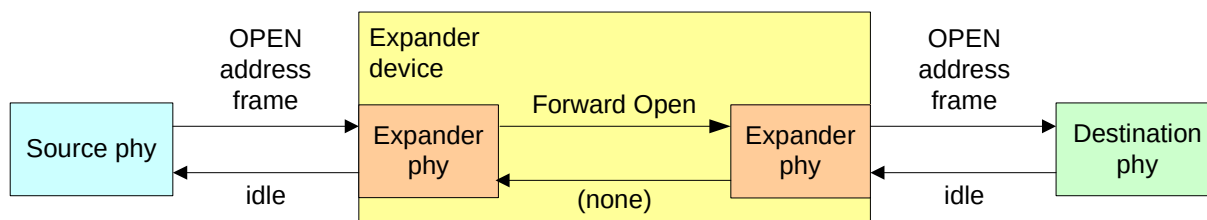
Case 1: OPEN address frame has not propagated through the expander device:



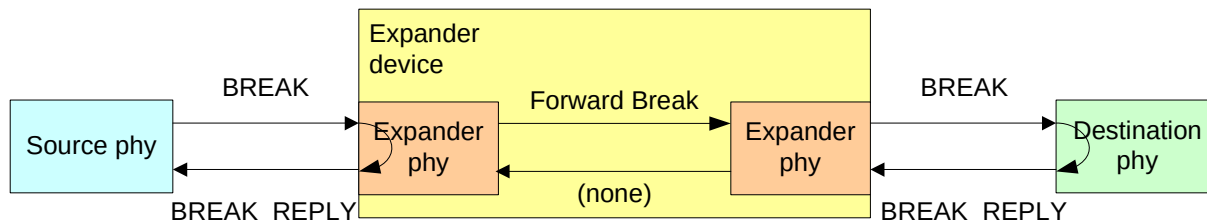
Case 1 result: expander device transmits BREAK_REPLY to the source phy



Case 2: OPEN address frame has propagated through the expander device:



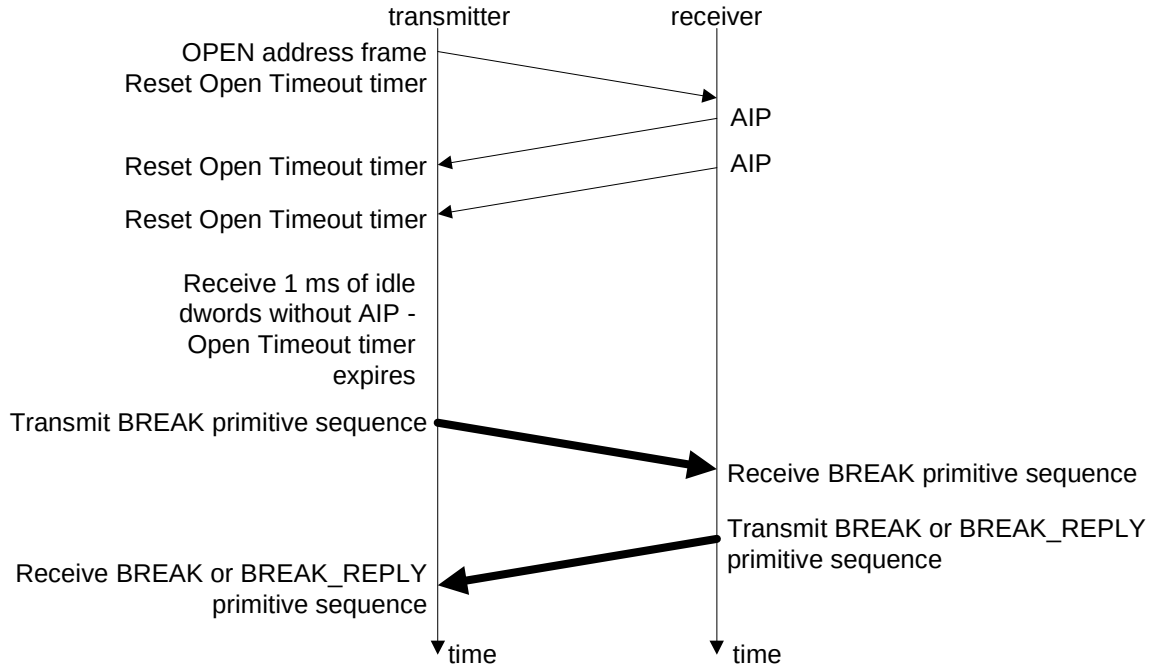
Case 2 result: Expander device transmits BREAK_REPLY to the source phy and BREAK to the destination phy, then waits for BREAK_REPLY from the destination phy



NOTE: If the BREAK_REPLY method of responding to BREAK primitive sequences is disabled, phys transmit BREAK rather than BREAK_REPLY in response to BREAK.

Figure 168 — Aborting a connection request with BREAK

Figure 169 shows the sequence for a connection request where the Open Timeout timer expires.



NOTE: If the BREAK_REPLY method of responding to BREAK primitive sequences is disabled, phys transmit BREAK rather than BREAK_REPLY in response to BREAK.

Figure 169 — Connection request timeout example

7.12.7 Closing a connection

CLOSE is used to close a connection of any protocol. See 7.16.7 for details on closing SSP connections, 7.17.6 for details on closing STP connections, and 7.18.4 for details on closing SMP connections.

After transmitting CLOSE, the source phy shall initialize a Close Timeout timer to 1 ms and start the Close Timeout timer.

After a phy transmits CLOSE to close a connection, it shall expect one of the results listed in table 139.

Table 139 — Results of closing a connection

Result	Description
Receive CLOSE	This confirms that the connection has been closed.
Close Timeout timer expires	The originating phy shall attempt to break the connection (see 7.12.8).

No additional dwords for the connection shall follow the CLOSE. Expander devices shall close the full-duplex connection upon forwarding a CLOSE in each direction.

When a phy has both transmitted and received CLOSE, it shall consider the connection closed.

Figure 170 shows example sequences for closing a connection.

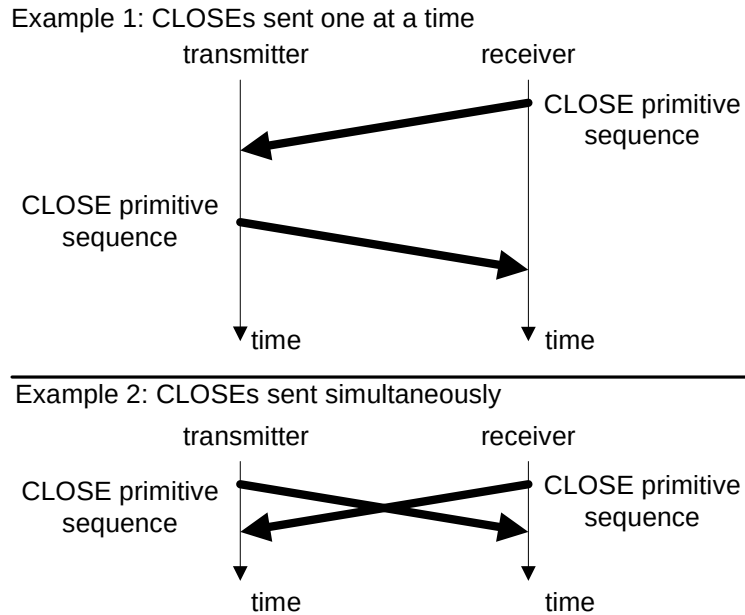


Figure 170 — Closing a connection example

7.12.8 Breaking a connection

In addition to aborting a connection request, BREAK may also be used to break a connection, in cases where CLOSE is not available. After transmitting BREAK, the originating phy shall ignore all incoming dwords except for BREAKs and BREAK_REPLYS.

After transmitting BREAK, the source phy shall initialize a Break Timeout timer to 1 ms and start the Break Timeout timer.

After a phy transmits a BREAK to break a connection, it shall expect one of the results listed in table 140.

Table 140 — Results of breaking a connection

BREAK_REPLY method of responding to received BREAK primitive sequences	Result	Description
Disabled	Receive BREAK	This confirms that the connection has been broken.
	Receive BREAK_REPLY	Ignore.
Enabled	Receive BREAK	The originating phy shall transmit BREAK_REPLY and wait to receive BREAK_REPLY or for the BREAK Timeout timer to expire.
	Receive BREAK_REPLY	This confirms that the connection has been broken.
Enabled or disabled	Break Timeout timer expires	The originating phy shall assume the connection has been broken. The originating phy may perform a link reset sequence.

In addition to a BREAK, a connection is considered broken if a link reset sequence starts (i.e., the SP state machine transitions from SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)).

See 7.16.6 for additional rules on breaking an SSP connection.

7.13 Rate matching

Each successful connection request contains the connection rate (see 4.1.12) of the pathway.

Each logical phy in the pathway shall insert deletable primitives between dwords if its logical link rate is faster than the connection rate as described in table 141.

Table 141 — Rate matching deletable primitive insertion requirements

Logical link rate	Connection rate	Requirement
1,5 Gbps	1,5 Gbps	None
3 Gbps	1,5 Gbps	One deletable primitive within every 2 dwords that are not physical link rate tolerance management deletable primitives (i.e., every overlapping window of 2 dwords)(e.g., a repeating pattern of a deletable primitive followed by a dword, or a repeating pattern of a dword followed by an deletable primitive)
	3 Gbps	None
6 Gbps	1,5 Gbps	Three deletable primitives within every 4 dwords that are not physical link rate tolerance management deletable primitives (i.e., 3 in every overlapping window of 4 dwords)
	3 Gbps	One deletable primitive within every 2 dwords that are not physical link rate tolerance management deletable primitives (i.e., every overlapping window of 2 dwords)(e.g., a repeating pattern of a deletable primitive followed by a dword, or a repeating pattern of a dword followed by an deletable primitive)
	6 Gbps	None

Deletable primitives inserted for rate matching are in addition to deletable primitives inserted for physical link rate tolerance management (see 7.3). See Annex H for a summary of their combined requirements.

Figure 171 shows an example of rate matching between a 3 Gbps source phy and a 3 Gbps destination phy, with an intermediate 1,5 Gbps physical link in between them.

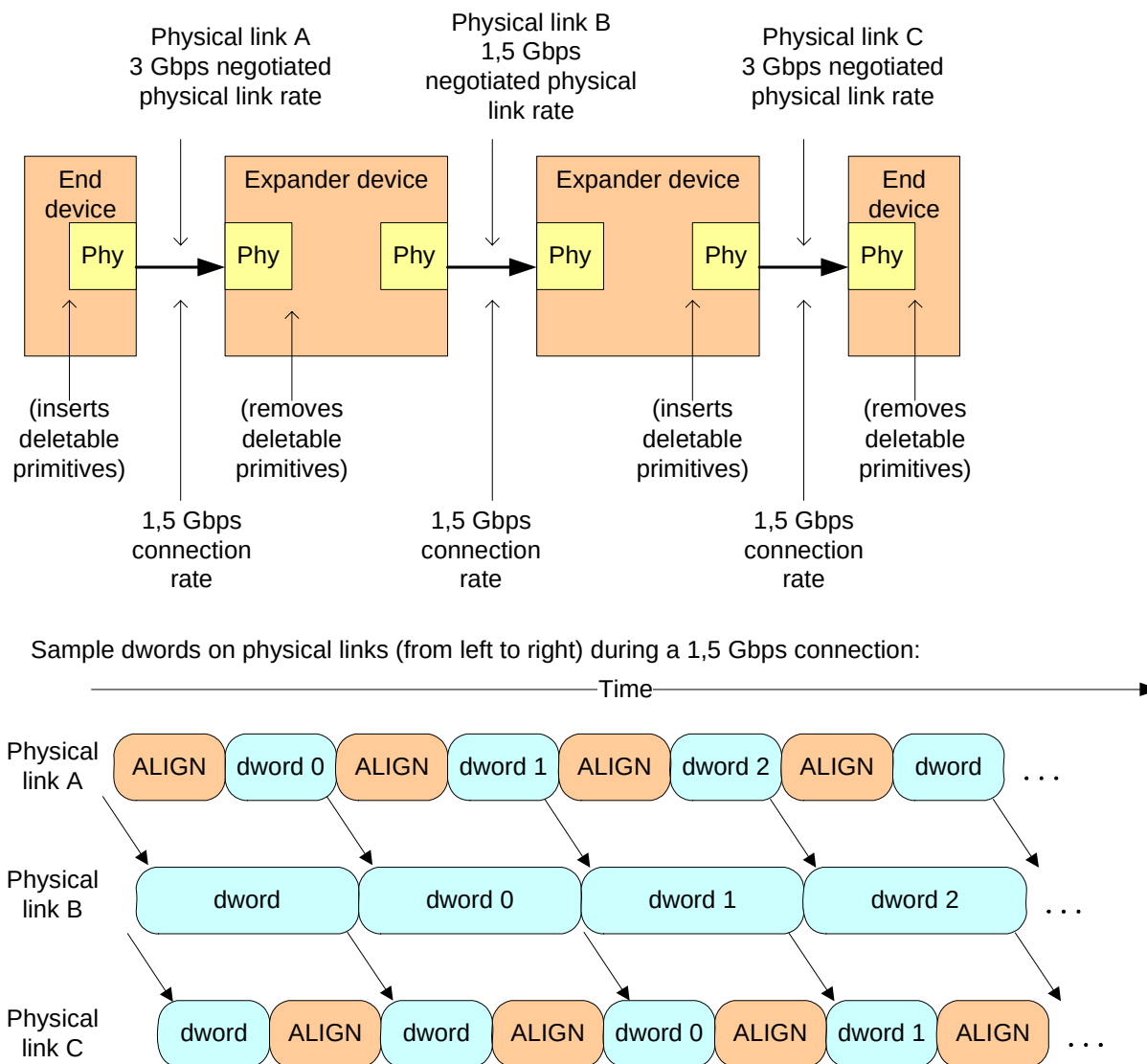


Figure 171 — Rate matching example

A phy originating dwords shall start rate matching at the selected connection rate starting with the first dword that is not a deletable primitive inserted for physical link rate tolerance management following:

- transmitting the EOAF for an OPEN address frame; or
- transmitting an OPEN_ACCEPT.

An expander phy forwarding dwords shall not insert deletable primitives for rate matching based on counting dwords transmitted. It shall insert deletable primitives whenever it underflows.

The source phy transmits idle dwords including deletable primitives at the selected connection rate while waiting for the connection response. This enables each expander device to start forwarding dwords from the source phy to the destination phy after forwarding an OPEN_ACCEPT.

A phy shall stop inserting deletable primitives for rate matching after:

- transmitting the first dword in a CLOSE;
- transmitting the first dword in a BREAK;
- transmitting the first dword in a BREAK_REPLY;
- receiving an OPEN_REJECT for a connection request; or

- e) losing arbitration to a received OPEN address frame.

7.14 SL (link layer for SAS logical phys) state machines

7.14.1 SL state machines overview

The SL (link layer for SAS logical phys) state machines controls connections, handling both connection requests (OPEN address frames), CLOSEs, and BREAKs. The SL state machines are as follows:

- a) SL_RA (receive OPEN address frame) state machine (see 7.14.3); and
- b) SL_CC (connection control) state machine (see 7.14.4).

All the SL state machines shall begin after receiving an Enable Disable SAS Link (Enable) message from the SL_IR state machines.

If a state machine consists of multiple states the initial state is as indicated in the state machine description.

Figure 172 shows the SL state machines.

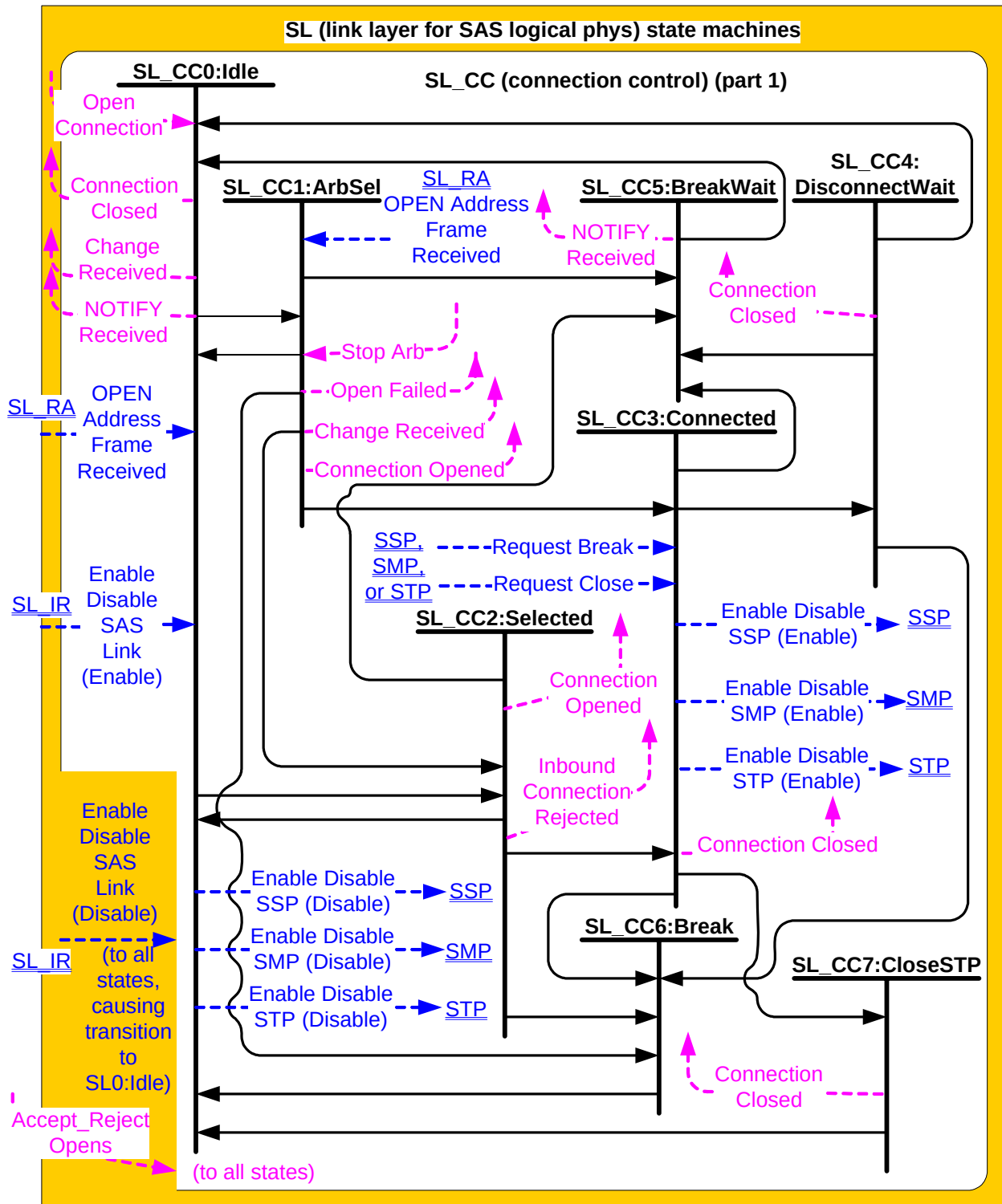


Figure 172 — SL (link layer for SAS logical phys) state machines (part 1)

Figure 173 shows the messages sent to the SL transmitter and received from the SL receiver.

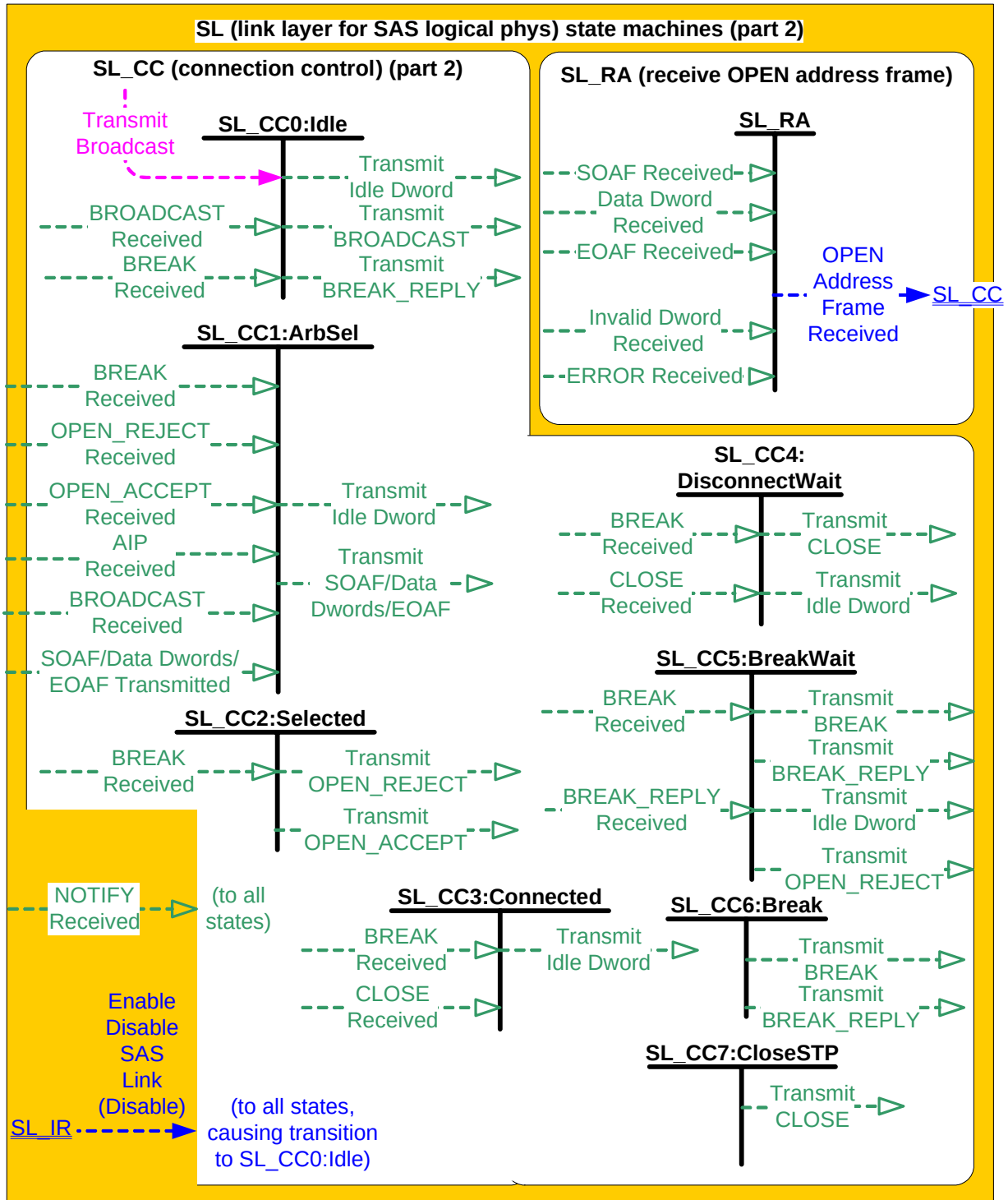


Figure 173 — SL (link layer for SAS logical phys) state machines (part 2)

7.14.2 SL transmitter and receiver

The SL transmitter receives the following messages from the SL state machines specifying primitive sequences, frames, and dwords to transmit:

- Transmit Idle Dword;
- Transmit SOAF/Data Dwords/EOAF;

- c) Transmit OPEN_ACCEPT;
- d) Transmit OPEN_REJECT with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (Retry));
- e) Transmit BREAK;
- f) Transmit BREAK_REPLY;
- g) Transmit BROADCAST; and
- h) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal)).

When the SL transmitter is requested to transmit a dword from any state within any of the SL state machines, it shall transmit that dword. If there are multiple requests to transmit, the following priority should be followed when selecting the dword to transmit:

- 1) BREAK_REPLY;
- 2) BREAK;
- 3) CLOSE;
- 4) OPEN_ACCEPT or OPEN_REJECT;
- 5) SOAF or data dword or EOF; then
- 6) idle dword.

When there is no outstanding message specifying a dword to transmit, the SL transmitter shall transmit idle dwords.

The SL transmitter sends the following messages to the SL state machines based on dwords that have been transmitted:

- a) SOAF/Data Dwords/EOF Transmitted.

The SL receiver sends the following messages to the SL state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

- a) SOAF Received;
- b) Data Dword Received;
- c) EOF Received;
- d) BROADCAST Received with an argument indicating the specific type (e.g., BROADCAST Received (Change));
- e) BREAK Received;
- f) BREAK_REPLY Received;
- g) OPEN_ACCEPT Received;
- h) OPEN_REJECT Received with an argument indicating the specific type (e.g., OPEN_REJECT Received (No Destination));
- i) AIP Received;
- j) CLOSE Received with an argument indicating the specific type (e.g., CLOSE Received (Normal));
- k) NOTIFY Received (Power Loss Expected);
- l) ERROR Received; and
- m) Invalid Dword Received.

The SL receiver shall ignore all other dwords.

The SL transmitter relationship to other transmitters is defined in 4.3.2. The SL receiver relationship to other receivers is defined in 4.3.3.

7.14.3 SL_RA (receive OPEN address frame) state machine

The SL_RA state machine's function is to receive address frames and determine if the received address frame is an OPEN address frame and whether or not it was received successfully. This state machine consists of one state.

This state machine receives SOAFs, dwords of an OPEN address frames, and EOFs.

This state machine shall ignore all messages except SOAF Received, Data Dword Received, and EOF Received.

If this state machine receives a subsequent SOAF Received message after receiving an SOAF Received message but before receiving an EOAF Received message, then this state machine shall discard the Data Dword Received messages received before the subsequent SOAF Received message.

If this state machine receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state machine shall discard the address frame.

If this state machine receives an Invalid Dword Received message or an ERROR Received message after an SOAF Received message and before an EOAF Received message, then this state machine shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the address frame.

After receiving an EOAF Received message, this state machine shall check if the address frame is a valid OPEN address frame.

This state machine shall accept an address frame if:

- a) the ADDRESS FRAME TYPE field is set to 1h (i.e., OPEN);
- b) the number of data dwords between the SOAF and EOAF is 8; and
- c) the CRC field contains a good CRC.

Otherwise, this state machine shall discard the address frame. If the frame is not discarded then this state machine shall send a OPEN Address Frame Received message to the SL_CC0:Idle state and the SL_CC1:ArbSel state with an argument that contains all the data dwords received in the OPEN address frame.

7.14.4 SL_CC (connection control) state machine

7.14.4.1 SL_CC state machine overview

The state machine consists of the following states:

- a) SL_CC0:Idle (see 7.14.4.2)(initial state);
- b) SL_CC1:ArbSel (see 7.14.4.3);
- c) SL_CC2:Selected (see 7.14.4.4);
- d) SL_CC3:Connected (see 7.14.4.5);
- e) SL_CC4:DisconnectWait (see 7.14.4.6);
- f) SL_CC5:BreakWait (see 7.14.4.7);
- g) SL_CC6:Break (see 7.14.4.8); and
- h) SL_CC7:CloseSTP (see 7.14.4.9).

The state machine shall start in the SL_CC0:Idle state. The state machine shall transition to the SL_CC0:Idle state from any other state after receiving an Enable Disable SAS Link (Disable) message from the SL_IR state machines (see 7.9.4).

The SL_CC state machine receives the following messages from the SSP link layer state machine (see 7.16.8), the STP link layer state machine, and SMP link layer state machine (see 7.18.5):

- a) Request Break; and
- b) Request Close.

The SL_CC state machine sends the following messages to the SSP link layer state machine, the STP link layer state machine, and SMP link layer state machine:

- a) Enable Disable SSP (Enable);
- b) Enable Disable SSP (Disable);
- c) Enable Disable STP (Enable);
- d) Enable Disable STP (Disable);
- e) Enable Disable SMP (Enable); and
- f) Enable Disable SMP (Disable).

The SL_CC state machine receives the following messages from the SL_IR state machines (see 7.9.4):

- a) Enable Disable SAS Link (Enable); and

- b) Enable Disable SAS Link (Disable).

Any message received by a state that is not referred to in the description of that state or in this subclause shall be ignored.

If this state machine receives an Accept_Reject Opens (Accept SSP) request, then this state machine shall set the Reject SSP Opens state machine variable to NO. If this state machine receives an Accept_Reject Opens (Reject SSP) request, then this state machine shall set the Reject SSP Opens state machine variable to YES.

If this state machine receives an Accept_Reject Opens (Accept SMP) request, then this state machine shall set the Reject SMP Opens state machine variable to NO. If this state machine receives an Accept_Reject Opens (Reject SMP) request, then this state machine shall set the Reject SMP Opens state machine variable to YES.

If this state machine receives an Accept_Reject Opens (Accept STP) request, then this state machine shall set the Reject STP Opens state machine variable to NO. If this state machine receives an Accept_Reject Opens (Reject STP) request, then this state machine shall set the Reject STP Opens state machine variable to YES.

Any detection of an internal error shall cause the SL_CC state machine to transition to the SL_CC5:BreakWait state.

The SL_CC state machine shall maintain the timers listed in table 142.

Table 142 — SL_CC timers

Timer	Initial value
Open Timeout timer	1 ms
Close Timeout timer	1 ms
Break Timeout timer	1 ms

The SL_CC state machine shall maintain the state machine variables listed in table 143.

Table 143 — SL_CC state machine variables

State machine variable	Description
Reject SSP Opens	Used to determine if the SCSI application layer is permitting SSP connection requests to be accepted on this phy.
Reject SMP Opens	Used to determine if the management application layer is permitting SMP connection requests to be accepted on this phy.
Reject STP Opens	Used to determine if the ATA application layer is permitting STP connection requests to be accepted on this phy.

7.14.4.2 SL_CC0:Idle state

7.14.4.2.1 State description

This state is the initial state and is the state that is used when there is no connection pending or established.

Upon entry into this state, this state shall send:

- an Enable Disable SSP (Disable) message to the SSP link layer state machines;
- an Enable Disable SMP (Disable) message to the SMP link layer state machines;
- an Enable Disable STP (Disable) message to the STP link layer state machines; and
- a Connection Closed (Transition to Idle) confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter (see 7.4).

If a BROADCAST Received (Change) message, BROADCAST Received (Reserved Change 0) message, or BROADCAST Received (Reserved Change 1) message is received, this state shall send a Change Received confirmation to the management layer.

If a Transmit Broadcast request is received with any argument, this state shall send a Transmit BROADCAST message with the same argument to the SL transmitter.

If a BREAK Received message is received and the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5), then this state shall send a Transmit BREAK_REPLY message to the SL transmitter.

After this state receives an Enable Disable SAS Link (Enable) confirmation, this state shall:

- a) set the Reject SSP Opens state machine variable to a vendor-specific default value (i.e., YES or NO);
- b) set the Reject SMP Opens state machine variable to a vendor-specific default value (i.e., YES or NO); and
- c) set the Reject STP Opens state machine variable to a vendor-specific default value (i.e., YES or NO).

If this state receives a NOTIFY Received (Power Loss Expected) message and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this state shall send a Notify Received (Power Loss Expected) confirmation to the port layer.

7.14.4.2.2 Transition SL_CC0:Idle to SL_CC1:ArbSel

This transition shall occur after receiving both an Enable Disable SAS Link (Enable) confirmation and an Open Connection request. The Open Connection request includes these arguments:

- a) initiator port bit;
- b) protocol;
- c) connection rate;
- d) initiator connection tag.
- e) destination SAS address;
- f) source SAS address;
- g) pathway blocked count; and
- h) arbitration wait time.

7.14.4.2.3 Transition SL_CC0:Idle to SL_CC2:Selected

This transition shall occur after receiving both an Enable Disable SAS Link (Enable) confirmation and an OPEN Address Frame Received message.

7.14.4.3 SL_CC1:ArbSel state

7.14.4.3.1 State description

This state is used to make a connection request.

Upon entry into this state, this state shall:

- 1) request an OPEN address frame be transmitted by sending a Transmit SOAF/Data Dwords/EOAF message to the SL transmitter with the dwords containing the OPEN address frame with its fields set to the arguments received with the Open Connection request;
- 2) initialize and start the Open Timeout timer; and
- 3) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter.

This state shall ignore OPEN_REJECT Received and OPEN_ACCEPT Received messages from the time a Transmit SOAF/Data Dwords/EOAF message is sent to the SL transmitter until an SOAF/Data Dwords/EOAF Transmitted message is received from the SL transmitter.

If a BROADCAST Received (Change) message, BROADCAST Received (Reserved Change 0) message, or BROADCAST Received (Reserved Change 1) message is received this state shall send a Change Received confirmation to the management layer.

If an AIP Received message is received after requesting the OPEN address frame be transmitted, this state shall reinitialize and restart the Open Timeout timer. The state machine shall not enforce a limit on the number of AIPs received.

If this state receives an OPEN_REJECT Received message listed in table 144 after transmitting the OPEN address frame, this state shall send the corresponding Open Failed confirmation listed in table 144 to the port layer.

Table 144 — OPEN_REJECT Received message to Open Failed confirmation mapping

OPEN_REJECT Received message	Open Failed confirmation
OPEN_REJECT Received (Bad Destination)	Open Failed (Bad Destination)
OPEN_REJECT Received (Connection Rate Not Supported)	Open Failed (Connection Rate Not Supported)
OPEN_REJECT Received (Protocol Not Supported)	Open Failed (Protocol Not Supported)
OPEN_REJECT Received (Reserved Abandon 1)	Open Failed (Reserved Abandon 1)
OPEN_REJECT Received (Reserved Abandon 2)	Open Failed (Reserved Abandon 2)
OPEN_REJECT Received (Reserved Abandon 3)	Open Failed (Reserved Abandon 3)
OPEN_REJECT Received (STP Resources Busy)	Open Failed (STP Resources Busy)
OPEN_REJECT Received (Wrong Destination)	Open Failed (Wrong Destination)
OPEN_REJECT Received (Zone Violation)	Open Failed (Zone Violation)
OPEN_REJECT Received (No Destination)	Open Failed (No Destination)
OPEN_REJECT Received (Pathway Blocked)	Open Failed (Pathway Blocked)
OPEN_REJECT Received (Reserved Continue 0)	Open Failed (Reserved Continue 0)
OPEN_REJECT Received (Reserved Continue 1)	Open Failed (Reserved Continue 1)
OPEN_REJECT Received (Reserved Initialize 0)	Open Failed (Reserved Initialize 0)
OPEN_REJECT Received (Reserved Initialize 1)	Open Failed (Reserved Initialize 1)
OPEN_REJECT Received (Reserved Stop 0)	Open Failed (Reserved Stop 0)
OPEN_REJECT Received (Reserved Stop 1)	Open Failed (Reserved Stop 1)
OPEN_REJECT Received (Retry)	Open Failed (Retry)

7.14.4.3.2 Transition SL_CC1:ArbSel to SL_CC0:Idle

This transition shall occur after sending an Open Failed confirmation.

7.14.4.3.3 Transition SL_CC1:ArbSel to SL_CC2:Selected

This transition shall occur after receiving a SOAF/Data Dwords/EOAF Transmitted message if:

- one or more AIP Received messages have been received before an OPEN Address Frame Received message is received (i.e., the incoming OPEN address frame overrides the outgoing OPEN address frame); or
- no AIP Received messages have been received before an OPEN Address Frame Received message is received, and the arbitration fairness rules (see 7.12.3) indicate the received OPEN address frame overrides the outgoing OPEN address frame.

The arbitration fairness comparison shall compare:

- a) the value of the arbitration wait time argument in the Open Connection request for the outgoing OPEN address frame; and
- b) the value of the ARBITRATION WAIT TIME field received in the incoming OPEN address frame.

7.14.4.3.4 Transition SL_CC1:ArbSel to SL_CC3:Connected

This transition shall occur if this state receives an SOAF/Data Dwords/EOAF Transmitted message followed by an OPEN_ACCEPT Received message.

If the PROTOCOL field in the transmitted OPEN address frame was set to STP, then this state shall send a Connection Opened (STP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open STP Connection argument. At this point an STP connection has been opened between the source phy and the destination phy.

If the PROTOCOL field in the transmitted OPEN address frame was set to SSP, then this state shall send a Connection Opened (SSP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open SSP Connection argument. At this point an SSP connection has been opened between the source phy and the destination phy.

If the PROTOCOL field in the transmitted OPEN address frame was set to SMP, then this state shall send a Connection Opened (SMP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open SMP Connection argument. At this point an SMP connection has been opened between the source phy and the destination phy.

7.14.4.3.5 Transition SL_CC1:ArbSel to SL_CC5:BreakWait

This transition shall occur after receiving a SOAF/Data Dwords/EOAF Transmitted message if a BREAK Received message has not been received and after:

- a) a Stop Arb request is received and after sending an Open Failed (Port Layer Request) confirmation to the port layer;
- b) there is no response to the OPEN address frame before the Open Timeout timer expires and after sending an Open Failed (Open Timeout Occurred) confirmation to the port layer; or
- c) a NOTIFY Received (Power Loss Expected) message is received.

If a NOTIFY Received (Power Loss Expected) message was received and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this transition shall include a Power Loss Expected argument.

7.14.4.3.6 Transition SL_CC1:ArbSel to SL_CC6:Break

This transition shall occur after:

- a) receiving a SOAF/Data Dwords/EOAF Transmitted message;
- b) receiving a BREAK Received message; and
- c) sending an Open Failed (Break Received) confirmation to the port layer.

7.14.4.4 SL_CC2:Selected state

7.14.4.4.1 State description

This state completes the establishment of an SSP, SMP, or STP connection when an incoming connection request has won arbitration by sending a Transmit OPEN_ACCEPT message, or rejects opening a connection by sending a Transmit OPEN_REJECT message to the SL transmitter.

This state shall respond to an incoming OPEN address frame using the following rules:

- 1) If the OPEN address frame DESTINATION SAS ADDRESS field does not match the SAS address of this port, this state shall send a Transmit OPEN_REJECT (Wrong Destination) message to the SL transmitter (see 7.14.4.4.2);

- 2) If the OPEN address frame INITIATOR PORT bit, PROTOCOL field, FEATURES field, and/or INITIATOR CONNECTION TAG field are set to values that are not supported (e.g., a connection request from an SMP target port), this state shall send a Transmit OPEN_REJECT (Protocol Not Supported) message to the SL transmitter (see 7.14.4.4.2);
- 3) If the OPEN address frame CONNECTION RATE field is set to a connection rate that is not supported, this state shall send a Transmit OPEN_REJECT (Connection Rate Not Supported) message to the SL transmitter (see 7.14.4.4.2);
- 4) If the OPEN address frame PROTOCOL field is set to STP, the STP target port supports affiliations, and the source SAS address is not that of an STP initiator port with an affiliation established or the source SAS address is not that of an STP initiator port with ATA task file register set resources (see 7.17.4), this state shall send a Transmit OPEN_REJECT (STP Resources Busy) message to the SL transmitter (see 7.14.4.4.2);
- 5) If the OPEN address frame PROTOCOL field is set to SSP and the Reject SSP Opens state machine variable is set to YES, this state shall send a Transmit OPEN_REJECT (Retry) message to the SL transmitter (see 7.14.4.4.2);
- 6) If the OPEN address frame PROTOCOL field is set to SMP and the Reject SMP Opens state machine variable is set to YES, this state shall send a Transmit OPEN_REJECT (Retry) message to the SL transmitter (see 7.14.4.4.2);
- 7) If the OPEN address frame PROTOCOL field is set to STP and the Reject STP Opens state machine variable is set to YES, this state shall send a Transmit OPEN_REJECT (Retry) message to the SL transmitter (see 7.14.4.4.2);
- 8) If the OPEN address frame PROTOCOL field is set to SSP and the Reject SSP Opens state machine variable is set to NO, this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (SSP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3);
- 9) If the OPEN address frame PROTOCOL field is set to SMP and the Reject SMP Opens state machine variable is set to NO, this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (SMP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3); or
- 10) If the OPEN address frame PROTOCOL field is set to STP and the Reject STP Opens state machine variable is set to NO, this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (STP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3).

If this state sends a Transmit OPEN_REJECT message to the SL transmitter, it shall also send an Inbound Connection Rejected confirmation to the port layer.

NOTE 56 - Possible livelock scenarios occur if the BREAK_REPLY method of responding to BREAK primitive sequences is disabled and a SAS logical phy transmits BREAK to abort a connection request (e.g., if its Open Timeout timer expires). SAS logical phys should respond to OPEN Address frames faster than 1 ms to reduce susceptibility to this problem.

7.14.4.4.2 Transition SL_CC2:Selected to SL_CC0:Idle

This transition shall occur after this state sends a Transmit OPEN_REJECT message to the SL transmitter.

7.14.4.4.3 Transition SL_CC2:Selected to SL_CC3:Connected

This transition shall occur after sending a Connection Opened confirmation.

This transition shall include:

- a) an Open SSP Connection, Open STP Connection, or Open SMP Connection argument based on the requested protocol; and
- b) the received OPEN address frame.

7.14.4.4.4 Transition SL_CC2:Selected to SL_CC5:BreakWait

If the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this transition shall occur after receiving a NOTIFY Received (Power Loss Expected) message and shall include a Power Loss Expected argument.

7.14.4.4.5 Transition SL_CC2:Selected to SL_CC6:Break

This transition shall occur after a BREAK Received message is received.

7.14.4.5 SL_CC3:Connected state**7.14.4.5.1 State description**

This state enables the SSP, STP, or SMP link layer state machine to transmit dwords during a connection. See 7.13 for details on rate matching during the connection.

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open SMP Connection then this state shall send an Enable Disable SMP (Enable) message to the SMP link layer state machines (see 7.18.5).

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open SSP Connection then this state shall send an Enable Disable SSP (Enable) message to the SSP link layer state machines (see 7.16.8).

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open STP Connection then this state shall send an Enable Disable STP (Enable) message to the STP link layer state machines (see 7.17.8).

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter until the SSP, SMP, or STP link layer state machine starts transmitting.

A CLOSE Received message may be received at any time while in this state, but shall be ignored during SSP and SMP connections. If a CLOSE Received (Clear Affiliation) is received during an STP connection, this state shall clear any affiliation (see 7.17.4).

7.14.4.5.2 Transition SL_CC3:Connected to SL_CC4:DisconnectWait

This transition shall occur if a Request Close message is received.

7.14.4.5.3 Transition SL_CC3:Connected to SL_CC5:BreakWait

This transition shall occur after sending a Connection Closed (Break Requested) confirmation to the port layer if:

- a) a Request Break message is received and a BREAK Received message has not been received; or
- b) a NOTIFY Received (Power Loss Expected) message is received.

If a NOTIFY Received (Power Loss Expected) message was received and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this state shall include a Power Loss Expected argument.

7.14.4.5.4 Transition SL_CC3:Connected to SL_CC6:Break

This transition shall occur if a BREAK Received message is received and after sending a Connection Closed (Break Received) confirmation to the port layer.

7.14.4.5.5 Transition SL_CC3:Connected to SL_CC7:CloseSTP

This transition shall occur if a CLOSE Received message is received during an STP connection.

7.14.4.6 SL_CC4:DisconnectWait state**7.14.4.6.1 State description**

This state closes the connection and releases all resources associated with the connection.

Upon entry into this state, this state shall:

- 1) send a Transmit CLOSE (Normal) message or Transmit CLOSE (Clear Affiliation) message to the SL transmitter (see 7.17.6); and
- 2) initialize and start the Close Timeout timer.

A CLOSE Received message may be received at any time while in this state. If a CLOSE Received (Clear Affiliation) is received during an STP connection, this state shall clear any affiliation (see 7.17.4).

NOTE 57 - Possible livelock scenarios occur if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled and a SAS logical phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). SAS logical phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

7.14.4.6.2 Transition SL_CC4:DisconnectWait to SL_CC0:Idle

This transition shall occur after:

- a) sending a Transmit CLOSE message to the SL transmitter;
- b) receiving a CLOSE Received message; and
- c) sending a Connection Closed (Normal) confirmation to the port layer.

7.14.4.6.3 Transition SL_CC4:DisconnectWait to SL_CC5:BreakWait

This transition shall occur if a NOTIFY Received (Power Loss Expected) message is received or if:

- a) a BREAK Received message has not been received;
- b) no CLOSE Received message is received in response to a Transmit CLOSE message before the Close Timeout timer expires; and
- c) after sending a Connection Closed (Close Timeout) confirmation to the port layer.

If a NOTIFY Received (Power Loss Expected) message was received and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this state shall include a Power Loss Expected argument.

7.14.4.6.4 Transition SL_CC4:DisconnectWait to SL_CC6:Break

This transition shall occur after receiving a BREAK Received message and after sending a Connection Closed (Break Received) confirmation to the port layer.

7.14.4.7 SL_CC5:BreakWait state**7.14.4.7.1 State description**

This state closes the connection if one is established and releases all resources associated with the connection.

Upon entry into this state, this state shall:

- 1) send a Transmit BREAK message to the SL transmitter; and
- 2) initialize and start the Break Timeout timer.

If this state is entered with a Power Loss Expected message or this state receives a NOTIFY Received (Power Loss Expected) message and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this state shall send a Notify Received (Power Loss Expected) confirmation to the port layer.

If a BREAK Received message is received and the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5), then this state shall send a Transmit BREAK_REPLY message to the SL transmitter.

NOTE 58 - Some SAS logical phys compliant with previous versions of this standard send a Transmit OPEN_REJECT (Retry) message to the SL transmitter in response to each OPEN Address Frame Received message received while in this state.

7.14.4.7.2 Transition SL_CC5:BreakWait to SL_CC0:Idle

This transition shall occur after:

- a) receiving a BREAK_REPLY Received message if the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5);
- b) receiving a BREAK Received message if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled (see 7.12.5); or
- c) the Break Timeout timer expires.

7.14.4.8 SL_CC6:Break state

7.14.4.8.1 State description

This state closes any connection and releases all resources associated with this connection.

Upon entry into this state:

- a) if the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5), this state shall send a Transmit BREAK_REPLY message to the SL transmitter (see 7.14.4.8.2); and
- b) if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled (see 7.12.5), this state shall send a Transmit BREAK message to the SL transmitter.

If this state receives a NOTIFY Received (Power Loss Expected) message and the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port), then this state shall send a Notify Received (Power Loss Expected) confirmation to the port layer.

7.14.4.8.2 Transition SL_CC6:Break to SL_CC0:Idle

This transition shall occur after sending a Transmit BREAK message or a Transmit BREAK_REPLY message to the SL transmitter.

7.14.4.9 SL_CC7:CloseSTP state

7.14.4.9.1 State description

This state closes an STP connection and releases all resources associated with the connection.

Upon entry into this state, this state shall:

- 1) send a Transmit CLOSE (Normal) message or Transmit CLOSE (Clear Affiliation) message to the SL transmitter (see 7.17.6); and
- 2) send a Connection Closed (Normal) confirmation to the port layer (see 7.14.4.9.2).

NOTE 59 - Possible livelock scenarios occur if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled and a SAS logical phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). SAS logical phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

7.14.4.9.2 Transition SL_CC7:CloseSTP to SL_CC0:Idle

This transition shall occur after sending a Connection Closed (Normal) confirmation to the port layer.

7.15 XL (link layer for expander logical phys) state machine

7.15.1 XL state machine overview

The XL state machine controls the flow of dwords on the logical link and establishes and maintains connections with another XL state machine as facilitated by the expander function (e.g., the ECM and ECR).

This state machine consists of the following states:

- a) XL0:Idle (see 7.15.3)(initial state);
- b) XL1:Request_Path (see 7.15.4);
- c) XL2:Request_Open (see 7.15.5);
- d) XL3:Open_Confirm_Wait (see 7.15.6);
- e) XL4:Open_Reject (see 7.15.7);
- f) XL5:Forward_Open (see 7.15.8);
- g) XL6:Open_Response_Wait (see 7.15.9);
- h) XL7:Connected (see 7.15.10);
- i) XL8:Close_Wait (see 7.15.11);
- j) XL9:Break (see 7.15.12); and
- k) XL10:Break_Wait (see 7.15.13).

The XL state machine shall start in the XL0:Idle state. The XL state machine shall transition to the XL0:Idle state from any other state after receiving an Enable Disable SAS Link (Disable) message from the SL_IR state machines (see 7.9.4).

The XL state machine receives the following messages from the SL_IR state machines:

- a) Enable Disable SAS Link (Enable); and
- b) Enable Disable SAS Link (Disable).

Any message received by a state that is not referred to in the description of that state shall be ignored.

The XL state machine shall maintain the timers listed in table 145.

Table 145 — XL timers

Timer	Initial value
Partial Pathway Timeout timer	Partial pathway timeout value (see 7.12.4.4)
Break Timeout timer	1 ms

Figure 174 shows several states in the XL state machine.

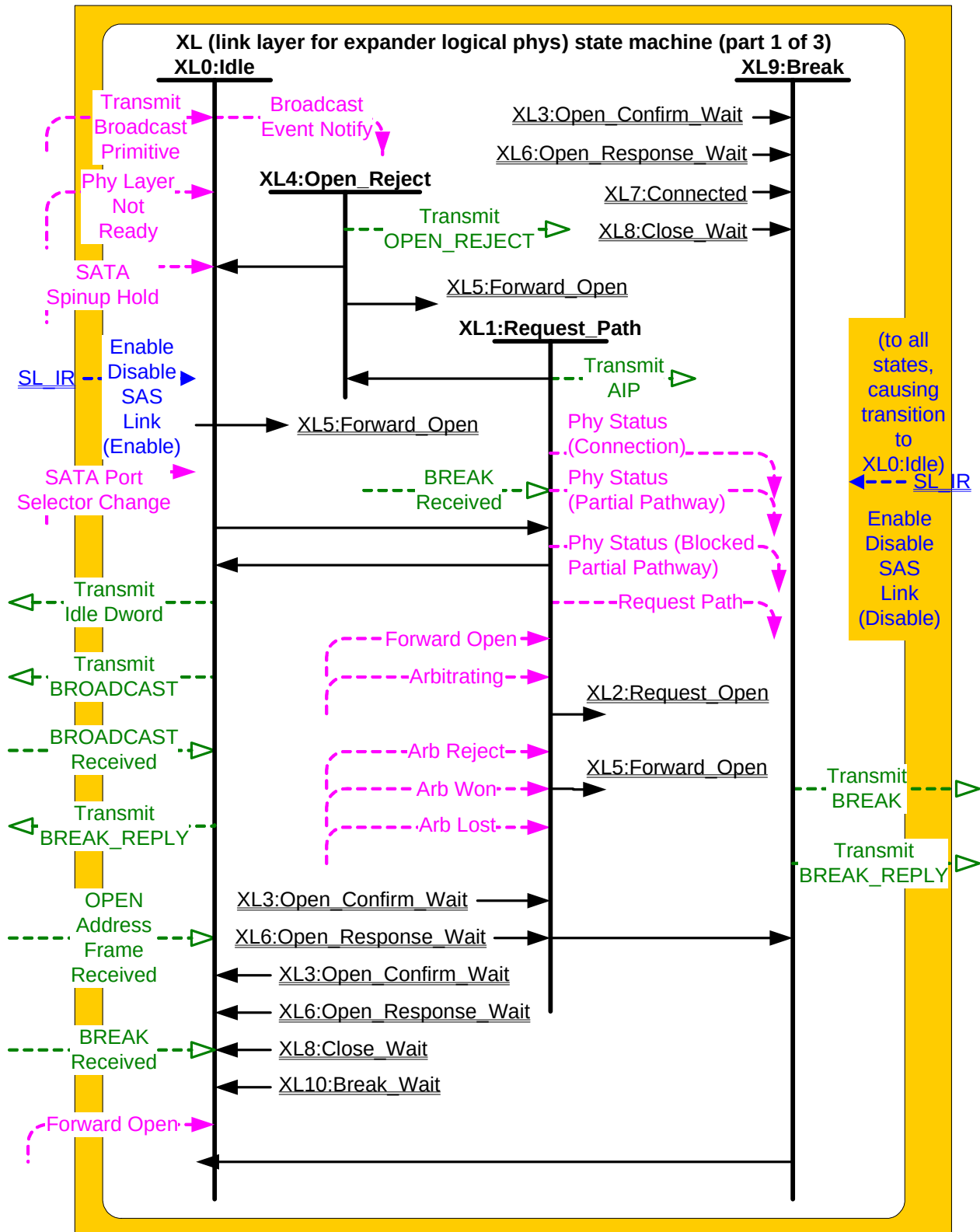


Figure 174 — XL (link layer for expander logical phys) state machine (part 1)

Figure 175 shows additional states in the XL state machine.

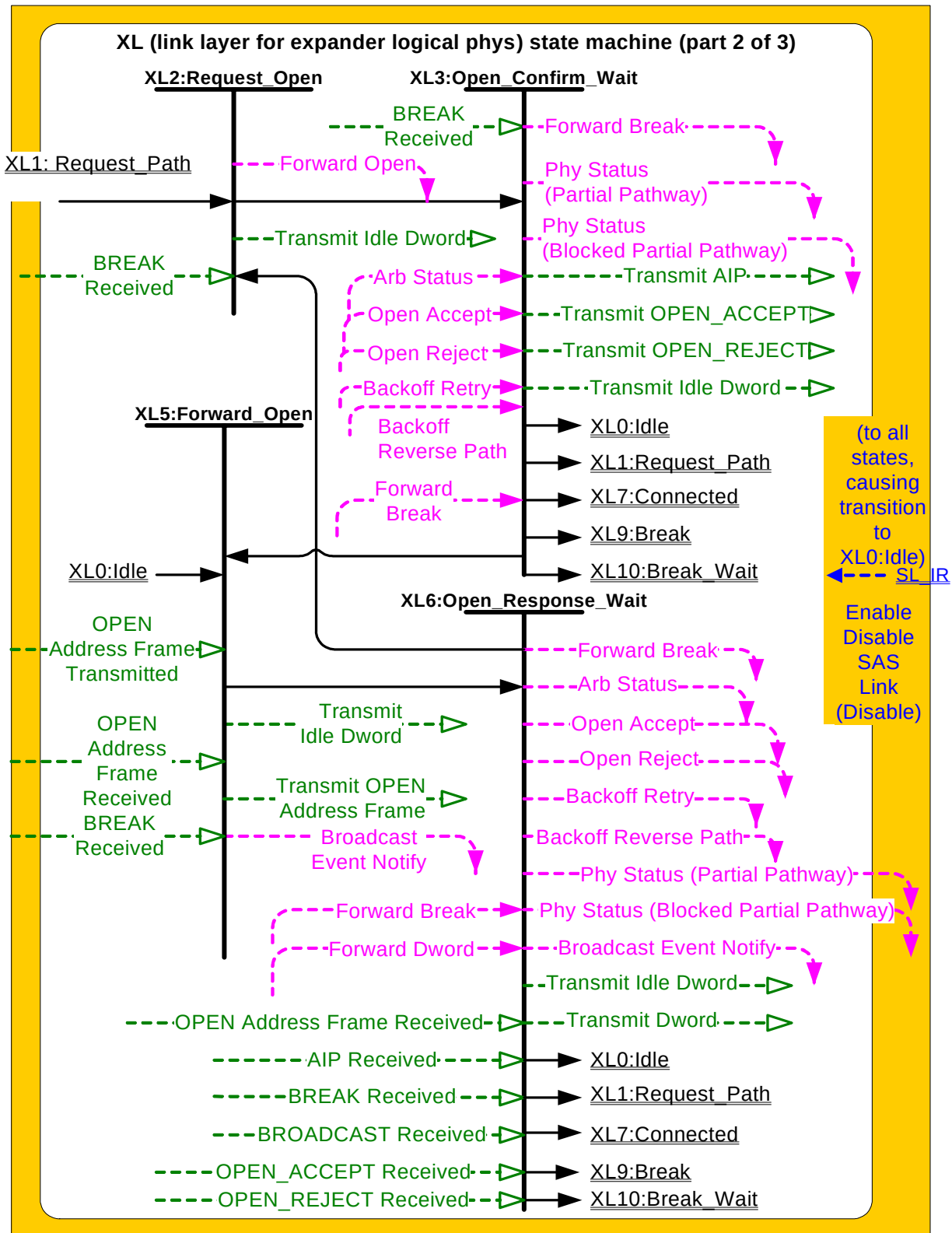


Figure 175 — XL (link layer for expander logical phys) state machine (part 2)

Figure 176 shows additional states in the XL state machine.

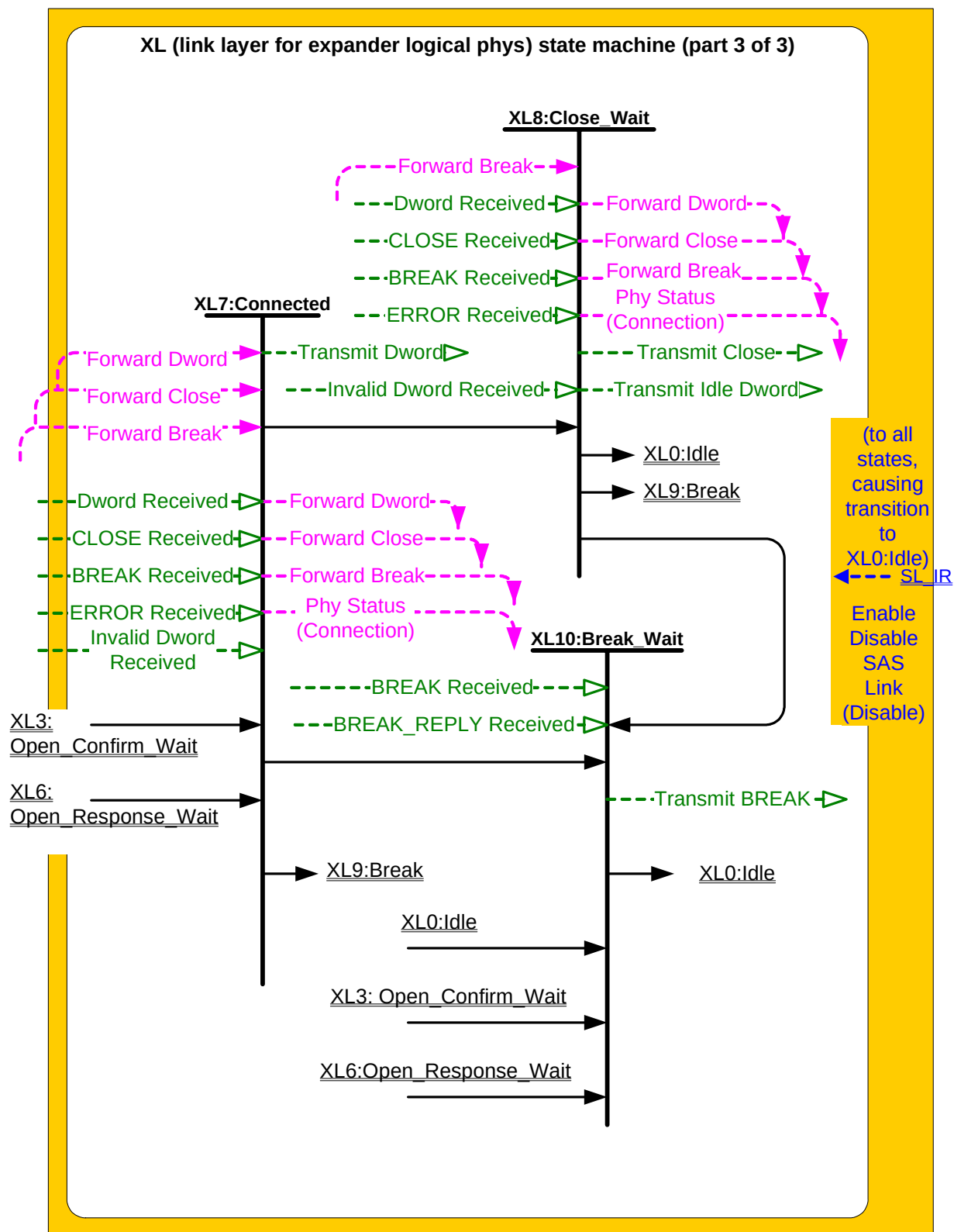


Figure 176 — XL (link layer for expander logical phys) state machine (part 3)

7.15.2 XL transmitter and receiver

The XL transmitter receives the following messages from the XL state machine specifying primitive sequences, frames, and dwords to transmit:

- a) Transmit Idle Dword;
- b) Transmit AIP with an argument indicating the specific type (e.g., Transmit AIP (Normal));
- c) Transmit BREAK;
- d) Transmit BREAK_REPLY;
- e) Transmit BROADCAST with an argument indicating the specific type (e.g., Transmit BROADCAST (Change));
- f) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal));
- g) Transmit OPEN_ACCEPT;
- h) Transmit OPEN_REJECT, with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (No Destination));
- i) Transmit OPEN Address Frame; and
- j) Transmit Dword.

The XL transmitter sends the following messages to the XL state machine based on dwords that have been transmitted:

- a) OPEN Address Frame Transmitted.

The XL transmitter shall ensure physical link rate tolerance management requirements are met (see 7.3) while originating dwords.

The XL transmitter shall ensure physical link rate tolerance management requirements are met while forwarding dwords (i.e., during a connection) by inserting or deleting as many deletable primitives as required to match the transmit and receive connection rates (see 7.3.2).

The XL transmitter shall ensure physical link rate tolerance management requirements are met (see 7.3) during and after switching from forwarding dwords to originating dwords, including, for example:

- a) when transmitting BREAK;
- b) when transmitting BREAK_REPLY;
- c) when transmitting CLOSE;
- d) when transmitting an idle dword after closing a connection (i.e., after receiving BREAK, BREAK_REPLY, or CLOSE);
- e) while transmitting a SATA frame to a SAS logical link during an STP connection, when transmitting the first SATA_HOLD in response to detection of SATA_HOLD; and
- f) while receiving dwords of a SATA frame from a SAS logical link during an STP connection, when transmitting SATA_HOLD.

NOTE 60 - The XL transmitter may always insert a deletable primitive before transmitting a BREAK, BREAK_REPLY, CLOSE, or SATA_HOLD to meet physical link rate tolerance management requirements.

The XL transmitter shall insert a deletable primitive before switching from originating dwords to forwarding dwords, including, for example:

- a) when transmitting OPEN_ACCEPT;
- b) when transmitting the last idle dword before a connection is established (i.e., after receiving OPEN_ACCEPT);
- c) while transmitting a SATA frame to a SAS logical link during an STP connection, when transmitting the last dword from the STP flow control buffer in response to release of SATA_HOLD;
- d) while transmitting a SATA frame to a SAS logical link during an STP connection, when transmitting the last SATA_HOLD in response to release of SATA_HOLD (e.g., if the STP flow control buffer is empty); and
- e) while receiving dwords of a SATA frame from a SAS logical link during an STP connection, when transmitting the last SATA_HOLD.

NOTE 61 - This ensures that physical link rate tolerance management requirements are met, even if the forwarded dword stream does not include a deletable primitive until the last possible dword.

The XL transmitter shall ensure rate matching requirements are met during a connection (see 7.13).

When there is no outstanding message specifying a dword to transmit, the XL transmitter shall transmit idle dwords.

The XL receiver sends the following messages to the XL state machine indicating primitive sequences, frames, and dwords received from the SP_DWS receiver (see 6.9.2):

- a) AIP Received with an argument indicating the specific type (e.g., AIP Received (Normal));
- b) BREAK Received;
- c) BREAK_REPLY Received;
- d) BROADCAST Received;
- e) CLOSE Received;
- f) OPEN_ACCEPT Received;
- g) OPEN_REJECT Received;
- h) OPEN Address Frame Received;
- i) Dword Received with an argument indicating the data dword or primitive received; and
- j) Invalid Dword Received.

The XL receiver shall ignore all other dwords.

While receiving an address frame, if the XL receiver receives an invalid dword or ERROR, then the XL receiver shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the address frame.

The XL transmitter relationship to other transmitters is defined in 4.3.2. The XL receiver relationship to other receivers is defined in 4.3.3.

7.15.3 XL0:Idle state

7.15.3.1 State description

This state is the initial state and is the state that is used when there is no connection pending or established.

If a Phy Layer Not Ready confirmation is received, this state shall send a Broadcast Event Notify (Phy Not Ready) request to the BPP.

If a SATA Spinup Hold confirmation is received, this state shall send a Broadcast Event Notify (SATA Spinup Hold) request to the BPP.

If an Enable Disable SAS Link (Enable) message is received, this state shall send a Broadcast Event Notify (Identification Sequence Complete) request to the BPP.

If a SATA Port Selector Change confirmation is received, this state shall send a Broadcast Event Notify (SATA Port Selector Change) request to the BPP.

If a BROADCAST Received message is received, this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., Change Received).

If a Transmit Broadcast indication is received, this state shall send a Transmit BROADCAST message to the XL transmitter with an argument specifying the specific type from the Transmit Broadcast indication.

Otherwise, this state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

If a BREAK Received message is received and the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5), then this state shall send a Transmit BREAK_REPLY message to the XL transmitter.

7.15.3.2 Transition XL0:Idle to XL1:Request_Path

This transition shall occur if:

- a) an Enable Disable SAS Link (Enable) message has been received;

- b) a Forward Open indication is not being received; and
- c) an OPEN Address Frame Received message is received.

This state shall include an OPEN Address Frame Received argument with the transition.

7.15.3.3 Transition XL0:Idle to XL5:Forward_Open

This transition shall occur if:

- a) an Enable Disable SAS Link (Enable) message has been received; and
- b) a Forward Open indication is received.

This transition shall include a set of arguments containing the arguments received in the Forward Open indication.

If an OPEN Address Frame Received message is received, this state shall include an OPEN Address Frame Received argument with the transition.

7.15.4 XL1:Request_Path state

7.15.4.1 State description

This state is used to arbitrate for connection resources and to specify the destination of the connection.

If an Arbitrating (Normal) confirmation is received, this state shall repeatedly send Transmit AIP (Normal) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.3).

If an Arbitrating (Waiting On Partial) or Arbitrating (Blocked On Partial) confirmation is received, this state shall repeatedly send Transmit AIP (Waiting On Partial) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.3).

If an Arbitrating (Waiting On Partial) confirmation is received, this state shall repeatedly send a Phy Status (Partial Pathway) response to the ECM.

If an Arbitrating (Blocked On Partial) confirmation is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) response to the ECM.

If an Arbitrating (Waiting On Connection) confirmation is received, this state shall repeatedly send Transmit AIP (Waiting On Connection) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.3).

If an Arbitrating (Waiting On Connection) confirmation is received, this state shall repeatedly send a Phy Status (Connection) response to the ECM.

If this state is entered from the XL6:Open_Response_Wait state, the Retry Priority Status argument shall be set to IGNORE AWT. If this state is entered from any other state, the Retry Priority Status argument shall be set to NORMAL.

Upon entry into this state, this state shall send a Request Path request to the ECM with the following arguments:

- a) initiator port bit;
- b) protocol;
- c) connection rate;
- d) initiator connection tag;
- e) destination SAS address;
- f) source SAS address;
- g) pathway blocked count;
- h) arbitration wait time; and
- i) retry priority status.

This state maintains the Partial Pathway Timeout timer.

If the Partial Pathway Timeout timer is not already running, the Partial Pathway Timeout timer shall be initialized and started when an Arbitrating (Blocked On Partial) confirmation is received.

If the Partial Pathway Timeout timer is already running, the Partial Pathway Timeout timer shall continue to run if an Arbitrating (Blocked On Partial) confirmation is received.

The Partial Pathway Timeout timer shall be stopped when one of the following confirmations is received:

- a) Arbitrating (Waiting On Partial); or
- b) Arbitrating (Waiting On Connection).

If the Partial Pathway Timeout timer expires, this state shall send a Partial Pathway Timeout Timer Expired request to the ECM.

7.15.4.2 Transition XL1:Request_Path to XL0:Idle

This transition shall occur if:

- a) a BREAK Received message has not been received; and
- b) an Arb Lost confirmation is received.

7.15.4.3 Transition XL1:Request_Path to XL2:Request_Open

This transition shall occur if:

- a) a BREAK Received message has not been received; and
- b) an Arb Won confirmation is received.

7.15.4.4 Transition XL1:Request_Path to XL4:Open_Reject

This transition shall occur if:

- a) a BREAK Received message has not been received; and
- b) an Arb Reject confirmation is received.

This transition shall include an Arb Reject argument corresponding to the Arb Reject confirmation.

7.15.4.5 Transition XL1:Request_Path to XL5:Forward_Open

This transition shall occur if a Forward Open indication is received and none of the following confirmations have been received:

- a) Arbitrating (Normal);
- b) Arbitrating (Waiting On Partial);
- c) Arbitrating (Blocked On Partial);
- d) Arbitrating (Waiting On Connection);
- e) Arb Won;
- f) Arb Lost;
- g) Arb Reject (No Destination);
- h) Arb Reject (Bad Destination);
- i) Arb Reject (Connection Rate Not Supported);
- j) Arb Reject (Zone Violation);
- k) Arb Reject (Pathway Blocked); or
- l) Arb Reject (Retry).

This transition shall include:

- a) an OPEN Address Frame Received argument containing the arguments received in the Forward Open indication; and
- b) a BREAK Received argument if a BREAK Received message was received.

7.15.4.6 Transition XL1:Request_Path to XL9:Break

This transition shall occur after receiving a BREAK Received message if a Forward Open indication has not been received.

7.15.5 XL2:Request_Open state

7.15.5.1 State description

This state is used to forward an OPEN address frame through the ECR to a destination phy.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

Upon entry into this state, this state shall send a Forward Open request to the ECR, received by the destination phy as a Forward Open indication (see 7.15.5.2). The arguments to the Forward Open request are:

- a) initiator port bit;
- b) protocol;
- c) features;
- d) connection rate;
- e) initiator connection tag;
- f) destination SAS address;
- g) source SAS address;
- h) source zone group;
- i) pathway blocked count;
- j) arbitration wait time; and
- k) more compatible features.

7.15.5.2 Transition XL2:Request_Open to XL3:Open_Confirm_Wait

This transition shall occur after sending a Forward Open request to the ECR.

If a BREAK Received message is received, this state shall include a BREAK Received argument with the transition.

7.15.6 XL3:Open_Confirm_Wait state

7.15.6.1 State description

This state waits for confirmation to an OPEN address frame sent on a destination phy.

This state shall send the following messages to the XL transmitter:

- a) Transmit AIP (Normal) when an Arb Status (Normal) confirmation is received;
- b) Transmit AIP (Waiting On Partial) when an Arb Status (Waiting On Partial) confirmation is received;
- c) Transmit AIP (Waiting On Connection) when an Arb Status (Waiting On Connection) confirmation is received;
- d) Transmit AIP (Waiting On Device) when an Arb Status (Waiting On Device) confirmation is received;
- e) Transmit OPEN_ACCEPT when an Open Accept confirmation is received (see 7.15.6.5);
- f) Transmit OPEN_REJECT when an Open Reject confirmation is received with the argument from the Open Reject confirmation, after releasing path resources (see 7.15.6.2); or
- g) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages when none of the previous conditions are present.

If a Backoff Retry confirmation is received, this state shall release path resources.

If a BREAK Received message is received or a BREAK Received argument is included in the transition into this state, this state shall send a Forward Break request to the ECR (see 7.15.6.6).

This state shall repeatedly send a Phy Status (Partial Pathway) response to the ECM until an Arb Status (Waiting On Partial) confirmation is received. After an Arb Status (Waiting on Partial) confirmation is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) response to the ECM.

Editor's Note 39: XL3 can also send Forward Dword requests (per all the example figures in the annex, and the fact that XL6 receives it). This needs to be added here and to the state machine

figure.

7.15.6.2 Transition XL3:Open_Confirm_Wait to XL0:Idle

This transition shall occur after sending a Transmit OPEN_REJECT message to the XL transmitter if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

7.15.6.3 Transition XL3:Open_Confirm_Wait to XL1:Request_Path

This transition shall occur after receiving a Backoff Retry confirmation, after releasing path resources if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

7.15.6.4 Transition XL3:Open_Confirm_Wait to XL5:Forward_Open

This transition shall occur after receiving a Backoff Reverse Path confirmation if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

The transition shall include the Backoff Reverse Path arguments (i.e., the OPEN address frame).

7.15.6.5 Transition XL3:Open_Confirm_Wait to XL7:Connected

This transition shall occur after sending a Transmit OPEN_ACCEPT message to the XL transmitter if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

7.15.6.6 Transition XL3:Open_Confirm_Wait to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.6.7 Transition XL3:Open_Confirm_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

7.15.7 XL4:Open_Reject state

7.15.7.1 State description

This state is used to reject a connection request.

This state shall send one of the following messages to the XL transmitter (see 7.15.7.2):

- a) a Transmit OPEN_REJECT (No Destination) message when an Arb Reject (No Destination) argument is received with the transition into this state;
- b) a Transmit OPEN_REJECT (Bad Destination) message when an Arb Reject (Bad Destination) argument is received with the transition into this state;
- c) a Transmit OPEN_REJECT (Connection Rate Not Supported) message when an Arb Reject (Connection Rate Not Supported) argument is received with the transition into this state;
- d) a Transmit OPEN_REJECT (Zone Violation) message when an Arb Reject (Zone Violation) argument is received with the transition into this state;
- e) a Transmit OPEN_REJECT (Pathway Blocked) message when an Arb Reject (Pathway Blocked) argument is received with the transition into this state; or
- f) a Transmit OPEN_REJECT (Retry) message when an Arb Reject (Retry) argument is received with the transition into this state.

g)

7.15.7.2 Transition XL4:Open_Reject to XL0:Idle

This transition shall occur after sending a Transmit OPEN_REJECT message to the XL transmitter.

7.15.7.3 Transition XL4:Open_Reject to XL5:Forward_Open

This transition shall occur if a Forward Open indication is received. This transition shall include an OPEN Address Frame Received argument containing the arguments received in the Forward Open indication.

7.15.8 XL5:Forward_Open state

7.15.8.1 State description

This state is used to transmit an OPEN address frame passed with the transition into this state.

If a BROADCAST Received message is received, this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., Change Received).

Upon entry into this state, this state shall send a Transmit OPEN Address Frame message to the XL transmitter with the fields set to the values specified with the transition into this state.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

7.15.8.2 Transition XL5:Forward_Open to XL6:Open_Response_Wait

This transition shall occur after receiving an OPEN Address Frame Transmitted message.

If an OPEN Address Frame Received message or argument is received, this state shall include an OPEN Address Frame Received argument with the transition.

If a BREAK Received message or argument is received, this state shall include a BREAK Received argument with the transition.

7.15.9 XL6:Open_Response_Wait state

7.15.9.1 State description

This state waits for a response to a transmitted OPEN address frame and determines the appropriate action to take based on the response.

This state shall either:

- a) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter, honoring ALIGN insertion rules for rate matching and physical link rate tolerance management; or
- b) send Transmit Dword messages to the XL transmitter to transmit all dwords received with Forward Dword indications. During an STP connection, the expander device may expand or contract a repeated or continued primitive sequence (see 7.2.4).

If a BROADCAST Received message is received before an AIP Received message is received this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., Broadcasts).

This state shall send the following responses to the ECR, which are received by the source phy as confirmations:

- a) an Open Accept response when an OPEN_ACCEPT Received message is received (see 7.15.9.5);
- b) an Open Reject response when an OPEN_REJECT Received message is received, after releasing any path resources (see 7.15.9.2);
- c) a Backoff Retry response, after releasing path resources (see 7.15.9.3), when:
 - A) an AIP Received message has not been received;

- B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state containing a higher priority OPEN address frame according to the arbitration fairness comparison (see 7.12.3); and
 - C) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame;
 - d) a Backoff Retry response, after releasing path resources (see 7.15.9.3), when:
 - A) an AIP Received message has been received;
 - B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state; and
 - C) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame;
 - e) a Backoff Reverse Path response (see 7.15.9.4) when:
 - A) an AIP Received message has not been received,
 - B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state containing a higher priority OPEN address frame according to the arbitration fairness comparison (see 7.12.3); and
 - C) the destination SAS address and connection rate of the received OPEN address frame are equal to the source SAS address and connection rate of the transmitted OPEN address frame;
- and
- f) a Backoff Reverse Path response (see 7.15.9.4) when:
 - A) an AIP Received message has been received;
 - B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state; and
 - C) the destination SAS address and connection rate of the received OPEN address frame are equal to the source SAS address and connection rate of the transmitted OPEN address frame.

A Backoff Reverse Path response shall include the contents of the OPEN Address Frame Received message or argument.

This state shall send the following responses to the ECR, which are received by the source phy as confirmations:

- a) an Arb Status (Waiting On Device) response upon entry into this state;
- b) an Arb Status (Normal) response when an AIP Received (Normal) message is received;
- c) an Arb Status (Waiting On Partial) response when an AIP Received (Waiting On Partial) message is received;
- d) an Arb Status (Waiting On Connection) response when an AIP Received (Waiting On Connection) message is received; and
- e) an Arb Status (Waiting On Device) response when an AIP Received (Waiting On Device) message is received.

If a BREAK Received message is received or a BREAK Received argument is included in the transition into this state, this state shall send a Forward Break request to the ECR (see 7.15.9.6).

This state shall repeatedly send a Phy Status (Partial Pathway) response to the ECM until an AIP Received (Waiting On Partial) message is received. After an AIP Received (Waiting On Partial) message is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) response to the ECM.

7.15.9.2 Transition XL6:Open_Response_Wait to XL0:Idle

This transition shall occur after sending an Open Reject response to the ECR.

7.15.9.3 Transition XL6:Open_Response_Wait to XL1:Request_Path

This transition shall occur after sending a Backoff Retry response to the ECR.

7.15.9.4 Transition XL6:Open_Response_Wait to XL2:Request_Open

This transition shall occur after sending a Backoff Reverse Path response to the ECR.

7.15.9.5 Transition XL6:Open_Response_Wait to XL7:Connected

This transition shall occur after sending an Open Accept response to the ECR.

7.15.9.6 Transition XL6:Open_Response_Wait to XL9:Break

This transition shall occur after sending a Forward Break response to the ECR.

7.15.9.7 Transition XL6:Open_Response_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if:

- a) a BREAK Received message has not been received; and
- b) a BREAK Received argument was not included in the transition into this state.

7.15.10 XL7:Connected state**7.15.10.1 State description**

This state provides a full-duplex circuit between two phys within an expander device.

This state shall send Transmit Dword messages to the XL transmitter to transmit all dwords received with Forward Dword indications. During an STP connection, the expander device may expand or contract a repeated or continued primitive sequence (see 7.2.4).

If this state has not sent a Forward Close request to the ECR, this state shall send Forward Dword requests to the ECR containing each valid dword except BREAK and CLOSE primitives received with Dword Received messages. During an STP connection, the expander device may expand or contract a repeated or continued primitive sequence (see 7.2.4).

If:

- a) an Invalid Dword Received message is received; and
- b) the expander phy is forwarding to an expander logical phy attached to a SAS logical link,

the expander logical phy shall:

- a) send an ERROR primitive with the Forward Dword request instead of the invalid dword; or
- b) delete the invalid dword.

If:

- a) an ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
- b) the expander phy is forwarding to an expander logical phy attached to a SATA phy,

the expander logical phy shall:

- a) send a SATA_ERROR with the Forward Dword request instead of the invalid dword or ERROR primitive; or
- b) delete the ERROR primitive or invalid dword.

If a CLOSE Received message is received, this state shall send a Forward Close request to the ECR with the argument from the CLOSE Received message.

If a BREAK Received message is received, this state shall send a Forward Break request to the ECR (see 7.15.10.3).

This state shall repeatedly send a Phy Status (Connection) response to the ECM.

7.15.10.2 Transition XL7:Connected to XL8:Close_Wait

This transition shall occur after receiving a Forward Close indication if a BREAK Received message has not been received.

7.15.10.3 Transition XL7:Connected to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.10.4 Transition XL7:Connected to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if a BREAK Received message has not been received.

7.15.11 XL8:Close_Wait state**7.15.11.1 State description**

This state closes a connection and releases path resources.

Upon entry into this state, this state shall send a Transmit CLOSE message to the XL transmitter with the argument from the Forward Close indication, then shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

NOTE 62 - Possible livelock scenarios occur if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled and a phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). Phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

If a Dword Received message is received containing a valid dword except a BREAK or CLOSE primitive, this state shall send Forward Dword requests to the ECR containing that dword. During an STP connection, the expander device may expand or contract a repeated or continued primitive sequence (see 7.2.4).

If:

- a) an Invalid Dword Received message is received; and
- b) the expander logical phy is forwarding to an expander logical phy attached to a SAS logical link,

the expander logical phy shall:

- a) send an ERROR primitive with the Forward Dword request instead of the invalid dword; or
- b) delete the invalid dword.

If:

- a) an ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
- b) the expander logical phy is forwarding to an expander phy attached to a SATA physical link,

the expander logical phy shall:

- a) send a SATA_ERROR with the Forward Dword request instead of the invalid dword or ERROR primitive; or
- b) delete the ERROR primitive or invalid dword.

If a CLOSE Received message is received, this state shall release path resources and send a Forward Close request to the ECR with the argument from the CLOSE Received message (see 7.15.11.2).

If a BREAK Received message is received, this state shall send a Forward Break request to the ECR (see 7.15.11.3).

This state shall repeatedly send a Phy Status (Connection) response to the ECM.

7.15.11.2 Transition XL8:Close_Wait to XL0:Idle

This transition shall occur after sending a Forward Close request to the ECR.

7.15.11.3 Transition XL8:Close_Wait to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.11.4 Transition XL8:Close_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if a BREAK Received message has not been received.

7.15.12 XL9:Break state**7.15.12.1 State description**

This state closes the connection if there is one and releases all path resources associated with the connection.

This state shall send a Transmit BREAK message to the XL transmitter (see 7.15.12.2).

This state shall:

- a) send a Transmit BREAK_REPLY message to the XL transmitter if the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5); and
- b) send a Transmit BREAK message to the XL transmitter if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled (see 7.12.5).

7.15.12.2 Transition XL9:Break to XL0:Idle

This transition shall occur after sending a Transmit BREAK message or a Transmit BREAK_REPLY message to the XL transmitter.

7.15.13 XL10:Break_Wait state**7.15.13.1 State description**

This state closes the connection if there is one and releases path resources associated with the connection.

Upon entry into this state, this state shall:

- 1) send a Transmit BREAK message to the XL transmitter; and
- 2) initialize and start the Break Timeout timer.

If a BREAK Received message is received and the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5), then this state shall send a Transmit BREAK_REPLY message to the XL transmitter.

7.15.13.2 Transition XL10:Break_Wait to XL0:Idle

This transition shall occur after:

- a) a BREAK_REPLY Received message is received if the BREAK_REPLY method of responding to received BREAK primitive sequences is enabled (see 7.12.5);
- b) a BREAK Received message is received if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled (see 7.12.5); or
- c) the Break Timeout timer expires.

7.16 SSP link layer**7.16.1 Opening an SSP connection**

An SSP phy that accepts a connection request (i.e., an OPEN address frame) shall transmit at least one RRDY in that connection within 1 ms of transmitting an OPEN_ACCEPT. If the SSP phy is not able to grant credit, it shall respond with OPEN_REJECT (RETRY) and not accept the connection request.

To prevent livelocks (e.g., where ports are waiting on each other to accept a connection request):

- a) a SAS phy shall not reject an incoming connection request to an SSP initiator port with OPEN_REJECT (RETRY) because the SAS port containing that SAS phy needs an outgoing connection request to be accepted (e.g., if the SAS phy is used by an SSP initiator port and an SSP

- target port, they share a buffer, that buffer is being used by the SSP target port, and the SSP target port needs to transmit a frame to another SSP initiator port before it is able to free that buffer);
- b) a SAS phy may reject an incoming connection request to an SSP initiator port with OPEN_REJECT (RETRY) for any reason that is not dependent on the SAS port containing that SAS phy having an outgoing connection request accepted (e.g., a temporary buffer full condition); and
 - c) a SAS phy may reject an incoming connection request to an SSP target port with OPEN_REJECT (RETRY) for any reason, including because the SAS port containing that SAS phy needs an outgoing connection request to be accepted (e.g., to transmit a frame and empty a buffer).

7.16.2 Full duplex

SSP is a full duplex protocol. An SSP phy may receive an SSP frame or primitive in a connection while it is transmitting an SSP frame or primitive in the same connection. A wide SSP port may send and/or receive SSP frames or primitives concurrently on different connections (i.e., on different phys).

When a connection is open and an SSP phy has no more SSP frames to transmit on that connection, it transmits a DONE to start closing the connection (see 8.2.2.3.5). The other direction may still be active, so the DONE may be followed by one or more of the following primitives: CREDIT_BLOCKED, RRDY, ACK, or NAK.

7.16.3 SSP frame transmission and reception

During an SSP connection, SSP frames are preceded by SOF and followed by EOF as shown in figure 177.

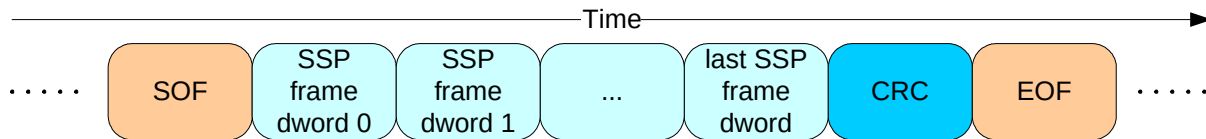


Figure 177 — SSP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5). The SSP link layer state machine checks that the frame is not too short and that the CRC is valid (see 7.16.8.7).

Receiving SSP phys shall acknowledge SSP frames within 1 ms, if not discarded as described in 7.16.8.7, with either:

- a) ACK (i.e., positive acknowledgement) if the SSP frame was received into a frame buffer without errors; or
- b) NAK (CRC ERROR) (i.e., negative acknowledgement) if the SSP frame was received with a CRC error (i.e., a bad CRC), an invalid dword, or an ERROR primitive.

NOTE 63 - It is not required that frame recipients generate NAK (CRC ERROR) from invalid dwords and ERRORS (see 7.16.8.2).

Either the transport layer (see 9.2.4) retries sending SSP frames that encounter a link layer error (e.g., are NAKed or create an ACK/NAK timeout), or the application layer aborts the SCSI command associated with the SSP frame that encountered a link layer error.

7.16.4 SSP flow control

An SSP phy uses RRDY to grant credit for permission for the other SSP phy in the connection to transmit frames. Each RRDY increments credit by one frame. Frame transmission decrements credit by one frame. Credit of zero frames is established at the beginning of each connection.

SSP phys shall not increment credit past 255 frames.

To prevent deadlocks where an SSP initiator port and SSP target port are both waiting on each other to provide credit, an SSP initiator port shall not refuse to provide credit by withholding RRDY because it needs to transmit a frame itself. It may refuse to provide credit for other reasons (e.g., temporary buffer full conditions).

An SSP target port may refuse to provide credit for any reason, including because it needs to transmit a frame itself.

If credit is zero, SSP phys that are going to be unable to provide credit for 1 ms may send CREDIT_BLOCKED. The other phy may use this to avoid waiting 1 ms to transmit DONE (CREDIT_TIMEOUT) (see 7.16.8).

If credit is nonzero, SSP phys that are going to be unable to provide additional credit for 1 ms, even if they receive frames per the existing credit, may transmit CREDIT_BLOCKED.

After sending CREDIT_BLOCKED, an SSP phy shall not transmit any additional RRDYs in the connection.

7.16.5 Interlocked frames

Table 146 shows which SSP frames shall be interlocked and which are non-interlocked.

Table 146 — SSP frame interlock requirements

SSP frame type	Interlock requirement
COMMAND	Interlocked
TASK	Interlocked
XFER_RDY	Interlocked
DATA	Non-interlocked
RESPONSE	Interlocked
See 9.2 for SSP frame type definitions.	

Before transmitting an interlocked frame, an SSP phy shall wait for all SSP frames to be acknowledged with ACK or NAK, even if credit is available. After transmitting an interlocked frame, an SSP phy shall not transmit another SSP frame until it has been acknowledged with ACK or NAK, even if credit is available.

Before transmitting a non-interlocked frame, an SSP phy shall wait for:

- a) all non-interlocked frames with different tags; and
- b) all interlocked frames;

to be acknowledged with ACK or NAK, even if credit is available.

After transmitting a non-interlocked frame, an SSP phy may transmit another non-interlocked frame with the same tag if credit is available. The phy shall not transmit:

- a) a non-interlocked frame with a different tag; or
- b) an interlocked frame;

until all SSP frames have been acknowledged with ACK or NAK, even if credit is available.

Interlocking does not prevent transmitting and receiving interlocked frames simultaneously (e.g., an SSP initiator phy could be transmitting a COMMAND frame while receiving XFER_RDY, DATA, or RESPONSE frames for a different command).

An SSP phy may transmit primitives responding to traffic it is receiving (e.g., an ACK or NAK to acknowledge an SSP frame, an RRDY to grant more receive credit, or a CREDIT_BLOCKED to specify that no more RRDYs are going to be transmitted in the connection) while waiting for an interlocked frame it transmitted to be acknowledged. These primitives may also be interspersed within an SSP frame.

Figure 178 shows an example of interlocked frame transmission.

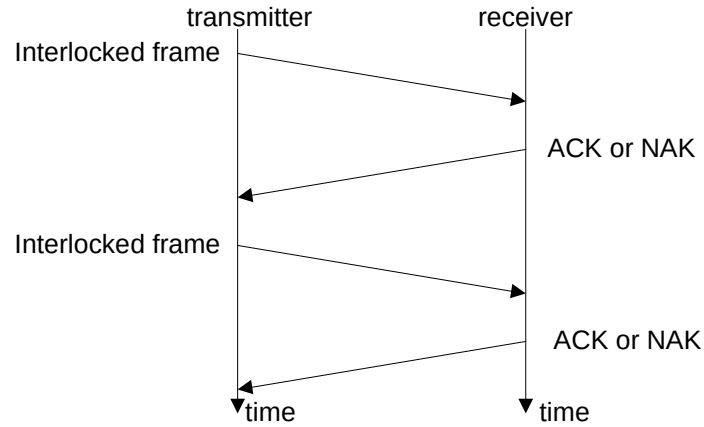


Figure 178 — Interlocked frames

Figure 179 shows an example of non-interlocked frame transmission with the same tags.

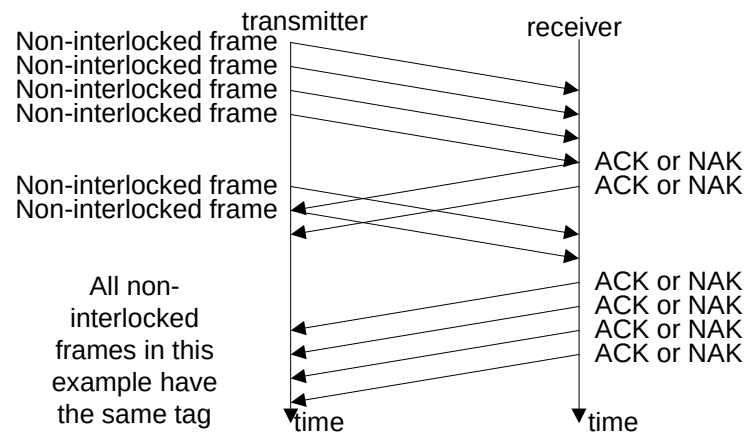


Figure 179 — Non-interlocked frames with the same tag

Figure 180 shows an example of non-interlocked frame transmission with different tags.

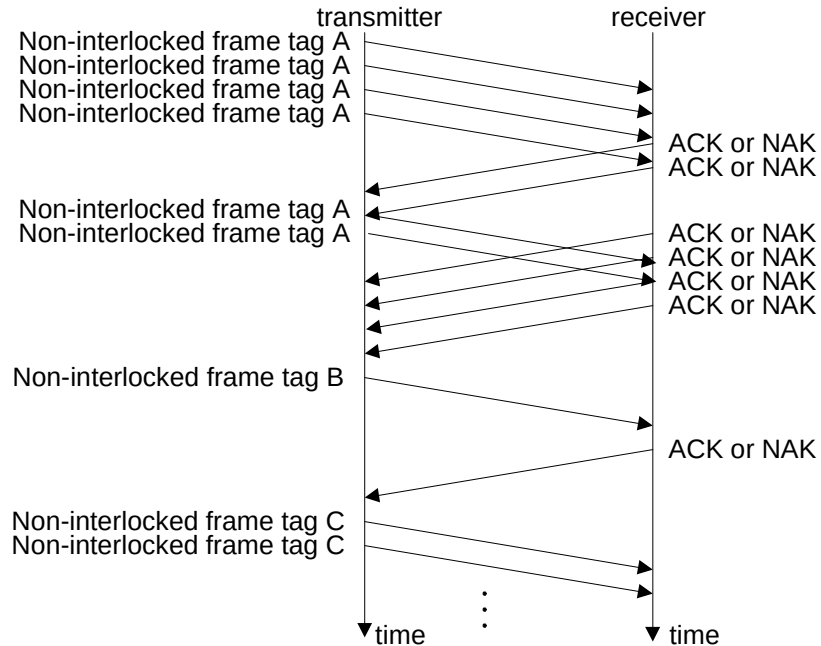


Figure 180 — Non-interlocked frames with different tags

7.16.6 Breaking an SSP connection

In addition to the actions described in 7.12.8, the following shall be the responses by an SSP phy to a broken connection:

- Received frames having no CRC error may be considered valid regardless of whether an ACK has been transmitted in response to the frame prior to the broken connection;
- Transmitted frames for which an ACK has been received prior to a broken connection shall be considered successfully transmitted; and
- Transmitted frames for which an ACK or NAK has not been received prior to a broken connection shall be considered not successfully transmitted.

7.16.7 Closing an SSP connection

DONE shall be exchanged prior to closing an SSP connection (see 8.2.2.3.5). There are several versions of the DONE primitive indicating additional information about why the SSP connection is being closed:

- DONE (NORMAL) specifies normal completion; the transmitter has no more SSP frames to transmit;
- DONE (CREDIT TIMEOUT) specifies that the transmitter still has SSP frames to transmit but did not receive an RRDY granting frame credit within 1 ms, or the transmitter has received a CREDIT_BLOCKED and has consumed all RRDYs received; and
- DONE (ACK/NAK TIMEOUT) specifies that the transmitter transmitted an SSP frame but did not receive the corresponding ACK or NAK within 1 ms. As a result, the ACK/NAK count is not balanced and the transmitter is going to transmit a BREAK in 1 ms unless the recipient replies with DONE and the connection is closed.

If the transmitter has no more SSP frames to transmit and receives a CREDIT_BLOCKED, it may transmit either DONE (NORMAL) or DONE (CREDIT TIMEOUT).

After transmitting DONE, the transmitting phy initializes and starts a 1 ms DONE Timeout timer (see 7.16.8.5).

After transmitting DONE, the transmitting phy shall not transmit any more SSP frames during this connection. However, the phy may transmit ACK, NAK, RRDY, and CREDIT_BLOCKED as needed after transmitting DONE if the other phy is still transmitting SSP frames in the reverse direction. Once an SSP phy has both transmitted and received DONE, it shall close the connection by transmitting CLOSE (NORMAL) (see 7.12.7).

Figure 181 shows the sequence for a closing an SSP connection.

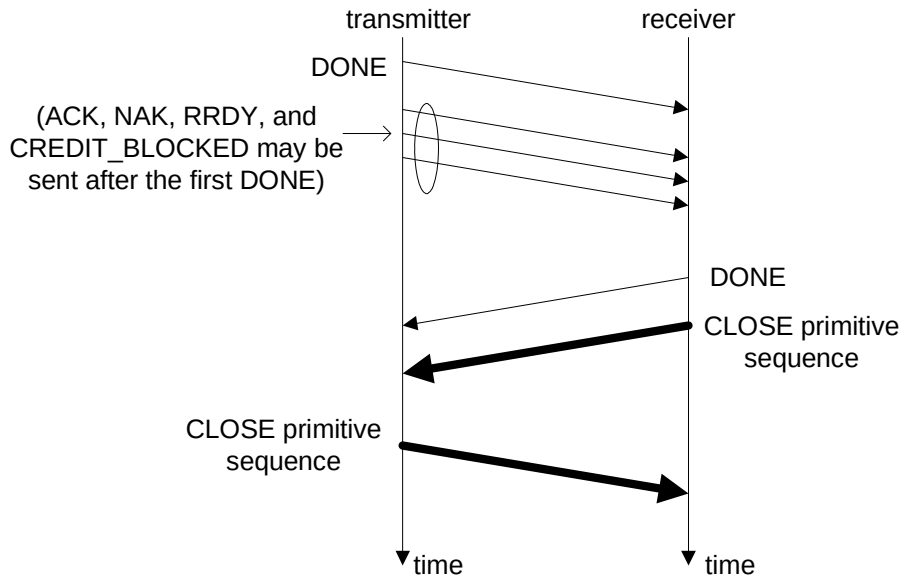


Figure 181 — Closing an SSP connection example

7.16.8 SSP (link layer for SSP phys) state machines

7.16.8.1 SSP state machines overview

The SSP link layer contains several state machines that run in parallel to control the flow of dwords on the physical link during an SSP connection. The SSP state machines are as follows:

- SSP_TIM (transmit interlocked frame monitor) state machine (see 7.16.8.3);
- SSP_TCM (transmit frame credit monitor) state machine (see 7.16.8.4);
- SSP_D (DONE control) state machine (see 7.16.8.5);
- SSP_TF (transmit frame control) state machine (see 7.16.8.6);
- SSP_RF (receive frame control) state machine (see 7.16.8.7);
- SSP_RCM (receive frame credit monitor) state machine (see 7.16.8.8);
- SSP_RIM (receive interlocked frame monitor) state machine (see 7.16.8.9);
- SSP_TC (transmit credit control) state machine (see 7.16.8.10); and
- SSP_TAN (transmit ACK/NAK control) state machine (see 7.16.8.11).

All the SSP state machines shall start after receiving an Enable Disable SSP (Enable) message from the SL state machines (see 7.14).

All the SSP state machines shall terminate after:

- receiving an Enable Disable SSP (Disable) message from the SL state machines;
- receiving a Request Close message from the SSP_D state machine indicating that the connection has been closed;
- receiving a Request Break message from the SSP_D state machine indicating that a BREAK has been transmitted; or
- receiving a NOTIFY Received (Power Loss Expected) message from the SP_DWS receiver if the SAS port that contains this state machine supports NOTIFY (Power Loss Expected) (e.g., the SAS port is a SSP target port).

If a state machine consists of multiple states the initial state is as indicated in the state machine description in this subclause.

The SSP state machines shall maintain the timers listed in table 147.

Table 147 — SSP link layer timers

Timer	Initial value	State machine	Reference
ACK/NAK Timeout timer	1 ms	SSP_TIM	7.16.8.3
DONE Timeout timer	1 ms	SSP_D	7.16.8.5
Credit Timeout timer	1 ms	SSP_TF	7.16.8.6

Figure 182 shows the SSP state machines and states related to frame transmission.

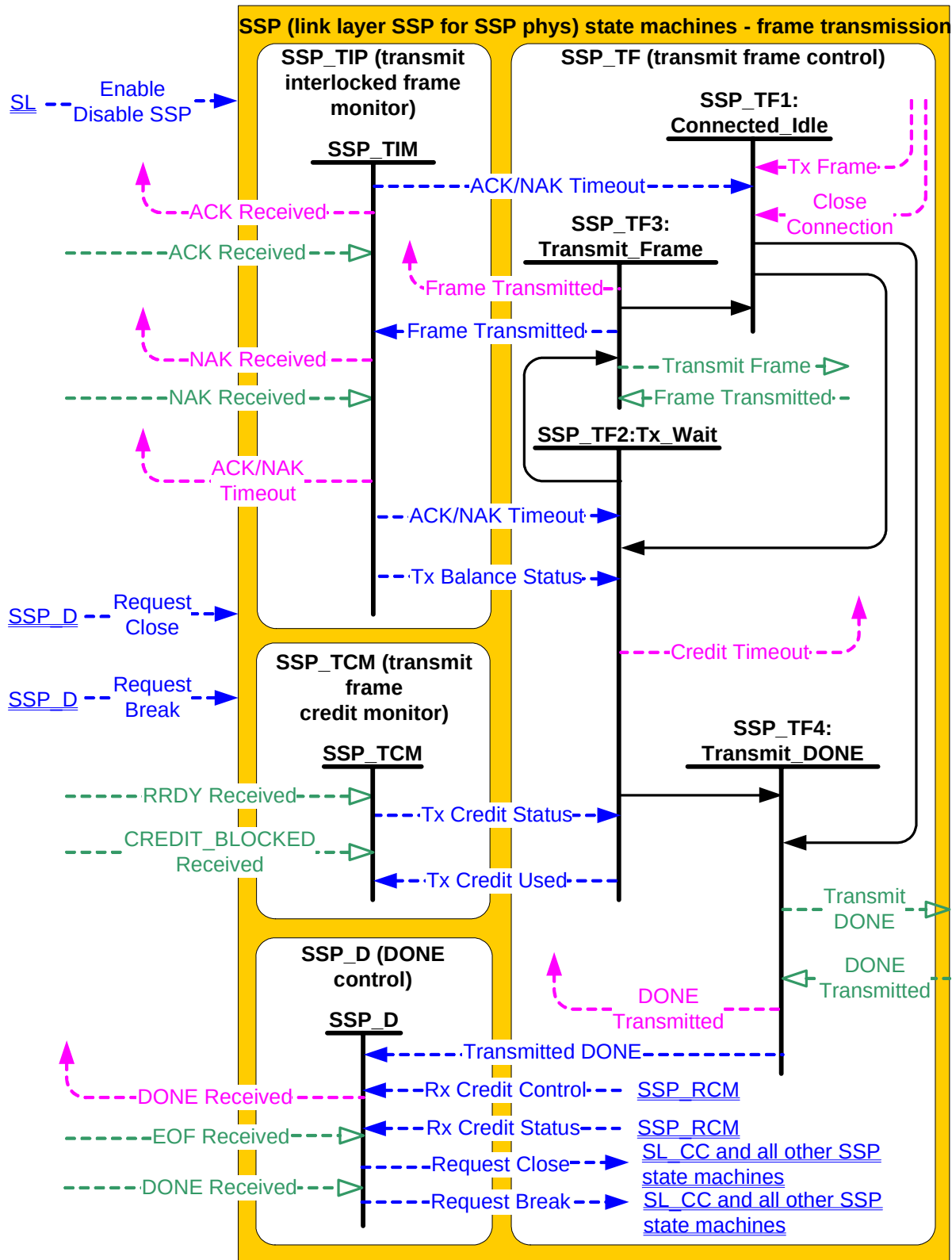


Figure 182 — SSP (link layer for SSP phys) state machines (part 1 - frame transmission)

Figure 183 shows the SSP state machines and states related to frame reception.

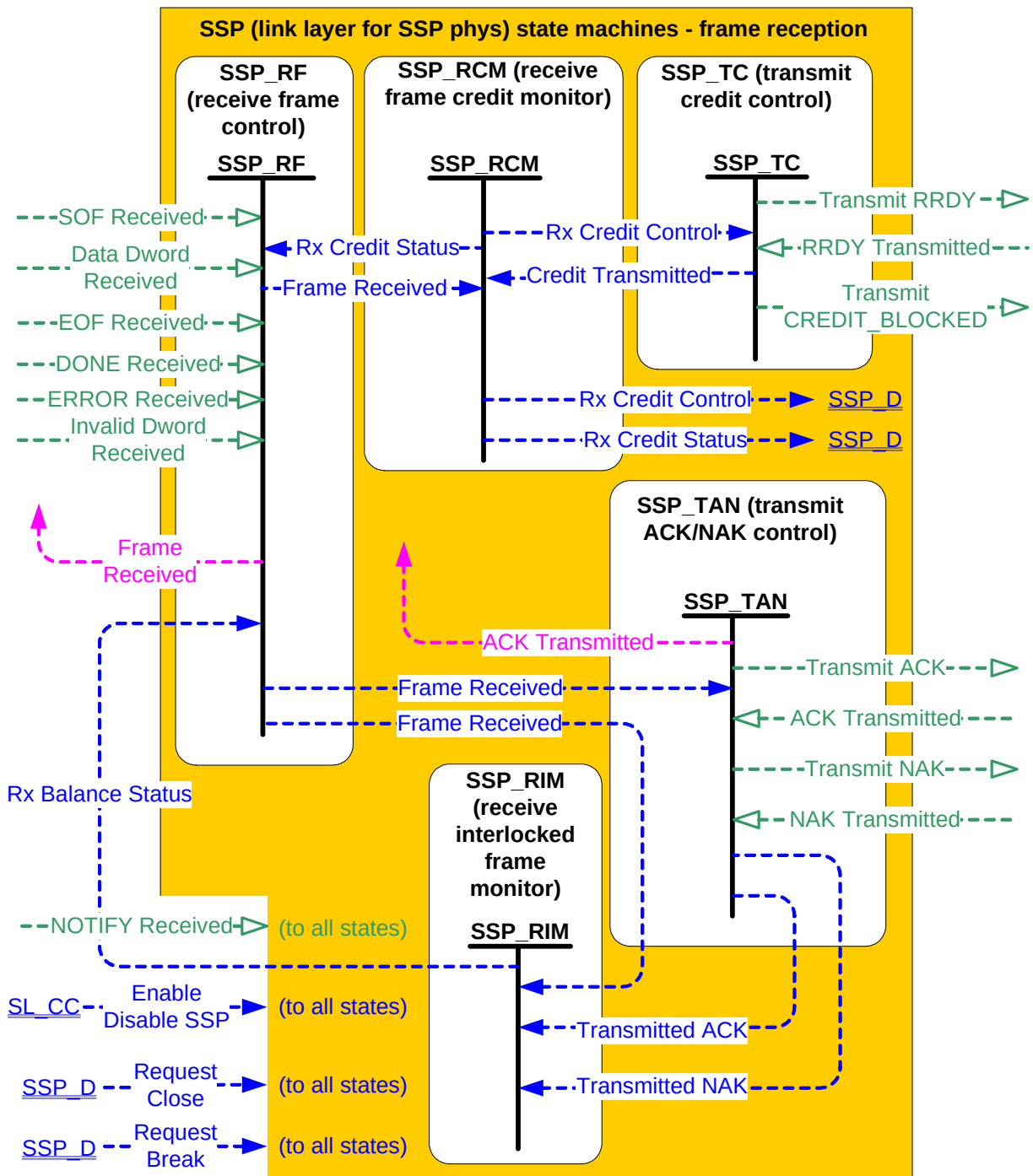


Figure 183 — SSP (link layer for SSP phys) state machines (part 2 - frame reception)

7.16.8.2 SSP transmitter and receiver

The SSP transmitter receives the following messages from the SSP state machines specifying primitive sequences and frames to transmit:

- Transmit RRDY with an argument indicating the specific type (e.g., Transmit RRDY (Normal));
- Transmit CREDIT_BLOCKED;
- Transmit ACK;

- d) Transmit NAK with an argument indicating the specific type (e.g., Transmit NAK (CRC Error));
- e) Transmit Frame with an argument containing the frame contents; and
- f) Transmit DONE with an argument indicating the specific type (e.g., Transmit DONE (Normal)).

In response to the Transmit Frame message, the SSP transmitter transmits:

- 1) SOF;
- 2) the frame contents;
- 3) CRC; and
- 4) EOF.

The SSP transmitter sends the following messages to the SSP state machines based on dwords that have been transmitted:

- a) DONE Transmitted;
- b) RRDY Transmitted;
- c) CREDIT_BLOCKED Transmitted;
- d) ACK Transmitted;
- e) NAK Transmitted; and
- f) Frame Transmitted.

When there is no outstanding message specifying a dword to transmit, the SSP transmitter shall transmit idle dwords.

The SSP receiver sends the following messages to the SSP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

- a) ACK Received;
- b) NAK Received;
- c) RRDY Received;
- d) CREDIT_BLOCKED Received;
- e) DONE Received with an argument indicating the specific type (e.g., DONE Received (Normal));
- f) SOF Received;
- g) Data Dword Received;
- h) EOF Received;
- i) NOTIFY Received (Power Loss Expected);
- j) ERROR Received; and
- k) Invalid Dword Received.

The SSP receiver shall ignore all other dwords.

The SSP transmitter relationship to other transmitters is defined in 4.3.2. The SSP receiver relationship to other receivers is defined in 4.3.3.

7.16.8.3 SSP_TIM (transmit interlocked frame monitor) state machine

The SSP_TIM state machine's function is to ensure that ACKs or NAKs are received for each transmitted frame before the ACK/NAK timeout. This state machine consists of one state.

This state machine monitors the number of frames transmitted with a Number Of Frames Transmitted counter and monitors the number of ACKs and NAKs received with a Number Of ACKs/NAKs Received counter. This state machine ensures that an ACK or NAK is received for each frame transmitted and reports an ACK/NAK timeout if they are not.

When the Number Of Frames Transmitted counter equals the Number Of ACKs/NAKs Received counter, the ACK/NAK count is balanced and this state machine shall send the Tx Balance Status (Balanced) message to the SSP_TF2:Tx_Wait state. When the Number Of Frames Transmitted counter does not equal the Number Of ACKs/NAKs Received counter, the ACK/NAK count is not balanced and this state machine shall send the Tx Balance Status (Not Balanced) message to the SSP_TF2:Tx_Wait state.

Each time a Frame Transmitted message is received, this state machine shall increment the Number Of Frames Transmitted counter.

If the ACK/NAK count is not balanced, each time an ACK Received message is received, this state machine shall:

- a) increment the Number Of ACKs/NAKs Received counter; and
- b) send an ACK Received confirmation to the port layer.

If the ACK/NAK count is not balanced, each time a NAK Received message is received, this state machine shall:

- a) increment the Number Of ACKs/NAKs Received counter; and
- b) send an NAK Received confirmation to the port layer.

If the ACK/NAK count is balanced, the ACK Received message and NAK Received message shall be ignored and the ACK/NAK Timeout timer shall be stopped.

Each time the ACK/NAK count is not balanced, the ACK/NAK Timeout timer shall be initialized and started. The ACK/NAK Timeout timer shall be re-initialized each time the Number Of ACKs/NAKs Received counter is incremented. If the ACK/NAK Timeout timer expires, this state machine shall send the ACK/NAK Timeout confirmation to the port layer and to the following states:

- a) SSP_TF1:Connected_Idle; and
- b) SSP_TF2:Tx_Wait state.

When this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the Number Of Frames Transmitted counter shall be set to zero and the Number Of ACKs/NAKs Received counter shall be set to zero.

7.16.8.4 SSP_TCM (transmit frame credit monitor) state machine

The SSP_TCM state machine's function is to ensure that transmit frame credit is available before a frame is transmitted. This state machine consists of one state.

This state machine shall keep track of the number of transmit frame credits available. This state machine shall add one transmit frame credit for each RRDY Received message received and subtract one transmit frame credit for each Tx Credit Used message received.

The CREDIT_BLOCKED Received message indicates that transmit frame credit is blocked. After receiving a CREDIT_BLOCKED Received message, this state machine may ignore additional RRDY Received messages until it receives a Request Close message or a Request Break message.

When transmit frame credit is available, this state machine shall send the Tx Credit Status (Available) message to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and transmit frame credit is not blocked, this state machine shall send the Tx Credit Status (Not Available) message to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and transmit frame credit is blocked, this state machine shall send the Tx Credit Status (Blocked) message to the SSP_TF2:Tx_Wait state.

When this state machine receives an Enable Disable SSP (Enable) message, a Request Close message, or a Request Break message, this state shall set transmit frame credit to not available and transmit frame credit shall be set to not blocked.

7.16.8.5 SSP_D (DONE control) state machine

The SSP_D state machine's function is to ensure a DONE has been received and transmitted before the SL_CC state machine disables the SSP state machines. This state machine consists of one state.

This state machine ensures that a DONE is received and transmitted before the connection is closed. The DONE may be transmitted and received in any order.

If the DONE Received message has been received before the Transmitted DONE message is received, this state machine shall send the Request Close message to the SL_CC state machine (see 7.14) and all the SSP state machines after receiving the Transmitted DONE message.

If a DONE Received message, the Transmitted DONE (Normal) message, or the Transmitted DONE (Credit Timeout) message has not been received and the Rx Credit Status (Extended) message or the Rx Credit Control (Blocked) message has been received, then this state shall initialize and start the DONE Timeout timer after receiving the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message.

If the DONE Received message has not been received and the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, this state machine shall initialize and start the DONE Timeout timer each time:

- a) the Rx Credit Status (Extended) message is received; or
- b) the Rx Credit Control (Blocked) message is received.

If the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, the DONE Timeout timer shall be reinitialized each time the EOF Received message is received.

If the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, the DONE Timeout timer shall be stopped after:

- a) the Rx Credit Status (Exhausted) message is received; and
- b) the Rx Credit Control (Blocked) message has not been received.

NOTE 64 - Stopping the timer ensures that, if credit remains exhausted long enough that the Credit Timeout timer of the other phy in the connection expires, the other phy is able to transmit a DONE (CREDIT TIMEOUT).

If the DONE Received message has not been received and the Transmitted DONE (ACK/NAK Timeout) message has been received:

- a) this state machine shall initialize and start the DONE Timeout timer; and
- b) this state shall not reinitialize the DONE Timeout timer if an EOF Received message is received.

If the DONE Received message is received before the DONE Timeout timer expires, this state machine shall send the Request Close message to the SL_CC state machine and all the SSP state machines.

If the DONE Received message is not received before the DONE Timeout timer expires, this state machine shall send a Request Break message to the SL_CC state machine and all the SSP state machines.

Any time a DONE Received message is received this state machine shall send a DONE Received confirmation to the port layer. A DONE Received (ACK/NAK Timeout) confirmation informs the port layer that the SSP transmitter is going to close the connection within 1 ms; other DONE Received confirmations (e.g., DONE Received (Close Connection) and DONE Received (Credit Timeout)) may be used by the application layer to decide when to reuse tags.

NOTE 65 - The DONE Timeout timer in one phy (e.g., phy A) may expire concurrently with the ACK/NAK Timeout timer in the other phy (e.g., phy B) in a connection.

For example, if phy A receives DONE (NORMAL) indicating phy B has no more frames to transmit, and phy A then transmits a series of non-interlocked frames where one or more of the SOFs is corrupted, then phy A waits to receive all the ACKs and/or NAKs after transmitting the series of non-interlocked frames. However, since phy B did not receive the full number of SOFs, it does not transmit as many ACKs and/or NAKs as phy A is expecting. The ACK/NAK Timeout timer in phy A expires and phy A transmits DONE (ACK/NAK TIMEOUT). Meanwhile, despite having transmitted DONE, phy B stops receiving frames while phy A is waiting for the final ACKs and/or NAKs. Since phy B does not receive DONE or any more frames, its DONE Timeout timer expires and phy B transmits BREAK.

Since the timers may expire at slightly different times (e.g., due to timer resolution differences), the DONE (ACK/NAK TIMEOUT) may be transmitted before, concurrently with, or after the BREAK. Nevertheless, the phys handle the link layer error (i.e., the ACK/NAK timeout or the DONE timeout) the same way (see 9.2.4.5).

7.16.8.6 SSP_TF (transmit frame control) state machine

7.16.8.6.1 SSP_TF state machine overview

The SSP_TF state machine's function is to control when the SSP transmitter transmits SOF, frame dwords, EOF, and DONE. This state machine consists of the following states:

- a) SSP_TF1:Connected_Idle (see 7.16.8.6.2)(initial state);
- b) SSP_TF2:Tx_Wait (see 7.16.8.6.3);
- c) SSP_TF3:Transmit_Frame (see 7.16.8.6.4); and
- d) SSP_TF4:Transmit_DONE (see 7.16.8.6.5).

This state machine shall start in the SSP_TF1:Connected_Idle state.

7.16.8.6.2 SSP_TF1:Connected_Idle state

7.16.8.6.2.1 State description

This state waits for a request to transmit a frame or to close the connection.

7.16.8.6.2.2 Transition SSP_TF1:Connected_Idle to SSP_TF2:Tx_Wait

This transition shall occur after a Tx Frame request is received or a Close Connection request is received.

If a Tx Frame (Balance Required) request was received this transition shall include a Transmit Frame Balance Required argument.

If a Tx Frame (Balance Not Required) request was received this transition shall include a Transmit Frame Balance Not Required argument.

If a Close Connection request was received this transition shall include a Close Connection argument.

7.16.8.6.2.3 Transition SSP_TF1:Connected_Idle to SSP_TF4:Transmit_DONE

This transition shall occur if an ACK/NAK Timeout message is received. This transition shall include an ACK/NAK Timeout argument.

7.16.8.6.3 SSP_TF2:Tx_Wait state

7.16.8.6.3.1 State description

This state monitors the Tx Balance Status message and the Tx Credit Status message to ensure that frames are transmitted and connections are closed at the proper time.

If this state is entered from the SSP_TF1:Connected_Idle state with a Transmit Frame Balance Required argument or a Transmit Frame Balance Not Required argument, and:

- a) if the last Tx Credit Status message received had an argument of Not Available, this state shall initialize and start the Credit Timeout timer; or
- b) if the last Tx Credit Status message had an argument other than Not Available, this state shall stop the Credit Timeout timer.

7.16.8.6.3.2 Transition SSP_TF2:Tx_Wait to SSP_TF3:Transmit_Frame

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Frame Balance Required if:

- a) the last Tx Balance Status message received had an argument of Balanced; and
- b) the last Tx Credit Status message received had an argument of Available.

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Frame Balance Not Required and if the last Tx Credit Status message received had an argument of Available.

This transition shall occur after sending a Tx Credit Used message to the SSP_TCM state machine.

7.16.8.6.3.3 Transition SSP_TF2:Tx_Wait to SSP_TF4:Transmit_DONE

This transition shall occur and include an ACK/NAK Timeout argument if an ACK/NAK Timeout message is received.

This transition shall occur and include a Close Connection argument if:

- a) this state was entered from the SSP_TF1:Connected_Idle state with an argument of Close Connection; and
- b) the last Tx Balance Status message received had an argument of Balanced.

This transition shall occur after sending a Credit Timeout confirmation and include a Credit Timeout argument if:

- a) this state was entered from the SSP_TF1:Connected_Idle state with a Transmit Frame Balance Required argument or a Transmit Frame Balance Not Required argument;
- b) the Credit Timeout timer expired before a Tx Credit Status message was received with an argument of Available, or the last Tx Credit Status message received had an argument of Blocked;
- c) a Tx Balance Status message was received with an argument of Balanced (i.e., the Credit Timeout argument shall not be included in this transition for this reason unless the ACK/NAK count is balanced); and
- d) an ACK/NAK Timeout message was not received.

7.16.8.6.4 SSP_TF3:Transmit_Frame state**7.16.8.6.4.1 State description**

This state shall request a frame transmission by sending a Transmit Frame message to the SSP transmitter with an argument containing the frame contents. Each time a Transmit Frame message is sent to the SSP transmitter, one SSP frame (i.e., SOF, frame contents, CRC, and EOF) is transmitted.

In this state receiving a Frame Transmitted message indicates that the frame has been transmitted.

7.16.8.6.4.2 Transition SSP_TF3:Transmit_Frame to SSP_TF1:Connected_Idle

This transition shall occur after:

- a) receiving a Frame Transmitted message;
- b) sending an Frame Transmitted message to the SSP_TIM state machine; and
- c) sending a Frame Transmitted confirmation to the port layer.

7.16.8.6.5 SSP_TF4:Transmit_DONE state

This state shall send one of the following messages to an SSP transmitter:

- a) a Transmit DONE (Normal) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Close Connection;
- b) a Transmit DONE (ACK/NAK Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state or the SSP_TF1:Connected_Idle state with an argument of ACK/NAK Timeout; or
- c) a Transmit DONE (Credit Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Credit Timeout.

NOTE 66 - Possible livelock scenarios occur if the BREAK_REPLY method of responding to received BREAK primitive sequences is disabled and a SAS logical phy transmits BREAK to break a connection (e.g., if its Done Timeout timer expires). SAS logical phys should respond to DONE faster than 1 ms to reduce susceptibility to this problem.

After a DONE Transmitted message is received this state shall send the DONE Transmitted confirmation to the port layer and send one of the following messages to the SSP_D state machine:

- a) a Transmitted DONE (Normal) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Close Connection;

- b) a Transmitted DONE (ACK/NAK Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state or the SSP_TF1:Connected_Idle state with an argument of ACK/NAK Timeout; or
- c) a Transmitted DONE (Credit Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Credit Timeout.

7.16.8.7 SSP_RF (receive frame control) state machine

The SSP_RF state machine's function is to receive frames and determine whether or not those frames were received successfully. This state machine consists of one state.

This state machine:

- a) checks the frame to determine if the frame should be accepted or discarded;
- b) checks the frame to determine if an ACK or NAK should be transmitted; and
- c) sends a Frame Received confirmation to the port layer.

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame if:

- a) this state receives more than 263 Data Dword Received messages after an SOF Received message and before an EOF Received message;
- b) this state receives fewer than 7 Data Dword Received messages after an SOF Received message and before an EOF Received message;
- c) this state receives an Rx Credit Status (Credit Exhausted) message; or
- d) this state receives a DONE Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after receiving an SOF Received message and before receiving an EOF Received message, then this state machine shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the frame, send a Frame Received message to the SSP_RCM state machine, send a Frame Received message to the SSP_RIM state machine, and send a Frame Received (Unsuccessful) message to the SSP_TAN state machine.

If the frame is not discarded and the frame CRC is bad, this state machine shall:

- a) send a Frame Received message to the SSP_RCM state machine;
- b) send a Frame Received message to the SSP_RIM state machine; and
- c) send a Frame Received (Unsuccessful) message to the SSP_TAN state machine.

If the frame is not discarded and the frame CRC is good, this state machine shall send a Frame Received (Successful) message to the SSP_TAN state machine and:

- a) send a Frame Received message to the SSP_RCM state machine;
- b) send a Frame Received message to the SSP_RIM state machine; and
- c) send a Frame Received (Successful) message to the SSP_TAN state machine, and:
 - A) if the last Rx Balance Status message received had an argument of Balanced, send a Frame Received (ACK/NAK Balanced) confirmation to the port layer; or
 - B) if the last Rx Balance Status message received had an argument of Not Balanced, send a Frame Received (ACK/NAK Not Balanced) confirmation to the port layer.

7.16.8.8 SSP_RCM (receive frame credit monitor) state machine

The SSP_RCM state machine's function is to ensure that there was credit given to the originator for every frame that is received. This state machine consists of one state.

This state machine monitors the receiver's resources and keeps track of the number of RRDYs transmitted versus the number of frames received.

Any time resources are released or become available, if this state machine has not sent the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine, this state machine shall send the Rx Credit Control (Available) message to the SSP_TC state machine. This state machine shall only send the Rx Credit Control (Available) message to the SSP_TC state machine after frame receive resources become available. The specifications for when or how resources become available is outside the scope of this standard.

This state machine may send the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine when no further receive frame credit is going to become available within a credit timeout (i.e., less than 1 ms), even if frames are received per the existing receive frame credit. After sending the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine, this state machine shall not send the Rx Credit Control (Available) message to the SSP_TC state machine or the SSP_D state machine for the duration of the current connection.

This state machine shall indicate through the Rx Credit Control message only the amount of resources available to handle received frames (e.g., if this state machine has resources for five frames the maximum number of Rx Credit Control requests with the Available argument outstanding is five).

This state machine shall use the Credit Transmitted message to keep track of the number of RRDYs transmitted. This state machine shall use the Frame Received message to keep a track of the number of frames received.

Any time the number of Credit Transmitted messages received exceeds the number of Frame Received messages received this state machine shall send a Rx Credit Status (Extended) message to the SSP_RF state machine and the SSP_D state machine.

Any time the number of Credit Transmitted messages received equals the number of Frame Received messages received this state machine shall send a Rx Credit Status (Exhausted) message to the SSP_RF state machine and the SSP_D state machine.

If this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the frame receive resources shall be initialized to the no credit value for the current connection.

7.16.8.9 SSP_RIM (receive interlocked frame monitor) state machine

The SSP_RIM state machine's function is to inform the SSP_RF state machine when the number of ACKs and NAKs transmitted equals the number of the EOFs received. This state machine consists of one state.

This state machine monitors the number of frames received with a Number Of Frames Received counter and monitors the number of ACKs and NAKs transmitted with a Number Of ACKs/NAKs Transmitted counter.

Each time a Frame Received message is received, this state machine shall increment the Number Of Frames Received counter.

Each time an ACK Transmitted message or a NAK Transmitted is received, this state machine shall increment the Number Of ACKs/NAKs Transmitted counter.

When the Number Of Frames Received counter equals the Number Of ACKs/NAKs Transmitted counter, this state machine shall send an Rx Balance Status (Balanced) message to the SSP_RF state machine.

When the Number Of Frames Received counter does not equal the Number Of ACKs/NAKs Transmitted counter, this state machine shall send an Rx Balance Status (Not Balanced) message to the SSP_RF state machine.

When this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the Number Of Frames Received counter shall be set to zero and the Number Of ACKs/NAKs Transmitted counter shall be set to zero.

7.16.8.10 SSP_TC (transmit credit control) state machine

The SSP_TC state machine's function is to control the sending of requests to transmit an RRDY or CREDIT_BLOCKED. This state machine consists of one state.

Any time this state machine receives a Rx Credit Control (Available) message it shall send a number of Transmit RRDY (Normal) messages to the SSP transmitter as indicated by the amount of resources available to handle received frames (e.g., if the Available argument indicates five RRDYs are to be transmitted this state machine sends five Transmit RRDY (Normal) messages to the SSP transmitter).

Any time this state machine receives a RRDY Transmitted message it shall send a Credit Transmitted message to the SSP_RCM state machine.

Any time this state machine receives a Rx Credit Control (Blocked) message it shall send a Transmit CREDIT_BLOCKED message to the SSP transmitter.

7.16.8.11 SSP_TAN (transmit ACK/NAK control) state machine

The SSP_TAN state machine's function is to control the sending of requests to transmit an ACK or NAK to the SSP transmitter. This state machine consists of one state.

Any time this state machine receives a Frame Received (Successful) message it shall send a Transmit ACK message to the SSP transmitter.

Any time this state machine receives a Frame Received (Unsuccessful) message it shall send a Transmit NAK (CRC Error) message to the SSP transmitter.

If multiple Frame Received (Unsuccessful) messages and Frame Received (Successful) messages are received, then the order in which the Transmit ACK messages and Transmit NAK messages are sent to the SSP transmitter shall be the same order as the Frame Received (Unsuccessful) messages and Frame Received (Successful) messages were received.

Any time this state machine receives an ACK Transmitted message it shall:

- a) send a Transmitted ACK message to the SSP_RIM state machine; and
- b) send an ACK Transmitted confirmation to the port layer.

Any time this state receives a NAK Transmitted argument it shall send a Transmitted NAK message to the SSP_RIM state machine.

7.17 STP link layer

7.17.1 STP frame transmission and reception

STP frame transmission is defined by SATA (see SATA-2). During an STP connection, frames are preceded by SATA_SOFTWARE and followed by SATA_EOF as shown in figure 184.

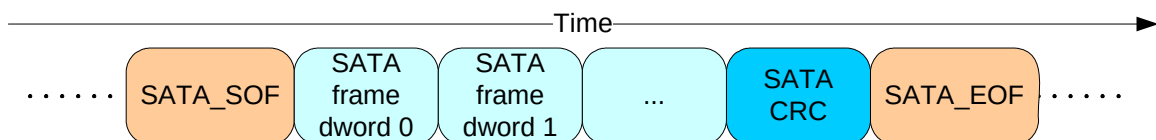


Figure 184 — STP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5).

Other primitives may be interspersed during the connection as defined by SATA.

STP encapsulates SATA with connection management. Table 148 summarizes STP link layer differences from the SATA link layer (see SATA-2) that affect behavior during an STP connection.

Table 148 — STP link layer differences from SATA link layer during an STP connection

Feature	Description	Reference
STP flow control	Flow control through an STP connection is point-to-point, not end-to-end. Expander devices accept dwords in an STP flow control buffer after transmitting SATA_HOLD to avoid losing data en-route before the transmitting phy acknowledges the SATA_HOLD with SATA_HOLDA.	7.17.2
Continued primitive sequence	Sustain the continued primitive sequence if a SATA_CONT appears after the continued primitive sequence has begun.	7.17.3

7.17.2 STP flow control

Each STP phy (i.e., STP initiator phy and STP target phy) and expander logical phy through which the STP connection is routed shall implement the SATA flow control protocol on each logical link in the pathway. The flow control primitives are not forwarded through expander devices like other dwords.

When an STP phy or expander phy during an STP connection is receiving a SATA frame and its STP flow control buffer begins to fill up, it shall transmit SATA_HOLD. After transmitting SATA_HOLD, it shall accept at least the following number of data dwords for the SATA frame into its STP flow control buffer:

- 24 data dwords at the 1,5 Gbps connection rate;
- 28 data dwords at the 3 Gbps connection rate; and
- 36 data dwords at the 6 Gbps connection rate.

and shall expect to receive SATA_HOLDA within that number of data dwords. While receiving SATA_HOLDA, it does not place any data dwords into the STP flow control buffer. When the STP flow control buffer empties enough to hold at least that number of data dwords, it shall stop transmitting SATA_HOLD.

When an STP phy or expander phy during an STP connection is transmitting a SATA frame and receives SATA_HOLD, it shall transmit no more than 20 data dwords for the SATA frame and respond with SATA_HOLDA.

NOTE 67 - The STP flow control buffer requirements are based on $(20 + (4 \times 2^n))$ where n is 0 for 1,5 Gbps, 1 for 3 Gbps, and 2 for 6 Gbps. The 20 portion of this equation is based on the frame transmitter requirements (see SATA-2). The (4×2^n) portion of this equation is based on:

- One-way propagation time on a 10 m cable = $(5 \text{ ns/m propagation delay}) \times (10 \text{ m cable}) = 50 \text{ ns}$;
- Round-trip propagation time on a 10 m cable = 100 ns (e.g., time to send SATA_HOLD and receive SATA_HOLD A);
- Time to transmit a 1,5 Gbps dword = $(0,667 \text{ ns/bit unit interval}) \times (40 \text{ bits/dword}) = 26,667 \text{ ns}$; and
- Number of 1,5 Gbps dwords on the wire during round-trip propagation time = $(100 \text{ ns} / 26,667 \text{ ns}) = 3,75$.

Receivers may support longer cables by providing larger STP flow control buffer sizes.

When a SATA host phy in an STP/SATA bridge is receiving a SATA frame from a SATA physical link, it shall transmit a SATA_HOLD when it is only capable of receiving 21 more data dwords. It shall stop transmitting SATA_HOLD (e.g., return to transmitting SATA_R_IP) when it is capable of receiving at least 21 more data dwords.

NOTE 68 - SATA requires that frame transmission cease and SATA_HOLD A be transmitted within 20 data dwords of receiving SATA_HOLD. Since the SATA physical link has non-zero propagation time, one dword of margin is included.

When a SATA host phy in an STP/SATA bridge is transmitting a SATA frame to a SATA physical link, it shall transmit no more than 19 data dwords after receiving SATA_HOLD.

NOTE 69 - SATA assumes that once a SATA_HOLD is transmitted, frame transmission ceases and SATA_HOLDDA arrives within 20 dwords. Since the SATA physical link has non-zero propagation time, one dword of margin is included.

Figure 185 shows STP flow control between:

- a) an STP initiator phy receiving a frame;
- b) an expander device (the first expander device);
- c) an expander device with an STP/SATA bridge (the second expander device); and
- d) a SATA device phy transmitting a frame.

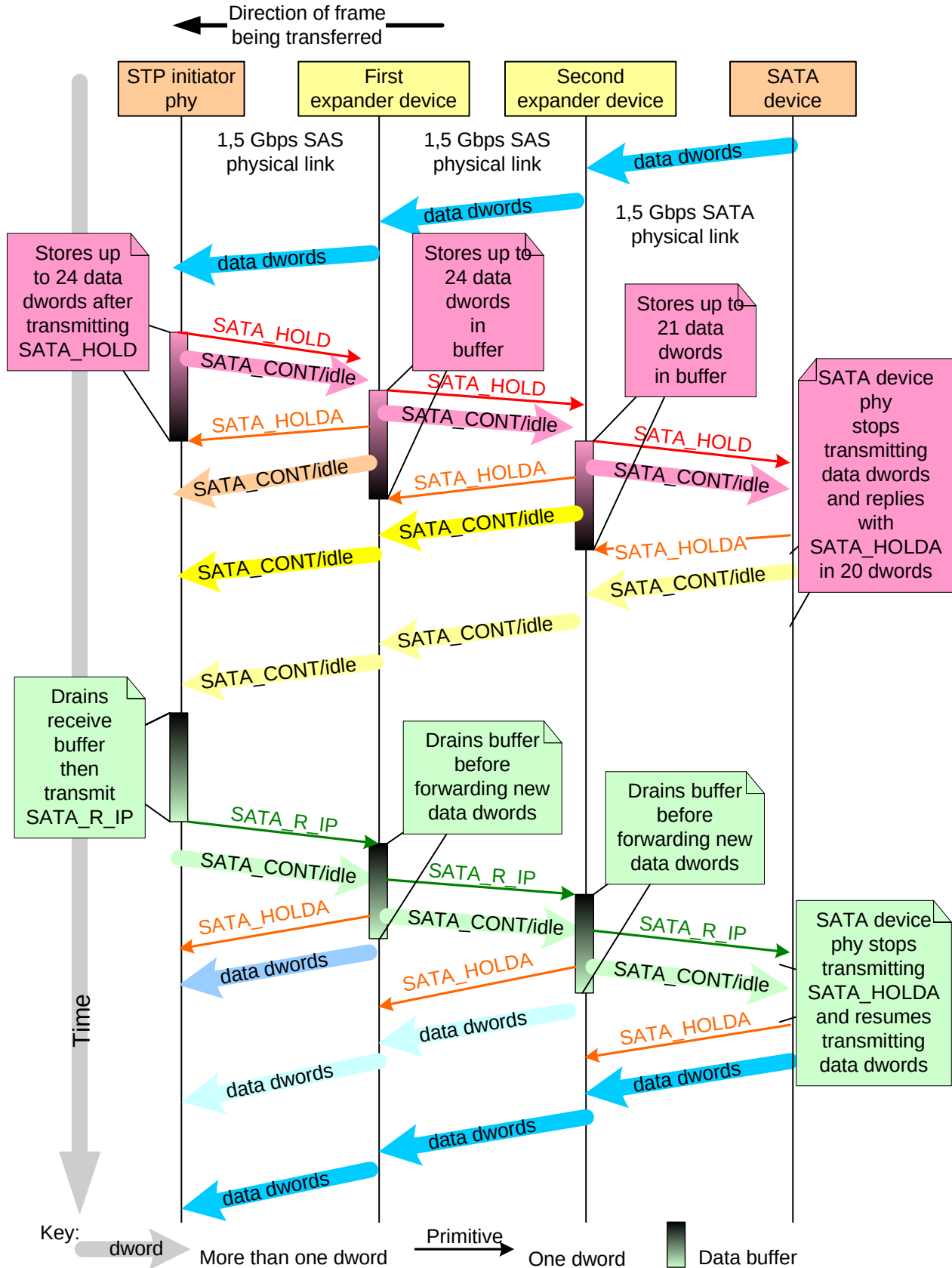


Figure 185 — STP flow control

After the STP initiator phy transmits SATA_HOLD, it receives a SATA_HOLDA reply from the first expander device within 24 dwords. The first expander device transmits SATA_HOLD to the second expander device and receives SATA_HOLD within 24 dwords, buffering data dwords in the STP flow control buffer that it is no longer able to forward to the STP initiator phy. The second expander device transmits SATA_HOLD to the SATA device phy and receives SATA_HOLD within 21 dwords, buffering data dwords in the STP flow control buffer that it is no longer able to forward to the first expander device. When the SATA device phy stops transmitting data dwords, its previous data dwords are stored in the STP flow control buffers in both expander devices and the STP initiator phy.

After the STP initiator phy drains its STP flow control buffer and transmits SATA_R_IP, it receives data dwords from the first expander device's STP flow control buffer, followed by data dwords from the second expander device's STP flow control buffer, followed by data dwords from the SATA device phy.

7.17.3 Continued primitive sequence

Primitives that form continued primitive sequences (e.g., SATA_HOLD) shall be:

- 1) transmitted two times;
- 2) be followed by SATA_CONT, if needed; and
- 3) be followed by vendor-specific scrambled data dwords, if needed.

Deletable primitives may be sent inside continued primitive sequences as described in 7.2.4.1.

After the SATA_CONT, during the vendor-specific scrambled data dwords:

- a) a SATA_CONT continues the continued primitive sequence; and
- b) any other STP primitive, including the primitive that is being continued, ends the continued primitive sequence.

Figure 186 shows an example of transmitting a continued primitive sequence.

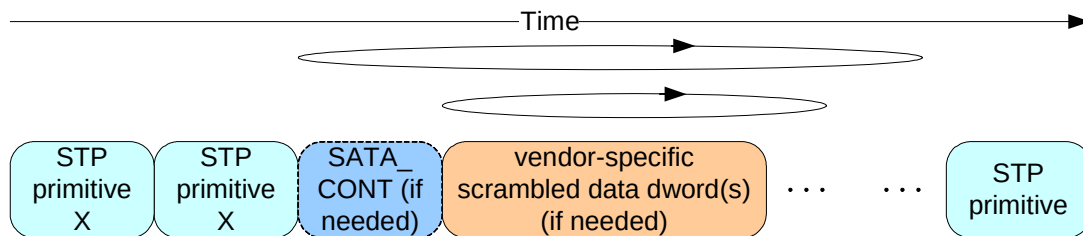


Figure 186 — Transmitting a continued primitive sequence

Receivers shall detect a continued primitive sequence after at least one primitive is received. The primitive may be followed by one or more of the same primitive. The primitive may be followed by one or more SATA_CONTs, each of which may be followed by vendor-specific data dwords. Receivers shall ignore invalid dwords before, during, or after the SATA_CONT(s). Receivers do not count the number of times the continued primitive, the SATA_CONTs, or the vendor-specific data dwords are received (i.e., receivers are simply in the state of receiving the primitive).

Expanders forwarding dwords may or may not detect an incoming sequence of the same primitive and convert it into a continued primitive sequence.

Figure 187 shows an example of receiving a continued primitive sequence.

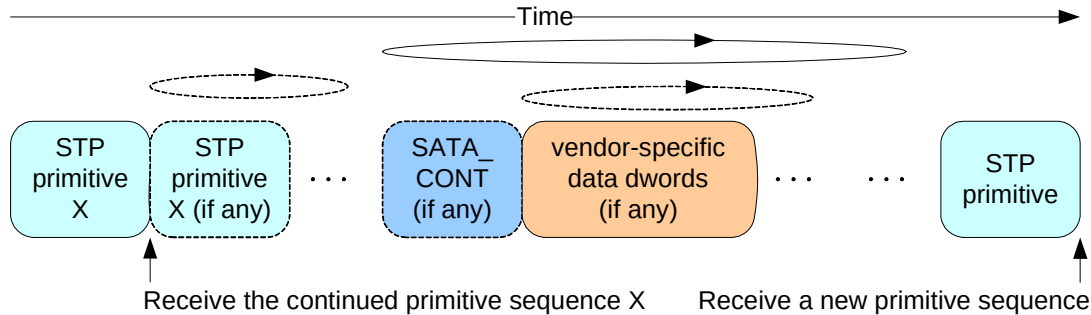


Figure 187 — Receiving a continued primitive sequence

An expander device forwarding a continued primitive sequence may transmit more dwords in the continued primitive sequence than it receives (i.e., expand) or transmit fewer dwords in the continued primitive sequence than it receives (i.e., contract).

7.17.4 Affiliations

The STP target port shall provide coherent access to an ATA task file register set for each STP initiator port from which it accepts connections. It may use affiliations to limit the number of STP initiator ports that establish connections and for which it maintains an ATA task file register set. An affiliation is a state entered by an STP target port where it refuses to accept connection requests from STP initiator ports other than those that have established an affiliation.

STP target ports implement one of the affiliation policies defined in table 149.

Table 149 — Affiliation policies

Affiliation policy	Description
No affiliations	An unlimited number of STP initiator ports are allowed to access the STP target port concurrently. The STP target port is cognizant of the SAS address of the STP initiator port that sends each ATA command.
Multiple affiliations	The STP target port implements more than one affiliation, so a limited number of STP initiator ports are allowed to access the STP target port concurrently. The STP target port implements no more than one affiliation context per STP initiator port.
Single affiliation	The STP target port implements one affiliation, so one STP initiator port is allowed to access the STP target port at a time.

An STP/SATA bridge that supports multiple affiliations shall:

- ensure that the SATA NCQ tags in commands issued to the SATA device are unique across all affiliations;
- ensure that a non-queued command received in one affiliation context is not issued to the SATA device while another affiliation context has a queued command outstanding to the drive (e.g., the STP target port shall allow all queued commands in the SATA device to complete prior to issuing the non-queued command); and
- ensure that a queued command received in one affiliation context is not issued to the SATA device while another affiliation context has a non-queued command outstanding to the drive (e.g., the STP target port shall allow any non-queued command in the SATA device to complete prior to issuing the queued commands).

An STP/SATA bridge that supports multiple affiliations may modify the queue depth reported in the ATA IDENTIFY DEVICE data (see ATA8-ACS) to each STP initiator port to ensure that all the STP initiator ports with affiliations do not send more commands than the SATA device supports.

An STP target port that supports affiliations shall establish an affiliation whenever it accepts a connection request from an STP initiator port that does not already have an affiliation. When the maximum number of affiliations have been established (i.e., all affiliation contexts are in use), the STP target port shall reject all subsequent connection requests from other STP initiator ports with OPEN_REJECT (STP RESOURCES BUSY).

An STP target port shall maintain an affiliation until any of the following occurs:

- a) power on;
- b) the management device server receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of HARD RESET (see 10.4.3.27) from any SMP initiator port;
- c) the management device server receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of TRANSMIT SATA PORT SELECTION SIGNAL (see 10.4.3.27) from any SMP initiator port;
- d) the management device server receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of CLEAR AFFILIATION (see 10.4.3.27) from the same SAS initiator port that has the affiliation.

If a connection is already established to the STP target port on one phy while an SMP PHY CONTROL request specifying a phy operation of CLEAR AFFILIATION is processed by an SMP target port on another phy, the affiliation shall be cleared and the STP target port shall respond to new connection attempts with:

- A) AIP (WAITING ON CONNECTION) and/or OPEN_REJECT (RETRY), if the STP target port is in an expander device; or
 - B) OPEN_REJECT (RETRY), if the STP target port is in a SAS device;
- rather than OPEN_REJECT (STP RESOURCES BUSY);
- e) an STP connection to a phy in the STP target port is closed with CLOSE (CLEAR AFFILIATION); or
 - f) the STP target port encounters an I_T nexus loss.

The STP initiator port shall maintain an affiliation starting with the connection in which a command is transmitted until all frames for the command have been delivered. An STP initiator port implementing command queuing (see ATA8-ACS and SATA-2) shall maintain an affiliation while any commands are outstanding. STP initiator ports may keep affiliations for longer tenures, but this is discouraged.

An STP target port that implements affiliations shall implement at least one affiliation context per STP target port. Multiple phys on the same STP target port shall use the same set of affiliation contexts. Support for affiliations is indicated in the SMP REPORT PHY SATA response (see 10.4.3.11).

An STP target port implementing multiple affiliations shall sort the affiliation contexts in a vendor-specific order. In the SMP REPORT PHY SATA response, if the SMP initiator port has the same SAS address as an affiliated STP initiator port, the management device server shall report the affiliation for that SAS address as relative identifier 0 and shall report all additional affiliations with incrementing relative identifiers following the sorted order. If the SMP initiator port does not have the same SAS address as an affiliated STP initiator port, the management device server shall report the affiliation contexts in the vendor-specific order.

For example, if the STP target port supports four affiliation contexts sorted in order A, B, C, and D, when returning the SMP REPORT PHY SATA response to an SMP initiator port, the management device server shall report the affiliation contexts as described in table 150.

Table 150 — Affiliation context relative identifier example

Affiliation context containing the SAS address of the SMP initiator port	Affiliation context relative identifier assignment			
	0	1	2	3
A	A	B	C	D
B	B	C	D	A
C	C	D	A	B
D	D	A	B	C
None	A	B	C	D

7.17.5 Opening an STP connection

When the SATA host port in an STP/SATA bridge receives a SATA_X_RDY from the attached SATA device, the STP target port in the STP/SATA bridge shall establish an STP connection to the appropriate STP initiator port.

[Editor's Note 40: If there is no affiliation established when this occurs, what happens?](#)

[Editor's Note 41: In a multiple-affiliation STP/SATA bridge, if the tag doesn't match any affiliation, what happens?](#)

If an STP/SATA bridge receives a connection request for a SATA device that has not successfully delivered the initial Register – Device to Host FIS, it shall return an OPEN_REJECT (NO DESTINATION).

NOTE 70 - If there is a problem receiving the expected initial Register - Device to Host FIS, the STP/SATA bridge should use SATA_R_ERR to retry until it succeeds. In the DISCOVER response, the ATTACHED SATA DEVICE bit is set to one and the ATTACHED SAS ADDRESS field is valid, but the ATTACHED DEVICE TYPE field is set to 000b (i.e., no device attached) during this time.

If an STP/SATA bridge that retrieves IDENTIFY (PACKET) DEVICE data receives a connection request for a SATA device before it has retrieved the IDENTIFY (PACKET) DEVICE data, it shall return an OPEN_REJECT (NO DESTINATION). If it has a problem retrieving the IDENTIFY (PACKET) DEVICE data (e.g., word 255 (i.e., the Integrity Word) is not correct), the STP/SATA bridge shall set the ATTACHED DEVICE NAME field to zero, set the ATTACHED DEVICE TYPE field to 001b (i.e., end device), and start accepting connections.

A wide STP initiator port shall not request more than one connection at a time to a specific STP target port.

While a wide STP initiator port is waiting for a response to a connection request to an STP target port, a SAS phy in the STP initiator port shall not reject an incoming connection request from that STP target port with OPEN_REJECT (RETRY) because the SAS port containing that SAS phy needs an outgoing connection request to be accepted. The SAS phy may reject an incoming connection request from that STP target port with OPEN_REJECT (RETRY) for any reason that is not dependent on the SAS port containing that SAS phy having an outgoing connection request accepted (e.g., because of a temporary buffer full condition).

If a wide STP initiator port receives an incoming connection request from an STP target port while it has a connection established with that STP target port, it shall reject the request with OPEN_REJECT (RETRY).

A wide STP target port shall not request more than one connection at a time to a specific STP initiator port.

While a wide STP target port is waiting for a response to a connection request or has established a connection to an STP initiator port, it shall:

- a) reject incoming connection requests from that STP initiator port with OPEN_REJECT (RETRY); and
- b) if affiliations are supported and the maximum number of affiliations has been established (i.e., all affiliation contexts are in use), reject incoming connection requests from other STP initiator ports that do not have affiliations with OPEN_REJECT (STP RESOURCES BUSY).

A SAS phy may reject an incoming connection request (i.e., an OPEN address frame) to an STP target port with OPEN_REJECT (RETRY) for any reason, including because the SAS port containing that SAS phy needs an outgoing connection request to be accepted (e.g., to transmit a frame and empty a buffer).

An expander device should not allow its STP ports (e.g., the STP target ports in STP/SATA bridges and any STP initiator ports in the expander device) to attempt to establish more connections to a specific destination port than the destination port width or the width of the narrowest physical link on the pathway to the destination port. This does not apply to connection requests being forwarded by the expander device.

An expander device should not allow its STP ports (e.g., the STP target ports in STP/SATA bridges and any STP initiator ports in the expander device) to attempt to establish more connections than the width of the narrowest common physical link on the pathways to the destination ports of those connections. This does not apply to connection requests being forwarded by the expander device.

Figure 188 shows an example of the simultaneous connection recommendations for an expander device containing STP ports.

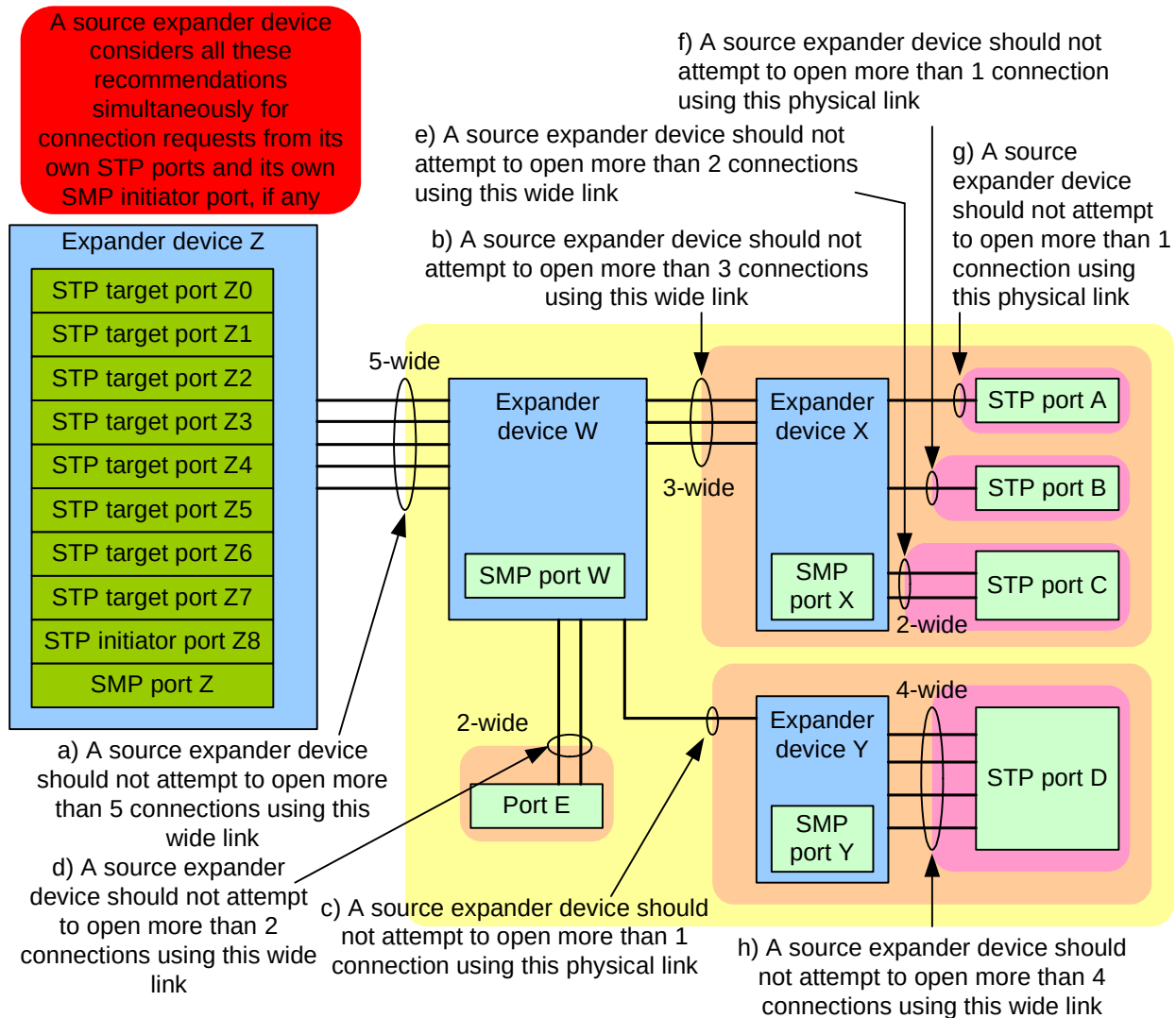


Figure 188 — Example simultaneous connection recommendations for an expander device

In figure 188, some of the recommendations are combined as follows:

- a) recommendations a), b), and e) together mean expander device Z should not attempt to open more than 2 connections to port C;
- b) recommendations a), b), e), f), and g) together mean that if expander device Z has 2 connections open to ports A, B, and X, it should not attempt to open more than 1 connection to port C. If it has 4 connections open to ports A, B, D, E, W, X, and Y, it should not attempt to open more than 1 connection to port C; and
- c) recommendations a), c), and h) together mean expander device Z should not attempt to open more than 1 connection to port D. If it has a connection open to port Y, it should not attempt to open another connection to port D until the first connection is closed.

The first dword that an STP phy sends inside an STP connection after OPEN_ACCEPT that is not a deletable primitive shall be an STP primitive (e.g., SATA_SYNC).

7.17.6 Closing an STP connection

Either STP port (i.e., either the STP initiator port or the STP target port) may originate closing an STP connection. An STP port shall not originate closing an STP connection after sending a SATA_X_RDY or

SATA_R_RDY until after both sending and receiving SATA_SYNC. An STP port shall transmit CLOSE after receiving a CLOSE if it has not already transmitted CLOSE.

If an STP port receives a CLOSE after transmitting a SATA_X_RDY but before receiving a SATA_R_RDY, the STP port shall complete closing the connection (i.e., transmit CLOSE) and retransmit the SATA_X_RDY in a new connection.

When an STP initiator port closes an STP connection, it shall transmit a CLOSE (NORMAL) or CLOSE (CLEAR AFFILIATION). When an STP target port closes an STP connection, it shall transmit a CLOSE (NORMAL).

An STP initiator port may issue CLOSE (CLEAR AFFILIATION) in place of a CLOSE (NORMAL) to cause the STP target port to clear the affiliation (see 7.17.4) along with closing the connection. If an STP target port receives CLOSE (CLEAR AFFILIATION), the STP target port shall clear the affiliation for the STP initiator port that sent the CLOSE (CLEAR AFFILIATION).

See 7.12.7 for additional details on closing connections.

An STP/SATA bridge shall break an STP connection if its SATA host phy loses dword synchronization (see 7.12.8).

7.17.7 STP connection management examples

The STP/SATA bridge adds the outgoing OPEN address frames and CLOSEs so the STP initiator port sees an STP target port. The STP/SATA bridge removes incoming OPEN address frame and CLOSEs so the SATA device port sees only a SATA host port. While the connection is open, the STP/SATA bridge passes through all dwords without modification. Both STP initiator port and STP target port use SATA, with SATA flow control (see 7.17.2), while the connection is open.

Figure 189 — STP initiator port opening an STP connection



Figure 190 shows a SATA device transmitting a SATA frame. In this example, the STP target port in the STP/SATA bridge opens a connection to an STP initiator port to send just one frame, then closes the connection.

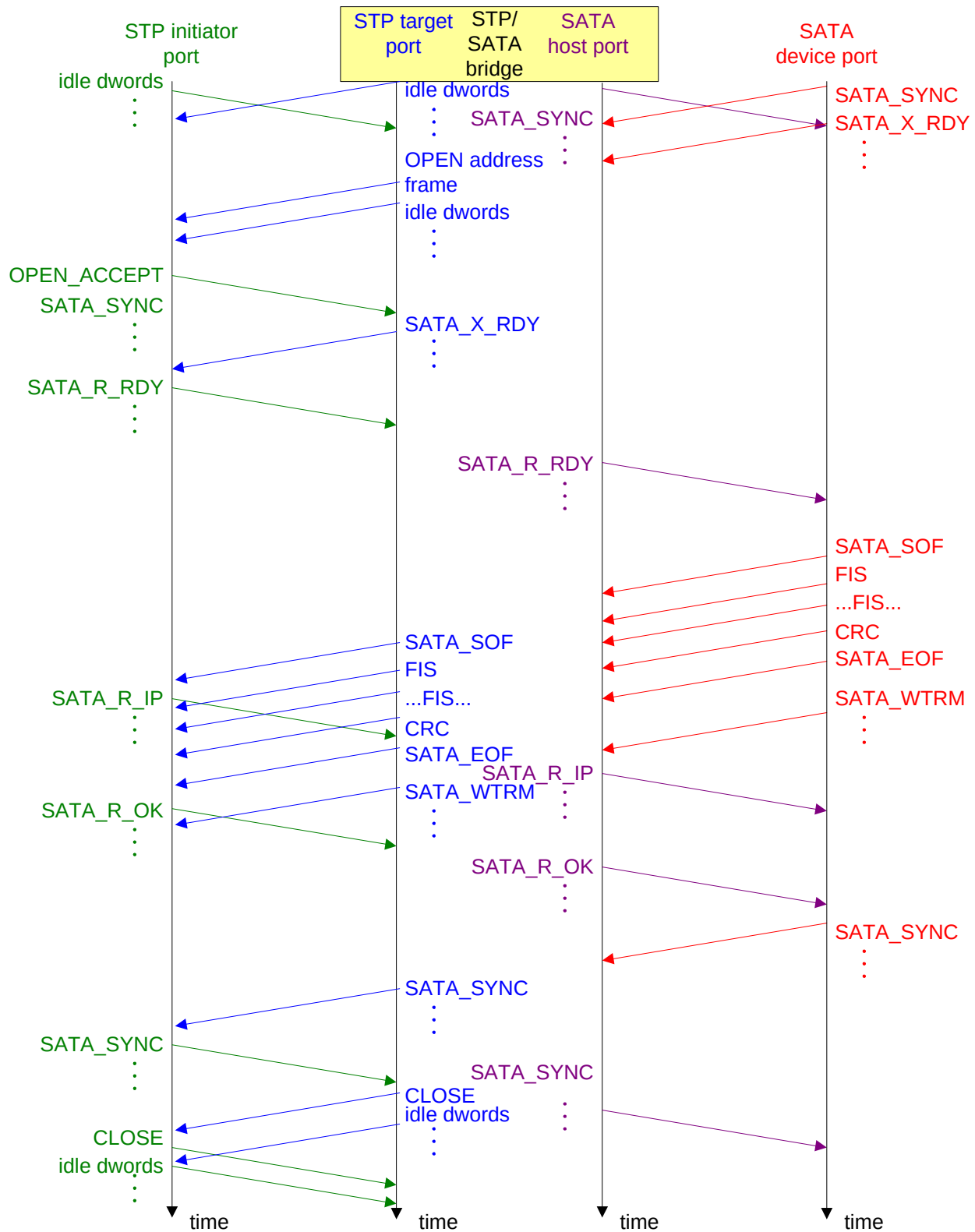


Figure 190 — STP target port opening an STP connection

7.17.8 STP (link layer for STP phys) state machines

The STP link layer uses the SATA link layer state machines (see SATA-2), modified to:

- communicate with the port layer rather than directly with the transport layer;
- interface with the SL state machines for connection management (e.g., to select when to open and close STP connections, and to tolerate idle dwords between an OPEN address frame and the first SATA primitive);
- communicate with an STP transmitter and receiver; and
- support an affiliation policy (see 7.17.4).

These modifications are not described in this standard.

The STP transmitter relationship to other transmitters is defined in 4.3.2. The STP receiver relationship to other receivers is defined in 4.3.3.

7.17.9 SMP target port support

A SAS device that contains an STP target port shall also contain an SMP target port.

7.18 SMP link layer

7.18.1 SMP frame transmission and reception

Inside an SMP connection, the SMP initiator phy transmits a single SMP_REQUEST frame within 100 μ s and the SMP target phy responds with a single SMP_RESPONSE frame (see 9.4) within 1 900 μ s.

Frames are surrounded by SOF and EOF as shown in figure 191. See 7.18.5 for error handling details.

NOTE 71 - Unlike SSP, there is no acknowledgement of SMP frames with ACK and NAK and there is no credit exchange with RRDY.

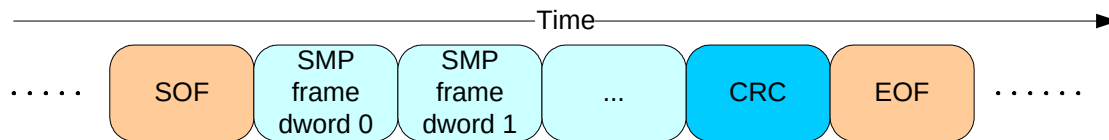


Figure 191 — SMP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5). The SMP link layer state machine checks that the frame is not too short and that the CRC is valid (see 7.18.5).

7.18.2 SMP flow control

By accepting an SMP connection, the SMP target phy indicates it is ready to receive one SMP_REQUEST frame.

After the SMP initiator phy transmits one SMP_REQUEST frame, it shall be ready to receive one SMP_RESPONSE frame.

7.18.3 Opening an SMP connection

An SMP target port shall not attempt to establish an SMP connection.

A SAS phy may reject an incoming connection request (i.e., OPEN address frame) to an SMP target port with OPEN_REJECT (RETRY) for any reason, including because the SAS port containing that SAS phy needs an outgoing connection request to be accepted (e.g., to transmit a frame and empty a buffer).

7.18.4 Closing an SMP connection

After receiving the SMP_RESPONSE frame, the SMP initiator phy shall transmit a CLOSE (NORMAL) to close the connection.

After transmitting the SMP_RESPONSE frame, the SMP target phy shall reply with a CLOSE (NORMAL).

See 7.12.7 for additional details on closing connections.

7.18.5 SMP (link layer for SMP phys) state machines

7.18.5.1 SMP state machines overview

The SMP state machines control the flow of dwords on the physical link during an SMP connection. The SMP state machines are as follows:

- a) SMP_IP (link layer for SMP initiator phys) state machine (see 7.18.5.3); and
- b) SMP_TP (link layer for SMP target phys) state machine (see 7.18.5.4).

7.18.5.2 SMP transmitter and receiver

The SMP transmitter receives the following messages from the SMP state machines specifying dwords and frames to transmit:

- a) Transmit Idle Dword; and
- b) Transmit Frame with an argument containing the frame contents.

In response to the Transmit Frame message, the SMP transmitter transmits:

- 1) SOF;
- 2) the frame contents;
- 3) CRC; and
- 4) EOF.

The SMP transmitter sends the following messages to the SMP state machines based on dwords that have been transmitted:

- a) Frame Transmitted.

When there is no outstanding message specifying a dword to transmit, the SMP transmitter shall transmit idle dwords.

The SMP receiver sends the following messages to the SMP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

- a) SOF Received;
- b) Data Dword Received;
- c) EOF Received;
- d) ERROR Received; and
- e) Invalid Dword Received.

The SMP receiver shall ignore all other dwords.

The SMP transmitter relationship to other transmitters is defined in 4.3.2. The SMP receiver relationship to other receivers is defined in 4.3.3.

7.18.5.3 SMP_IP (link layer for SMP initiator phys) state machine

7.18.5.3.1 SMP_IP state machine overview

The SMP_IP state machine's function is to transmit an SMP request frame and then receive the corresponding response frame. This state machine consists of the following states:

- a) SMP_IP1:Idle (see 7.18.5.3.2)(initial state);
- b) SMP_IP2:Transmit_Frame (see 7.18.5.3.3); and
- c) SMP_IP3:Receive_Frame (see 7.18.5.3.4).

This state machine shall start in the SMP_IP1:Idle state on receipt of an Enable Disable SMP (Enable) message from the SL state machines (see 7.14).

The SMP_IP state machine shall terminate after receiving an Enable Disable SMP (Disable) message from the SL state machines.

Figure 192 shows the SMP_IP state machine.

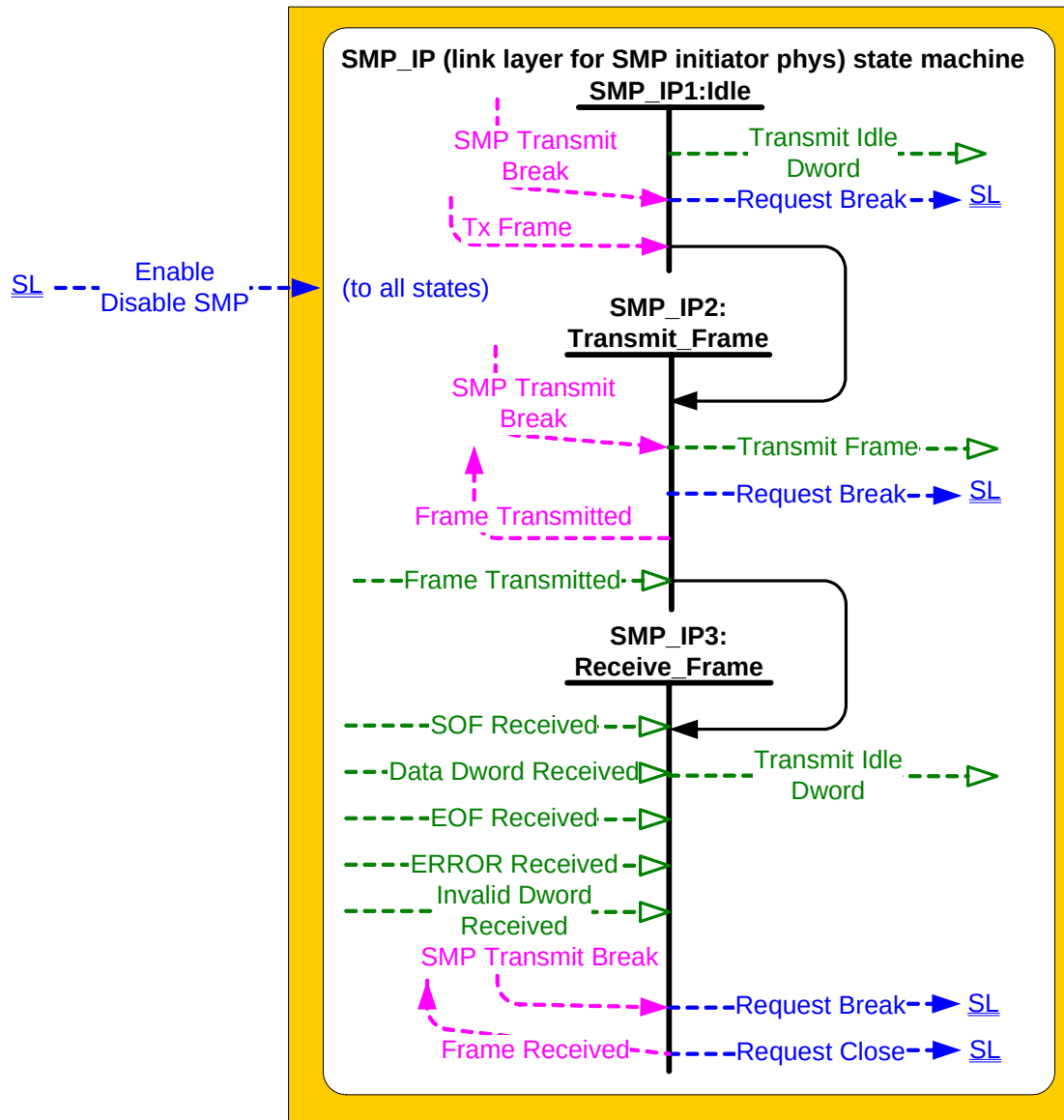


Figure 192 — SMP_IP (link layer for SMP initiator phys) state machine

7.18.5.3.2 SMP_IP1:Idle state

7.18.5.3.2.1 State description

This state is the initial state.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines (see 7.14).

7.18.5.3.2.2 Transition SMP_IP1:Idle to SMP_IP2:Transmit_Frame

This transition shall occur after a Tx Frame request is received.

7.18.5.3.3 SMP_IP2:Transmit_Frame state**7.18.5.3.3.1 State description**

This state shall send a Transmit Frame message to the SMP transmitter with an argument containing the frame contents.

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines (see 7.14) and terminate.

After the Frame Transmitted message is received, this state shall send a Frame Transmitted confirmation to the port layer.

7.18.5.3.3.2 Transition SMP_IP2:Transmit_Frame to SMP_IP3:Receive_Frame

This transition shall occur after sending a Frame Transmitted confirmation to the port layer.

7.18.5.3.4 SMP_IP3:Receive_Frame state

This state checks the SMP response frame and determines if the SMP response frame was successfully received (e.g., no CRC error).

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine if:

- a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
- b) this state receives fewer than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine.

If the SMP response frame is received with a CRC error, this state shall discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine.

If the SMP response frame is received with no CRC error and the SMP response frame is valid, this state shall:

- a) send a Frame Received confirmation to the port layer; and
- b) send a Request Close message to the SL state machines (see 7.14).

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines and this state machine shall terminate.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

7.18.5.4 SMP_TP (link layer for SMP target phys) state machine

7.18.5.4.1 SMP_TP state machine overview

The SMP_TP state machine's function is to receive an SMP request frame and then transmit the corresponding SMP response frame. The SMP_TP state machine consists of the following states:

- SMP_TP1:Receive_Frame (see 7.18.5.4.2)(initial state); and
- SMP_TP2:Transmit_Frame (see 7.18.5.4.3).

This state machine shall start in the SMP_TP1:Receive_Frame state after receiving an Enable Disable SMP (Enable) message from the SL state machines (see 7.14).

The SMP_TP state machine shall terminate after receiving an Enable Disable SMP (Disable) message from the SL state machines.

Figure 193 shows the SMP_TP state machine.

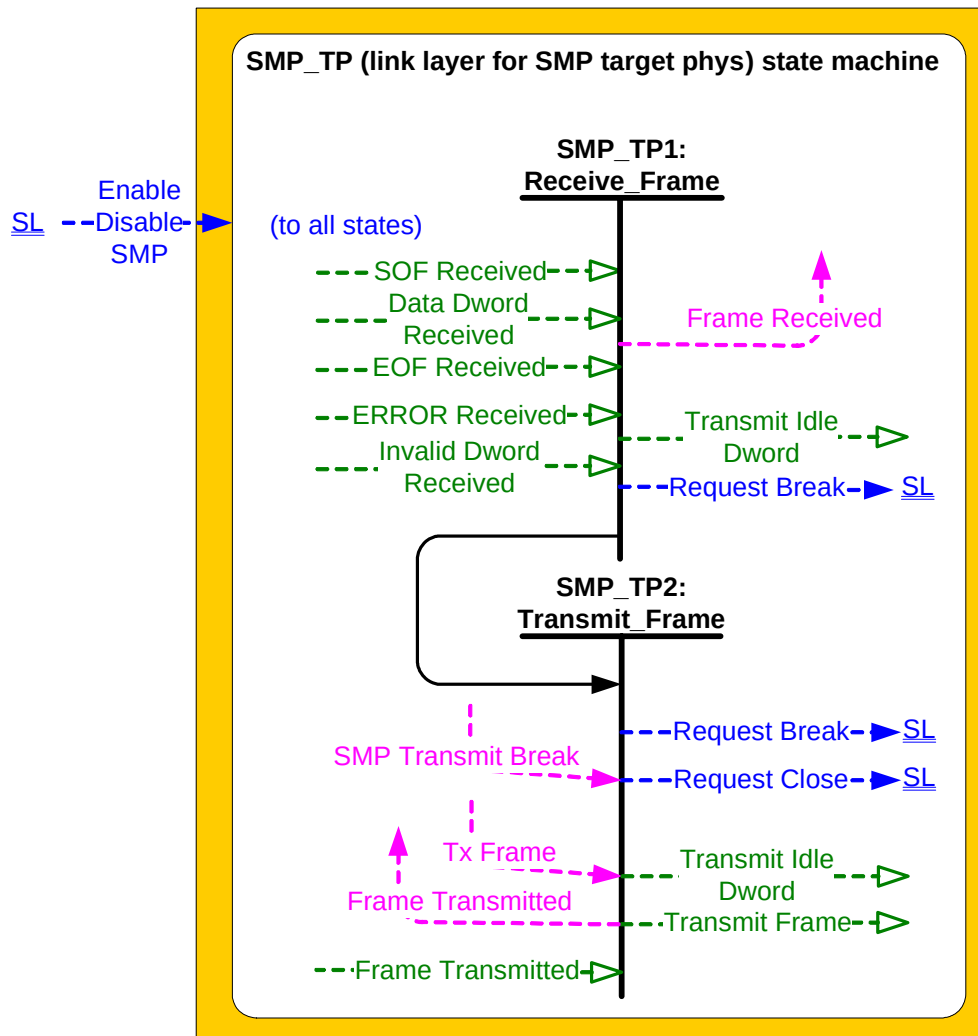


Figure 193 — SMP_TP (link layer for SMP target phys) state machine

7.18.5.4.2 SMP_TP1:Receive_Frame state

7.18.5.4.2.1 State description

This state waits for an SMP frame and determines if the SMP frame was successfully received (e.g., no CRC error).

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine if:

- a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
- b) this state receives fewer than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

- a) ignore the invalid dword or ERROR; or
- b) discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

If the SMP request frame is received with a CRC error, this state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

Otherwise, this state shall send a Frame Received confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

7.18.5.4.2.2 Transition SMP_TP1:Receive_Frame to SMP_TP2:Transmit_Frame

This transition shall occur after sending a Frame Received confirmation to the port layer.

7.18.5.4.3 SMP_TP2:Transmit_Frame state

If this state receives an SMP Transmit Break request, this state shall send a Request Break message to the SL state machines and terminate.

If this state receives a Tx Frame request, this state shall send a Transmit Frame message to the SMP transmitter with an argument containing the frame contents, then wait for a Frame Transmitted message. After receiving a Frame Transmitted message, this state shall send a Frame Transmitted confirmation to the port layer, send a Request Close message to the SL state machines (see 7.14) and terminate.

After sending Transmit Frame message to the SMP transmitter, this state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

8 Port layer

8.1 Port layer overview

The port layer (PL) state machines interface with one or more SAS link layer state machines and one or more SSP, SMP, and STP transport layer state machines to establish port connections and disconnections. The port layer state machines also interpret or pass transmit data, receive data, commands, and confirmations between the link and transport layers.

8.2 PL (port layer) state machines

8.2.1 PL state machines overview

The PL (port layer) consists of state machines that run in parallel and perform the following functions:

- a) receive requests from the SSP, SMP, and STP transport layer state machines for connection management (e.g., requests to open or close connections) and frame transmission;
- b) send requests to the SAS link layer state machines for connection management and frame transmission;
- c) receive confirmation from the SAS link layer state machines; and
- d) send confirmations to the SSP, SMP, and STP transport layer state machines.

The port layer state machines are as follows:

- a) PL_OC (port layer overall control) state machines (see 8.2.2); and
- b) PL_PM (port layer phy manager) state machines (see 8.2.3).

There is one PL_OC state machine per port (see 4.1.4). There is one PL_PM state machine for each phy contained in the port. Phys are assigned to ports by the management application layer. More than one port in a SAS device may have the same SAS address.

Figure 194 shows examples of the port layer state machines and their interaction with the transport and link layers.

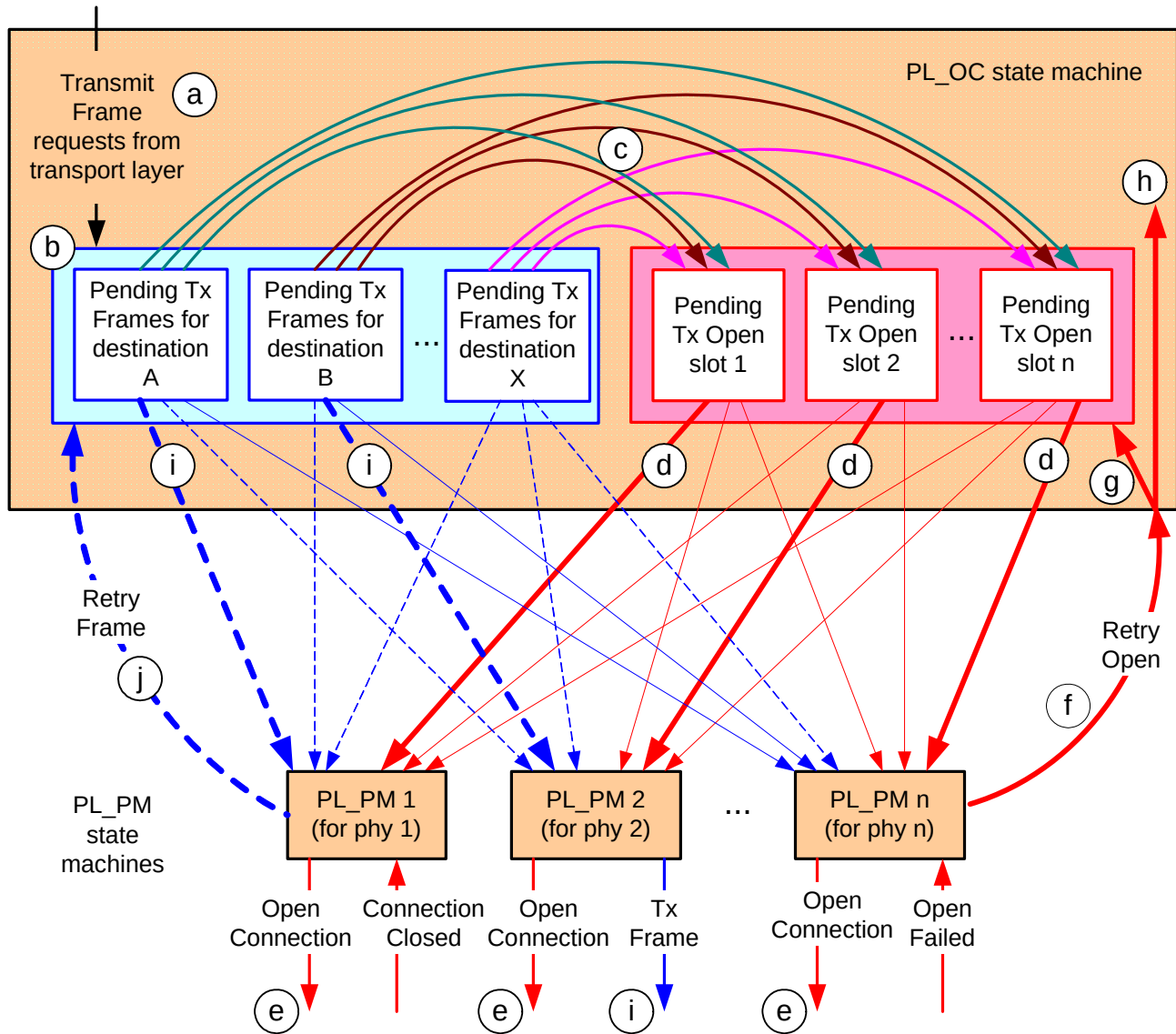


Figure 194 — Port layer examples

The following is a description of the example processes in figure 194. These example processes do not describe all of the possible condition or actions.

- Transmit Frame requests are received by the PL_OC state machine;
- the PL_OC state machine converts Transmit Frame requests into pending Tx Frame messages associated with the destination SAS address;
- the PL_OC state machine generates a pending Tx Open message for a pending Tx Frame message when there is a pending Tx Open slot available (i.e., the number of pending Tx Open messages is less than or equal to the number of phys);
- the PL_OC state machine sends a pending Tx Open message as a Tx Open message to a PL_PM state machine when a PL_PM machine is available; a slot is then available for a new pending Tx Open message;
- when a PL_PM state machine receives a Tx Open message, the PL_PM state machine attempts to establish a connection with the destination SAS address through the link layer;
- if a PL_PM state machine is unable to establish a connection with the destination SAS address, then the PL_PM state machine sends a Retry Open message to the PL_OC state machine;

- g) the PL_OC state machine converts a Retry Open message to a pending Tx Open message if there is a pending Tx Open slot available; if the PL_OC state machine converts a Retry Open message into a pending Tx Open message, then the pathway blocked count and arbitration wait time context from the Retry Open message are applied to the pending Tx Open message;

Editor's Note 42: 07-391 will reword g) to be structured like "if there is a pending Tx slot available, then the PL_OC state machine converts... with the PBC and AWT set to ..."

- h) if the PL_OC state machine does not convert a Retry Open to a pending Tx Open frame, then the PL_OC discards the Retry Open message. The PL_OC state machine may create a new Tx Open message for the same pending Tx Frame at a later time. If the PL_OC state machine discards a Retry Open message, then the pathway blocked count and arbitration wait time context from the Retry Open message are also discarded;
- i) after a PL_PM state machine establishes a connection with a destination SAS address, the PL_OC state machine sends pending Tx Frame messages for the destination to the PL_PM state machine as Tx Frame messages;
- j) if a PL_PM state machine is unable to send a Tx Frame message to the link layer as a Tx Frame request (e.g., due to a credit timeout), then the PL_PM state machine sends a Retry Frame message to the PL_OC state machine, and the PL_OC state machine converts the Retry Frame message into a pending Tx Frame message; and
- k) if the PL_PM state machine is able to send a Tx Frame message as a Tx Frame request to the link layer, then the PL_PM state machine sends a Transmission Status confirmation to the transport layer.

The Transmission Status confirmation from either the PL_OC state machine or a PL_PM state machine shall include the following as arguments:

- a) tag;
- b) destination SAS address; and
- c) source SAS address.

8.2.2 PL_OC (port layer overall control) state machine

8.2.2.1 PL_OC state machine overview

A PL_OC state machine:

- a) receives requests from the SSP, SMP, and STP transport layers;
- b) sends messages to the PL_PM state machine;
- c) receives messages from the PL_PM state machine;
- d) selects frames to transmit;
- e) selects phys on which to transmit frames;
- f) receives confirmations from the link layer;
- g) sends confirmations to the transport layer;
- h) has Arbitration Wait Time timers; and
- i) has I_T Nexus Loss timers.

This state machine consists of the following states:

- a) PL_OC1:Idle (see 8.2.2.2) (initial state); and
- b) PL_OC2:Overall_Control (see 8.2.2.3).

After power on this state machine shall start in the PL_OC1:Idle state.

The PL_OC state machine shall maintain:

- a) a pool of pending Tx Frame messages for each destination SAS address; and
- b) as many pending Tx Open message slots as there are phys in the port.

The PL_OC state machine shall maintain the timers listed in table 151.

Table 151 — PL_OC state machine timers

Timer	Maximum number of timers	Initial value
I_T Nexus Loss timer	One per destination SAS address	Depending on the protocol used by the port: a) for SSP target ports, the value in the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page (see 10.2.7.4); b) for SSP initiator ports, the value in the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page for the SSP target port with that destination SAS address (see 10.2.7.4); c) for STP target ports, the value in the STP SMP I_T NEXUS LOSS TIME field in the SMP CONFIGURE GENERAL function (see 10.4.3.17); d) for STP initiator ports, the value in the STP SMP I_T NEXUS LOSS TIME field in the SMP REPORT GENERAL function (see 10.4.3.3) for the STP target port with that destination SAS address; or e) for SMP initiator ports, the value in the STP SMP I_T NEXUS LOSS TIME field in the SMP REPORT GENERAL function (see 10.4.3.3).
Arbitration Wait Time timer	One per pending Tx Open message	0000h, a vendor-specific value less than 8000h (see 7.12.3), or the value received with a Retry Open message.

Figure 195 shows the PL_OC state machine.

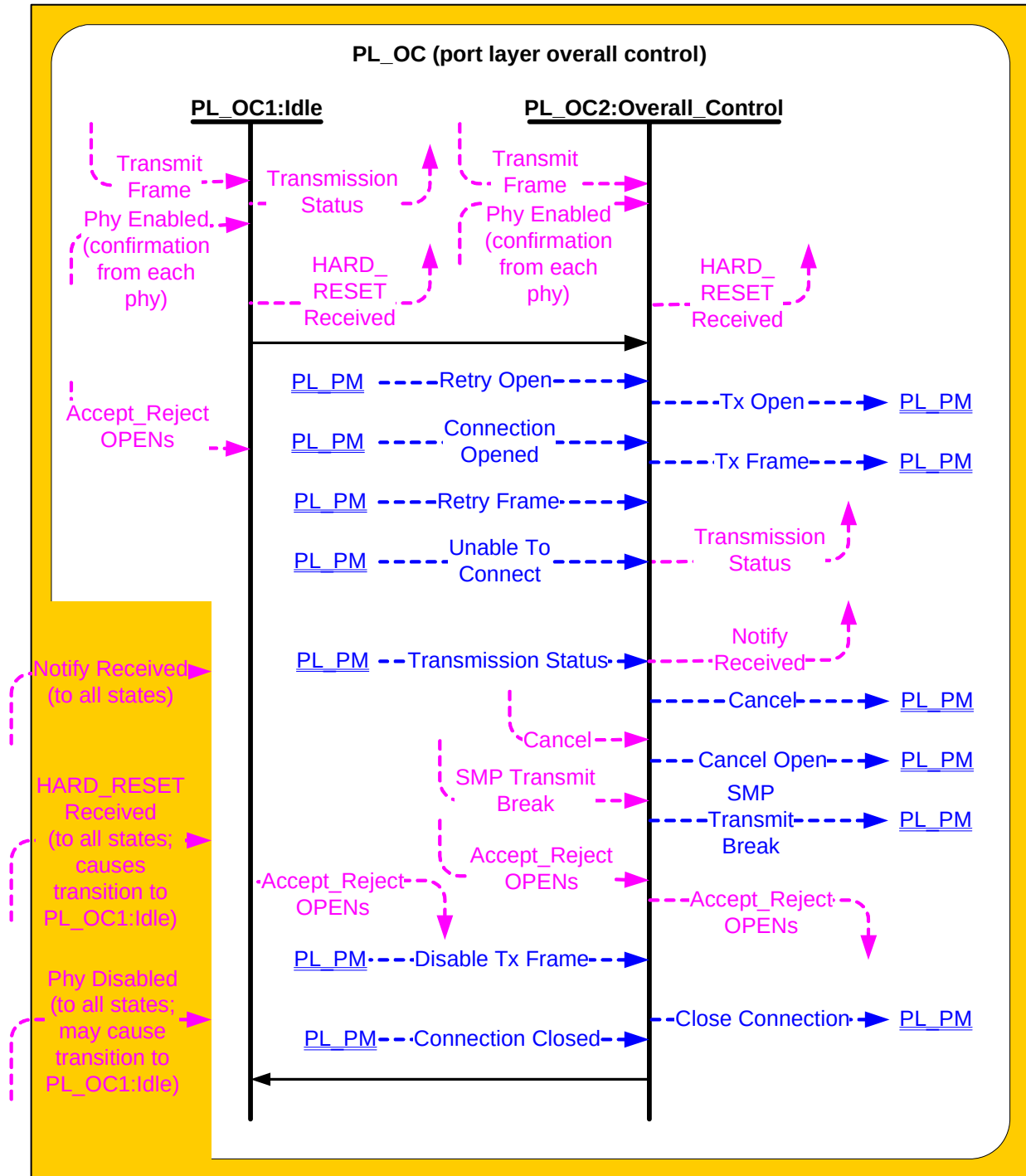


Figure 195 — PL_OC (port layer overall control) state machine

8.2.2.2 PL_OC1:Idle state

8.2.2.2.1 PL_OC1:Idle state description

This state is the initial state of the PL_OC state machine.

If this state receives a HARD_RESET Received confirmation, then this state shall send a HARD_RESET Received confirmation to the transport layer.

Editor's Note 43: PL_OC should be the sole recipient of HARD_RESET Received confirmations from the link layer. PL_PMs should not look at that confirmation from all the phys in the port (as it is implied they are doing, as written). PL_OC should distribute it as a message to all the PL_PMs in the port.

If this state receives a Notify Received (Power Loss Expected) confirmation, then this state shall send a Notify Received (Power Loss Expected) confirmation to the transport layer.

If this state receives an Accept_Reject Opens request, then this state shall send an Accept_Reject Opens request to all link layers in the port.

If this state receives a Transmit Frame request, then this state shall send a Transmission Status (No Phys In Port) confirmation to the transport layer.

If an I_T Nexus Loss timer expires for a destination SAS address, this state shall perform the following:

- a) delete the I_T Nexus Loss timer for the SAS address;
- b) send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message for the SAS address; and
- c) discard each pending Tx Frame message for the SAS address and any corresponding pending Tx Open messages.

If the port is an STP target port or an STP initiator port, the port shall handle all pending commands as described in 4.5.

8.2.2.2 Transition PL_OC1:Idle to PL_OC2:Overall_Control

This transition shall occur after a Phy Enabled confirmation is received for at least one phy assigned to the port.

8.2.2.3 PL_OC2:Overall_Control state

8.2.2.3.1 PL_OC2:Overall_Control state overview

This state may receive Transmit Frame requests from the transport layers (i.e., SSP and SMP) and Retry frame messages from PL_PM state machines. This state shall create a pending Tx Frame message for each received Transmit Frame request and Retry Frame message. There may be more than one pending Tx Frame message at a time for each SSP transport layer. There shall be only one pending Tx Frame message at a time for each SMP transport layer.

This state selects PL_PM state machines through which connections are established. This state shall only attempt to establish connections through PL_PM state machines whose phys are enabled. In a vendor-specific manner, this state selects PL_PM state machines on which connections are established to transmit frames. This state shall receive a response to a message from a PL_PM state machine before sending another message to that PL_PM state machine.

This state also:

- a) receives connection management requests from the transport layers;
- b) sends connection management messages to PL_PM state machines;
- c) receives connection management messages from PL_PM state machines; and
- d) sends connection management confirmations to the transport layers.

After receiving a Transmit Frame request for a destination SAS address for which there is no connection established and for which no I_T Nexus Loss timer has been created, this state shall create an I_T Nexus Loss timer for that SAS address if:

- a) the protocol is SSP, the port is an SSP target port, the Protocol-Specific Port mode page is implemented, and the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page (see 10.2.7.4) is not set to 0000h;

- b) the protocol is STP, the port is an STP target port, and the STP SMP I_T NEXUS LOSS TIME field in the SMP CONFIGURE GENERAL function is not set to 0000h; or
- c) the protocol is SMP, the port is an SMP initiator port, and the STP SMP I_T NEXUS LOSS TIME field in the SMP CONFIGURE GENERAL function is not set to 0000h.

This state may create an I_T Nexus Loss timer for that SAS address if:

- a) the protocol is SSP and the port is an SSP initiator port; or;
- b) the protocol is STP and the port is an STP initiator port.

When this state creates an I_T Nexus Loss timer it shall:

- 1) initialize the I_T Nexus Loss timer as specified in table 151 (see 8.2.2.1); and
- 2) not start the I_T Nexus Loss timer.

If this state machine is in an SSP initiator port, then this state may create an I_T Nexus Loss timer for the SAS address. If a state machine in an SSP initiator port and creates an I_T Nexus Loss timer, then the state machine should use the value in the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page for the SSP target port (see 10.2.7.4) as the initial value for its I_T Nexus Loss timer.

If there are no pending Tx Frame messages for a destination SAS address and an I_T Nexus Loss timer has been created for that destination SAS address, then this state shall delete the I_T Nexus Loss timer for that destination SAS address.

If this state receives a HARD_RESET Received confirmation, then this state shall discard all pending Tx Frame messages and delete all I_T Nexus Loss timers and send a HARD_RESET Received confirmation to the transport layer.

If this state receives a Notify Received (Power Loss Expected) confirmation, then this state shall:

- a) discard all pending Tx Frame messages, if any;
- b) delete all I_T Nexus Loss timers, if any;
- c) send a Close Connection message to all the PL_PM state machines;
- d) send a Cancel Open message to all the PL_PM state machines; and
- e) send a Notify Received (Power Loss Expected) confirmation to the transport layer.

8.2.2.3.2 PL_OC2:Overall_Control state establishing connections

This state receives Phy Enabled confirmations indicating when a phy is available.

This state receives Retry Open messages from a PL_PM state machine.

This state creates pending Tx Open messages based on pending Tx Frame messages and Retry Open messages. Pending Tx Open messages are sent to a PL_PM state machine as Tx Open messages.

If this state receives a Retry Open (Retry) message, then this state shall process the Retry Open message.

If this state receives a Retry Open (No Destination) or a Retry Open (Open Timeout Occurred) message and an I_T Nexus Loss timer has not been created for the destination SAS address (e.g., an SSP target port does not support the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page or the field is set to 0000h), then this state shall process the Retry Open message as either a Retry Open message or an Unable To Connect message. This selection is vendor-specific.

If this state receives a Retry Open (Pathway Blocked) message and an I_T Nexus Loss timer has not been created for the destination SAS address, then this state shall process the Retry Open message.

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message, and an I_T Nexus Loss timer has been created for the destination SAS address with an initial value of FFFFh, then this state shall process the Retry Open message (i.e., the Retry Open message is never processed as an Unable to Connect message).

If this state receives a Retry Open (No Destination) or a Retry Open (Open Timeout Occurred) message, an I_T Nexus Loss timer has been created for the destination SAS address, and there is no connection established with the destination SAS address, then this state shall check the I_T Nexus Loss timer, and:

- a) if the I_T Nexus Loss timer is not running, the I_T nexus loss time is not set to FFFFh, and the CONFIGURING bit is set to zero in the REPORT GENERAL response (see 10.4.3.3) for each expander device between this port and the destination port that is two or more levels away from this port, then this state shall start the timer;
- b) if the I_T Nexus Loss timer is not running and the I_T nexus loss time is not set to FFFFh, then this state shall start the timer;
- c) if the I_T Nexus Loss timer is running, then this state shall not stop the timer; and
- d) if the I_T Nexus Loss timer has expired, then this state shall process the Retry Open message as if it were an Unable To Connect message (see 8.2.2.3.4).

If this state receives a Retry Open (Pathway Blocked) message, an I_T Nexus Loss timer has been created for the destination SAS address, and there is no connection established with the destination SAS address, then this state shall check the I_T Nexus Loss timer, and:

- a) if the I_T Nexus Loss timer is running, then this state shall not stop the timer; and
- b) if the I_T Nexus Loss timer has expired, then this state shall process the Retry Open message as if it were an Unable To Connect message (see 8.2.2.3.4).

If this state receives a Retry Open (Retry) and an I_T Nexus Loss timer is running for the destination SAS address, then this state shall:

- a) stop the I_T Nexus Loss timer (if the timer has been running); and
- b) initialize the I_T Nexus Loss timer.

This state shall create a pending Tx Open message if:

- a) this state has a pending Tx Frame message or has received a Retry Open message;
- b) this state has fewer pending Tx Open messages than the number of PL_PM state machines (i.e., the number of phys in the port);
- c) there is no pending Tx Open message for the destination SAS address; and
- d) there is no connection established with the destination SAS address.

This state may create a pending Tx Open message if:

- a) this state has a pending Tx Frame message, or this state has received a Retry Open message and has not processed the message by sending a confirmation; and
- b) this state has fewer pending Tx Open messages than the number of PL_PM state machines.

This state shall have no more pending Tx Open messages than the number of PL_PM state machines.

If this state receives a Retry Open message and there are pending Tx Frame messages for which pending Tx Open messages have not been created, then this state should create a pending Tx Open message from the Retry Open message.

If this state does not create a pending Tx Open message from a Retry Open message (e.g., the current number of pending Tx Open messages equals the number of phys), then this state shall discard the Retry Open message. This state may create a new pending Tx Open message at a later time for the pending Tx Frame message that resulted in the Retry Open message.

If this state receives a Retry Open (Opened By Destination) message and the initiator port bit and protocol arguments match those in the Tx Open messages that resulted in the Retry Open message, then this state may discard the Retry Open message and use the established connection to send pending Tx Frame messages as Tx Frame messages to the destination SAS address. If this state receives a Retry Open (Opened By Destination) message, then, if this state has a pending Tx Open slot available, this state may create a pending Tx Open message from the Retry Open message.

NOTE 72 - If a connection is established by another port as indicated by a Retry Open (Opened By Destination) message, credit may not be granted for frame transmission. In this case this state may create a pending Tx Open message from a Retry Open message in order to establish a connection where credit is granted.

This state shall send a pending Tx Open message as a Tx Open message to a PL_PM state machine that has an enabled phy and does not have a connection established. If there is more than one pending Tx Open message, this state should send a Tx Open message for the pending Tx Open message that has been pending for the longest time first.

If this state creates a pending Tx Open message from one of the following messages:

- a) Retry Open (Opened By Destination);
- b) Retry Open (Opened By Other);
- c) Retry Open (Collided);
- d) Retry Open (Pathway Blocked), or
- e) if the CONTINUE AWT bit is set to one in the Protocol-specific Port mode page (see 10.2.7.4), Retry Open (Retry),

then this state shall:

- 1) create an Arbitration Wait Time timer for the pending Tx Open message;
- 2) set the Arbitration Wait Time timer for the pending Tx Open message to the arbitration wait time argument from the Retry Open message; and
- 3) start the Arbitration Wait Time timer for the pending Tx Open message.

When a pending Tx Open message is sent to a PL_PM state machine as a Tx Open message, the Tx Open message shall contain the following arguments to be used in an OPEN address frame:

- a) initiator port bit from the Transmit Frame request;
- b) protocol from the Transmit Frame request;
- c) connection rate from the Transmit Frame request;
- d) initiator connection tag from the Transmit Frame request;
- e) destination SAS address from the Transmit Frame request;
- f) source SAS address from the Transmit Frame request;
- g) pathway blocked count; and
- h) arbitration wait time.

If this state creates a pending Tx Open message from one of the following:

- a) a Transmit Frame request;
- b) a Retry Open (No Destination) message;
- c) a Retry Open (Open Timeout Occurred) message; or
- d) if the CONTINUE AWT bit is set to one in the Protocol-specific Port mode page (see 10.2.7.4), a Retry Open (Retry) message,

then this state shall:

- a) set the pathway blocked count argument in the Tx Open message to zero; and
- b) set the arbitration wait time argument in the Tx Open message to zero or a vendor-specific value less than 8000h (see 7.12.3).

If a pending Tx Open message was created as the result this state receiving a Retry Open (Pathway Blocked) message, then this state shall set the pathway blocked count argument in the Tx Open message to the value of the pathway blocked count argument received with the message plus one, unless the pathway blocked count received with the argument is FFh.

If a pending Tx Open message was created as the result of this state receiving one of the following:

- a) a Retry Open (Opened By Destination) message;
- b) a Retry Open (Opened By Other) message;
- c) a Retry Open (Collided) message;
- d) a Retry Open (Pathway Blocked) message; or
- e) if the CONTINUE AWT bit is set to one in the Protocol-specific Port mode page (see 10.2.7.4), a Retry Open (Retry) message,

then this state shall set the arbitration wait time argument in the Tx Open message to be the value from the Arbitration Wait Time timer created as a result of the Retry Open message.

After this state sends a Tx Open message, this state shall discard the pending Tx Open message from which the Tx Open messages was created. After this state discards a pending Tx Open message, this state may create a new pending Tx Open message.

If this state receives a Connection Opened message and the initiator port bit and protocol arguments match those in any pending Tx Frame messages, then this state may use the established connection to send pending Tx Frame messages as Tx Frame messages to the destination SAS address.

Editor's Note 44: 07-391 will mention that after receiving a Connection Opened message, if credit is granted, the state may delete a pending Tx Open if it no longer needs to open its own connection.

8.2.2.3.3 PL_OC2:Overall_Control state connection established

If this state receives a Connection Opened message or a Retry Open (Opened By Destination) message for a SAS address, and an I_T Nexus Loss timer has been created for the SAS address, then this state shall:

- a) stop the I_T Nexus Loss timer for the SAS address, if the timer has been running; and
- b) initialize the I_T Nexus Loss timer.

8.2.2.3.4 PL_OC2:Overall_Control state unable to establish a connection

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message and the I_T Nexus Loss timer for the SAS address has expired, then this state shall perform the following:

- a) delete the I_T Nexus Loss timer for the SAS address;
- b) discard the Retry Open message;
- c) send a Transmission Status (I_T Nexus Loss) confirmation for the pending Tx Frame message from which the Retry Open message resulted;
- d) discard the pending Tx Frame message from which the Retry Open message resulted;
- e) if this state has any pending Tx Frame messages with the same destination SAS address and protocol as the Retry Open message, and this state has not sent a Tx Open message to a PL_PM state machine for the messages, then this state shall send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message and discard the pending Tx Frame messages and any corresponding pending Tx Open messages; and
- f) if this state has any pending Tx Frame messages with the same destination SAS address and protocol as the Retry Open message, and this state has sent a Tx Open message to a PL_PM state machine for a message, then this state shall send a Cancel Open message to each PL_PM state machine to which it has sent a Tx Open message. After receiving an Unable To Connect (Cancel Acknowledge) message from a PL_PM state machine in response to the Cancel Open message, then this state shall send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message and discard the pending Tx Frame messages and any corresponding pending Tx Open messages.

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message and processes it as an Unable To Connect message, or this state receives an

Unable To Connect message, then this state shall send a Transmission Status confirmation as defined in table 152.

Table 152 — Confirmations from Unable To Connect or Retry Open messages

Message received	Confirmation to be sent to transport layer
Retry Open (No Destination)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired, or Transmission Status (No Destination) if it has not
Retry Open (Open Timeout Occurred)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired, or Transmission Status (Open Timeout Occurred) if it has not
Retry Open (Pathway Blocked)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired
Unable To Connect (Break Received)	Transmission Status (Break Received)
Unable To Connect (Port Layer Request)	Transmission Status (Cancel Acknowledge)
Unable to Connect (Bad Destination)	Transmission Status (Bad Destination)
Unable To Connect (Connection Rate Not Supported)	Transmission Status (Connection Rate Not Supported)
Unable To Connect (Protocol Not Supported)	Transmission Status (Protocol Not Supported)
Unable To Connect (Reserved Abandon 1)	Transmission Status (Reserved Abandon 1)
Unable To Connect (Reserved Abandon 2)	Transmission Status (Reserved Abandon 2)
Unable To Connect (Reserved Abandon 3)	Transmission Status (Reserved Abandon 3)
Unable To Connect (STP Resources Busy)	Transmission Status (STP Resources Busy)
Unable To Connect (Wrong Destination)	Transmission Status (Wrong Destination)
Unable To Connect (Zone Violation)	Transmission Status (Zone Violation)

[Editor's Note 45: why no Bad Destination or Wrong Destination below?](#)

If this state receives an Unable To Connect (Connection Rate Not Supported), Unable To Connect (Protocol Not Supported), Unable To Connect (Zone Violation), Unable To Connect (Reserved Abandon 1), Unable To Connect (Reserved Abandon 2), Unable To Connect (Reserved Abandon 3), or Unable To Connect (STP Resources Busy) message and an I_T Nexus Loss timer is running for the SAS address, then this state shall:

- a) stop the I_T Nexus Loss timer, if the timer has been running; and
- b) initialize the I_T Nexus Loss timer.

This state shall discard the pending Tx Frame message for which the Transmission Status confirmation was sent.

8.2.2.3.5 PL_OC2:Overall_Control state connection management

If this state receives an Accept_Reject Opens request, then this state shall send an Accept_Reject Opens request to all phys in the port.

If this state receives an SMP Transmit Break request, then this state shall send an SMP Transmit Break message to the PL_PM state machine associated with the corresponding SMP transport state machine. If there is no PL_PM state machine associated with the request, the PM_OC state shall ignore the request.

If this state receives one of the following:

- a) a Connection Closed (Close Timeout) message;
- b) a Connection Closed (Break Requested) message; or
- c) a Connection Closed (Break Received) message,

then this state shall not send a Tx Open or Tx Frame message to the PL_PM state machine that sent the message until this state receives a Connection Closed (Transition to Idle) message from that PL_PM state machine.

If this state receives a Connection Closed (Normal) message or a Connection Closed (Transition to Idle) message indicating that a connection with a destination SAS address is no longer open and this state has pending Tx Open messages, then this state may send a Tx Open message to the PL_PM state machine that sent the Connection Closed message.

If this port is a wide SSP port, then this state shall not reject an incoming connection request on one phy because it has an outgoing connection request on another phy.

If this port is an SSP port, there are no pending Tx Frame messages for a destination SAS address with which a PL_PM state machine has established a connection, and the connection was established by a message from this state, then this state should send a Close Connection message to the PL_PM state machine.

If this port is an SSP port, has no pending Tx Frame messages for a destination SAS address with which a PL_PM state machine has established a connection, and the connection was established by the destination, then this state may wait a vendor-specific time and then shall send a Close Connection message to the PL_PM state machine.

If this state has received a Disable Tx Frame message from a PL_PM state machine, then this state should send a Close Connection message to the PL_PM state machine.

NOTE 73 - The PL_PM state machine sends a Close Connection request to the link layer upon receipt of a Close Connection message or on expiration of the Bus Inactivity Time Limit timer (see 8.2.3.4.1).

8.2.2.3.6 PL_OC2:Overall_Control state frame transmission

In order to prevent livelocks, If this port is a wide SSP port, has multiple connections established, and has a pending Tx Frame message, then this state shall send at least one Tx Frame message to a PL_PM state machine before sending a Close Connection message to the PL_PM state machine.

After this state receives a Connection Opened message from a PL_PM state machine, this state selects pending Tx Frame messages for the destination SAS address with the same initiator port bit and protocol arguments, and, as an option, the same connection rate argument, and sends the messages to the PL_PM state machine as Tx Frame messages.

This state may send a Tx Frame message to any PL_PM state machine that has established a connection with the destination SAS address when the initiator port bit and protocol arguments match those in the Tx Frame message.

After this state sends a Tx Frame message to a PL_PM state machine, it shall not send another Tx Frame message to that PL_PM state machine until it receives a Transmission Status (Frame Transmitted) message.

This state shall not send a Tx Frame message containing a Request Fence argument or Response Fence argument to any PL_PM state machine until this state has received one of the following messages for each Tx Frame message with the same nexus as specified by that Request Fence argument or Response Fence argument:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout); or
- d) Transmission Status (Connection Lost Without ACK/NAK).

After this state sends a Tx Frame message containing a Request Fence argument or Response Fence argument, it shall not send another Tx Frame message with the same nexus as specified by that Request Fence argument or Response Fence argument until it has received one of the following messages:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout); or
- d) Transmission Status (Connection Lost Without ACK/NAK).

Once this state has sent a Tx Frame message containing a Non-Interlocked argument to a PL_PM state machine, this state shall not send a Tx Frame message containing a Non-Interlocked argument with the same I_T_L_Q nexus to another PL_PM state machine until this state has received one of the following messages for each Tx Frame message containing a Non-Interlocked argument for the same I_T_L_Q nexus:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout); or
- d) Transmission Status (Connection Lost Without ACK/NAK).

Frames with the Non-Interlocked argument for the same I_T_L_Q nexus may be transmitted and received simultaneously on the same or different phys.

If this port is an SMP initiator port, then this state shall send the Tx Frame message containing the SMP REQUEST frame to the PL_PM state machine on which the connection was established for the Tx Open message. If this port is an SMP target port, then this state shall send the Tx Frame message containing the SMP RESPONSE frame to the PL_PM state machine on which the connection was established for the Tx Open message. See 7.18 for additional information about SMP connections.

Characteristics of STP connections are defined by SATA (also see 7.17).

The following arguments shall be included with the Tx Frame message:

- a) the frame to be transmitted; and
- b) Balance Required or Balance Not Required.

A Balance Not Required argument shall only be included if:

- a) the request was a Transmit Frame (Non-Interlocked) request (i.e., the request included a DATA frame); and
- b) the last Tx Frame message sent to this PL_PM state machine while this connection has been established was for a DATA frame having the same logical unit number and tag value as the DATA frame in this Tx Frame message.

If a Balance Not Required argument is not included in the Tx Frame message, then a Balance Required argument shall be included.

If this state receives a Disable Tx Frames message from a PL_PM state machine, then this state should send no more Tx Frame messages to that state machine until a new connection is established.

8.2.2.3.7 PL_OC2:Overall_Control state frame transmission cancellations

Cancel requests cause this state to cancel previous Transmit Frame requests. A Cancel request includes the following arguments:

- a) destination SAS address; and
- b) tag.

If this state receives a Cancel request and has not already sent a Tx Frame message for the Transmit Frame request to a PL_PM state machine for the Transmit Frame request specified by the Cancel request, then this state shall:

- a) discard all Transmit Frame requests for the specified destination SAS address and tag; and
- b) send a Transmission Status (Cancel Acknowledge) confirmation to the transport layer.

If this state receives a Cancel request and has already sent a Tx Frame message to a PL_PM state machine for the Transmit Frame request specified by the Cancel request, then this state shall send a Cancel message to the PL_PM state machine to which the Tx Frame message was sent. The Cancel message shall include the tag.

8.2.2.3.8 Transition PL_OC2:Overall_Control to PL_OC1:Idle

This transition shall occur after:

- a) sending a HARD_RESET Received confirmation to the transport layer;
- b) a Phy Disabled confirmation is received from all of the link layers in the port; or
- c) sending a Notify Received (Power Loss Expected) confirmation to the transport layer.

8.2.3 PL_PM (port layer phy manager) state machine

8.2.3.1 PL_PM state machine overview

A PL_PM state machine:

- a) receives messages from the PL_OC state machine;
- b) sends requests to the link layer;
- c) receives confirmations from the link layer;
- d) sends confirmations to the transport layer;
- e) sends messages to PL_OC state machine;
- f) has an Arbitration Wait Time timer;
- g) may have a Bus Inactivity Time Limit timer; and
- h) may have Maximum Connect Time Limit timer.

This state machine consist of the following states:

- a) PL_PM1:Idle (see 8.2.3.2) (initial state);
- b) PL_PM2:Req_Wait (see 8.2.3.3);
- c) PL_PM3:Connected (see 8.2.3.4); and
- d) PL_PM4:Wait_For_Close (see 8.2.3.5).

After power on this state machine shall start in the PL_PM1:Idle state.

The PL_PM state machine shall maintain the timers listed in table 153.

Table 153 — PL_PM state machine timers

Timer	Initial value
Arbitration Wait Time timer	The arbitration wait time argument from a Retry Open message (see 8.2.2.3.1).
Bus Inactivity Time Limit timer	Depending on the protocol used by the port: a) for SSP target ports, the value in the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.2); or b) for STP target ports, the value in the STP BUS INACTIVITY TIME LIMIT field in the SMP REPORT GENERAL response (see 10.4.3.3).
Maximum Connect Time Limit timer	Depending on the protocol used by the port: a) for an SSP target port, the value in the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.2); or b) for an STP target port, the value in the STP MAXIMUM CONNECT TIME LIMIT field in the SMP REPORT GENERAL response.

Figure 196 shows part 1 of the PL_PM state machine.

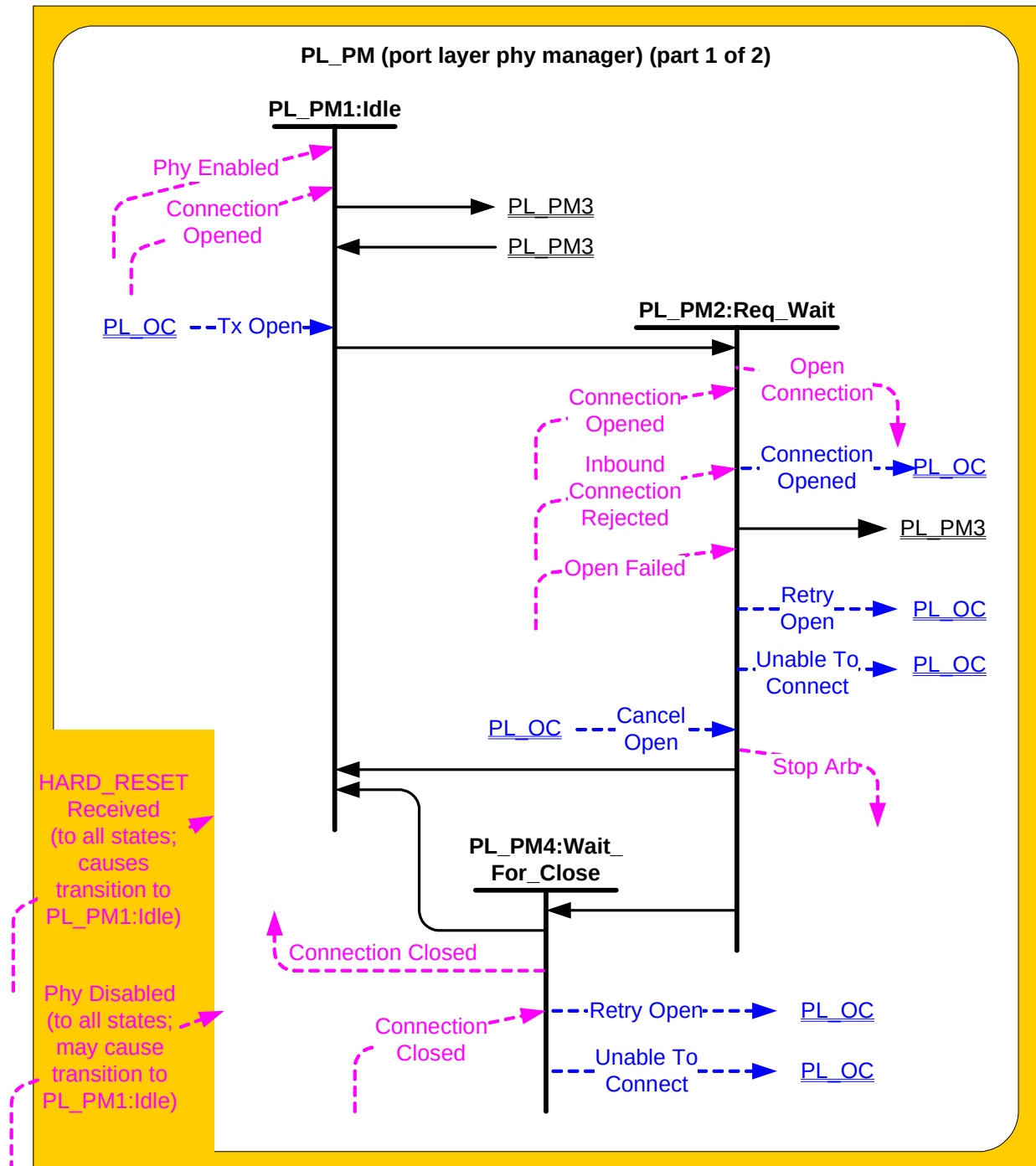


Figure 196 — PL_PM (port layer phy manager) state machine (part 1)

8.2.3.2.2 Transition PL_PM1:Idle to PL_PM2:Req_Wait

This transition shall occur after:

- a) a Phy Enabled confirmation is received; and
- b) a Tx Open message is received.

8.2.3.2.3 Transition PL_PM1:Idle to PL_PM3:Connected

This transition shall occur after a Connection Opened confirmation is received. The transition shall include the received OPEN address frame.

8.2.3.3 PL_PM2:Req_Wait state

8.2.3.3.1 PL_PM2:Req_Wait state overview

This state sends an Open Connection request to the link layer and waits for a confirmation. This state sends and receives connection management messages to and from the PL_OC state machine.

If this state receives a HARD_RESET Received confirmation, then this state shall terminate all operations.

8.2.3.3.2 PL_PM2:Req_Wait establishing a connection

Upon entry into this state, this state shall:

- 1) create an Arbitration Wait Time timer;
- 2) initialize the Arbitration Wait Time timer to the arbitration wait time argument received with the Tx Open message;
- 3) start the Arbitration Wait Time timer; and
- 4) send an Open Connection request to the link layer.

The Open Connection request shall contain the following arguments from the Tx Open message to be used in an OPEN address frame:

- a) initiator port bit;
- b) protocol;
- c) connection rate;
- d) initiator connection tag;
- e) destination SAS address;
- f) source SAS address;
- g) pathway blocked count; and
- h) arbitration wait time.

8.2.3.3.3 PL_PM2:Req_Wait connection established

If this state receives a Connection Opened confirmation, then this state shall send a Connection Opened message to the PL_OC state machine.

If this state receives a Connection Opened confirmation and the confirmation was not in response to an Open Connection request from this state (i.e., the connection was established in response to an OPEN address frame from another device), then this state shall discard any Open Connection request and send a Retry Open message to the PL_OC state machine. If the Connection Opened confirmation was from the destination of the Open Connection request, then this state shall send a Retry Open (Opened By Destination) message. If the Connection Opened confirmation was from a destination other than the destination of the Open Connection request, then this state shall send a Retry Open (Opened By Other) message.

A Retry Open (Opened By Destination) or Retry Open (Opened By Other) message shall contain the following arguments:

- a) initiator port bit set to the value received with the Tx Open message;
- b) protocol set to the value received with the Tx Open message;
- c) connection rate set to the value received with the Tx Open message;
- d) initiator connection tag set to the value received with the Tx Open message;

- e) destination SAS address set to the value received with the Tx Open message;
- f) source SAS address set to the value received with the Tx Open message;
- g) pathway blocked count set to the value received with the Tx Open message; and
- h) arbitration wait time set to the value of the Arbitration Wait Time timer.

8.2.3.3.4 PL_PM2:Req_Wait unable to establish a connection

If this state receives one of the Open Failed confirmations listed in table 154, then this state shall send either a Retry Open message or an Unable To Connect message to the PL_OC state machine.

Table 154 defines the message to be sent for each Open Failed confirmation.

Table 154 — Messages from Open Failed confirmations

Confirmation received	Message to be sent to PL_OC
Open Failed (No Destination)	Retry Open (No Destination)
Open Failed (Pathway Blocked)	Retry Open (Pathway Blocked)
Open Failed (Reserved Continue 0)	Retry Open (Retry)
Open Failed (Reserved Continue 1)	Retry Open (Retry)
Open Failed (Reserved Initialize 0)	Retry Open (No Destination)
Open Failed (Reserved Initialize 1)	Retry Open (No Destination)
Open Failed (Reserved Stop 0)	Retry Open (Pathway Blocked)
Open Failed (Reserved Stop 1)	Retry Open (Pathway Blocked)
Open Failed (Retry)	Retry Open (Retry)
Open Failed (Bad Destination)	Unable To Connect (Bad Destination)
Open Failed (Connection Rate Not Supported)	Unable To Connect (Connection Rate Not Supported)
Open Failed (Protocol Not Supported)	Unable To Connect (Protocol Not Supported)
Open Failed (Reserved Abandon 1)	Unable To Connect (Reserved Abandon 1)
Open Failed (Reserved Abandon 2)	Unable To Connect (Reserved Abandon 2)
Open Failed (Reserved Abandon 3)	Unable To Connect (Reserved Abandon 3)
Open Failed (STP Resources Busy)	Unable To Connect (STP Resources Busy)
Open Failed (Wrong Destination)	Unable To Connect (Wrong Destination)
Open Failed (Zone Violation)	Unable To Connect (Zone Violation)

If this state receives an Inbound Connection Rejected confirmation after sending an Open Connection request, then this state shall discard the Open Connection request and send a Retry Open (Collided) message to the PL_OC state machine.

A Retry Open message shall include the following arguments:

- a) initiator port bit set to the value received with the Tx Open message;
- b) protocol set to the value received with the Tx Open message;
- c) connection rate set to the value received with the Tx Open message;
- d) initiator connection tag set to the value received with the Tx Open message;
- e) destination SAS address set to the value received with the Tx Open message;
- f) source SAS address set to the value received with the Tx Open message;

- g) pathway blocked count argument set to the value received with the Tx Open message; and
- h) arbitration wait time set to the value of the Arbitration Wait Time timer.

An Unable To Connect message shall include the following arguments:

- a) initiator connection tag set to the value received with the Tx Open message;
- b) destination SAS address set to the value received with the Tx Open message; and
- c) source SAS address set to the value received with the Tx Open message.

8.2.3.3.5 PL_PM2:Req_Wait connection management

If this state receives a Cancel Open message and a Connection Opened confirmation has not been received, then this state shall send a Stop Arb request to the link layer.

8.2.3.3.6 Transition PL_PM2:Req_Wait to PL_PM1:Idle

This transition shall occur after:

- a) a Retry Open message is sent to the PL_OC state machine;
- b) an Unable To Connect message is sent to the PL_OC state machine;
- c) all operations have been terminated after a HARD_RESET Received confirmation is received; or
- d) a Phy Disabled confirmation is received.

8.2.3.3.7 Transition PL_PM2:Req_Wait to PL_PM3:Connected

This transition shall occur after a Connection Opened confirmation is received.

8.2.3.3.8 Transition PL_PM2:Req_Wait to PL_PM4:Wait_For_Close

This transition shall occur after one of the following confirmations is received:

- a) an Open Failed (Open Timeout Occurred);
- b) an Open Failed (Break Received); or
- c) an Open Failed (Port Layer Request).

8.2.3.4 PL_PM3:Connected state

8.2.3.4.1 PL_PM3:Connected state description

If this state was entered from the PL_PM1:Idle state, then this state shall send a Connection Opened message to the PL_OC state machine that includes as an argument the received OPEN address frame.

If:

- a) the protocol for the connection is SSP, the port is an SSP target port, the Disconnect-Reconnect mode page is implemented, and the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.2) is not set to zero;
- b) the protocol for the connection is SMP and the port is an SMP target port; or
- c) the protocol for the connection is STP, the port is an STP target port, and the STP MAXIMUM CONNECT TIME LIMIT field is not set to zero in the SMP REPORT GENERAL response (see 10.4.3.3),

then, upon entry into this state, this state shall:

- 1) create a Maximum Connect Time Limit timer;
- 2) initialize the Maximum Connect Time Limit timer as specified in table 153 (see 8.2.3.1); and
- 3) start the Maximum Connect Time Limit timer.

If:

- a) the protocol for the connection is SSP, the port is an SSP initiator port, and the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.2) for the destination SSP target port is not set to zero; or

- b) the protocol for the connection is STP, the port is an STP initiator port, and the STP MAXIMUM CONNECT TIME LIMIT field is not set to zero in the SMP REPORT GENERAL response (see 10.4.3.3) for the destination STP target port,

then, upon entry into this state, this state may:

- 1) create a Maximum Connect Time Limit timer;
- 2) initialize the Maximum Connect Time Limit timer as specified in table 153 (see 8.2.3.1); and
- 3) start the Maximum Connect Time Limit timer.

If:

- a) the protocol for the connection is SSP, the port is an SSP target port, and the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.2) is set to a non-zero value; or
- b) the protocol for the connection is STP, the port is an STP initiator port, and the STP BUS INACTIVITY TIME LIMIT field is not set to zero in the SMP REPORT GENERAL response for the destination STP target port,

then, upon entry into this state, this state shall:

- 1) create a Bus Inactivity Time Limit timer;
- 2) initialize the Bus Inactivity Time Limit timer as specified in table 153 (see 8.2.3.1); and
- 3) start the Bus Inactivity Time Limit timer.

If a Bus Inactivity Time Limit timer has been created and:

- a) the connection is SSP or SMP and this state receives a Tx Frame message; or
- b) the connection is STP and the phy is not both transmitting and receiving SATA_SYNC,

then this state shall:

- 1) stop the Bus Inactivity Time Limit timer, if it is running;
- 2) initialize the Bus Inactivity Time Limit timer as specified in table 153 (see 8.2.3.1); and
- 3) start the Bus Inactivity Time Limit timer.

If this state receives a Tx Frame message, this state shall send a Tx Frame request to the link layer. The following arguments from the Tx Frame message shall be included with the Tx Frame request:

- a) the frame to be transmitted; and
- b) if this port is an SSP port, Balance Required or Balance Not Required.

For STP connections, this state connects the STP transport layer to the STP link layer.

If a Bus Inactivity Time Limit timer expires:

- a) if the connection is SSP and there is no Tx Frame request outstanding (i.e., this state is not waiting for an ACK Received or NAK Received confirmation), then this state shall send a Close Connection request to the link layer;
- b) if the connection is SSP and there is a Tx Frame request outstanding (i.e., this state is waiting for an ACK Received or NAK Received confirmation), then this state shall send a Close Connection request to the link layer after receiving an ACK Received or NAK Received confirmation; and
- c) if the connection is STP, then this state shall send a Close Connection request to the link layer.

If a Maximum Connect Time Limit timer expires:

- a) if the connection is SSP and there is no Tx Frame request outstanding (i.e., this state is not waiting for an ACK Received or NAK Received confirmation), then this state shall send a Close Connection request to the link layer;
- b) if the connection is SSP and there is a Tx Frame request outstanding (i.e., this state is waiting for an ACK Received or NAK Received confirmation), then this state shall send a Close Connection request to the link layer after receiving an ACK Received or NAK Received confirmation;
- c) if the connection is SMP, then this state shall send an SMP Transmit Break request to the link layer; and
- d) if the connection is STP, then this state shall send a Close Connection request to the link layer after the phy is both transmitting and receiving SATA_SYNC.

If this state receives a Tx Frame message after sending a Close Connection request but before receiving a Connection Closed confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

If this state receives a Frame Received confirmation, then this state shall send a Frame Received confirmation to the transport layer. The confirmation shall include the arguments received with the confirmation (e.g., the frame).

If this state receives an ACK Transmitted confirmation, then this state shall send an ACK Transmitted confirmation to the transport layer including the tag of the frame that was ACKed.

If this state receives a Frame Transmitted confirmation, then this state shall send:

- a) a Transmission Status (Frame Transmitted) confirmation to the transport layer; and
- b) a Transmission Status (Frame Transmitted) message to the PL_OC state machine.

If this state receives an ACK Received confirmation, then this state shall send:

- a) a Transmission Status (ACK Received) confirmation to the transport layer; and
- b) a Transmission Status (ACK Received) message to the PL_OC state machine.

If this state receives a NAK Received confirmation, then this state shall send:

- a) a Transmission Status (NAK Received) confirmation to the transport layer; and
- b) a Transmission Status (NAK Received) message to the PL_OC state machine.

If this state receives an ACK/NAK Timeout confirmation, then this state shall send:

- a) a Transmission Status (ACK/NAK Timeout) confirmation to the transport layer; and
- b) a Transmission Status (ACK/NAK Timeout) message to the PL_OC state machine.

If this state receives a Cancel message, then this state shall:

- a) discard all Tx Frame requests for the specified tag;
- b) send a Transmission Status (Cancel Acknowledge) confirmation to the transport layer including the destination SAS address and the tag as arguments; and
- c) discard any subsequent confirmations for previous Tx Frame requests sent for the tag.

If this state receives a Close Connection message from the PL_OC state machine, then this state shall send a Close Connection request to the link layer.

If this state receives one of the following:

- a) a Connection Closed (Normal) confirmation;
- b) a Connection Closed (Close Timeout) confirmation;
- c) a Connection Closed (Break Requested) confirmation;
- d) a Connection Closed (Break Received) confirmation; or
- e) a Connection Closed (Transition to Idle) confirmation,

then this state shall send a Connection Closed message to the PL_OC state machine including the argument received with the confirmation.

If this state receives a Connection Closed (Transition to Idle) confirmation after receiving:

- a) a Connection Closed (Break Received) confirmation; or
- b) a Connection Closed (Break Requested) confirmation,

then this state shall send a Transmission Status (Break Received) confirmation to the transport layer.

If this state receives a Connection Closed (Normal) confirmation, a Connection Closed (Transition to Idle) confirmation, or a Phy Disabled confirmation after sending a Transmission Status (Frame Transmitted) confirmation, but before this state receives an ACK Received or NAK Received confirmation, then this state shall send:

- a) a Transmission Status (Connection Lost Without ACK/NAK) confirmation to the transport layer; and
- b) a Transmission Status (Connection Lost Without ACK/NAK) message to the PL_OC state machine.

If this state receives a Connection Closed (Normal) confirmation, a Connection Closed (Transition to Idle) confirmation, or a Phy Disabled confirmation after sending a Tx Frame request but before receiving a Frame Transmitted confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

If this state receives a Connection Closed confirmation during an SMP connection, this state shall send a Connection Closed confirmation to the transport layer.

If this state receives a Credit Timeout confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

A Retry Frame message shall include the following arguments from the Tx Frame message:

- a) initiator port bit;
- b) protocol;
- c) connection rate;
- d) initiator connection tag;
- e) destination SAS address;
- f) source SAS address; and
- g) frame.

After this state receives a DONE Received (Normal) or DONE Received (Credit Blocked) confirmation, if it does not receive a Tx Frame message within 1 ms, then this state shall send a Disable Tx Frames message to the PL_OC state machine.

If this state receives a DONE Received (ACK/NAK Timeout) or DONE Transmitted confirmation, then this state shall send a Disable Tx Frames message to the PL_OC state machine.

If this state receives an SMP Transmit Break message, then this state shall send an SMP Transmit Break request to the link layer.

If this state receives a HARD_RESET Received confirmation, then this state machine shall terminate all operations.

8.2.3.4.2 Transition PL_PM3:Connected to PL_PM1:Idle

This transition shall occur after:

- a) a Connection Closed (Transition to Idle) message is sent to the PL_OC state machine; or
- b) all operations are terminated after a HARD_RESET Received confirmation is received.

8.2.3.5 PL_PM4:Wait_For_Close state

8.2.3.5.1 PL_PM4:Wait_For_Close state description

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered as the result of the PL_PM2:Req_Wait state receiving an Open Failed (Open Timeout Occurred) confirmation, then this state shall send a Retry Open (Open Timeout Occurred) message to the PL_OC state machine. The Retry Open message shall include the following arguments:

- a) initiator port bit set to the value received with the Tx Open message;
- b) protocol set to the value received with the Tx Open message;
- c) connection rate set to the value received with the Tx Open message;
- d) initiator connection tag set to the value received with the Tx Open message;
- e) destination SAS address set to the value received with the Tx Open message;
- f) source SAS address set to the value received with the Tx Open message;
- g) pathway blocked count argument set to the value received with the Tx Open message; and
- h) arbitration wait time set to the value of the Arbitration Wait Time timer.

If this state receives a Connection Closed confirmation and the connection request was for an SMP connection, this state shall send a Connection Closed confirmation to the transport layer.

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered after the PL_PM2:Req_Wait state received an Open Failed (Port Layer Request) confirmation (i.e., as the result of the

PL_PM2:Req_Wait state sending a Stop Arb request), then this state shall send an Unable to Connect (Port Layer Request) message to the PL_OC state machine.

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered as the result of the PL_PM2:Req_Wait state receiving an Open Failed (Break Received) confirmation, then this state shall send an Unable to Connect (Break Received) message to the PL_OC state machine.

The Unable To Connect message shall include the following arguments:

- a) initiator connection tag set to the value received with the Tx Open message;
- b) destination SAS address set to the value received with the Tx Open message; and
- c) source SAS address set to the value received with the Tx Open message.

If this state receives a HARD_RESET Received confirmation, then this state shall terminate all operations.

8.2.3.5.2 Transition PL_PM4:Wait_For_Close to PL_PM1:Idle

This transition shall occur after:

- a) a Retry Open or Unable To Connect message is sent to the PL_OC state machine; or
- b) all operations are terminated after a HARD_RESET Received confirmation is received.

9 Transport layer

9.1 Transport layer overview

The transport layer defines frame formats. Transport layer state machines interface to the application layer and port layer and construct and parses frame contents. For SSP, the transport layer only receives frames from the port layer for which an ACK is going to be transmitted by the link layer.

9.2 SSP transport layer

9.2.1 SSP frame format

Table 155 defines the SSP frame format.

Table 155 — SSP frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	FRAME TYPE							
1	(MSB)	HASHED DESTINATION SAS ADDRESS						(LSB)
3								
4	Reserved							
5	(MSB)	HASHED SOURCE SAS ADDRESS						(LSB)
7								
8	Reserved							
9	Reserved							
10	Reserved			TLR CONTROL		RETRY DATA FRAMES	RETRANSMIT	CHANGING DATA POINTER
11	Reserved						NUMBER OF FILL BYTES	
12	Reserved							
13	Reserved							
15								
16	(MSB)	TAG						(LSB)
17								
18	(MSB)	TARGET PORT TRANSFER TAG						(LSB)
19								
20	(MSB)	DATA OFFSET						(LSB)
23								
24	INFORMATION UNIT							
m	(e.g., see table 157, table 159, table 161, table 162, or table 163)							
	Fill bytes, if needed							
n - 3	(MSB)	CRC						(LSB)
n								

Editor's Note 46: each Reserved field in the frame header and in each information unit should be defined as either Features or Compatible Features, so there is no confusion when adding features in the future about how existing recipients will behave (e.g. guaranteed to check, or guaranteed to

not check fields they don't know about).

Table 156 defines the FRAME TYPE field, which defines the format of the INFORMATION UNIT field.

Table 156 — FRAME TYPE field

Code	Name of frame	Type of information unit	Originator	Information unit size (bytes)	Reference
01h	DATA frame (i.e., write DATA frame or read DATA frame)	Data information unit (i.e., write Data information unit or read Data information unit)	SSP initiator port or SSP target port	1 to 1 024	9.2.2.4
05h	XFER_RDY frame	Transfer Ready information unit	SSP target port	12	9.2.2.3
06h	COMMAND frame	Command information unit	SSP initiator port	28 to 280	9.2.2.1
07h	RESPONSE frame	Response information unit	SSP target port	24 to 1 024	9.2.2.5
16h	TASK frame	Task Management Function information unit	SSP initiator port	28	9.2.2.2
F0h - FFh	Vendor specific				
All others	Reserved				

The HASHED DESTINATION SAS ADDRESS field contains the hashed value of the destination SAS address (see 4.2.3). See 9.2.6.2.2 and 9.2.6.3.2 for transport layer requirements on checking this field.

The HASHED SOURCE SAS ADDRESS field contains the hashed value of the source SAS address (see 4.2.3). See 9.2.6.2.2 and 9.2.6.3.2 for transport layer requirements on checking this field.

If an initiator port supports transport layer retries, it shall set the TLR CONTROL field to 01b in each COMMAND frame it sends unless it has determined that the I_T_L nexus does not support the TLR CONTROL field.

If an initiator port does not support transport layer retries, it shall set the TLR CONTROL field to 10b in each COMMAND frame it sends unless it has determined that the I_T_L nexus does not support the TLR CONTROL field.

An initiator port determines that an I_T_L nexus does not support the TLR CONTROL field if it sends a COMMAND frame with the TLR CONTROL field set to 01b or 10b and receives a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to 02h (i.e., INVALID FRAME). After determining that an I_T_L nexus does not support the TLR CONTROL field, an initiator port shall set the TLR CONTROL field to 00b for subsequent COMMAND frames for that I_T_L nexus.

NOTE 74 - Initiator ports compliant with previous versions of this standard always set the TLR CONTROL field to 00b.

NOTE 75 - The TLR CONTROL SUPPORTED field in the Protocol-Specific Logical Unit Information VPD page (see 10.2.11.3) indicates if the target port supports the TLR CONTROL field set to a non-zero value.

A TLR CONTROL field set to 00b or 11b in a COMMAND frame specifies that the SSP target port shall use the TRANSPORT LAYER RETRIES bit in the Protocol-Specific Logical Unit mode page (see 10.2.7.3) to enable or disable transport layer retries for this command:

- a) if the TRANSPORT LAYER RETRIES bit is set to one, the target port shall set the RETRY DATA FRAMES bit to one in any XFER_RDY frames that it transmits for this command; and
- b) if the TRANSPORT LAYER RETRIES bit is set to zero, the target port shall set the RETRY DATA FRAMES bit to zero in any XFER_RDY frames that it transmits for this command.

A TLR CONTROL field set to 01b in a COMMAND frame specifies that the SSP target port may enable transport layer retries for this command and:

- a) if it enables transport layer retries, the target port shall set the RETRY DATA FRAMES bit to one in any XFER_RDY frames that it transmits for this command; and
- b) if it does not enable transport layer retries, the target port shall set the RETRY DATA FRAMES bit to zero in any XFER_RDY frames that it transmits for this command.

A TLR CONTROL field set to 10b in a COMMAND frame specifies that the SSP target port shall:

- a) disable transport layer retries for this command; and
- b) set the RETRY DATA FRAMES bit to zero in any XFER_RDY frames that it transmits for this command.

The TLR CONTROL field is reserved for frames other than COMMAND frames.

A target port sets the RETRY DATA FRAMES bit in an XFER_RDY frame based on the TLR CONTROL field received in the COMMAND frame for the command and the TRANSPORT LAYER RETRIES bit in the Protocol-Specific Logical Unit mode page (see 10.2.7.3).

A RETRY DATA FRAMES bit set to one in an XFER_RDY frame specifies that the SSP initiator port shall enable transport layer retries for write DATA transfers related to this XFER_RDY.

A RETRY DATA FRAMES bit set to zero in an XFER_RDY frame specifies that the SSP initiator port shall disable transport layer retries for write DATA transfers related to this XFER_RDY.

The RETRY DATA FRAMES bit is reserved for frames other than XFER_RDY frames.

A RETRANSMIT bit set to one specifies that the frame is a retransmission after the SSP port failed in its previous attempt to transmit the frame. The RETRANSMIT bit is set to one for TASK frames, RESPONSE frames, and XFER_RDY frames under the conditions defined in 9.2.4 and shall be set to zero for all other frame types.

A CHANGING DATA POINTER bit set to one this bit specifies that the frame is a retransmission after the SSP port failed in its previous attempt to transmit the frame or a subsequent frame and the DATA OFFSET field of the frame may not be sequentially increased from that of the previous frame. The CHANGING DATA POINTER bit is set to one for DATA frames under the conditions defined in 9.2.4 and shall be set to zero for all other frame types.

The NUMBER OF FILL BYTES field specifies the number of fill bytes between the INFORMATION UNIT field and the CRC field. The NUMBER OF FILL BYTES field shall be set to zero for all frame types except DATA frames as specified in 9.2.2.4 and RESPONSE frames as specified in 9.2.2.5 (i.e., all other frame types are already four-byte aligned).

The TAG field contains a value that allows the SSP initiator port to establish a context for commands and task management functions.

For COMMAND frames and TASK frames, the SSP initiator port shall set the TAG field to a value that is unique for the I_T nexus established by the connection (see 7.12). An SSP initiator port shall not reuse the same tag when transmitting COMMAND frames or TASK frames to different LUNs in the same SSP target port. An SSP initiator port may reuse a tag when transmitting frames to different SSP target ports. An SSP initiator port does not reuse a tag until it receives indication from the SSP target port that the tag is no longer in use (see 9.2.4, 9.2.5, and 10.2.2).

The TAG field in a COMMAND frame contains the task tag defined in SAM-4. The TAG field in a TASK frame does not correspond to a SAM-4 task tag, but corresponds to an SAM-4 association (see 10.2.1). The tag space used in the TAG fields is shared across COMMAND frames and TASK frames (e.g., if a tag is used for a COMMAND frame, it is not also used for a concurrent TASK frame).

For DATA, XFER_RDY, and RESPONSE frames, the SSP target port shall set the TAG field to the tag of the command or task management function to which the frame pertains.

The TARGET PORT TRANSFER TAG field provides an optional method for an SSP target port to establish the write data context when receiving a write DATA frame (i.e., determine the command to which the write data corresponds). Unlike the TAG field, which was assigned by the SSP initiator port, the TARGET PORT TRANSFER TAG field in a write DATA frame contains a value assigned by the SSP target port that was delivered to the SSP initiator port in the XFER_RDY frame requesting the write data.

NOTE 76 - The TARGET PORT TRANSFER TAG field may be useful when the SSP target port has more than one XFER_RDY frame outstanding (i.e., the SSP target port has transmitted an XFER_RDY frame for each of two or more commands and has not yet received all the write data for them).

SSP target ports may set the TARGET PORT TRANSFER TAG field to any value when transmitting any SSP frame. SSP target ports that use this field should set the TARGET PORT TRANSFER TAG field in every XFER_RDY frame to a value that is unique for the L_Q portion of the I_T_L_Q nexus (i.e., that is unique for every XFER_RDY that is outstanding from the SSP target port).

SSP initiator ports shall set the TARGET PORT TRANSFER TAG field as follows:

- a) For each write DATA frame that is sent in response to an XFER_RDY frame, the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to the value that was in the corresponding XFER_RDY frame;
- b) For each write DATA frame that is sent containing first burst data (see 9.2.2.4), the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh; and
- c) For frames other than write DATA frames, the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh.

For DATA frames, the DATA OFFSET field is described in 9.2.2.4. For all other frame types, the DATA OFFSET field shall be ignored.

The INFORMATION UNIT field contains the information unit, the format of which is defined by the FRAME TYPE field (see table 156). The maximum size of the INFORMATION UNIT field is 1 024 bytes, making the maximum size of the frame 1 052 bytes (1 024 bytes of data + 24 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the INFORMATION UNIT field so the CRC field is aligned on a four byte boundary. The number of fill bytes are specified by the NUMBER OF FILL BYTES field. The contents of the fill bytes are vendor specific.

The CRC field contains a CRC value (see 7.5) that is computed over the entire SSP frame prior to the CRC field including the fill bytes (i.e., all data dwords between the SOF and EOF). The CRC field is checked by the link layer (see 7.16), not the transport layer.

9.2.2 Information units

9.2.2.1 COMMAND frame - Command information unit

The COMMAND frame is sent by an SSP initiator port to request that a command be processed by the device server in a logical unit (see 9.2.3.3, 9.2.3.4, 9.2.3.5, and 9.2.3.6).

Table 157 defines the Command information unit used in the COMMAND frame.

Table 157 — COMMAND frame - Command information unit

Byte\Bit	7	6	5	4	3	2	1	0
0	LOGICAL UNIT NUMBER							
7								
8	Reserved							
9	ENABLE FIRST BURST	TASK PRIORITY				TASK ATTRIBUTE		
10	Reserved							
11	ADDITIONAL CDB LENGTH (n dwords)						Reserved	
12	CDB							
27								
28	ADDITIONAL CDB BYTES							
27+n×4								

The LOGICAL UNIT NUMBER field contains the address of the logical unit. The structure of the LOGICAL UNIT NUMBER field shall be as defined in SAM-4. If the addressed logical unit does not exist, the task manager shall follow the rules for selection of incorrect logical units defined in SAM-4.

An ENABLE FIRST BURST bit set to one specifies that the SSP target port shall expect first burst data for the command as defined by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2). An ENABLE FIRST BURST bit set to zero specifies that the SSP target port shall not expect first burst data for the command (i.e., that the FIRST BURST SIZE field in the Disconnect-Reconnect mode page shall be ignored). Application clients shall only set the ENABLE FIRST BURST bit to one if:

- the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is non-zero or changeable; and
- the logical unit and target port comply with this standard (e.g., as reported in the standard INQUIRY data version descriptors (see SPC-4)).

The TASK PRIORITY field specifies the relative scheduling of the task containing this command in relation to other tasks already in the task set, if the tasks have SIMPLE task attributes (see SAM-4).

The TASK ATTRIBUTE field is defined in table 158.

Table 158 — TASK ATTRIBUTE field

Code	Task attribute	Description
000b	SIMPLE	Specifies that the task be managed according to the rules for a simple task attribute (see SAM-4).
001b	HEAD OF QUEUE	Specifies that the task be managed according to the rules for a head of queue task attribute (see SAM-4).
010b	ORDERED	Specifies that the task be managed according to the rules for an ordered task attribute (see SAM-4).
011b	Reserved	
100b	ACA	Specifies that the task be managed according to the rules for an automatic contingent allegiance task attribute (see SAM-4).
101b-111b	Reserved	

The ADDITIONAL CDB LENGTH field contains the length in dwords (four bytes) of the ADDITIONAL CDB field.

The CDB and ADDITIONAL CDB BYTES fields together contain the CDB to be interpreted by the addressed logical unit. Any bytes between the end of the CDB and the end of the two fields shall be ignored (e.g., a six-byte CDB occupies the first six bytes of the CDB field, the remaining ten bytes of the CDB field are ignored, and the ADDITIONAL CDB BYTES field is not present).

The contents of the CDB are defined in the SCSI command standards (e.g., SPC-4).

9.2.2.2 TASK frame - Task Management Function information unit

The TASK frame is sent by an SSP initiator port to request that a task management function be processed by the task manager in a logical unit (see 9.2.3.2).

Table 159 defines the Task Management Function information unit used in the TASK frame.

Table 159 — TASK frame - Task Management Function information unit

Byte\Bit	7	6	5	4	3	2	1	0						
0	LOGICAL UNIT NUMBER													
7														
8	Reserved													
9	Reserved													
10	TASK MANAGEMENT FUNCTION													
11	Reserved													
12	(MSB)	TAG OF TASK TO BE MANAGED												
13								(LSB)						
14	Reserved													
27														

The LOGICAL UNIT NUMBER field contains the address of the logical unit. The structure of the logical unit number field shall be as defined in SAM-4. If the addressed logical unit does not exist, the task manager shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and its RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER.

Table 160 defines the TASK MANAGEMENT FUNCTION field.

Table 160 — TASK MANAGEMENT FUNCTION field

Code	Task management function	Uses LOGICAL UNIT NUMBER field	Uses TAG OF TASK TO BE MANAGED field	Description
01h	ABORT TASK	yes	yes	The task manager shall perform the ABORT TASK task management function with L set to the value of the LOGICAL UNIT NUMBER field and Q set to the value of the TAG OF TASK TO BE MANAGED field (see SAM-4). ^a
02h	ABORT TASK SET	yes	no	The task manager shall perform the ABORT TASK SET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
04h	CLEAR TASK SET	yes	no	The task manager shall perform the CLEAR TASK SET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
08h	LOGICAL UNIT RESET	yes	no	The task manager shall perform the LOGICAL UNIT RESET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
10h	I_T NEXUS RESET	no	no	The task manager shall perform the I_T NEXUS RESET task management function (see SAM-4). ^a
20h	Reserved			
40h	CLEAR ACA	yes	no	The task manager shall perform the CLEAR ACA task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
80h	QUERY TASK	yes	yes	The task manager shall perform the QUERY TASK task management function with L set to the value of the LOGICAL UNIT NUMBER field and Q set to the value of the TAG OF TASK TO BE MANAGED field (see SAM-4). ^a
81h	QUERY TASK SET	yes	no	The task manager shall perform the QUERY TASK SET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
82h	QUERY UNIT ATTENTION	yes	no	The task manager shall perform the QUERY UNIT ATTENTION task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-4). ^a
All others	Reserved			
^a The task manager shall perform the specified task management function with the I and T arguments set to the initiator port and target port involved in the connection used to deliver the TASK frame.				

If the TASK MANAGEMENT FUNCTION field contains a reserved or unsupported value, the task manager shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED.

If the TASK MANAGEMENT FUNCTION field is set to ABORT TASK or QUERY TASK, the TAG OF TASK TO BE MANAGED field specifies the TAG value from the COMMAND frame that contained the task to be aborted or checked. For all other task management functions, the TAG OF TASK TO BE MANAGED field shall be ignored.

9.2.2.3 XFER_RDY frame - Transfer Ready information unit

The XFER_RDY frame is sent by an SSP target port to request write data from the SSP initiator port during a write command or a bidirectional command (see 9.2.3.4 and 9.2.3.6).

Table 161 defines the Transfer Ready information unit used in the XFER_RDY frame.

Table 161 — XFER_RDY frame - Transfer Ready information unit

Byte/Bit	7	6	5	4	3	2	1	0
0	(MSB)							
3	REQUESTED OFFSET							(LSB)
4	(MSB)							
7	WRITE DATA LENGTH							(LSB)
8	Reserved							
11								

The REQUESTED OFFSET field contains the application client buffer offset of the segment of write data in the data-out buffer that the SSP initiator port may transmit to the logical unit using write DATA frames. The requested offset shall be a multiple of four (i.e., each write DATA frame shall begin transferring data on a dword boundary).

The REQUESTED OFFSET field shall be zero for the first XFER_RDY frame of a command unless:

- the ENABLE FIRST BURST field in the COMMAND frame (see 9.2.2.1) was set to one; and
- the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2.5) is not set to zero.

If the ENABLE FIRST BURST field in the COMMAND frame (see 9.2.2.1) was set to one, then in the initial XFER_RDY frame for the command, the SSP target port shall set the REQUESTED OFFSET field to the application client buffer offset of the segment of write data following the first burst data defined by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2.5).

If any additional XFER_RDY frames are required for the command and transport-layer retries are not being used, the REQUESTED OFFSET field shall be set to the sum of the requested offset and write data length of the previous XFER_RDY frame.

The WRITE DATA LENGTH field contains the number of bytes of write data the SSP initiator port may transmit to the logical unit using write DATA frames from the application client data-out buffer starting at the requested offset. The SSP target port shall set the WRITE DATA LENGTH field to a value greater than or equal to 00000001h. If the value in the MAXIMUM BURST SIZE field in the Disconnect-Reconnect mode page is not zero, the SSP target port shall set the WRITE DATA LENGTH field to a value less than or equal to the value in the MAXIMUM BURST SIZE field (see 10.2.7.2.4).

If an SSP target port transmits an XFER_RDY frame containing a WRITE DATA LENGTH field that is not divisible by four, the SSP target port shall not transmit any subsequent XFER_RDY frames for that command (i.e., only the last XFER_RDY for a command may request a non-dword multiple write data length).

9.2.2.4 DATA frame - Data information unit

During a write command or a bidirectional command (see 9.2.3.4 and 9.2.3.6), one or more write DATA frames are sent by an SSP initiator port to deliver write data.

During a read command or a bidirectional command (see 9.2.3.5 and 9.2.3.6), one or more read DATA frames are sent by an SSP target port to deliver read data.

Table 162 defines the Data information unit used in the DATA frame.

Table 162 — DATA frame - Data information unit

Byte\Bit	7	6	5	4	3	2	1	0
0	DATA							
n								

The DATA field contains the read data (i.e., data to the application client's data-in buffer) or write data (i.e., data from the application client's data-out buffer). The size of the DATA field (i.e., the data length) is determined by the NUMBER OF FILL BYTES field in the frame header (see 9.2.1) and the link layer reception of EOF (see 7.16.3).

The maximum size of the Data information unit (i.e., the DATA field) is the maximum size of any information unit in an SSP frame (see 9.2.1). The minimum size of the Data information unit is one byte.

An SSP initiator port shall only transmit a write DATA frame:

- a) in response to an XFER_RDY frame; or
- b) after transmitting a COMMAND frame if the ENABLE FIRST BURST field in the COMMAND frame was set to one (see 9.2.2.1) and the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is not zero (see 10.2.7.2.5).

If the value in the MAXIMUM BURST SIZE field on the Disconnect-Reconnect mode page is not zero, the maximum amount of data that is transferred at one time by an SSP target port per I_T_L_Q nexus is limited by the value in the MAXIMUM BURST SIZE field (see 10.2.7.2.4).

A write DATA frame shall only contain write data for a single XFER_RDY frame.

An SSP initiator port shall set the NUMBER OF FILL BYTES field to zero in the frame header (see 9.2.1) in all write DATA frames that it transmits in response to an XFER_RDY frame except the last write DATA frame for that XFER_RDY frame. An SSP initiator port may set the NUMBER OF FILL BYTES field to a non-zero value in the last DATA frame that it transmits in response to an XFER_RDY.

NOTE 77 - Combined with the restrictions on WRITE DATA LENGTH in the XFER_RDY frame (see 9.2.2.3), this ensures that only the last write DATA frame for a command may have data with a length that is not a multiple of four).

An SSP target port shall set the NUMBER OF FILL BYTES field to zero in the frame header (see 9.2.1) in all read DATA frames for a command except the last read DATA frame for that command. The SSP target port may set the NUMBER OF FILL BYTES field to a non-zero value in the last read DATA frame for a command (i.e., only the last read DATA frame for a command may contain data with a length that is not a multiple of four).

An SSP initiator port shall not transmit a write DATA frame for a given I_T_L_Q nexus after it has sent a TASK frame that terminates that task (e.g., an ABORT TASK).

The DATA OFFSET field in the frame header (see 9.2.1) contains the application client buffer offset as described by SAM-4. For read DATA frames, this is the offset into the application client's data-in buffer; for write DATA frames, this is the offset into the application client's data-out buffer. The data offset shall be a multiple of four (i.e., each DATA frame shall transfer data beginning on a dword boundary).

The DATA OFFSET field shall be set to zero in the initial read DATA frame for a command. If any additional read DATA frames are required for the command and transport-layer retries are not being used, the DATA OFFSET field shall be set to the sum of the data offset and data length of the previous read DATA frame.

The DATA OFFSET field shall be set to zero in the initial write DATA frame for a command. If any additional write DATA frames are required for the command and transport-layer retries are not being used, the DATA OFFSET field shall be set to the sum of the data offset and data length of the previous write DATA frame.

9.2.2.5 RESPONSE frame - Response information unit

9.2.2.5.1 RESPONSE frame - Response information unit overview

The RESPONSE frame is sent by an SSP target port to deliver:

- a) service response, SCSI status (e.g., GOOD or CHECK CONDITION) and sense data, if any, for a command (see 9.2.3.3, 9.2.3.4, 9.2.3.5, and 9.2.3.6);
- b) service response for a task management function (see 9.2.3.2); or
- c) SSP-specific response (e.g., illegal frame format).

Table 163 defines the Response information unit used in the RESPONSE frame.

Table 163 — RESPONSE frame - Response information unit

Byte\Bit	7	6	5	4	3	2	1	0	
0	Reserved								
7									
8	(MSB)	RETRY DELAY TIMER						(LSB)	
9									
10	Reserved					DATAPRES			
11	STATUS								
12	Reserved								
15									
16	(MSB)	SENSE DATA LENGTH (n bytes)						(LSB)	
19									
20	(MSB)	RESPONSE DATA LENGTH (m bytes)						(LSB)	
23									
24	RESPONSE DATA (see table 165 in 9.2.2.5.3)(if any)								
23+m									
24+m	SENSE DATA (if any)								
23+m+n									

The RETRY DELAY TIMER field contains the retry delay timer code (see SAM-4).

Table 164 defines the DATAPRES field, which specifies the format and content of the STATUS field, SENSE DATA LENGTH field, RESPONSE DATA LENGTH field, RESPONSE DATA field, and SENSE DATA field.

Table 164 — DATAPRES field

Code	Name	Description	Reference
00b	NO_DATA	No response data or sense data present	9.2.2.5.2
01b	RESPONSE_DATA	Response data present	9.2.2.5.3
10b	SENSE_DATA	Sense data present	9.2.2.5.4
11b	Reserved		

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to NO_DATA if a command completes without response data or sense data to return.

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA in response to every TASK frame and in response to errors that occur while the transport layer is processing a COMMAND frame (see 9.2.5.3).

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to SENSE_DATA if a command completes with sense data to return (e.g., CHECK CONDITION status).

If the DATAPRES field is set to a reserved value, then the SSP initiator port shall discard the RESPONSE frame.

9.2.2.5.2 Response information unit - NO_DATA format

If the DATAPRES field is set to NO_DATA, then:

- the SSP target port shall set the STATUS field to the status code for a command that has ended (see SAM-4 for a list of status codes);
- the SSP target port shall set the SENSE DATA LENGTH field to zero and the RESPONSE DATA LENGTH field to zero;
- the SSP initiator port shall ignore the SENSE DATA LENGTH field and the RESPONSE DATA LENGTH field; and
- the SSP target port shall not include the SENSE DATA field and the RESPONSE DATA field.

9.2.2.5.3 Response information unit - RESPONSE_DATA format

If the DATAPRES field is set to RESPONSE_DATA, then:

- the SSP target port shall set the STATUS field to zero and the SENSE DATA LENGTH field to zero;
- the SSP initiator port shall ignore the STATUS field and the SENSE DATA LENGTH field;
- the SSP target port shall not include the SENSE DATA field;
- the SSP target port shall set the RESPONSE DATA LENGTH field to 00000004h; and
- the SSP target port shall include the RESPONSE DATA field.

Table 165 defines the RESPONSE DATA field. The RESPONSE DATA field shall be present if the SSP target port detects any of the conditions described by a non-zero value in the RESPONSE CODE field and shall be present for a RESPONSE frame sent in response to a TASK frame.

Table 165 — RESPONSE DATA field

Byte\Bit	7	6	5	4	3	2	1	0
0	ADDITIONAL RESPONSE INFORMATION							
2								
3	RESPONSE CODE							

The ADDITIONAL RESPONSE INFORMATION field contains additional response information for certain task management functions (e.g., QUERY UNIT ATTENTION) as defined in SAM-4. If the task management function does not define additional response information or the logical unit does not support additional response information, the SSP target port shall set the ADDITIONAL RESPONSE INFORMATION field to 000000h.

Table 166 defines the RESPONSE CODE field, which specifies the error condition or the completion status of a task management function. See 10.2.1.5 and 10.2.1.15 for the mapping of these response codes to SCSI service responses.

Table 166 — RESPONSE CODE field

Code	Description
00h	TASK MANAGEMENT FUNCTION COMPLETE ^a
02h	INVALID FRAME
04h	TASK MANAGEMENT FUNCTION NOT SUPPORTED ^a
05h	TASK MANAGEMENT FUNCTION FAILED ^a
08h	TASK MANAGEMENT FUNCTION SUCCEEDED ^a
09h	INCORRECT LOGICAL UNIT NUMBER ^a
0Ah	OVERLAPPED TAG ATTEMPTED ^b
All others	Reserved
^a Only valid when responding to a TASK frame ^b Returned in case of command/task management function or task management function/task management function tag conflicts.	

9.2.2.5.4 Response information unit - SENSE_DATA format

If the DATAPRES field is set to SENSE_DATA, then:

- the SSP target port shall set the STATUS field to the status code for a command that has ended (see SAM-4 for a list of status codes);
- the SSP target port shall set the RESPONSE DATA LENGTH field to zero;
- the SSP initiator port shall ignore the RESPONSE DATA LENGTH field;
- the SSP target port shall not include the RESPONSE DATA field;
- the SSP target port shall set the SENSE DATA LENGTH field to a non-zero value indicating the number of bytes in the SENSE DATA field. The value in the SENSE DATA LENGTH field shall not be greater than 1 000 (see table 156 in 9.2.1); and
- the SSP target port shall set the SENSE DATA field to the sense data (see SAM-4).

The value in the SENSE DATA LENGTH field is not required to be a multiple of four. If it is not, the value in the NUMBER OF FILL BYTES field in the SSP frame header is non-zero and fill bytes are present.

9.2.3 Sequences of SSP frames

9.2.3.1 Sequences of SSP frames overview

Table 167 lists the sequences of SSP frames supporting the SCSI transport protocol services described in 10.2.1.

Table 167 — Sequences of SSP frames

Sequence	Reference
Task management function	9.2.3.2
Non-data command	9.2.3.3
Write command	9.2.3.4
Read command	9.2.3.5
Bidirectional command	9.2.3.6

When multiple commands and/or task management functions are outstanding, frames from each of the individual sequences may be interleaved in any order. RESPONSE frames may be returned in any order (i.e., the order in which TASK frames and COMMAND frames are sent has no effect on the order that RESPONSE frames are returned).

Frames in a sequence may be transmitted during one or more connections (see 7.12)(e.g., for a write command using a single XFER_RDY frame, the COMMAND frame could be transmitted in a connection originated by the SSP initiator port, the XFER_RDY frame in a connection originated by the SSP target port, the DATA frames in one or more connections originated by the SSP initiator port, and the RESPONSE frame in a connection originated by the SSP target port. Or, they could all be transmitted in one connection.).

9.2.3.2 Task management function sequence of SSP frames

Figure 198 shows the sequence of SSP frames for a task management function (e.g., ABORT TASK (see SAM-4)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.

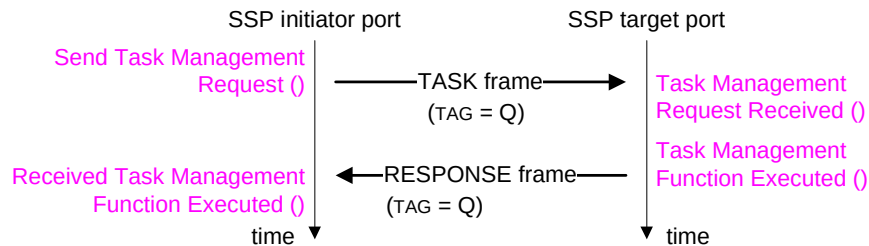


Figure 198 — Task management function sequence of SSP frames

9.2.3.3 Non-data command sequence of SSP frames

Figure 199 shows the sequence of SSP frames for a non-data command (e.g., TEST UNIT READY (see SPC-4)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.

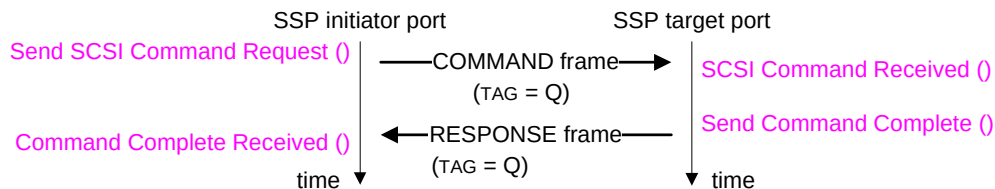


Figure 199 — Non-data command sequence of SSP frames

9.2.3.4 Write command sequence of SSP frames

Figure 200 shows the sequence of SSP frames for a write command (e.g., MODE SELECT (see SPC-4)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.

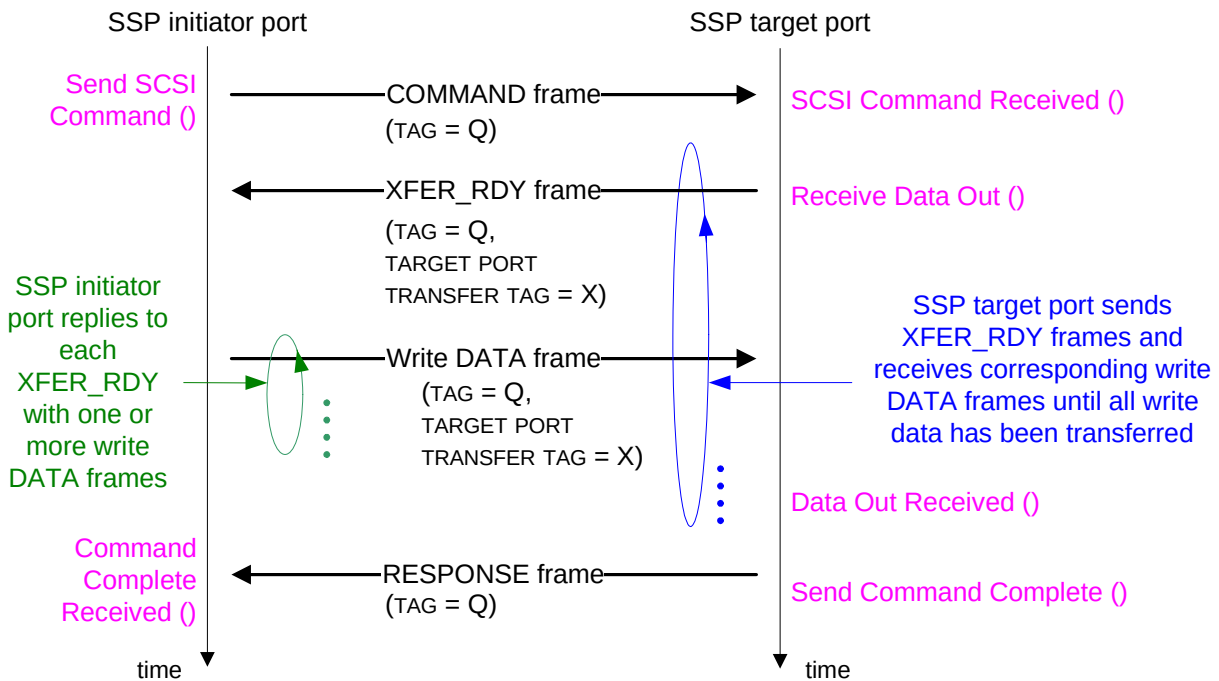


Figure 200 — Write command sequence of SSP frames

9.2.3.5 Read command sequence of SSP frames

Figure 201 shows the sequence of SSP frames for a read command (e.g., INQUIRY, REPORT LUNS, or MODE SENSE (see SPC-4)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.

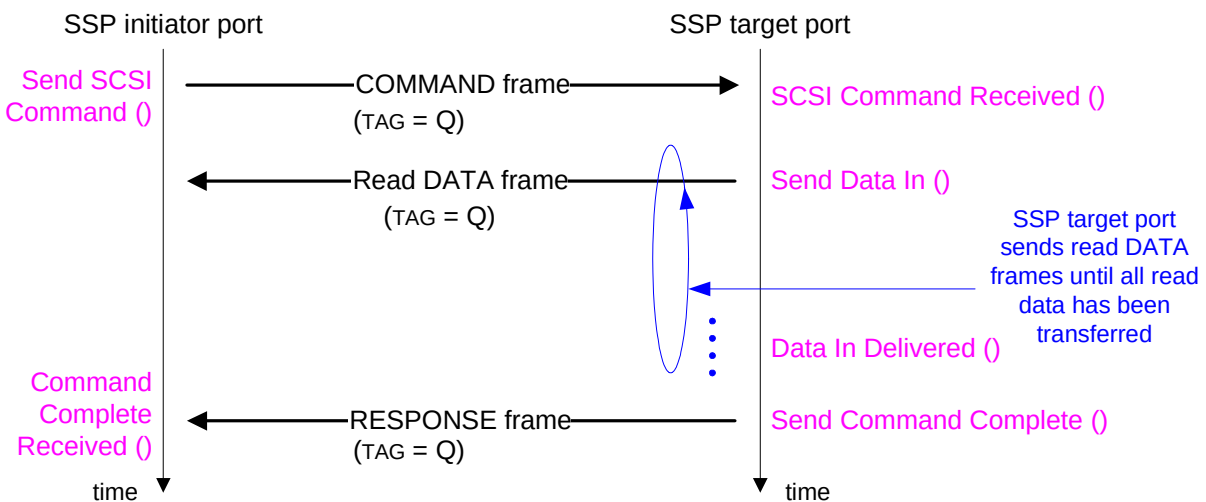


Figure 201 — Read command sequence of SSP frames

9.2.3.6 Bidirectional command sequence of SSP frames

Figure 202 shows the sequence of SSP frames for a bidirectional command (e.g., XDWRITEREAD (see SBC-3)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.

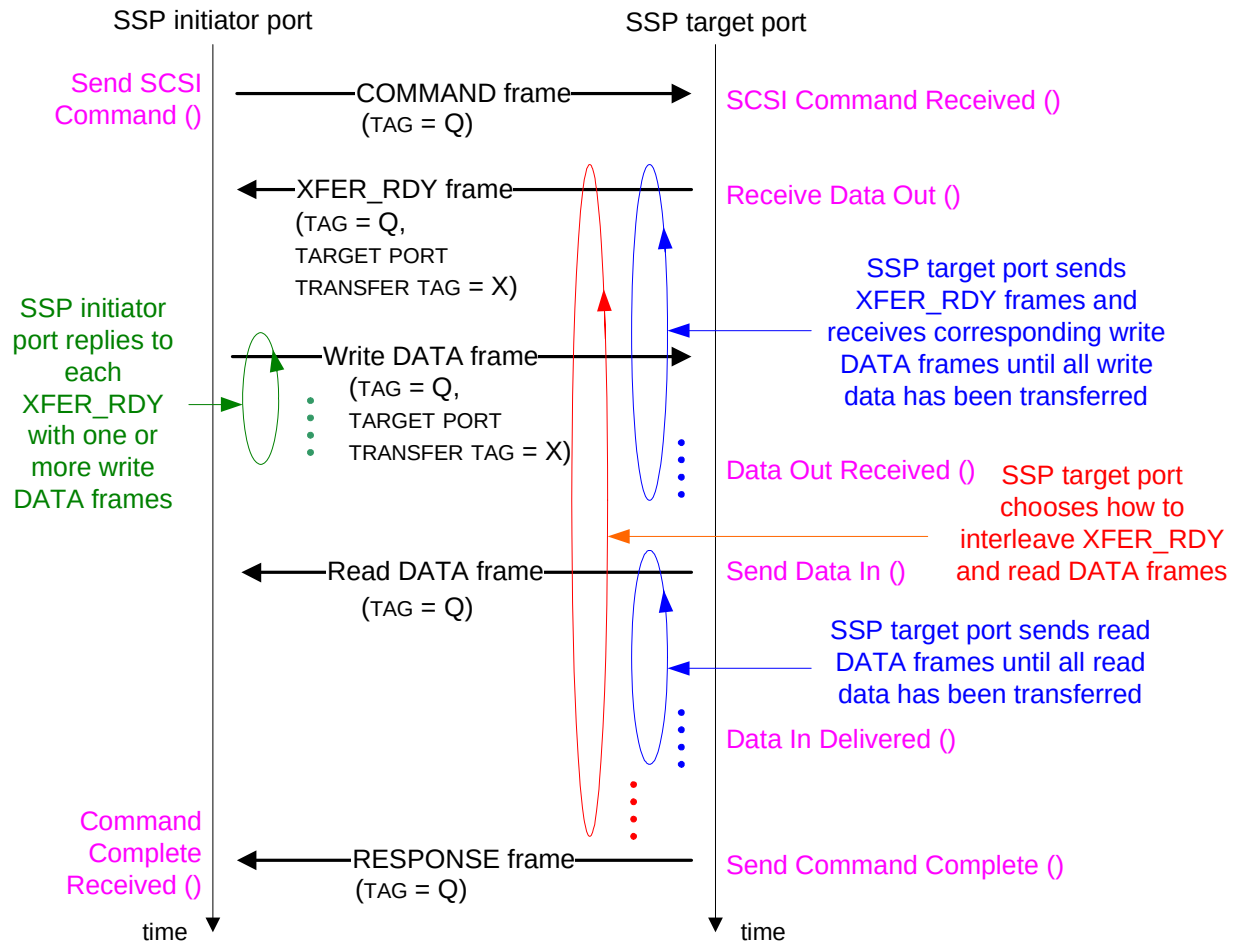


Figure 202 — Bidirectional command sequence of SSP frames

The SSP target port may transmit read DATA frames for a bidirectional command at the same time it is receiving write DATA frames for the same bidirectional command.

9.2.4 SSP transport layer handling of link layer errors

9.2.4.1 SSP transport layer handling of link layer errors overview

The transport layer, sometimes assisted by the application layer, handles some link layer errors (e.g., NAKs and ACK/NAK timeouts). See 9.2.5 for transport layer handling of transport layer errors (e.g., invalid frame contents).

Link layer errors that occur when transmitting XFER_RDY and DATA frames are handled differently based on the TLR CONTROL field in the COMMAND frame header (see 9.2.1) and the TRANSPORT LAYER RETRIES bit in the Protocol-Specific Logical Unit mode page (see 10.2.7.3) of the logical unit that is the source of the frame.

If transport layer retries are disabled, the logical unit:

- sets the RETRY DATA FRAMES bit to zero in each XFER_RDY frame;
- may or may not select a different value for the TARGET PORT TRANSFER TAG field in each XFER_RDY frame than that used in the previous XFER_RDY frame for that I_T_L_Q nexus;
- processes XFER_RDY frame link layer errors as described in 9.2.4.4.3; and
- processes DATA frame link layer errors as described in 9.2.4.5.3.

If transport layer retries are enabled, the logical unit:

- a) supports the QUERY TASK task management function (see SAM-4);
- b) sets the RETRY DATA FRAMES bit to one in each XFER_RDY frame;
- c) selects a different value for the TARGET PORT TRANSFER TAG field in each XFER_RDY frame than that used in the previous XFER_RDY frame for that I_T_L_Q nexus;
- d) processes XFER_RDY frame link layer errors as described in 9.2.4.4.2; and
- e) processes DATA frame link layer errors as described in 9.2.4.5.2.

9.2.4.2 COMMAND frame - handling of link layer errors

If an SSP initiator port transmits a COMMAND frame and receives a NAK for that frame, then the COMMAND frame was not received. The SSP initiator port should retransmit, in the same or in a new connection, the COMMAND frame at least one time (see 9.2.6.2.3.3). The SSP initiator port may reuse the tag.

If an SSP initiator port transmits a COMMAND frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
- 2) to determine whether the command was received, the application client calls Send Task Management Function Request () (see 10.2.2) with:
 - A) Nexus set to the I_T_L_Q nexus of the COMMAND frame; and
 - B) Function Identifier set to QUERY TASK;
 and
- 3) the SSP initiator port transmits the TASK frame in a new connection to the SSP target port.

If the SSP initiator port receives an XFER_RDY frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received and is being processed by the target port, and the XFER_RDY frame is valid.

If the SSP initiator port receives a read DATA frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received and is being processed by the target port, and the read DATA frame is valid.

If the SSP initiator port receives a RESPONSE frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received by the target port, the RESPONSE frame is valid, and the command processing is complete. The SSP initiator port may reuse the tag of the COMMAND frame.

If the SSP initiator port receives a RESPONSE frame for the QUERY TASK with a response code of TASK MANAGEMENT FUNCTION SUCCEEDED, then the COMMAND frame was received by the SSP target port (i.e., ACKed) and the command is being processed.

If the SSP initiator port receives a RESPONSE frame for the QUERY TASK with a response code of TASK MANAGEMENT FUNCTION COMPLETE, then the COMMAND frame is not being processed. If neither an XFER_RDY frame, a read DATA frame, nor a RESPONSE frame has been received for the I_T_L_Q nexus of the command, then the COMMAND frame was not received. The SSP initiator port should retransmit the COMMAND frame at least one time. The SSP initiator port may reuse the tag of the COMMAND frame.

9.2.4.3 TASK frame - handling of link layer errors

If an SSP initiator port transmits a TASK frame and receives a NAK for that frame, then the TASK frame was not received. The SSP initiator port should retransmit, in the same or in a new connection, the TASK frame at least one time with the RETRANSMIT bit set to one (see 9.2.6.2.2.2). The SSP initiator port may reuse the tag.

If an SSP initiator port transmits a TASK frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the application client calls Send Task Management Request () using the same tag (see 10.2.2);

- 3) the SSP initiator port transmits the TASK frame with the RETRANSMIT bit set to one in a new connection to the SSP target port (see 9.2.6.2.2.2).

If the SSP initiator port receives a RESPONSE frame for the TASK frame that arrives before the ACK or NAK for the TASK frame, then the TASK frame was received by the SSP target port (i.e., ACKed), the RESPONSE frame is valid, and the task management function is complete (see 9.2.6.2.2.3). The initiator port may reuse the tag of the TASK frame.

9.2.4.4 XFER_RDY frame - handling of link layer errors

9.2.4.4.1 XFER_RDY frame overview

If transport layer retries are enabled, then the SSP target port processes link layer errors that occur while transmitting XFER_RDY frames as described in 9.2.4.4.2.

If transport layer retries are disabled, then the SSP target port processes link layer errors that occur while transmitting XFER_RDY frames as described in 9.2.4.4.3.

9.2.4.4.2 XFER_RDY frame with transport layer retries enabled

If an SSP target port transmits an XFER_RDY frame and receives a NAK for that frame, the SSP target port retransmits, in the same or a new connection, the XFER_RDY frame with a different value in the TARGET PORT TRANSFER TAG field, with the RETRANSMIT bit set to one, and with the other fields set to the same values as in the original XFER_RDY frame (see 9.2.6.3.3.3).

If an SSP target port transmits an XFER_RDY frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the SSP target port retransmits, in a new connection, the XFER_RDY frame with:
 - A) the TARGET PORT TRANSFER TAG field set to a different value than in the original XFER_RDY frame;
 - B) the RETRANSMIT bit set to one; and
 - C) the other fields set to the same values as in the original XFER_RDY frame (see 9.2.6.3.3.3).

If an SSP initiator port receives a new XFER_RDY frame with the RETRANSMIT bit set to one while processing the previous XFER_RDY frame for that I_T_L_Q nexus, the ST_ITS state machine stops processing the previous XFER_RDY frame (i.e., stops transmitting write DATA frames) and starts servicing the new XFER_RDY frame (see 9.2.6.2.3). The ST_ITS state machine does not transmit any write DATA frames for the previous XFER_RDY frame after transmitting a write DATA frame for the new XFER_RDY frame.

The SSP target port may reuse the value in the TARGET PORT TRANSFER TAG field from the previous XFER_RDY frame after it receives a write DATA frame for the new XFER_RDY frame.

An SSP target port retransmits each XFER_RDY frame that does not receive an ACK at least one time.

9.2.4.4.3 XFER_RDY frame with transport layer retries disabled

If an SSP target port transmits an XFER_RDY frame and receives a NAK for that frame:

- 1) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to NAK RECEIVED (see 10.2.3); and
- 2) the SSP target port transmits the RESPONSE frame in the same or a new connection.

If an SSP target port transmits an XFER_RDY frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
- 2) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to ACK/NAK TIMEOUT (see 10.2.3); and
- 3) the SSP target port transmits the RESPONSE frame in a new connection.

9.2.4.5 DATA frame - handling of link layer errors

9.2.4.5.1 DATA frame overview

If an SSP target port transmits a read DATA frame for a command with transport layer retries enabled, then the SSP target port processes link layer errors that occur while transmitting read DATA frames as described in 9.2.4.5.2.

If an SSP target port transmits a read DATA frame for a command with transport layer retries disabled, then the SSP target port processes link layer errors that occur while transmitting read DATA frames as described in 9.2.4.5.3.

An SSP initiator port processes link layer errors that occur while transmitting write DATA frames transmitted in response to an XFER_RDY frame that has its RETRY DATA FRAMES bit set to one as described in 9.2.4.5.2.

An SSP initiator port processes link layer errors that occur while transmitting write DATA frames in response to an XFER_RDY frame that has its RETRY DATA FRAMES bit set to zero as described in 9.2.4.5.3.

9.2.4.5.2 DATA frame with transport layer retries enabled

If an SSP target port transmits a read DATA frame and receives a NAK for that frame, then the read DATA frame was not received. The SSP target port retransmits, in the same or in a new connection, all the read DATA frames since a previous time when ACK/NAK balance occurred (see 9.2.6.3.3.4).

If an SSP target port transmits a read DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the ST_TTS state machine retransmits, in a new connection, all the read DATA frames since a previous time when ACK/NAK balance occurred (see 9.2.6.3.3.4).

If an SSP initiator port transmits a write DATA frame and receives a NAK for that frame, then the write DATA frame was not received. The SSP_ITS state machine retransmits, in the same or in a new connection, all the write DATA frames for the previous XFER_RDY frame (see 9.2.6.2.3.3.2).

If an SSP initiator port transmits a write DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the ST_ITS state machine retransmits, in a new connection, all the write DATA frames for the previous XFER_RDY frame (see 9.2.6.2.3.3.2).

If that SSP initiator port receives a new XFER_RDY frame or a RESPONSE frame for the command while retransmitting or preparing to retransmit the write DATA frames, the ST_IFR state machine and ST_ITS state machine process the XFER_RDY frame or RESPONSE frame and stop retransmitting the write DATA frames (see 9.2.6.2.2 and 9.2.6.2.3). The ST_ITS state machine does not transmit a write DATA frame for the previous XFER_RDY frame after transmitting a write DATA frame in response to the new XFER_RDY frame.

For both reads and writes, the CHANGING DATA POINTER bit is set to one in the first retransmitted DATA frame and the CHANGING DATA POINTER bit is set to zero in subsequent DATA frames.

The ST_ITS state machine and ST_TTS state machine retransmit each DATA frame that does not receive an ACK at least one time (see 9.2.6.2.3 and 9.2.6.3.3). The number of times they retransmit each DATA frame is vendor-specific.

9.2.4.5.3 DATA frame with transport layer retries disabled

If an SSP target port transmits a read DATA frame and receives a NAK for that frame:

- 1) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to NAK RECEIVED (see 10.2.3); and

- 2) the SSP target port transmits the RESPONSE frame in the same or a new connection.

If an SSP target port transmits a read DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
- 2) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to ACK/NAK TIMEOUT (see 10.2.3); and
- 3) the SSP target port transmits the RESPONSE frame in a new connection.

If an SSP initiator port transmits a write DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the application client aborts the command (see 10.2.2).

If an SSP initiator port transmits a write DATA frame and receives a NAK for that frame, the application client aborts the command (see 10.2.2).

9.2.4.6 RESPONSE frame - handling of link layer errors

If an SSP target port transmits a RESPONSE frame and receives a NAK for that frame, the SSP target port retransmits, in the same or a new connection, the RESPONSE frame at least one time with the RETRANSMIT bit set to one and with the other fields set to the same values as in the original RESPONSE frame (see 9.2.6.3.3.3).

If an SSP target port transmits a RESPONSE frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

- 1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
- 2) the SSP target port retransmits, in a new connection, the RESPONSE frame with:
 - A) the RETRANSMIT bit set to one; and
 - B) the other fields set to the same values as in the original RESPONSE frame (see 9.2.6.3.3.3).

The ST_TTS state machine retransmits each RESPONSE frame that does not receive an ACK at least one time (see 9.2.6.3.3). The number of times it retransmits each RESPONSE frame is vendor-specific.

If an SSP initiator port receives a RESPONSE frame with a RETRANSMIT bit set to one, and it has previously received a RESPONSE frame for the same I_T_L_Q nexus, the ST_IFR state machine discards the extra RESPONSE frame (see 9.2.6.3.2). If the ST_IFR state machine has not previously received a RESPONSE frame for the I_T_L_Q nexus, then it considers the RESPONSE frame to be the valid RESPONSE frame.

9.2.5 SSP transport layer error handling summary

9.2.5.1 SSP transport layer error handling summary introduction

This subclause contains a summary of how SSP ports process transport layer errors. This summary does not include every error case. See 9.2.4 for transport layer handling of link layer errors (e.g., using transport layer retries).

9.2.5.2 SSP initiator port transport layer error handling summary

If an SSP initiator port receives a COMMAND or TASK frame or an unsupported frame type, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3).

If an SSP initiator port receives an XFER_RDY, read DATA, or RESPONSE frame with an unknown TAG field value, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3). The application client may then abort the command with that tag.

If an SSP initiator port receives an XFER_RDY frame that is not 12 bytes long, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3). The application client may then abort the command.

If an SSP initiator port receives an XFER_RDY frame in response to a command with no write data, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives an XFER_RDY frame requesting more write data than expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives an XFER_RDY frame requesting zero bytes, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If transport layer retries are disabled and an SSP initiator port receives an XFER_RDY frame with a requested offset that was not expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives a read DATA frame in response to a command with no read data, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives a read DATA frame with more read data than expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If an SSP initiator port receives a read DATA frame with zero bytes, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If transport layer retries are disabled and an SSP initiator port receives a read DATA frame with a data offset that was not expected, the ST_ITS state machine discards that frame and any subsequent read DATA frames received for that command (see 9.2.6.2.3.7) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If an SSP initiator port receives a RESPONSE frame that is not the correct length, the ST_IFR state machine considers the command or task management function completed with an error and discards the frame (see 9.2.6.2.2.3).

9.2.5.3 SSP target port transport layer error handling summary

If an SSP target port receives an XFER_RDY or RESPONSE frame or another unsupported frame type, the ST_TFR state machine discards the frame (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame and:

- a) the frame is too short to contain a LOGICAL UNIT NUMBER field;
- b) the frame is too short to contain a CDB; or
- c) the ADDITIONAL CDB LENGTH field specifies that the frame should be a different length,

then the ST_TTS state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port receives a TASK frame that is too short, the ST_TTS state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame with a tag that is already in use for another command, the device server may return CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to OVERLAPPED COMMANDS ATTEMPTED (see 10.2.3).

If an SSP target port receives:

- a) a COMMAND frame with a tag that is already in use for a task management function; or
- b) a TASK frame with a tag that is already in use for a command or another task management function,

then the task router and task manager(s) return a RESPONSE frame with the RESPONSE CODE field set to OVERLAPPED TAG ATTEMPTED (see 10.2.4).

If an SSP target port receives a write DATA frame with an unknown tag, the ST_TFR state machine discards the frame (see 9.2.6.3.2).

If an SSP target port receives a write DATA frame that does not contain first burst data and for which there is no XFER_RDY frame outstanding (i.e., it has received all requested write data), the ST_TFR state machine discards the frame (see 9.2.6.3.2.2).

If an SSP target port receives a TASK frame with an unknown logical unit number, the ST_TFR state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame or TASK frame with a TARGET PORT TRANSFER TAG field set to a value other than FFFFh, the ST_TFR state machine may return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port is using target port transfer tags and receives a write DATA frame with an unknown target port transfer tag, the ST_TFR state machine discards the frame (see 9.2.6.3.3).

If transport layer retries are disabled and an SSP target port receives a write DATA frame with a data offset that was not expected, the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to DATA OFFSET ERROR (see 10.2.3).

If an SSP target port receives a write DATA frame with more write data than expected (i.e., the write DATA frame contains data in excess of that requested by an XFER_RDY frame or, for first burst data, indicated by the FIRST BURST LENGTH field in the Disconnect-Reconnect mode page), the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to TOO MUCH WRITE DATA (see 10.2.3).

If an SSP target port receives a write DATA frame with zero bytes, the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to INFORMATION UNIT TOO SHORT (see 10.2.3).

9.2.6 ST (transport layer for SSP ports) state machines

9.2.6.1 ST state machines overview

The ST state machines perform the following functions:

- a) receive and process transport protocol service requests and transport protocol service responses from the SCSI application layer;
- b) receive and process other SAS connection management requests from the application layer;
- c) send transport protocol service indications and transport protocol service confirmations to the SCSI application layer;
- d) send requests to the port layer to transmit frames and manage SAS connections; and
- e) receive confirmations from the port layer.

The following confirmations between the ST state machines and the port layer:

- a) Transmission Status; and
- b) Frame Received;

include the following as arguments:

- a) tag;
- b) destination SAS address; and
- c) source SAS address;

and are used to route the confirmations to the correct ST state machines.

NOTE 78 - Although allowed by this standard, the ST state machines do not handle bidirectional commands that result in concurrent write DATA frames and read DATA frames.

9.2.6.2 ST_I (transport layer for SSP initiator ports) state machines

9.2.6.2.1 ST_I state machines overview

The ST_I state machines are as follows:

- a) ST_IFR (initiator frame router) state machine (see 9.2.6.2.2); and
- b) ST_ITS (initiator transport server) state machine (see 9.2.6.2.3).

The SAS initiator port includes:

- a) one ST_IFR state machine; and
- b) one ST_ITS state machine for each possible task and task management function (i.e., for each tag).

Figure 203 shows the ST_I state machines.

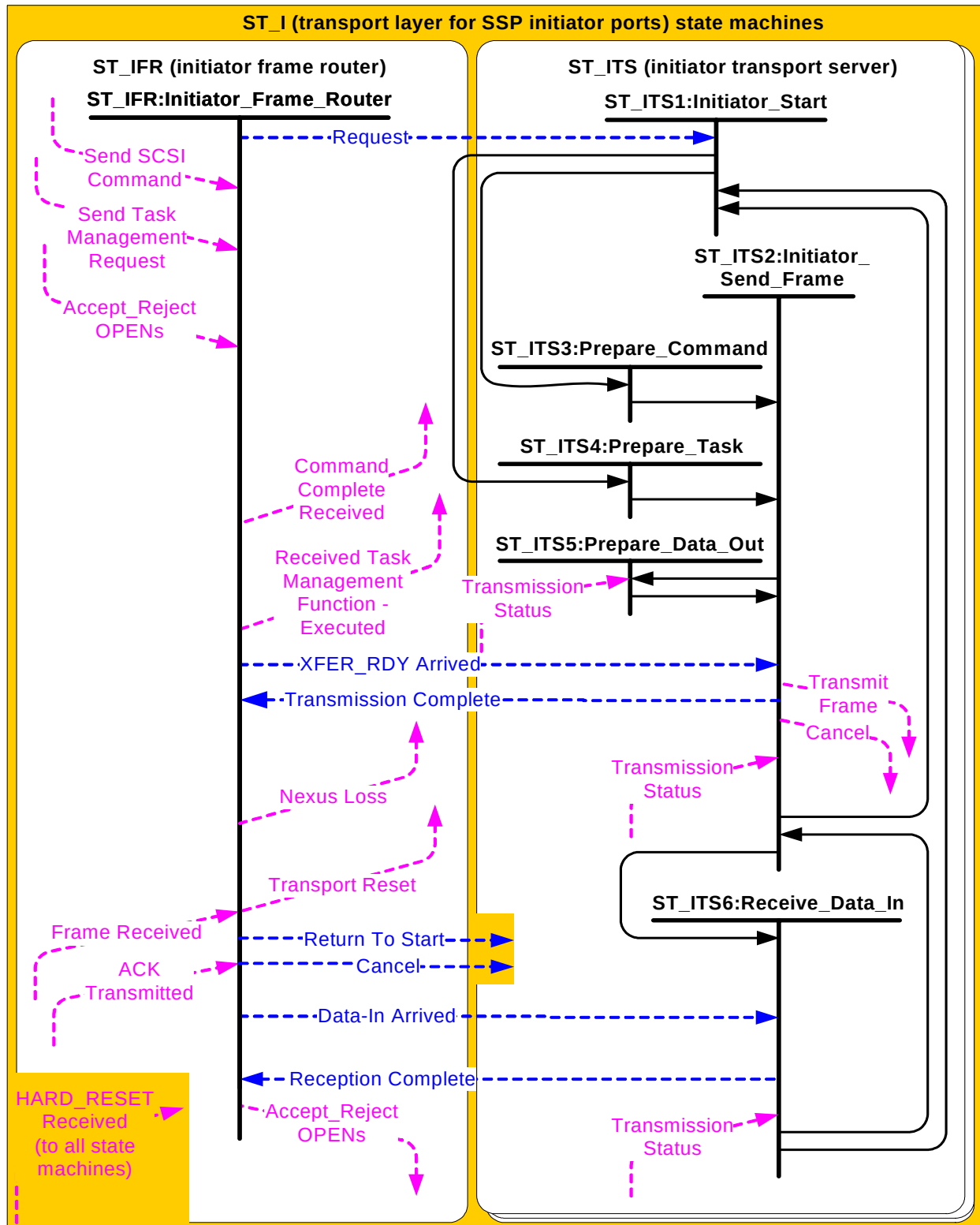


Figure 203 — ST_I (transport layer for SSP initiator ports) state machines

9.2.6.2.2 ST_IFR (initiator frame router) state machine

9.2.6.2.2.1 ST_IFR state machine overview

The ST_IFR state machine performs the following functions:

- a) receives Send SCSI Command and Send Task Management transport protocol service requests from the SCSI application layer;
- b) sends messages to the ST_ITS state machine;
- c) receives messages from the ST_ITS state machine;
- d) receives confirmations from the port layer;
- e) sends transport protocol service confirmations to the SCSI application layer;
- f) receives vendor-specific requests from the SCSI application layer;
- g) sends vendor-specific confirmations to the SCSI application layer;
- h) receives Accept_Reject OPENs requests from the SCSI application layer;
- i) sends Accept_Reject OPENs requests to the port layer;
- j) sends I_T Nexus Loss event notifications to the SCSI application layer; and
- k) sends Transport Reset event notifications to the SCSI application layer.

This state machine consists of one state.

This state machine shall be started after power on.

9.2.6.2.2.2 Processing transport protocol service requests

If this state machine receives a Send SCSI Command transport protocol service request then this state machine shall send a Request (Send Command) message with Command arguments and Application Client Buffer arguments to the ST_ITS state machine for the specified tag.

The following is the list of Command arguments:

- a) connection rate;
- b) initiator connection tag;
- c) destination SAS address;
- d) source SAS address set to the SAS address of the SSP initiator port;
- e) tag;
- f) logical unit number;
- g) task priority;
- h) task attribute;
- i) additional CDB length;
- j) CDB;
- k) additional CDB bytes, if any;
- l) first burst enabled; and
- m) request fence.

The following is the list of Application Client Buffer arguments:

- a) data-in buffer size;
- b) data-out buffer; and
- c) data-out buffer size.

If the command is performing a write operations and the Send SCSI Command transport service request contains a First Burst Enabled argument, then the Request (Send Command) message shall also include the Enable First Burst argument and the number of bytes for the First Burst Size argument.

If this state machine receives a Send Task Management Request transport protocol service request, then this state machine shall send a Request (Send Task) message with the Task arguments to the ST_ITS state machine for the specified tag.

The following is the list of Task arguments:

- a) connection rate;
- b) initiator connection tag;

- c) source SAS address set to the SAS address of the SSP initiator port;
- d) destination SAS address;
- e) retransmit bit;
- f) tag;
- g) logical unit number;
- h) task management function;
- i) tag of task to be managed; and
- j) request fence.

If the ST_ITS state machine for the tag specified in the Send Task Management Request is currently in use, then this state machine shall set the retransmit bit argument to one. If the ST_ITS state machine for the tag specified in the Send Task Management Request is not currently in use, then this state machine shall set the retransmit bit argument to zero.

9.2.6.2.2.3 Processing Frame Received confirmations

If this state machine receives a Frame Received (ACK/NAK Balanced) confirmation or Frame Received (ACK/NAK Not Balanced) confirmation, then this state machine shall compare the frame type of the frame received with the received confirmation (see table 156 in 9.2.1). If the confirmation was Frame Received (ACK/NAK Balanced) and the frame type is not XFER_RDY, RESPONSE, or DATA, then this state machine shall discard the frame. If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then this state machine shall discard the frame.

If the frame type is correct relative to the Frame Received confirmation, then this state machine may check that the hashed source SAS address matches the SAS address of the SAS port that transmitted the frame and that the hashed destination SAS address matches the SAS address of the SAS port that received the frame based on the connection information. If this state machine checks these SAS addresses, and they do not match, then this state machine:

- a) shall discard the frame; and
- b) may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is XFER_RDY then this state machine shall check the length of the information unit. If the length of the information unit is not correct, then this state machine:

- a) shall discard the frame; and
- b) may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is XFER_RDY and the tag is for a task with no write data, then this state machine shall:

- a) discard the frame;
- b) send a Command Complete Received transport protocol service confirmation with the Delivery Result argument set to Service Delivery or Target Failure - DATA Not Expected to the SCSI application layer; and
- c) if there is an ST_ITS state machine for the tag, send a Return To Start message to that state machine.

If the frame type is DATA and the tag is for a task with no read data, then this state machine shall:

- a) discard the frame;
- b) send a Command Complete Received transport protocol service confirmation with the Delivery Result argument set to Service Delivery or Target Failure - DATA Not Expected to the SCSI application layer; and
- c) if there is an ST_ITS state machine for the tag, send a Return To Start message to that state machine.

If the frame type is RESPONSE then this state machine shall check the length of the information unit. If the length of the information unit is not correct and the RESPONSE frame was for a command, then this state shall discard the frame and send a Command Complete Received confirmation to the SCSI application layer with the Service Response argument set to Service Delivery or Target Failure. If the length of the information unit is not correct and the RESPONSE frame was for a task management function, then this state shall

discard the frame and send a Received Task Management Function – Executed confirmation to the SCSI application layer with the Service Response argument set to Service Delivery or Target Failure.

If the frame type is correct relative to the Frame Received confirmation, then this state machine shall check the tag. If the tag does not specify a valid ST_ITS state machine, then this state machine shall discard the frame and may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is RESPONSE, and this state machine has received a RESPONSE frame for the I_T_L_Q nexus, then this state machine shall discard the frame.

If the frame type is RESPONSE, the fields checked in the frame are correct, and this state machine has not received a RESPONSE frame for this I_T_L_Q nexus, then this state machine shall send a Return To Start message to the ST_ITS state machine for the specified tag and:

- a) if the RESPONSE frame was for a command, then this state machine shall send a Command Complete Received protocol service confirmation to the SCSI application layer with the arguments set as specified in table 192 (see 10.2.1.5); or
- b) if the RESPONSE frame was for a task management request, then this state machine shall send a Received Task Management Function Executed protocol service confirmation to the SCSI application layer with the arguments set as specified in table 192 (see 10.2.1.5).

If the frame type is XFER_RDY and the fields checked in the frame are correct, then this state machine shall wait to receive an ACK Transmitted confirmation.

If this state machine receives an ACK Transmitted confirmation for an XFER_RDY frame, then this state machine shall send an XFER_RDY Arrived message to ST_ITS state machine specified by the tag. The message shall include the following Xfer_Rdy arguments:

- a) retry data frames;
- b) retransmit bit;
- c) target port transfer tag;
- d) requested offset; and
- e) write data length.

If the frame type is DATA and the fields checked in the frame are correct, then this state machine shall send a Data-In Arrived message to the ST_ITS state machine specified by the tag. The message shall include the following Read Data arguments:

- a) changing data pointer;
- b) number of fill bytes;
- c) data offset; and
- d) data.

9.2.6.2.2.4 Processing Transmission Complete and Reception Complete messages

If this state receives a Transmission Complete (I_T Nexus Loss), then this state machine shall send a Nexus Loss event notification to the SCSI application layer.

Table 168 defines the transport protocol service confirmation and its argument generated as a result of receiving a Transmission Complete message or a Reception Complete message that indicate an error occurred during the transmission or reception of a frame.

Table 168 — Confirmations sent to the SCSI application layer if a frame transmission or reception error occurs

Message received from ST_ITS state machine	Protocol service confirmation and Delivery Result argument sent to the SCSI application layer
Transmission Complete (Command Failed, ACK/NAK Timeout)	Command Complete Received (Service Delivery or Target Failure - ACK/NAK Timeout)
Transmission Complete (Command Failed, NAK Received)	Command Complete Received (Service Delivery or Target Failure - NAK Received)
Transmission Complete (Command Failed, Connection Failed)	Command Complete Received (Service Delivery or Target Failure - Connection Failed)
Transmission Complete (Task Failed, ACK/NAK Timeout)	Received Task Management Function - Executed (Service Delivery or Target Failure - ACK/NAK Timeout)
Transmission Complete (Task Failed, NAK Received)	Received Task Management Function - Executed (Service Delivery or Target Failure - NAK Received)
Transmission Complete (Task Failed, Connection Failed)	Received Task Management Function - Executed (Service Delivery or Target Failure - Connection Failed)
Transmission Complete (XFER_RDY Incorrect Write Data Length)	Command Complete Received (Service Delivery or Target Failure - XFER_RDY Incorrect Write Data Length)
Transmission Complete (XFER_RDY Requested Offset Error)	Command Complete Received (Service Delivery or Target Failure - XFER_RDY Requested Offset Error)
Transmission Complete (Cancel Acknowledged)	Command Complete Received (Service Delivery or Target Failure - Cancel Acknowledged)
Reception Complete (Data Offset Error)	Command Complete Received (Service Delivery or Target Failure - DATA Offset Error)
Reception Complete (Too Much Read Data)	Command Complete Received (Service Delivery or Target Failure - DATA Too Much Read Data)
Reception Complete (Information Unit Too Short)	Command Complete Received (Service Delivery or Target Failure - DATA Information Unit Too Short)
Reception Complete (Command Failed, ACK/NAK Timeout)	Command Complete Received (Service Delivery or Target Failure - ACK/NAK Timeout)
Reception Complete (Cancel Acknowledged)	Command Complete Received (Service Delivery or Target Failure - Cancel Acknowledged)

The protocol service confirmation shall include the tag as an argument.

9.2.6.2.2.5 Processing miscellaneous requests

If this state machine receives an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request, then this state machine shall send a corresponding Accept_Reject OPENs request to the port layer.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall send a Transport Reset event notification to the SCSI application layer.

This state machine may receive vendor-specific requests from the SCSI application layer that cause this state machine to send a Cancel message to an ST_ITS state machine.

9.2.6.2.3 ST_ITS (initiator transport server) state machine

9.2.6.2.3.1 ST_ITS state machine overview

The ST_ITS state machine performs the following functions:

- a) receives and processes messages from the ST_IFR state machine;
- b) sends messages to the ST_IFR state machine;
- c) sends request to the port layer regarding frame transmission;
- d) receives confirmations from the port layer regarding frame transmission; and
- e) receives HARD_RESET Received confirmations from the port layer.

This state machine consists of the following states:

- a) ST_ITS1:Initiator_Start state (see 9.2.6.2.3.2) (initial state);
- b) ST_ITS2:Initiator_Send_Frame state (see 9.2.6.2.3.3);
- c) ST_ITS3:Prepare_Command state (see 9.2.6.2.3.4);
- d) ST_ITS4:Prepare_Task state (see 9.2.6.2.3.5);
- e) ST_ITS5:Prepare_Data_Out state (see 9.2.6.2.3.6); and
- f) ST_ITS6:Receive_Data_In state (see 9.2.6.2.3.7).

This state machine shall start in the ST_ITS1:Initiator_Start state after power on.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall transition to the ST_ITS1:Initiator_Start state.

This state machine shall maintain the state machine variables defined in table 169.

Table 169 — ST_ITS state machine variables

State machine variable	Description
Data-In Buffer Offset	Current offset in the application client's data-in buffer (i.e., the application client buffer for read data)
Data-Out Buffer Offset	Current offset in the application client's data-out buffer (i.e., the application client buffer for write data)
Previous Requested Offset	Offset in the application client's data-out buffer (i.e., the application client buffer for write data) from the last XFER_RDY frame received
Previous Write Data Length	Write data length from the last XFER_RDY frame received

This state machine shall maintain the state machine arguments defined in table 170.

Table 170 — ST_ITS state machine arguments

State machine argument	Description
Command	Consists of the Command arguments received in the Request (Send Command) message
Task	Consists of the arguments received in the Request (Send Task) message
Xfer_Rdy	Consists of the arguments received in the XFER_RDY Arrived message
Data-Out Buffer	The location of the application client's data-out buffer (i.e., the application client buffer for write data)
Data-Out Buffer Size	The size in bytes of the application client's data-out buffer (i.e., the application client buffer for write data)
Data-In Buffer Size	The size in bytes of the application client's data-in buffer (i.e., the application client buffer for read data)

9.2.6.2.3.2 ST_ITS1:Initiator_Start state

9.2.6.2.3.2.1 State description

This state is the initial state of the ST_ITS state machine.

Upon entry into this state, this state shall set the Data-In Buffer Offset state machine variable to zero.

Upon entry into this state, this state shall set the Data-Out Buffer Offset state machine variable to zero.

9.2.6.2.3.2.2 Transition ST_ITS1:Initiator_Start to ST_ITS3:Prepare_Command

This transition shall occur after this state receives a Request (Send Command) message.

9.2.6.2.3.2.3 Transition ST_ITS1:Initiator_Start to ST_ITS4:Prepare_Task

This transition shall occur after this state receives a Request (Send Task) message.

9.2.6.2.3.3 ST_ITS2:Initiator_Send_Frame state

If this state is entered from the ST_ITS3:Prepare_Command state for transmission of a COMMAND frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS6:Receive_Data_In state, and the vendor-specific number of retries has not been reached for the COMMAND frame requesting a read operation, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS4:Prepare_Task state for transmission of an TASK frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS5:Prepare_Data_Out state for transmission of a write DATA frame, then this state shall send a Transmit Frame (Non-Interlocked) request to the port layer after this state has received an XFER_RDY Arrived message.

If this state is entered from the ST_ITS5:Prepare_Data_Out state for transmission of a write DATA frame and first bust is enabled, then this state shall send a Transmit Frame (Non-Interlocked) request to the port layer after this state has received a Transmission Status (Frame Transmitted) confirmation and a Transmission Status (ACK Received) confirmation for the COMMAND frame.

A Transmit Frame request shall include the COMMAND frame from the ST_ITS3:Prepare_Command state or from the ST_ITS6:Receive_Data_In state, the TASK frame from the ST_ITS4:Prepare_Task state, or the write

DATA frame from the ST_ITS5:Prepare_Data_Out state and the following arguments to be used for any OPEN address frame:

- a) initiator port bit set to one;
- b) protocol set to SSP;
- c) Connection Rate argument;
- d) Initiator Connection Tag argument;
- e) Destination SAS Address argument; and
- f) Source SAS Address argument.

After sending a Transmit Frame request this state shall wait to receive a Transmission Status confirmation.

If the confirmation is Transmission Status (I_T Nexus Loss), then this state shall send a Transmission Complete (I_T Nexus Loss) message to the ST_IFR state machine. This Transmission Complete message shall include the tag as an argument.

If the confirmation is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss) (see table 152 in 8.2.2.3.4), and the Transmit Frame request was for a COMMAND frame or a DATA frame, then this state shall send a Transmission Complete (Command Failed, Connection Failed) message to the ST_IFR state machine. The message shall include the tag.

If the confirmation is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss) (see table 152 in 8.2.2.3.4), and the Transmit Frame request was for a TASK frame, then this state shall send a Transmission Complete (Task Failed, Connection Failed) message to the ST_IFR state machine. The message shall include the tag.

If the confirmation is Transmission Status (Frame Transmitted), and the Transmit Frame request was for a COMMAND frame not requesting a read operation, a COMMAND frame not requesting a write operation, a TASK frame, or a write DATA frame where the number of data bytes that have been transmitted equal the Data-Out Buffer Size state machine argument, then this state shall wait to receive one of the following confirmations:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout); or
- d) Transmission Status (Connection Lost Without ACK/NAK).

If the confirmation is Transmission Status (Frame Transmitted), and the Transmit Frame request was for a COMMAND frame requesting a write operation, or a write DATA frame where the number of data bytes that have been transmitted is less than the Data-Out Buffer Size state machine argument and the write data length from the previous XFER_RDY frame, then this state shall wait to receive one of the following confirmations:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout);
- d) Transmission Status (Connection Lost Without ACK/NAK); or
- e) XFER_RDY Arrived message.

If a XFER_RDY Arrived message is received, then the ST_ITS shall respond to the XFER_RDY frame as if a Transmission Status (ACK Received) was received.

NOTE 79 - If the number of data bytes requested to be transmitted for the Send SCSI Command protocol service request are fewer than the number of bytes in the service request, then this state may send additional Transmit Frame requests for write DATA frames for the protocol service request before receiving a Transmission Status (ACK Received), Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK) confirmation for Transmit Frame requests for previous write DATA frames sent for the I_T_L_Q nexus.

After a Transmission Status (Frame Transmitted) is received, if a confirmation of Transmission Status (NAK Received) is received, the Transmit Frame request was for a COMMAND frame, and the vendor-specific number of retries has not been reached, then this state shall send a Transmit Frame (interlocked) request to the port layer (i.e., the last COMMAND frame is retransmitted).

After a Transmission Status (Frame Transmitted) is received, if a confirmation of Transmission Status (NAK Received) is received, the Transmit Frame request was for a TASK frame, and the vendor-specific number of retries has not been reached, then this state shall send a Transmit Frame (interlocked) request to the port layer (i.e., the last TASK frame is retransmitted).

Table 171 defines the messages that this state shall send to the ST_IFR state machine upon receipt of the listed confirmations, based on the conditions under which each confirmation was received.

Table 171 — Messages sent to the ST_IFR state machine

Confirmation received from the port layer	Conditions under which confirmation was received	Message sent to ST_IFR state machine
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)	The Transmit Frame request was for a COMMAND frame.	Transmission Complete (Command Failed, ACK/NAK Timeout)
	The Transmit Frame request was for a TASK frame.	Transmission Complete (Task Failed, ACK/NAK Timeout)
Transmission Status (NAK Received)	The Transmit Frame request was for a COMMAND frame and the vendor-specific number of retries has been reached.	Transmission Complete (Command Failed, NAK Received)
	The Transmit Frame request was for a TASK frame and the vendor-specific number of retries has been reached.	Transmission Complete (Task Failed, NAK Received)
Transmission Status (NAK Received)	The Transmit Frame request was for a write DATA frame and:	Transmission Complete (Data-Out Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)	a) the RETRY DATA FRAMES bit was set to zero in the XFER_RDY frame requesting the data; or	Transmission Complete (Data-Out Failed, ACK/NAK Timeout)
	b) the RETRY DATA FRAMES bit was set to one in the XFER_RDY frame requesting the data, and the vendor-specific number of retries has been reached.	

After this state sends a Transmission Complete (Command Failed, ACK/NAK Timeout) this state shall continue processing messages and confirmations.

NOTE 80 - The application client may determine the command was received and is being processed by the device server and allow the command to complete. The application client may accomplish this by the use of the QUERY TASK task management request.

If this state receives a Return to Start message or a Return to Start argument, and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to the port layer. This state may also send a Cancel request to the port layer to cancel a previous Transmit Frame request.

If this state receives a Cancel message or a Cancel argument, and this state has received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Transmission Complete (Cancel Acknowledged) message to the ST_IFR state machine.

If this state receives a Cancel message or a Cancel argument, and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to the port layer. This state may also send a Cancel request to the port layer to cancel a previous Transmit Frame request. The Cancel request shall include the following arguments:

- a) destination SAS address; and

b) tag.

NOTE 81 - The Cancel message results from a vendor-specific request from the SCSI application layer after the SCSI application layer has used a task management function to determine that the SAS target port did not receive the COMMAND frame.

If this state receives a Transmission Status (Cancel Acknowledged) confirmation, then this state shall send a Transmission Complete (Cancel Acknowledged) message to the ST_IFR state machine.

If this state receives an XFER_RDY Arrived message, then this state shall verify the Xfer_Rdy state machine argument as specified in table 172. If the verification fails, then this state sends the Transmission Complete message specified in table 172 to the ST_IFR state machine.

Table 172 — Transmission Complete messages for XFER_RDY frame verification failures

Message sent to ST_IFR ^a	Condition
Transmission Complete (XFER_RDY Incorrect Write Data Length)	The Write Data Length Xfer_Rdy state machine argument is zero.
	The Requested Offset Xfer_Rdy state machine argument plus the Write Data Length Xfer_Rdy state machine argument is greater than the Data-Out Buffer Size state machine argument.
Transmission Complete (XFER_RDY Requested Offset Error)	First burst is disabled, this is the first XFER_RDY frame for a command, and the value in the Requested Offset Xfer_Rdy state machine argument is not set to zero.
	First burst is enabled, this is the first XFER_RDY frame for a command, and the value in the Requested Offset Xfer_Rdy state machine argument is not equal to the value indicated by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2.5).
	Transport layer retries are disabled and the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable plus the Previous Write Data Length Field state machine variable.
	Transport layer retries are enabled, the Retransmit Bit Xfer_Rdy state machine argument is set to zero, and the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable plus the Previous Write Data Length state machine variable.
	Transport layer retries are enabled, this is not the first XFER_RDY frame for the command, the Retransmit Bit Xfer_Rdy state machine argument is set to one, and the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable.
^a If more than one condition is true, then this state shall send the Transmission Complete (XFER_RDY Incorrect Write Data Length) message to the ST_IFR state machine.	

After this state verifies an XFER_RDY frame, it shall:

- set the Data-Out Buffer Offset state machine variable to the Requested Offset Xfer_Rdy state machine argument;
- set the Previous Requested Offset state machine variable to the Requested Offset Xfer_Rdy state machine argument; and
- set the Previous Write Data Length state machine variable to the Requested Write Data Length Xfer_Rdy state machine argument.

9.2.6.2.3.3.1 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS1:Initiator_Start

This transition shall occur after:

- a) this state has sent one of the following to the ST_IFR state machine:
 - A) a Transmission Complete (Command Failed, NAK Received) message;
 - B) a Transmission Complete (Task Failed, ACK/NAK Timeout) message;
 - C) a Transmission Complete (Task Failed, NAK Received) message;
 - D) a Transmission Complete (Command Failed, ACK/NAK Timeout) message and the command was for a non-data operation;
 - E) a Transmission Complete (Data-Out Failed, NAK Received) message;
 - F) a Transmission Complete (Data-Out Failed, ACK/NAK Timeout) message;
 - G) a Transmission Complete (XFER_RDY Incorrect Write Data Length) message;
 - H) a Transmission Complete (XFER_RDY Requested Offset Error) message; or
 - I) a Transmission Complete (Cancel Acknowledged) message;
- or
- b) this state has received a Return To Start message or Return To Start argument, and has received:
 - A) confirmations for all Transmit Frame requests sent to the port layer; or
 - B) a Transmission Status (Cancel Acknowledged) confirmation.

9.2.6.2.3.3.2 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS5:Prepare_Data_Out

If first burst is enabled, this transition shall occur and include the First Burst argument after this state receives:

- a) a Transmission Status (Frame Transmitted) confirmation followed by a Transmission Status (ACK Received) for a COMMAND frame requesting a write operation; or
- b) a Transmission Status (Frame Transmitted) confirmation for a Transmit Frame (Non-interlocked) request if the Data-Out Buffer Offset state machine variable is less than the first burst size.

This transition shall occur after this state receives:

- a) an XFER_RDY Arrived message; or
- b) a Transmission Status (Frame Transmitted) confirmation for a Transmit Frame (Non-interlocked) request if the Data-Out Buffer Offset state machine variable is less than the Requested Offset Xfer_Rdy state machine argument plus the Write Data Length Xfer_Rdy state machine argument.

NOTE 82 - This transition occurs even if this state has not received a Transmission Status (ACK Received) for the write DATA frame.

This transition shall include a Retry argument and occur after:

- a) this state receives one of the following confirmations or arguments for a write DATA frame:
 - A) Transmission Status (NAK Received);
 - B) Transmission Status (ACK/NAK Timeout); or
 - C) Transmission Status (Connection Lost without ACK/NAK);
- b) the RETRY DATA FRAMES bit is set to one in the XFER_RDY frame for the write operation;
- c) the Data-Out Buffer Offset state machine variable is set to the Requested Offset Xfer_Rdy state machine argument;
- d) all write DATA frames that have received a Transmission Status (Frame Transmitted) confirmation have received a Transmission Status confirmation; and
- e) the vendor-specific number of retries, if any, for the write DATA frame has not been reached.

9.2.6.2.3.3.3 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS6:Process_Data_In

This transition shall occur after this state receives a Transmission Status (Frame Transmitted) confirmation for a COMMAND frame for a command requesting a read operation.

NOTE 83 - This transition occurs even if this state has not received a Transmission Status (ACK Received) for the COMMAND frame.

9.2.6.2.3.4 ST_ITS3:Prepare_Command state

9.2.6.2.3.4.1 State description

This state shall construct a COMMAND frame using the Command arguments:

- a) FRAME TYPE field set to 06h (i.e., COMMAND frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Commands argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
- d) RETRY DATA FRAMES bit set to zero;
- e) RETRANSMIT bit set to zero;
- f) CHANGING DATA POINTER bit set to zero;
- g) NUMBER OF FILL BYTES field set zero.
- h) TAG field set to the Tag Command argument;
- i) TARGET PORT TRANSFER TAG field set to FFFFh;
- j) DATA OFFSET field set to zero;
- k) in the information unit, LOGICAL UNIT NUMBER field set to the Logical Unit Number Command argument;
- l) in the information unit, ENABLE FIRST BURST bit set to the Enable First Burst Command argument;
- m) in the information unit, TASK PRIORITY field set to the Task Priority Command argument;
- n) in the information unit, TASK ATTRIBUTE field set to the Task Attribute Command argument;
- o) in the information unit, ADDITIONAL CDB LENGTH field set to the Additional CDB Length Command argument;
- p) in the information unit, CDB field set to the CDB Command argument;
- q) in the information unit, ADDITIONAL CDB BYTES field set to the Additional CDB Bytes Command argument, if any; and
- r) no fill bytes.

9.2.6.2.3.4.2 Transition ST_ITS3:Prepare_Command to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state:

- a) constructs a COMMAND frame;
- b) receives a Cancel message; or
- c) receives a Return To Start message.

This transition shall include the:

- a) COMMAND frame as an argument;
- b) if a Cancel message was received, then a Cancel argument; or
- c) if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.5 ST_ITS4:Prepare_Task state

9.2.6.2.3.5.1 State description

This state shall construct a TASK frame using the Task arguments:

- a) FRAME TYPE field set to 16h (i.e., TASK frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Task argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
- d) RETRY DATA FRAMES bit set to zero;
- e) RETRANSMIT bit set to the Retransmit Bit Task argument;
- f) CHANGING DATA POINTER bit set to zero;
- g) NUMBER OF FILL BYTES field set zero.
- h) TAG field set to the Tag Task argument;
- i) TARGET PORT TRANSFER TAG field set to FFFFh;
- j) DATA OFFSET field set to zero;
- k) in the information unit, LOGICAL UNIT NUMBER field set to the Logical Unit Number Task argument;

- l) in the information unit, TASK MANAGEMENT FUNCTION field set to the Task Management Function Task argument;
- m) in the information unit, TAG OF TASK TO BE MANAGED field set to the Tag Task argument of task to be managed and
- n) no fill bytes.

9.2.6.2.3.5.2 Transition ST_ITS4:Prepare_Task to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state:

- a) constructs a TASK frame;
- b) receives a Cancel message; or
- c) receives a Return To Start message.

This transition shall include the:

- a) TASK frame as an argument;
- b) if a Cancel message was received, then a Cancel argument; or
- c) if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.6 ST_ITS5:Prepare_Data_Out state

9.2.6.2.3.6.1 State description

This state shall construct a write DATA frame using the following Xfer_Rdy state machine arguments and Command state machine arguments:

- a) FRAME TYPE field set to 01h (i.e., DATA frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Command argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
- d) RETRY DATA FRAMES bit set to zero;
- e) RETRANSMIT bit set to zero;
- f) CHANGING DATA POINTER bit set as specified in this subclause;
- g) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the specified data;
- h) TAG field set to the Tag Command argument;
- i) TARGET PORT TRANSFER TAG field set to FFFFh if this state received a First Burst argument or the Target Port Transfer Tag Xfer_Rdy argument if this state did not receive a First Burst argument;
- j) DATA OFFSET field set to the Data-Out Buffer Offset state machine variable;
- k) in the information unit, DATA field set to the information that starts at the location in the Data-Out Buffer state machine argument pointed to by the Data-Out Buffer Offset state machine variable. If the number of bytes remaining to be transferred as defined by the following calculation:

bytes remaining to be transferred = Write Data Length Xfer_Rdy state machine argument - (Data-Out Buffer Offset state machine argument - Requested Offset Xfer_Rdy state machine argument)

is equal to the maximum size of the write Data information unit, then the amount of data shall be the maximum size of the write Data information unit. Otherwise, the amount of data shall be the lesser of:

- A) the bytes remaining to be transferred; and
- B) the maximum size of the Write information unit;

and

- l) fill bytes, if any.

If this state is entered without a Retry argument, then this state shall set the CHANGING DATA POINTER bit to zero.

If this state is entered with a Retry argument, then this state shall set the CHANGING DATA POINTER bit to one.

After constructing the write DATA frame, this state shall set the Data-Out Buffer Offset state machine variable to the value of the DATA OFFSET field plus the number of bytes in the DATA field in the write Data information unit.

9.2.6.2.3.6.2 Transition ST_ITS5:Prepare_Data_Out to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state:

- a) constructs a write DATA frame;
- b) receives a Cancel message; or
- c) receives a Return To Start message.

This transition shall include the received Transmission Status, if any, as an argument and the:

- a) write DATA frame as an argument;
- b) if a Cancel message was received, then a Cancel argument; or
- c) if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.7 ST_ITS6:Receive_Data_In state

9.2.6.2.3.7.1 State description

If this state receives a Data-In Arrived message, then this state shall verify the values in the read DATA frame received with the message as defined in table 173.

If the verification fails, then this state sends the Reception Complete message specified in table 173 to the ST_IFR state machine.

Table 173 — Reception Complete messages for read DATA frame verification failures

Message sent to ST_IFR ^a	Condition
Reception Complete (Data Offset Error)	Transport layer retries are disabled, and the DATA OFFSET field in the read DATA frame is not equal to the Data-In Buffer Offset state machine variable.
	The DATA OFFSET field in the read DATA frame is greater than the Data-In Buffer Size state machine argument.
Reception Complete (Too Much Read Data)	The number of bytes in the DATA field in the read Data information unit plus the Data-In Buffer Offset state machine variable is greater than the Data-In Buffer Size state machine argument.
Reception Complete (Incorrect Data Length)	The number of bytes in the DATA field in the read Data information unit is zero.
^a If more than one condition is true, then this state shall select which message to send to the ST_IFR state machine using the following order: 1) Reception Complete (Data Offset Error); 2) Reception Complete (Too Much Read Data); or 3) Reception Complete (Incorrect Data Length).	

If:

- a) transport layer retries are enabled;
- b) the CHANGING DATA POINTER bit is set to zero;
- c) the DATA OFFSET field is not set to the Data-In Buffer Offset state machine variable;
- d) the DATA OFFSET field is less than the Data-In Buffer Size state machine argument; and
- e) the DATA OFFSET field plus the number of bytes in the DATA field in the read Data information unit is less than or equal to the Data-In Buffer Size state machine argument,

then this state should discard all Data-In Arrived messages until a read DATA frame is received in which the CHANGING DATA POINTER bit is set to one. This state shall resume processing additional Data-In Arrived messages when it receives a Data-In Arrived message with the CHANGING DATA POINTER bit set to one.

Editor's Note 47: The above wording should parallel that for write DATA frames in

ST_TTS5:Receive_Data_Out. Items d)e) are part of "verification"

If the read DATA frame verification is successful or after this state resumes processing Data-In Arrived messages, then this state shall process the data received in the read DATA frame and set the Data-In Buffer Offset state machine variable to the DATA OFFSET field plus the number of bytes in the DATA field in the read Data information unit.

If data received in the read DATA frame overlaps data previously received and verified successfully, this state may either discard the overlapping data, or replace the previously received data with the new data.

If this state receives Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK), then this state shall send a Reception Complete (Command Failed, Connection Failed) to the ST_IFR state machine.

After this state sends a Reception Complete (Command Failed, Connection Failed) message, this state shall continue processing messages and confirmations.

NOTE 84 - The application client may determine the command was received and is being processed by the device server and allow the command to complete.

If this state receives a Cancel message, then this state shall send a Reception Complete (Cancel Acknowledged) message to the ST_IFR state machine. The Reception Complete message shall include the tag as an argument.

NOTE 85 - The Cancel message results from a vendor-specific request from the SCSI application layer after the SCSI application layer has used a task management function to determine that the SAS target port did not receive the COMMAND frame.

9.2.6.2.3.7.2 Transition ST_ITS6:Receive_Data_In to ST_ITS1:Initiator_Start

This transition shall occur after this state:

- a) sends one of the following to the ST_IFR state machine:
 - A) a Reception Complete (Data Offset Error) message;
 - B) a Reception Complete (Too Much Read Data) message;
 - C) a Reception Complete (Incorrect Data Length) message; or
 - D) a Reception Complete (Cancel Acknowledged) message;
- or
- b) receives a Return To Start message.

9.2.6.2.3.7.3 Transition ST_ITS6:Receve_Data_In to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state receives a Transmission Status (NAK Received) confirmation for a COMMAND frame for a command requesting a read operation.

9.2.6.3 ST_T (transport layer for SSP target ports) state machines

9.2.6.3.1 ST_T state machines overview

The ST_T state machines are as follows:

- a) ST_TFR (target frame router) state machine (see 9.2.6.3.2); and
- b) ST_TTS (target transport server) state machine (see 9.2.6.3.3).

The SAS target port includes:

- a) one ST_TFR state machine; and
- b) one ST_TTS state machine for each possible task and task management function (i.e., for each tag).

The ST_TTS state machine may maintain the timers listed in table 174.

Table 174 — ST_T state machine timers

Timer	Initial value
Initiator Response Timeout	The value in the INITIATOR RESPONSE TIMEOUT field in the Protocol-Specific Port mode page (see 10.2.7.4).

```

sequenceDiagram
    participant ST_TFR as ST_TFR (target frame router)  
ST_TFR:Target_Frame_Router
    participant ST_TTS1 as ST_TTS (target transport server)  
ST_TTS1:Target_Start
    participant ST_TTS2 as ST_TTS2:Target_Send_Frame
    participant ST_TTS3 as ST_TTS3:Prepare_Data_In
    participant ST_TTS4 as ST_TTS4:Prepare_Xfer_Rdy
    participant ST_TTS5 as ST_TTS5:Receive_Data_Out

    ST_TTS1-->>ST_TFR: Request
    ST_TFR-->>ST_TTS1: Send Data-In
    ST_TFR-->>ST_TTS1: Receive Data-Out
    ST_TFR-->>ST_TTS1: Send Command Complete
    ST_TFR-->>ST_TTS1: Task Management Function Executed
    ST_TFR-->>ST_TTS1: Terminate Data Transfer
    ST_TFR-->>ST_TTS1: Accept_Reject OPENS
    ST_TTS1-->>ST_TFR: Cancel
    ST_TFR-->>ST_TTS1: Data-Out Received
    ST_TFR-->>ST_TTS1: Data-In Delivered
    ST_TFR-->>ST_TTS1: Data Transfer Terminated
    ST_TFR-->>ST_TTS1: SCSI Command Received
    ST_TFR-->>ST_TTS1: Task Management Request Received
    ST_TTS1-->>ST_TFR: Transmission Complete
    ST_TFR-->>ST_TTS1: Frame Received
    ST_TFR-->>ST_TTS1: ACK Transmitted
    ST_TFR-->>ST_TTS1: Nexus Loss
    ST_TFR-->>ST_TTS1: Transport Reset
    ST_TTS1-->>ST_TFR: Reception Complete
    ST_TTS1-->>ST_TFR: Data-Out Arrived
    ST_TTS1-->>ST_TFR: ACK Transmitted
    ST_TFR-->>ST_TTS1: Accept_Reject OPENS
    ST_TFR-->>ST_TTS1: HARD_RESET Received (to all state machines)
  
```

Figure 204 — ST_T (transport layer for SSP target ports) state machines

9.2.6.3.2 ST_TFR (target frame router) state machine

9.2.6.3.2.1 ST_TFR state machine overview

The ST_TFR state machine performs the following functions:

- a) receives confirmations from the port layer;
- b) receives transport protocol service requests from the SCSI application layer;
- c) sends transport protocol service indications to the SCSI application layer;
- d) sends messages to the ST_TTS state machine;
- e) receives messages from the ST_TTS state machine;
- f) receives Accept_Reject OPENs requests from the SCSI application layer;
- g) sends Accept_Reject OPENs requests to the port layer;
- h) sends Nexus Loss event notifications to the SCSI application layer;
- i) sends Transport Reset event notifications to the SCSI application layer; and
- j) sends Power Loss Expected event notifications to the SCSI application layer.

This state machine consists of one state.

This state machine shall be started after power on.

If this state receives a Notify Received (Power Loss Expected) confirmation, then this state shall:

- a) send a Cancel message to all the ST_TTS state machines; and
- b) send a Power Loss Expected confirmation to the SCSI application layer.

9.2.6.3.2.2 Processing Frame Received confirmations

If this state machine receives a Frame Received (ACK/NAK Balanced) or Frame Received (ACK/NAK Not Balanced) confirmation, then this state machine shall check the frame type in the received frame (see table 156 in 9.2.1). If the frame type is not COMMAND, TASK, or DATA, then this state machine shall discard the frame. If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then this state machine shall discard the frame.

This state machine may check that reserved fields in the received frame are zero. If any reserved fields are checked and they are not set to zero, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
- c) the service response argument set to Invalid Frame.

The check of reserved fields within the frame shall not apply to the reserved fields within the CDB in a COMMAND frame. Checking of reserved fields in a CDB is described in SPC-4.

The following is the list of Transport Response arguments:

- a) connection rate;
- b) initiator connection tag;
- c) destination SAS address (i.e., the SAS address to which the RESPONSE frame is to be transmitted);
- d) source SAS address set to the SAS address of the SAS port containing the state machine;
- e) tag; and
- f) service response.

The response fence argument is not included in the Transport Response arguments.

If the frame type is correct relative to the Frame Received confirmation, then this state machine may check that the hashed source SAS address matches the SAS address of the SAS port that transmitted the frame and that the hashed destination SAS address matches the SAS address of the SAS port that received the frame based on the connection information. If this state machine checks these SAS addresses, and they do not match, then this state machine shall discard the frame.

If the frame type is COMMAND or TASK then this state machine shall check the length of the information unit. If the length of the information unit is not correct, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
- c) the service response argument set to Invalid Frame.

If the frame type is TASK, this state machine checks tags, the RETRANSMIT bit in the new TASK frame is set to one, and the tag for the new TASK frame is the same as the tag for a previous TASK frame where the task management function for the previous TASK frame is not complete, then this state machine shall discard the new TASK frame and not send a Task Management Request Received confirmation to the SCSI application layer.

If the frame type is TASK and this state machine does not check tags, then this state machine shall ignore the RETRANSMIT bit.

If the frame type is COMMAND or TASK, then this state machine may check the target port transfer tag. If this state checks the target port transfer tag and the tag is set to a value other than FFFFh, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
- c) the service response argument set to Invalid Frame.

If the frame type is TASK, then this state machine shall check the logical unit number. If the logical unit number is unknown, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
- c) the service response argument set to Incorrect Logical Unit Number.

If the frame type is DATA and this frame is for first burst data or this state machine did not assign a target port transfer tag for the data transfer, then this state machine may check the target port transfer tag. If target port transfer tag is set to a value other than FFFFh, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
- c) the service response argument set to Invalid Frame.

If the frame type is COMMAND or TASK and the fields checked in the frame are correct, then this state machine shall wait to receive an ACK Transmitted confirmation.

If the frame type is COMMAND, the fields checked in the frame are correct, and this state machine receives an ACK Transmitted confirmation, then this state machine shall send a SCSI Command Received transport protocol service indication with the following arguments to the SCSI application layer:

- a) source SAS address (i.e., the SAS address that transmitted the COMMAND frame);
- b) tag;
- c) logical unit number;
- d) task attribute;
- e) task priority;
- f) CDB; and
- g) additional CDB bytes, if any.

If the frame type is TASK, the fields checked in the frame are correct, and this state machine receives an ACK Transmitted confirmation, then this state machine shall send a Task Management Request Received transport protocol service indication with the following arguments to the SCSI application layer:

- a) source SAS address (i.e., the SAS address that transmitted the TASK frame);
- b) tag;
- c) logical unit number;
- d) task management function; and
- e) tag of the task to be managed.

If the frame type is DATA, and the tag does not match a tag for an outstanding command performing write operations, then this state machine shall discard the frame.

If the frame type is DATA, and the tag matches a tag for an outstanding command performing write operations when first burst is disabled or for which no Transmission Complete (Xfer_Rdy Delivered) message has been received from an ST_TTS state machine, then this state machine shall discard the frame.

If the frame type is DATA and a target port transfer tag was received in a Transmission Complete (Xfer_Rdy Delivered) message, then this state machine shall check the target port transfer tag. If the target port transfer tag received in the DATA frame does not match the Target Transport Tag argument in the Transmission Complete (Xfer_Rdy Delivered) message, then this state machine shall discard the frame.

If the frame type is DATA and the fields checked in the frame are correct, and first burst is enabled or this state machine has received a Transmission Complete (Xfer_Rdy Delivered) from the ST_TTS state machine for the request, then this state machine shall send a Data-Out Arrived message to the ST_TTS state machine specified by the tag in the frame. The message shall include the content of the write DATA frame.

9.2.6.3.2.3 Processing transport protocol service requests and responses

If this state machine receives a Send Data-In transport protocol service request from the SCSI application layer, then this state machine shall send a Request (Send Data-In) message to an ST_TTS state machine that does not have an active task. The message shall include the following Data-In arguments:

- a) connection rate;
- b) initiator connection tag;
- c) destination SAS address (i.e., the SAS address to which the read DATA frame is to be transmitted);
- d) source SAS address set to the SAS address of the SSP target port;
- e) tag;
- f) device server buffer;
- g) request byte count; and
- h) application client buffer offset.

If this state machine receives a Receive Data-Out transport protocol service request from the SCSI application layer, then this state machine shall send a Request (Receive Data-Out) message to an ST_TTS state machine that does not have an active task. The message shall include the following Data-Out state machine arguments:

- a) connection rate;
- b) initiator connection tag;
- c) destination SAS address (i.e., the SAS address to which the XFER_RDY frame is to be transmitted);
- d) source SAS address set to the SAS address of the SSP target port;
- e) tag;
- f) device server buffer;
- g) request byte count;
- h) application client buffer offset; and
- i) target port transfer tag.

If first burst is enabled, then the Request (Receive Data_Out) message shall also include the Enable First Burst argument and First Burst Size argument. The First Burst Size argument shall be set to the first burst size from the Disconnect-Reconnect mode page (see 10.2.7.2.5).

If this state machine receives a Send Command Complete transport protocol service response from the SCSI application layer with the Service Response argument set to TASK COMPLETE, then this state machine shall send a Request (Send Application Response) message to the ST_TTS state machine specified by the tag. The message shall include the following Application Response arguments:

- a) connection rate;
- b) initiator connection tag;
- c) destination SAS address (i.e., the SAS address to which the RESPONSE frame is to be transmitted);
- d) source SAS address set to the SAS address of the SSP target port;
- e) tag;
- f) status;
- g) sense data, if any; and
- h) response fence.

If this state machine receives a Task Management Function Executed transport protocol service response from the SCSI application layer, then this state machine shall send the following to the ST_TTS state machine specified by the tag:

- a) a Request (Send Transport Response) message with the Transport Response arguments;
- b) the service response argument set as specified in table 175; and
- c) the response fence argument set to the Task Management Function Executed protocol service response Response Fence argument.

Table 175 specifies which argument to send with Request (Send Transport Response) based on the Service Response argument that was received.

Table 175 — Task Management Function Executed Service Response argument mapping to Request (Send Transport Response) service response argument

Task Management Function Executed protocol service response Service Response argument received	Request (Send Transport Response) message service response argument
FUNCTION COMPLETE	Task Management Function Complete
FUNCTION SUCCEEDED	Task Management Function Succeeded
FUNCTION REJECTED	Task Management Function Not Supported
INCORRECT LOGICAL UNIT NUMBER	Incorrect Logical Unit Number
SERVICE DELIVERY OR TARGET FAILURE - Overlapped Tag Attempted	Overlapped Tag Attempted

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has not sent a Request message to a ST_TTS state machine for the Send Data-In or Receive Data-Out protocol service request to which the Terminate Data Transfer request applies, then this state machine shall:

- 1) discard the Terminate Data Transfer request and any corresponding Send Data-In or Receive Data-Out request; and
- 2) send a Data Transfer Terminated protocol service confirmation to the SCSI application layer.

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has sent a Request message to a ST_TTS state machine for the Send Data-In protocol service request to which the Terminate Data Transfer request applies, then this state machine shall send a Cancel message to the ST_TTS state machine specified by the tag and the Send Data-In protocol service request.

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has sent a Request message to a ST_TTS state machine for the Receive Data-Out protocol service request to which the Terminate Data Transfer request applies, then this state

machine shall send a Cancel message to the ST_TTS state machine specified by the tag and the Receive Data-Out protocol service request.

This state machine receives Transmission Complete and Reception Complete messages that may result in this state machine sending a Nexus Loss event notification or a protocol service confirmation to the SCSI application layer. If this state machine receives a Transmission Complete (I_T Nexus Loss) message, then this state machine shall send a Nexus Loss event notification to the SCSI application layer. Table 176 defines messages received from ST_TTS state machines and the corresponding service confirmations, if any, that shall be sent upon receipt of the message.

Table 176 — Confirmations sent to the SCSI application layer (part 1 of 2)

Message received from ST_TTS state machine	Protocol service confirmation sent to SCSI application layer
Transmission Complete (Xfer_Rdy Delivered)	None
Transmission Complete (Response Delivered)	None
Transmission Complete (Response Failed) ^a	None
Transmission Complete (Data Transfer Terminated)	Data Transfer Terminated
Transmission Complete (Data-In Delivered)	Data-In Delivered with the Delivery Result argument set to DELIVERY SUCCESSFUL
Transmission Complete (Xfer_Rdy Failed, NAK Received)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - NAK RECEIVED
Transmission Complete (Xfer_Rdy Failed, ACK/NAK Timeout)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Transmission Complete (Xfer_Rdy Failed, Connection Failed)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Transmission Complete (Data-In Failed, NAK Received)	Data-In Delivered with the Delivery Result argument set to DELIVERY FAILURE - NAK RECEIVED
Transmission Complete (Data-In Failed, ACK/NAK Timeout)	Data-In Delivered with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Transmission Complete (Data-In Failed, Connection Failed)	Data-In Delivered with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Reception Complete (Data-Out Received)	Data-Out Received with the Delivery Result argument set to DELIVERY SUCCESSFUL
Reception Complete (Data Offset Error)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - DATA OFFSET ERROR
^a SAM-4 does not define a mechanism for the device server to determine the result of its Send Command Complete and Task Management Function Executed transport protocol service response calls.	

Table 176 — Confirmations sent to the SCSI application layer (part 2 of 2)

Message received from ST_TTS state machine	Protocol service confirmation sent to SCSI application layer
Reception Complete (Too Much Write Data)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - TOO MUCH WRITE DATA
Reception Complete (Information Unit Too Short)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - INFORMATION UNIT TOO SHORT
Reception Complete (Initiator Response Timeout)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - INITIATOR RESPONSE TIMEOUT
Reception Complete (Data Transfer Terminated)	Data Transfer Terminated
^a SAM-4 does not define a mechanism for the device server to determine the result of its Send Command Complete and Task Management Function Executed transport protocol service response calls.	

Each protocol service confirmation shall include the tag as an argument.

9.2.6.3.2.4 Processing miscellaneous requests and confirmations

If this state machine receives an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request, then this state machine shall send a corresponding Accept_Reject OPENs request to the port layer.

If this state machine receives a HARD_RESET Received confirmation, then this state shall send a Transport Reset event notification to the SCSI application layer.

9.2.6.3.3 ST_TTS (target transport server) state machine

9.2.6.3.3.1 ST_TTS state machine overview

The ST_TTS state machine performs the following functions:

- receives and processes messages from the ST_TFR state machine;
- sends messages to the ST_TFR state machine;
- communicates with the port layer using requests and confirmations regarding frame transmission; and
- receives HARD_RESET Received confirmations from the port layer.

This state machine consists of the following states:

- ST_TTS1:Target_Start (see 9.2.6.3.3.2) (initial state);
- ST_TTS2:Target_Send_Frame (see 9.2.6.3.3.3);
- ST_TTS3:Prepare_Data_In (see 9.2.6.3.3.4);
- ST_TTS4:Prepare_Xfer_Rdy (see 9.2.6.3.3.5);
- ST_TTS5:Receive_Data_Out (see 9.2.6.3.3.6); and
- ST_TTS6:Prepare_Response (see 9.2.6.3.3.7).

This state machine shall start in the ST_TTS1:Target_Start state after power on.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall transition to the ST_TTS1:Target_Start state.

The state machine shall maintain the state machine variables defined in table 177.

Table 177 — ST_TTS state machine variables

State machine variable	Description
Read Data Offset	Offset into the application client's data-in buffer (i.e., the application client buffer for read data)
Balance Point Read Data Offset	Offset into the application client's data-in buffer (i.e., the application client buffer for read data) of last point at which the number of Transmission Status (ACK Received) confirmations or arguments was equal to the number of transmitted read DATA frames
Read Data Frames Transmitted	The number of Transmission Status (Frame Transmitted) confirmations received for read DATA frames
Read Data Frames ACKed	The number of Transmission Status (ACK Received) confirmations received for read DATA frames
Read Data Buffer End	One greater than the offset into the application client's data-in buffer (i.e., the application client buffer for read data) of the last location into which read data is to be placed.
Requested Write Data Offset	Device server requested offset in the application client buffer for write data
Requested Write Data Length	Amount of write data requested by the device server from the application client buffer
Write Data Offset	Offset into the application client's data-out buffer (i.e., the application client buffer containing write data)

This state machine shall maintain the state machine arguments defined in table 178.

Table 178 — ST_TTS state machine arguments

State machine argument	Description
Data-In	The Data-In arguments received in the Request (Send Data-In) message (see 9.2.6.3.2.3)
Data-Out	The Data-Out arguments received in the Request (Receive Data-Out) message (see 9.2.6.3.2.3)

9.2.6.3.3.2 ST_TTS1:Target_Start state

9.2.6.3.3.2.1 State description

This state is the initial state of the ST_TTS state machine.

Upon entry into this state, this state shall:

- set the Read Data Offset state machine variable to the Application Client Buffer Offset Data-In state machine argument;
- set the Balance Point Read Data Offset state machine variable to the Application Client Buffer Offset Data-In state machine argument;
- set the Read Data Frames Transmitted state machine variable to zero;
- set the Read Data Frames ACKed state machine variable to zero;
- set the Read Data Buffer End state machine variable to the Application Client Buffer Offset Data-In state machine argument plus the Request Byte Count Data-In state machine argument; and

- f) set the Requested Write Data Offset state machine variable to the Application Client Buffer Offset Data-Out state machine argument.

If this state was entered without an Enable First Burst Data-Out state machine argument, then the Requested Write Data Length state machine variable shall be set to the Request Byte Count Data-Out state machine argument.

If this state was entered with an Enable First Burst Data-Out state machine argument, then the Requested Write Data Length state machine variable shall be set to the First Burst Size Data-Out state machine argument.

9.2.6.3.3.2.2 Transition ST_TTS1:Target_Start to ST_TTS3:Prepare_Data_In

This transition shall occur after this state receives a Request (Send Data-In) message.

9.2.6.3.3.2.3 Transition ST_TTS1:Target_Start to ST_TTS4:Prepare_Xfer_Rdy

If this state was entered without an Enable First Burst Data-Out state machine argument, then this transition shall occur after a Request (Receive Data-Out) message is received.

9.2.6.3.3.2.4 Transition ST_TTS1:Target_Start to ST_TTS5:Receive_Data_Out

If this state was entered with an Enable First Burst Data-Out state machine argument, then this transition shall occur after a Request (Receive Data-Out) message is received.

9.2.6.3.3.2.5 Transition ST_TTS1:Target_Start to ST_TTS7:Prepare_Response

This transition shall occur after this state receives a Request (Send Transport Response) message.

The transition shall include the Transport Response arguments.

9.2.6.3.3.3 ST_TTS2:Target_Send_Frame state

9.2.6.3.3.3.1 State description

If this state is entered from the ST_TTS3:Prepare_Data_In state for transmission of a read DATA frame, then this state shall send a Transmit Frame (Non-Interlocked) request to the port layer.

If this state is entered from the ST_TTS4:Prepare_Xfer_Rdy state for transmission of an XFER_RDY frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_TTS6:Prepare_Response state for transmission of a RESPONSE frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

All Transmit Frame requests from this state shall include the read DATA frame from the ST_TTS3:Prepare_Data_In state, the XFER_RDY frame from the ST_TTS4:Prepare_Xfer_Rdy state, or the RESPONSE frame from the ST_TTS6:Prepare_Response state and the following arguments to be used for any OPEN address frame:

- a) initiator port bit set to zero;
- b) protocol set to SSP;
- c) Connection Rate argument;
- d) Initiator Connection Tag argument;
- e) Destination SAS Address argument; and
- f) Source SAS Address argument.

After sending a Transmit Frame request this state shall wait to receive a Transmission Status confirmation.

If the confirmation or argument is Transmission Status (I_T Nexus Loss), then this state shall send a Transmission Complete (I_T Nexus Loss) message to the ST_TFR state machine. The Transmission Complete message shall include the tag as an argument.

If the confirmation or argument is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss), then this state shall send the Transmission Complete message defined in table 179 to the ST_TFR state machine. The message shall include the following arguments:

- a) tag; and
- b) arguments received with the Transmission Status confirmation.

If the confirmation is Transmission Status (Frame Transmitted) and the Transmit Frame request was for:

- a) an XFER_RDY frame; or
- b) a RESPONSE frame,

then this state shall wait to receive one of the following confirmations:

- a) Transmission Status (ACK Received);
- b) Transmission Status (NAK Received);
- c) Transmission Status (ACK/NAK Timeout); or
- d) Transmission Status (Connection Lost Without ACK/NAK).

If the confirmation is Transmission Status (Frame Transmitted) and the Transmit Frame request was for a read DATA frame, then this state shall:

- a) increment the Read Data Frames Transmitted state machine variable by one; and
- b) set the Read Data Offset state machine variable to the current Read Data Offset state machine variable plus the number of read data bytes transmitted in the DATA frame associated with the Transmission Status (Frame Transmitted) confirmation.

If the confirmation is Transmission Status (ACK Received) and the Transmit Frame request was for a read DATA frame, then this state shall increment the Read Data Frames ACKed state machine variable by one.

If the confirmation is Transmission Status (Frame Transmitted), the Transmit Frame request was for a read DATA frame, and the Read Data Offset state machine variable is equal to the Read Data Buffer End state machine variable, then this state shall wait to receive:

- a) Transmission Status (ACK Received) confirmations or arguments for each outstanding read DATA frame (i.e., Read Data Frames Transmitted state machine variable equals the Read Data Frames ACKed state machine variable); or
- b) one of the following:
 - A) Transmission Status (NAK Received);
 - B) Transmission Status (ACK/NAK Timeout); or
 - C) Transmission Status (Connection Lost Without ACK/NAK).

NOTE 86 - If the number of data bytes that have been transmitted for a Request (Send Data-In) message are fewer than the Request Byte Count Data-In state machine argument, then this state transitions to the ST_TTS3:Prepare_Data_In state to construct the additional read DATA frames for the request before receiving a Transmission Status (ACK Received), Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK) confirmation.

When the Read Data Frames Transmitted state machine variable equals the Read Data Frames ACKed state machine variable and the Transmit Frame request was for a read DATA frame, this state shall:

- a) not modify the Balance Point Read Data Offset state machine variable (i.e., the balance point remains at the last point at which balance occurred); or
- b) set the Balance Point Read Data Offset state machine variable to the current Read Data Offset state machine variable.

If the Transmit Frame request was for a RESPONSE frame, the vendor-specific number of retries has not been reached, and this state receives one of the following confirmations:

- a) Transmission Status (NAK Received);
- b) Transmission Status (ACK/NAK Timeout); or
- c) Transmission Status (Connection Lost Without ACK/NAK),

then this state shall:

- a) set the RETRANSMIT bit to one;
- b) set the other fields to the same values as contained in the failed RESPONSE frame; and
- c) resend a Transmit Frame (Interlocked) request to the port layer for the failed RESPONSE frame.

If transport layer retries are enabled, the Transmit Frame request was for a XFER_RDY frame, the vendor-specific number of retries has not been reached, and this state receives one of the following confirmations:

- a) Transmission Status (NAK Received);
- b) Transmission Status (ACK/NAK Timeout); or
- c) Transmission Status (Connection Lost Without ACK/NAK),

then this state shall:

- a) set the RETRANSMIT bit to one;
- b) set the TARGET PORT TRANSFER TAG field to a value that is different than the target port transfer tag in the previous XFER_RDY frame associated with the Data-Out state machine arguments and is different than any other target port transfer tag currently in use. If write data is received for a subsequent XFER_RDY frame for a command, then all target port transfer tags used for previous XFER_RDY frames for the command are no longer in use;
- c) set the other fields to the same values contained in the failed XFER_RDY frame; and
- d) resend a Transmit Frame (Interlocked) request to the port layer for the failed XFER_RDY frame.

Table 179 defines messages that this state shall send to the ST_TFR state machine upon receipt of the listed confirmations and arguments, based on the conditions under which each confirmation or argument was received.

Table 179 — Messages sent to the ST_TFR state machine

Confirmation received from the port layer or argument received from ST_TTS3:Prepare_Data_In	Conditions under which confirmation was received	Message sent to the ST_TFR state machine
Transmission Status (ACK Received)	The Transmit Frame request was for an XFER_RDY frame.	Transmission Complete (Xfer_Rdy Delivered) with a Target Port Transfer Tag argument
	Transmit Frame request was for a RESPONSE frame	Transmission Complete (Response Delivered)
	The Transmit Frame request was for a read DATA frame and: a) the Read Data Offset state machine variable is equal to the Read Data Buffer End state machine variable; and b) the Read Data Offset state machine variable is equal to the Balance Point Read Data Offset state machine variable.	Transmission Complete (Data-In Delivered)
Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK)	The Transmit Frame request was for a RESPONSE frame and the vendor-specific number of retries has been reached.	Transmission Complete (Response Failed)
Transmission Status (NAK Received)	The Transmit Frame request was for an XFER_RDY frame and: a) if transport layer retries are disabled; or b) if transport layer retries are enabled and the vendor-specific number of retries has been reached.	Transmission Complete (Xfer_Rdy Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)		Transmission Complete (Xfer_Rdy Failed, Connection Failed)
Transmission Status (NAK Received)	The Transmit Frame request was for a read DATA frame and: a) if transport layer retries are disabled; or b) if transport layer retries are enabled and the vendor-specific number of retries has been reached.	Transmission Complete (Data-In Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)		Transmission Complete (Data-In Failed, Connection Failed)

Table 180 defines messages that this state shall send to the ST_TFR state machine upon receipt of the listed confirmations and arguments.

Table 180 — Additional messages sent to the ST_TFR state machine

Confirmation received from the port layer or argument received from ST_TTS3:Prepare_Data_In	Message sent to the ST_TFR state machine
Transmission Status (Bad Destination)	Transmission Complete (Connection Failed)
Transmission Status (Connection Rate Not Supported)	Transmission Complete (Connection Failed)
Transmission Status (Protocol Not Supported)	Transmission Complete (Connection Failed)
Transmission Status (Reserved Abandon 1)	Transmission Complete (Connection Failed)
Transmission Status (Reserved Abandon 2)	Transmission Complete (Connection Failed)
Transmission Status (Reserved Abandon 3)	Transmission Complete (Connection Failed)
Transmission Status (STP Resources Busy)	Transmission Complete (Connection Failed)
Transmission Status (Wrong Destination)	Transmission Complete (Connection Failed)
Transmission Status (Zone Violation)	Transmission Complete (Connection Failed)
Transmission Status (Break Received)	Transmission Complete (Data Transfer Terminated)

If this state receives a Cancel message or a Cancel argument and this state has received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Transmission Complete (Data Transfer Terminated) message to the ST_TFR state machine.

If this state receives a Cancel message or a Cancel argument and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to the port layer to cancel previous Transmit Frame requests. The Cancel request shall include the following arguments:

- a) destination SAS address; and
- b) tag.

Upon receipt of a Transmission Status (Cancel Acknowledged) confirmation or argument this state shall send a Transmission Complete (Data Transfer Terminated) message to the ST_TFR state machine.

A Transmission Complete message to the ST_TFR state machine shall include the following arguments:

- a) destination SAS address; and
- b) tag.

9.2.6.3.3.2 Transition ST_TTS2:Target_Send_Frame to ST_TTS1:Target_Start

This transition shall occur after this state sends a Transmission Complete message other than Transmission Complete (Xfer_Rdy Delivered) to the ST_TFR state machine.

9.2.6.3.3.3 Transition ST_TTS2:Target_Send_Frame to ST_TTS3:Prepare_Data_In

This transition shall occur after this state receives a Transmission Status (Frame Transmitted) confirmation for a read DATA frame if the Read Data Offset state machine variable is less than the Read Data Buffer End state machine variable (i.e., there is more read data to transfer).

If transport layer retries are enabled and the vendor-specific number of retries, if any, for the read DATA frame has not been reached, this transition shall occur and include a Retry argument after this state receives one of the following confirmations for a read DATA frame:

- a) Transmission Status (NAK Received);
- b) Transmission Status (ACK/NAK Timeout); or

- c) Transmission Status (Connection Lost Without ACK/NAK).

9.2.6.3.3.4 Transition ST_TTS2:Target_Send_Frame to ST_TTS5:Receive_Data_Out

This transition shall occur after this state sends a Transmission Complete (Xfer_Rdy Delivered) message to the ST_TFR state machine.

9.2.6.3.3.4 ST_TTS3:Prepare_Data_In state

9.2.6.3.3.4.1 State description

This state retrieves the data from the Device Server Buffer Data-In state machine argument and constructs a read DATA frame.

This state shall construct a read DATA frame using the Data-In state machine arguments as follows:

- a) FRAME TYPE field set to 01h (i.e., DATA frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Data-In state machine argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
- d) RETRY DATA FRAMES bit set to zero;
- e) RETRANSMIT bit set to zero;
- f) CHANGING DATA POINTER set as specified in this subclause;
- g) NUMBER OF FILL BYTES field set to the number of fill bytes needed for the specified read data;
- h) TAG field set to the Tag Data-In state machine argument;
- i) TARGET PORT TRANSFER TAG field set to a vendor-specific value;
- j) DATA OFFSET field set as specified in this subclause;
- k) in the information unit, DATA field set as specified in this subclause; and
- l) fill bytes, if required.

If this state is entered without a Retry argument then this state shall:

- a) set the CHANGING DATA POINTER bit set to zero;
- b) set the DATA OFFSET field to the Read Data Offset state machine variable; and
- c) in the information unit, set the DATA field to the information in the Device Server Buffer argument that corresponds to the read data to be transferred. If the Read Data Buffer End state machine variable minus the Read Data Offset state machine variable is equal to the maximum size of the read Data information unit, the amount of data shall be the maximum size of the read Data information unit. Otherwise, the amount of data shall be the lesser of:
 - A) the Read Data Buffer End state machine variable minus the Read Data Offset state machine variable; and
 - B) the maximum size of the read Data information unit for this Data-In request.

If this state is entered with a Retry argument then this state shall either:

- a) set the CHANGING DATA POINTER bit in the frame to one;
- b) set the DATA OFFSET field to the Balance Point Read Data Offset state machine variable;
- c) set the Read Data Offset state machine variable to the Balance Point Read Data Offset state machine variable;
- d) set the Read Data Frames Transmitted state machine variable to zero;
- e) set the Read Data Frames ACKed state machine variable to zero; and
- f) in the information unit, set the DATA field to the information in the Device Server Buffer argument that corresponds to the read data to be transferred. If the Read Data Buffer End state machine variable minus the Read Data Offset state machine variable is equal to the maximum size of the read Data information unit, the amount of data shall be the maximum size of the read Data information unit. Otherwise, the amount of data shall be the lesser of:
 - A) the Read Data Buffer End state machine variable minus the Balance Point Read Data Offset state machine variable; and
 - B) the maximum size of the read Data information unit for this Data-In request;

or:

- a) set the CHANGING DATA POINTER bit in the frame to one;

- b) set the DATA OFFSET field to the Application Client Buffer Offset Data-In state machine argument;
- c) set the Read Data Offset state machine variable to the Application Client Buffer Offset Data-In state machine argument;
- d) set the Read Data Frames Transmitted state machine variable to zero;
- e) set the Read Data Frames ACKed state machine variable to zero; and
- f) in the information unit, set the DATA field to the information in the Device Server Buffer argument that corresponds to the read data to be transferred. If the Request Byte Count Data-In state machine argument is equal to the maximum size of the read Data information unit, the amount of data shall be the maximum size of the read Data information unit. Otherwise, the amount of data shall be the lesser of:
 - A) the Request Byte Count Data-In state machine argument; and
 - B) the maximum size of the read Data information unit for this Data-In request.

9.2.6.3.3.4.2 Transition ST_TTS3:Prepare_Data_In to ST_TTS2:Target_Send_Frame

This transition shall occur after this state:

- a) constructs a read DATA frame; or
- b) receives a Cancel message.

This transition shall include the received Transmission Status, if any, as an argument and the:

- a) read DATA frame as an argument; or
- b) if a Cancel message was received, then a Cancel argument.

9.2.6.3.3.5 ST_TTS4:Prepare_Xfer_Rdy state

9.2.6.3.3.5.1 State description

This state shall construct an XFER_RDY frame using the Data-Out state machine arguments:

- a) FRAME TYPE field set to 05h (i.e., XFER_RDY frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Data-Out state machine argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
- d) RETRY DATA FRAMES bit set to one if transport layer retries are enabled and zero if transport layer retries are disabled;
- e) RETRANSMIT bit set to zero;
- f) CHANGING DATA POINTER bit set to zero;
- g) NUMBER OF FILL BYTES field set to zero;
- h) TAG field set to the Tag Data-Out state machine argument;
- i) if transport layer retries are disabled, TARGET PORT TRANSFER TAG field set to a vendor-specific value;
- j) if transport layer retries are enabled, TARGET PORT TRANSFER TAG field set to a vendor-specific value that is different from:
 - A) the target port transfer tag in the previous XFER_RDY frame associated with the Data-Out state machine arguments; and
 - B) any other target port transfer tag currently in use.

If write data is received for a subsequent XFER_RDY frame for a command, then all target port transfer tags used for previous XFER_RDY frames for the command are no longer in use;

- k) DATA OFFSET field set to zero; and
- l) in the information unit, REQUESTED OFFSET field set to the Requested Write Data Offset state machine variable;
- m) in the information unit, WRITE DATA LENGTH field set as specified in this subclause; and
- n) no fill bytes.

If the SSP target port has the resources available to receive all of the write data as indicated by the Requested Write Data Length state machine variable, then this state shall set the WRITE DATA LENGTH field in the XFER_RDY information unit to the Requested Write Data Length state machine variable.

If the SSP target port does not have the resources available to receive all of the write data as indicated by the Requested Write Data Length state machine variable (e.g., the SSP target port has a vendor specific limit as to how much write data may be received during one operation), then this state shall set the WRITE DATA LENGTH field in the XFER_RDY information unit and the Requested Write Data Length state machine variable to a value representing the amount of write data for which the SSP target port has available resources to receive.

9.2.6.3.3.5.2 Transition ST_TTS4:Prepare_Xfer_Rdy to ST_TTS2:Target_Send_Frame

This transition shall occur after this state:

- a) constructs an XFER_RDY frame; or
- b) receives a Cancel message.

This transition shall include the:

- a) if a Cancel message was received, then a Cancel argument; or
- b) XFER_RDY frame as an argument.

9.2.6.3.3.6 ST_TTS5:Receive_Data_Out state

9.2.6.3.3.6.1 State description

Upon entry into this state, the Write Data Offset state machine variable is set to the Requested Write Data Offset state machine variable.

If this state receives a Data-Out Arrived message, then this state shall verify the write DATA frame received with the Data-Out Arrived values as specified in table 181. If the verification test fails, then this state sends the message specified in table 181 to the ST_TFR state machine.

Table 181 — Reception Complete message for write DATA frame verification failures

Message sent to ST_TFR ^a	Condition
Reception Complete (Data Offset Error)	Transport layer retries are disabled, and the DATA OFFSET field is not equal to the Write Data Offset state machine variable.
	The DATA OFFSET field is: a) less than the Requested Write Data Offset state machine variable; or b) greater than or equal to the Requested Write Data Offset state machine variable plus the Requested Write Data Length state machine variable.
Reception Complete (Too Much Write Data)	The number of bytes in the DATA field in the write Data information unit plus the Write Data Offset state machine variable is greater than the Request Byte Count Data-Out state machine argument.
Reception Complete (Information Unit Too Short)	The number of bytes in the DATA field is zero.
^a If more than one condition is true, then this state shall select which message to send to the ST_TFR state machine using the following order: 1) Reception Complete (Data Offset Error); 2) Reception Complete (Too Much Write Data); or 3) Reception Complete (Information Unit Too Short).	

If:

- a) transport layer retries are enabled;
- b) the CHANGING DATA POINTER bit is set to zero; and
- c) the value in the DATA OFFSET field is not equal to the Write Data Offset state machine variable,

then this state should discard all Data-Out Arrived messages until the CHANGING DATA POINTER bit is set to one. This state shall resume processing additional Data-Out Arrived messages when it receives a Data-Out Arrived message with the CHANGING DATA POINTER bit set to one.

If the WRITE data frame verification is successful and the Data-Out Arrived message is not discarded, then this state shall:

- a) process the write data as indicated in the Data-Out state machine arguments using the Device Server Buffer (e.g., logical block address) to which the write data is to be transferred; and
- b) set the Write Data Offset state machine variable to the current Write Data Offset state machine variable plus the number of bytes received in the DATA field of the write Data information unit.

If the WRITE data frame verification is successful and the CHANGING DATA POINTER bit set to one, then this state shall:

- a) set the Write Data Offset state machine variable to the Requested Write Data Offset state machine variable plus the number of bytes received in the DATA field of the write Data information unit; and
- b) process the write data as indicated in the Data-Out state machine arguments using the Device Server Buffer (e.g., logical block address) to which the write data is to be transferred.

If data received in the write DATA frame overlaps data previously received and verified successfully, this state may either discard the overlapping data, or replace the previously received data with the new data.

If the Initiator Response Timeout timer is implemented, then this state shall initialize and start the Initiator Response Timeout timer:

- a) upon entry into this state; and
- b) when this state receives and verifies the write DATA frame received with the Data-Out Arrived values (i.e., Data-Out data was received and processed).

If the Initiator Response Timeout timer is running, then this state shall stop the timer before transitioning from this state.

If the Initiator Response Timeout timer expires, then this state shall send a Reception Complete (Initiator Response Timeout) message to the ST_TFR state machine.

If the Write Data Offset state machine variable equals the Request Byte Count Data-Out state machine argument plus the Application Client Buffer Offset Data-Out state machine argument, then this state shall send a Reception Complete (Data-Out Received) message to the ST_TFR state machine after a Reception Complete (ACK Transmitted) confirmation is received for each write DATA frame previously received.

If this state receives a Cancel message, then this state shall send a Reception Complete (Data Transfer Terminated) message to the ST_TFR state machine.

If this state receives Transmission Status (Break Received) confirmation, then this state shall send a Reception Complete (Data Transfer Terminated) to the ST_TFR state machine.

The Reception Complete message, if any, shall include the tag as an argument.

9.2.6.3.3.6.2 Transition ST_TTS5:Receive_Data_Out to ST_TTS1:Target_Start

This transition shall occur after this state sends a Reception Complete message to the ST_TFR state machine.

9.2.6.3.3.6.3 Transition ST_TTS5:Receive_Data_Out to ST_TTS4:Prepare_Xfer_Rdy

This transition shall occur:

- 1) if the Write Data Offset state machine variable is less than Request Byte Count Data-Out state machine argument plus the Application Client Buffer Offset Data-Out state machine argument and equal to the Requested Write Data Offset state machine variable plus the Requested Write Data Length state machine variable;
- 2) after a Reception Complete (ACK Transmitted) confirmation is received for each write DATA frame previously received;

- 3) after determining the amount of write data already transferred by subtracting the Application Client Buffer Offset Data-Out state machine argument from the Write Data Offset state machine variable;
- 4) after setting the Requested Write Data Length state machine variable to the Request Byte Count Data-Out state machine argument minus the amount of write data already transferred; and
- 5) after setting the Requested Write Data Offset state machine variable to the Write Data Offset state machine variable.

9.2.6.3.3.7 ST_TTS6:Prepare_Response state

9.2.6.3.3.7.1 State description

This state shall construct a RESPONSE frame using the received Application Response arguments or the received Transport Response arguments as follows:

- a) FRAME TYPE field set to 07h (i.e., RESPONSE frame);
- b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Application Response or Transport Response destination SAS address argument;
- c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
- d) RETRY DATA FRAMES bit set to zero;
- e) RETRANSMIT bit set to zero;
- f) CHANGING DATA POINTER bit set to zero;
- g) TAG field set to the Tag Application Response argument or the Tag Transport Response argument;
- h) TARGET PORT TRANSFER TAG field set to a vendor-specific value;
- i) DATA OFFSET field set to zero;
- j) information unit set as specified in this subclause; and
- k) fill bytes, if needed as specified in this subclause.

If this state was entered with the Transport Response arguments, then this state shall set the fields as follows:

- a) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the response data, if any;
- b) in the information unit, set the DATAPRES field to RESPONSE DATA;
- c) in the information unit, set the STATUS field to zero;
- d) in the information unit, set the SENSE DATA LENGTH field to zero;
- e) in the information unit, set the RESPONSE DATA LENGTH field to 00000004h;
- f) in the information unit, set the RESPONSE DATA field as specified in table 182;
- g) in the information unit, do not include the SENSE DATA field; and
- h) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the response data, if needed.

Table 182 defines how the RESPONSE DATA field shall be set based on the arguments received with the Request (Send Transport Response) message.

Table 182 — Request argument to RESPONSE frame response data field mapping

Request argument	RESPONSE frame response data field
Invalid Frame	INVALID FRAME
Function Complete	TASK MANAGEMENT FUNCTION COMPLETE
Function Succeeded	TASK MANAGEMENT FUNCTION SUCCEEDED
Function Not Supported	TASK MANAGEMENT FUNCTION NOT SUPPORTED
Function Failed	TASK MANAGEMENT FUNCTION FAILED
Incorrect Logical Unit Number	INCORRECT LOGICAL UNIT NUMBER
Overlapped Tag Attempted	OVERLAPPED TAG ATTEMPTED

If this state was entered with the Application Response arguments, then this state shall set the fields as follows:

- a) in the information unit, set the DATAPRES field to SENSE_DATA if sense data is to be included in the information unit or NO_DATA if sense data is not to be included in the information unit;
- b) in the information unit, set the STATUS field to the status;
- c) in the information unit, set the SENSE DATA LENGTH field to the length of the sense data, if any;
- d) in the information unit, set the RESPONSE DATA LENGTH field to zero;
- e) in the information unit, do not include the RESPONSE DATA field;
- f) in the information unit, set the SENSE DATA field to the sense data, if any; and
- g) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the sense data, if needed.

9.2.6.3.3.7.2 Transition ST_TTS6:Prepare_Response to ST_TTS2:Target_Send_Frame

This transition shall occur after this state constructs a RESPONSE frame.

This transition shall include:

- a) the RESPONSE frame; or
- b) if a Cancel message was received, then a Cancel argument.

9.3 STP transport layer

9.3.1 Initial FIS

A SATA device phy transmits a Register - Device to Host FIS after completing the link reset sequence (see G.5 for exceptions to this). The expander device shall update a set of shadow registers with the contents of this FIS and shall not deliver it to any STP initiator port. SMP initiator ports may read the shadow register contents using the SMP REPORT PHY SATA function (see 10.4.3.11). The expander device originates a Broadcast (Change) after receiving the Register - Device to Host FIS (see 7.11).

9.3.2 BIST Activate FIS

STP initiator ports and STP target ports shall not generate BIST Activate FISes and shall process any BIST Activate FISes received as frames having invalid FIS types (i.e., have the link layer generate SATA_R_ERR in response).

9.3.3 TT (transport layer for STP ports) state machines

The STP transport layer uses the transport layer state machines defined in SATA, modified to communicate with the port layer rather than directly with the link layer. These modifications are not described in this standard.

9.4 SMP transport layer

9.4.1 SMP transport layer overview

Table 183 defines the SMP frame format.

Table 183 — SMP frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE							
1	Frame-type dependent bytes							
	Fill bytes, if needed							
n - 3	(MSB) CRC (LSB)							
n								

Table 184 defines the SMP FRAME TYPE field, which defines the format of the frame-type dependent bytes.

Table 184 — SMP FRAME TYPE field

Code	Name	Frame type	Originator	Reference
40h	SMP_REQUEST	SMP function request	SMP initiator port	9.4.2
41h	SMP_RESPONSE	SMP function response	SMP target port	9.4.3
All others	Reserved.			

The SMP target port in an expander device shall support the SMP_REQUEST and SMP_RESPONSE frames. Other SMP target ports may support these frames.

Fill bytes shall be included after the frame-type dependent bytes so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field contains a CRC value (see 7.5) that is computed over the entire SMP frame prior to the CRC field, and shall begin on a four-byte boundary. The CRC field is checked by the SMP link layer (see 7.18).

9.4.2 SMP_REQUEST frame

The SMP_REQUEST frame is sent by an SMP initiator port to request an SMP function be performed by a management device server. Table 185 defines the SMP_REQUEST frame format.

Table 185 — SMP_REQUEST frame format

Byte/Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	REQUEST BYTES							
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

The SMP FRAME TYPE field shall be set to 40h indicating this is an SMP_REQUEST frame.

The REQUEST BYTES field definition and length is based on the SMP function (see 10.4.3.1). The maximum size of the REQUEST BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL REQUEST BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is defined in 9.4.1.

9.4.3 SMP_RESPONSE frame

The SMP_RESPONSE frame is sent by an SMP target port in response to an SMP_REQUEST frame. Table 186 defines the SMP_RESPONSE frame format.

Table 186 — SMP_RESPONSE frame format

Byte/Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	RESPONSE BYTES							
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

The SMP FRAME TYPE field shall be set to 41h indicating this is an SMP_RESPONSE frame.

The RESPONSE BYTES field definition and length is based on the SMP function (see 10.4.3.2). The maximum size of the RESPONSE BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL RESPONSE BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is defined in 9.4.1.

9.4.4 Sequence of SMP frames

Inside an SMP connection, the source phy transmits a single SMP_REQUEST frame and the destination phy replies with a single SMP_RESPONSE frame.

Figure 205 shows the sequence of SMP frames.

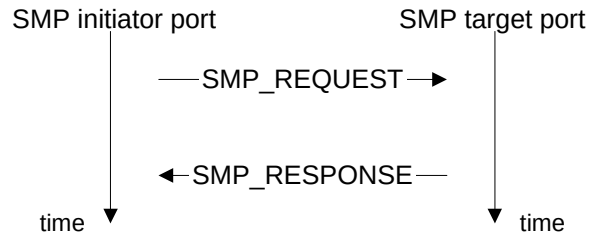


Figure 205 — Sequence of SMP frames

9.4.5 MT (transport layer for SMP ports) state machines

9.4.5.1 SMP transport layer state machines overview

The SMP transport layer contains state machines that process requests from the management application layer and returns confirmations to the management application layer. The SMP transport state machines are as follows:

- a) MT_IP (transport layer for SMP initiator ports) state machine (see 9.4.5.2); and
- b) MT_TP (transport layer for SMP target ports) state machine (see 9.4.5.3).

9.4.5.2 MT_IP (transport layer for SMP initiator ports) state machine

9.4.5.2.1 MT_IP state machine overview

The MT_IP state machine processes requests from the management application layer. These management requests are sent to the port layer and the resulting SMP frame or error condition is sent to the management application layer as a confirmation.

This state machine consists of the following states:

- a) MT_IP1:Idle (see 9.4.5.2.2)(initial state);
- b) MT_IP2:Send (see 9.4.5.2.3); and
- c) MT_IP3:Receive (see 9.4.5.2.4).

This state machine shall start in the MT_IP1:Idle state.

Figure 206 describes the MT_IP state machine.

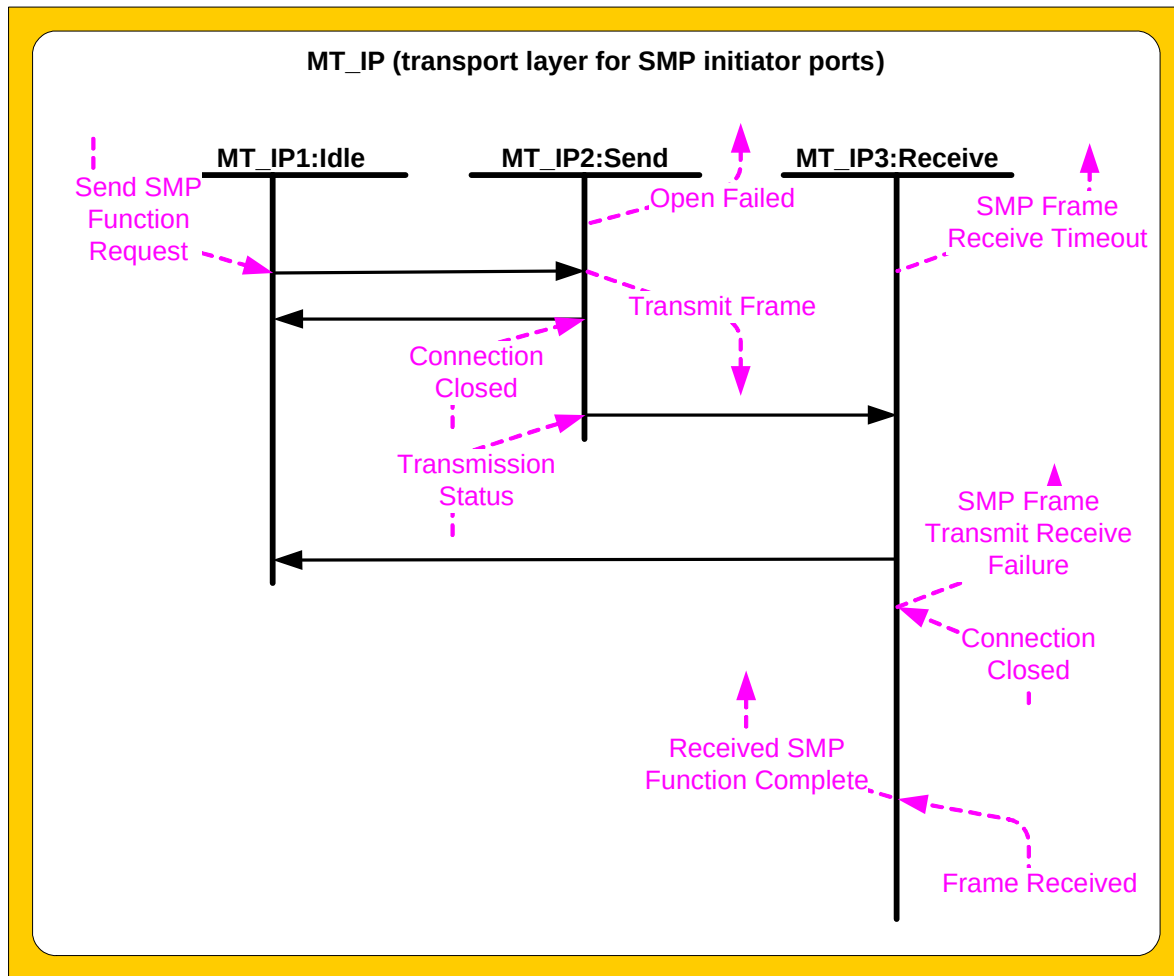


Figure 206 — MT_IP (transport layer for SMP initiator ports) state machine

9.4.5.2.2 MT_IP1:Idle state

9.4.5.2.2.1 State description

This state is the initial state of the MT_IP state machine.

This state waits for a Send SMP Function Request request, which includes the following arguments:

- connection rate;
- destination SAS address; and
- request bytes.

9.4.5.2.2.2 Transition MT_IP1:Idle to MT_IP2:Send

This transition shall occur after a Send SMP Function Request request is received. This transition shall include the following arguments:

- connection rate;
- destination SAS address; and
- request bytes.

9.4.5.2.3 MT_IP2:Send state**9.4.5.2.3.1 State description**

This state constructs an SMP_REQUEST frame using the following arguments received in the transition into this state:

- a) request bytes;

and sends a Transmit Frame request to the port layer with the following arguments:

- a) initiator port bit set to one;
- b) protocol set to SMP;
- c) connection rate;
- d) initiator connection tag set to FFFFh;
- e) destination SAS address;
- f) source SAS address set to the SAS address of the SMP initiator port; and
- g) request bytes.

9.4.5.2.3.2 Transition MT_IP2:Send to MT_IP1:Idle

This transition shall occur after receiving either a Connection Closed confirmation or a Transmission Status confirmation other than a Transmission Status (Frame Transmitted) confirmation, and after sending an Open Failed confirmation to the management application layer.

9.4.5.2.3.3 Transition MT_IP2:Send to MT_IP3:Receive

This transition shall occur after receiving a Transmission Status (Frame Transmitted) confirmation.

9.4.5.2.4 MT_IP3:Receive state**9.4.5.2.4.1 State description**

This state waits for a confirmation from the port layer that either an SMP frame has been received or a failure occurred.

If a Frame Received confirmation is received and the SMP frame type is equal to 41h, this state shall send a Received SMP Function Complete confirmation to the management application layer.

If a Frame Received confirmation is received and the SMP frame type is not equal to 41h, this state shall send a SMP Frame Transmit Receive Failure confirmation to the management application layer.

If a Connection Closed or Frame Received (SMP Failure) confirmation is received, this state shall send an SMP Frame Transmit Receive Failure confirmation to the management application layer.

9.4.5.2.4.2 Transition MT_IP3:Receive to MT_IP1:Idle

This transition shall occur after one of the following:

- a) sending a Received SMP Function Complete confirmation; or
- b) sending an SMP Frame Transmit Receive Failure confirmation.

9.4.5.3 MT_TP (transport layer for SMP target ports) state machine**9.4.5.3.1 MT_TP state machine overview**

The MT_TP state machine informs the management application layer of the receipt of an SMP frame. Confirmation of the receipt of an SMP frame is sent to the management application layer. The management application layer creates the corresponding SMP_RESPONSE frame and this state sends it to the port layer.

This state machine consists of the following states:

- a) MT_TP1:Idle (see 9.4.5.3.2)(initial state); and
- b) MT_TP2:Respond (see 9.4.5.3.3).

This state machine shall start in the MT_TP1:Idle state.

Figure 207 describes the MT_TP state machine.

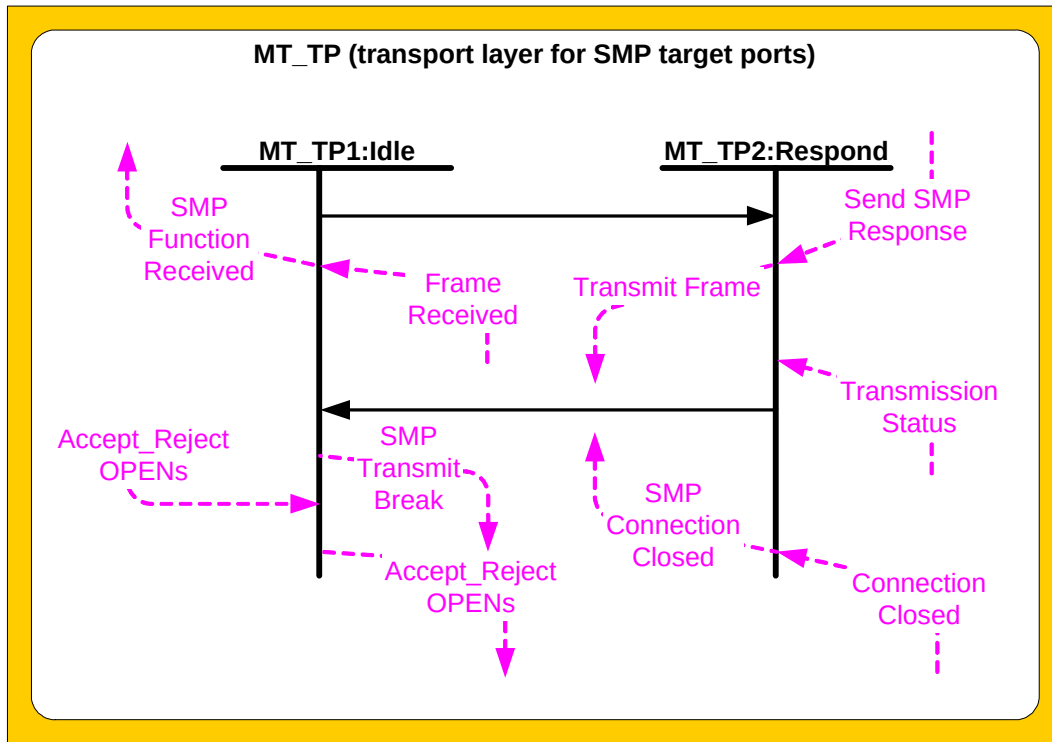


Figure 207 — MT_TP (transport layer for SMP target ports) state machine

The MT_TP state machine shall comply with the time limits listed in table 187.

Table 187 — MT_TP time limits

Time limit	Value	Description
SMP Response time limit	1 900 μs	Maximum time from receiving an SMP_REQUEST frame to transmitting an SMP_RESPONSE frame

9.4.5.3.2 MT_TP1:Idle state

9.4.5.3.2.1 State description

This state is the initial state of the MT_TP state machine.

This state waits for a Frame Received confirmation. If the SMP frame type is not equal to 40h, this state shall discard the frame and send a SMP Transmit Break request to the port layer. Otherwise, this state shall send an SMP Function Received confirmation to the management application layer.

If an Accept_Reject OPENS (Accept SMP) or Accept_Reject OPENS (Reject SMP) request is received, this state shall send an Accept_Reject OPENS request with the same arguments to the port layer.

9.4.5.3.2.2 Transition MT_TP1:Idle to MT_TP2:Respond

This transition shall occur after sending an SMP Function Received confirmation.

9.4.5.3.3 MT_TP2:Respond state**9.4.5.3.3.1 State description**

This state waits for a Send SMP Response request, which includes the following arguments:

- a) response bytes.

After receiving a Send SMP Response request, this state shall construct an SMP_RESPONSE frame using the arguments from the Send SMP Response request and send a Transmit Frame request to the port layer within the SMP Response time limit specified in table 187 (see 9.4.5.3.1).

If this state receives a Connection Closed confirmation, this state shall send an SMP Connection Closed confirmation to the management application layer.

9.4.5.3.3.2 Transition MT_TP2:Respond to MT_TP1:Idle

This transition shall occur after one of the following:

- a) receiving a Transmission Status (Frame Transmitted) confirmation; or
- b) sending an SMP Connection Closed confirmation.

10 Application layer

10.1 Application layer overview

The application layer defines SCSI, ATA, and management specific features.

10.2 SCSI application layer

10.2.1 SCSI transport protocol services

10.2.1.1 SCSI transport protocol services overview

An application client requests the processing of a SCSI command by invoking SCSI transport protocol services, the collective operation of which is conceptually modeled in the following remote procedure call (see SAM-4):

Service response = Execute Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Task Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status))

An application client requests the processing of a SCSI task management function by invoking SCSI transport protocol services, the collective operation of which is conceptually modeled in the following remote procedure calls (see SAM-4):

- a) Service Response = ABORT TASK (IN (Nexus));
- b) Service Response = ABORT TASK SET (IN (Nexus));
- c) Service Response = CLEAR ACA (IN (Nexus));
- d) Service Response = CLEAR TASK SET (IN (Nexus));
- e) Service Response = I_T NEXUS RESET (IN (Nexus));
- f) Service Response = LOGICAL UNIT RESET (IN (Nexus));
- g) Service Response = QUERY TASK (IN (Nexus));
- h) Service Response = QUERY TASK SET (IN (Nexus)); and
- i) Service Response = QUERY UNIT ATTENTION (IN (Nexus), OUT([Additional Response Information])).

SSP defines the transport protocol services required by SAM-4 in support of these remote procedure calls.

Table 188 describes the mapping of the remote procedure calls to transport protocol services and the SSP implementation of each transport protocol service.

Table 188 — SCSI architecture mapping

Remote procedure call	Type of transport protocol service	Transport protocol service interaction	Transport protocol service	I/T ^a	SSP implementation
Execute Command	Request/ Confirmation	Request	Send SCSI Command	I	COMMAND frame
		Indication	SCSI Command Received	T	Receipt of the COMMAND frame
		Response	Send Command Complete	T	RESPONSE frame
		Confirmation	Command Complete Received	I	Receipt of the RESPONSE frame or problem transmitting the COMMAND frame
	Data-In Transfer ^b	Request	Send Data-In	T	Read DATA frames
		Confirmation	Data-In Delivered	T	Receipt of ACKs for the read DATA frames
	Data-Out Transfer ^b	Request	Receive Data-Out	T	XFER_RDY frame
		Confirmation	Data-Out Received	T	Receipt of write DATA frames
	Terminate Data Transfer ^b	Request	Terminate Data Transfer	T	
		Confirmation	Data Transfer Terminated	T	
ABORT TASK, ABORT TASK SET, CLEAR ACA, CLEAR TASK SET, I_T NEXUS RESET, LOGICAL UNIT RESET, QUERY TASK, QUERY TASK SET, and QUERY UNIT ATTENTION	Request/ Confirmation	Request	Send Task Management Request	I	TASK frame
		Indication	Task Management Request Received	T	Receipt of the TASK frame
		Response	Task Management Function Executed	T	RESPONSE frame
		Confirmation	Received Task Management Function Executed	I	Receipt of the RESPONSE frame or problem transmitting the TASK frame
^a I/T indicates whether the SSP initiator port (I) or the SSP target port (T) implements the transport protocol service.					
^b Data transfer transport protocol services for SCSI initiator ports are not specified by SAM-4.					

These protocol services are used as the requests and confirmations to the SSP transport layer state machines (see 9.2.6) from the SCSI application layer.

10.2.1.2 Send SCSI Command transport protocol service

An application client uses the Send SCSI Command transport protocol service request to request that an SSP initiator port transmit a COMMAND frame.

Send SCSI Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Task Priority], [Command Reference Number], [First Burst Enabled], [Request Fence]))

Table 189 shows how the arguments to the Send SCSI Command transport protocol service are used.

Table 189 — Send SCSI Command transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to send the COMMAND frame; b) T specifies the target port to which the COMMAND frame is to be sent; c) L specifies the LOGICAL UNIT NUMBER field in the COMMAND frame header; and d) Q specifies the TAG field in the COMMAND frame header.
CDB	Specifies the CDB field in the COMMAND frame.
Task Attribute	Specifies the TASK ATTRIBUTE field in the COMMAND frame.
[Data-In Buffer Size]	Maximum of 2^{32}
[Data-Out Buffer]	Internal to the SSP initiator port.
[Data-Out Buffer Size]	Maximum of 2^{32}
[Task Priority]	Specifies the TASK PRIORITY field in the COMMAND frame.
[First Burst Enabled]	Specifies the ENABLE FIRST BURST field in the COMMAND frame and to cause the SSP initiator port to transmit the number of bytes indicated by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2.5) for the SCSI target port without waiting for an XFER_RDY frame.
[Request Fence]	If included, specifies an I_T nexus, I_T_L nexus, or I_T_L_Q nexus for which the COMMAND frame is fenced.

The application client shall set the Request Fence argument to the nexus containing any commands or task management functions that the command affects (e.g., for a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action, set the Response Fence argument to the I_T_L nexus) or upon which the command depends (e.g., when the Task Attribute argument is set to ORDERED, set the Response Fence argument to the I_T_L_Q nexus of the previous command). If the application client is not able to determine the nexus affected by the command or upon which the command depends, then it should set the Request Fence argument to the I_T nexus.

10.2.1.3 SCSI Command Received transport protocol service

An SSP target port uses the SCSI Command Received transport protocol service indication to notify a device server that it has received a COMMAND frame.

SCSI Command Received (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Task Priority], [Command Reference Number], [First Burst Enabled]))

Table 190 shows how the arguments to the SCSI Command Received transport protocol service are determined.

Table 190 — SCSI Command Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that sent the COMMAND frame; b) T indicates the target port that received the COMMAND frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the COMMAND frame header; and d) Q indicates the value of the TAG field in the COMMAND frame header.
CDB	Indicates the value of the CDB field in the COMMAND frame.
Task Attribute	Indicates the value of the TASK ATTRIBUTE field in the COMMAND frame.
[Task Priority]	Indicates the value of the TASK PRIORITY field in the COMMAND frame.
[Command Reference Number]	Ignored
[First Burst Enabled]	Indicates that first burst data is being delivered based on the ENABLE FIRST BURST field in the COMMAND frame and the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.2.5).

10.2.1.4 Send Command Complete transport protocol service

A device server uses the Send Command Complete transport protocol service response to request that an SSP target port transmit a RESPONSE frame.

Send Command Complete (IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, Service Response, [Response Fence]))

A device server shall only call Send Command Complete () after receiving SCSI Command Received ().

A device server shall not call Send Command Complete () for a given I_T_L_Q nexus until all its outstanding Receive Data-Out () calls have been responded to with Data-Out Received () and all its outstanding Send Data-In () calls have been responded to with Data-In Delivered ().

Table 191 shows how the arguments to the Send Command Complete transport protocol service are used.

Table 191 — Send Command Complete transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the RESPONSE frame is to be sent; b) T specifies the target port to send the RESPONSE frame; c) L specifies the LOGICAL UNIT NUMBER field in the RESPONSE frame header; and d) Q specifies the TAG field in the RESPONSE frame header.
[Sense Data]	Specifies the SENSE DATA field in the RESPONSE frame.
[Sense Data Length]	Specifies the SENSE DATA LENGTH field in the RESPONSE frame.
Status	Specifies the STATUS field in the RESPONSE frame.
Service Response	Specifies the DATAPRES field and STATUS field in the RESPONSE frame: a) TASK COMPLETE: The DATAPRES field is set to NO_DATA or SENSE_DATA; or b) SERVICE DELIVERY OR TARGET FAILURE: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to INVALID FRAME or OVERLAPPED TAG ATTEMPTED.
[Response Fence]	If included, specifies an I_T nexus, I_T_L nexus, or I_T_L_Q nexus for which the RESPONSE frame is fenced.

The device server shall set the Response Fence argument to the nexus containing any commands or task management functions that the command affects (e.g., for a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action, set the Response Fence argument to the I_T_L nexus) or upon which the command completion depends (e.g., when returning a unit attention condition with an additional sense code set to COMMANDS CLEARED BY ANOTHER INITIATOR, set the Response Fence argument to the I_T_L nexus). If the device server is not able to determine the nexus affected by the command or upon which the command depends, then it should set the Response Fence argument to the I_T nexus.

10.2.1.5 Command Complete Received transport protocol service

An SSP initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received a response for its COMMAND frame (e.g., a RESPONSE frame or a NAK) or terminated a command because of an error.

Command Complete Received (IN (I_T_L_Q Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 192 shows how the arguments to the Command Complete Received transport protocol service are determined.

Table 192 — Command Complete Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that received the RESPONSE frame; b) T indicates the target port that sent the RESPONSE frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the RESPONSE frame header or COMMAND frame header; and d) Q indicates the value of the TAG field in the RESPONSE frame header or COMMAND frame header.
[Data-In Buffer]	Internal to the SSP initiator port.
[Sense Data]	Indicates the value of the SENSE DATA field in the RESPONSE frame.
[Sense Data Length]	The smaller of the value of the SENSE DATA LENGTH field in the RESPONSE frame and the actual number of sense data bytes received by the SSP initiator port.
Status	Indicates the value of the STATUS field in the RESPONSE frame.
Service Response	From the DATAPRES field and STATUS field in the RESPONSE frame, or from a NAK on the COMMAND frame, or from the ST_IFR state machine detecting an error: a) TASK COMPLETE: The RESPONSE frame contains a DATAPRES field set to NO_DATA or SENSE_DATA; or b) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to INVALID FRAME or OVERLAPPED TAG ATTEMPTED, or a NAK was received for the COMMAND frame, or the length of the RESPONSE frame is incorrect, or the ST_IFR state machine detects an error as described in 9.2.6.2.2.3 and 9.2.6.2.2.4.

Editor's Note 48: Added the catch-all ST_IFR reference in sas2r12. NAK on a COMMAND frame is one of many errors that ST_IFR detects. It should probably be deleted from the list (all the error cases won't fit here, and that one is not more important than the others).

10.2.1.6 Send Data-In transport protocol service

A device server uses the Send Data-In transport protocol service request to request that an SSP target port transmit a read DATA frame.

Send Data-In (IN (I_T_L_Q Nexus, Device Server Buffer, Application Client Buffer Offset, Request Byte Count))

A device server shall only call Send Data-In () during a read or bidirectional command.

A device server shall not call Send Data-In () for a given I_T_L_Q nexus after it has called Send Command Complete () for that I_T_L_Q nexus (e.g., a RESPONSE frame with for that I_T_L_Q nexus) or called Task Management Function Executed for a task management function that terminates that task (e.g., an ABORT TASK).

Table 193 shows how the arguments to the Send Data-In transport protocol service are used.

Table 193 — Send Data-In transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the read DATA frame is to be sent; b) T specifies the target port to send the read DATA frame; c) L specifies the LOGICAL UNIT NUMBER field in the read DATA frame header; and d) Q specifies the TAG field in the read DATA frame header.
Device Server Buffer	Internal to the device server.
Application Client Buffer Offset	Specifies the DATA OFFSET field in the read DATA frame.
Request Byte Count	Specifies the size of the read DATA frame.

10.2.1.7 Data-In Delivered transport protocol service

An SSP target port uses the Data-In Delivered transport protocol service indication to notify a device server of the results of transmitting a read DATA frame.

Data-In Delivered (IN (I_T_L_Q Nexus, Delivery Result))

Table 194 shows how the arguments to the Data-In Delivered transport protocol service are determined.

Table 194 — Data-In Delivered transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that sent the read DATA frame; b) T indicates the target port that received the read DATA frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the read DATA frame header; and d) Q indicates the value of the TAG field in the read DATA frame header.
Delivery Result	From the response to the outgoing read DATA frame: a) DELIVERY SUCCESSFUL: The read DATA frame received an ACK; or b) DELIVERY FAILURE: The read DATA frame received a NAK or no response.

10.2.1.8 Receive Data-Out transport protocol service

A device server uses the Receive Data-Out transport protocol service request to request that an SSP target port transmit an XFER_RDY frame.

Receive Data-Out (IN (I_T_L_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device Server Buffer))

A device server shall only call Receive Data-Out () during a write or bidirectional command.

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus until Data-Out Received () has completed successfully for the previous Receive Data-Out () call (i.e., no XFER_RDY frame until all write DATA frames for the previous XFER_RDY frame, if any, and has provided link layer acknowledgement for all of the previous write DATA frames for that I_T_L_Q nexus).

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus after a Send Command Complete () has been called for that I_T_L_Q nexus or after a Task Management Function Executed () has been called for a task management function that terminates that task (e.g., an ABORT TASK).

Table 195 shows how the arguments to the Receive Data-Out transport protocol service are used.

Table 195 — Receive Data-Out transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the XFER_RDY frame is to be sent; b) T specifies the target port to send the XFER_RDY frame; c) L specifies the LOGICAL UNIT NUMBER field in the XFER_RDY frame header; and d) Q specifies the TAG field in the XFER_RDY frame header.
Application Client Buffer Offset	Specifies the REQUESTED OFFSET field in the XFER_RDY frame.
Request Byte Count	Specifies WRITE DATA LENGTH field in the XFER_RDY frame.
Device Server Buffer	Internal to the device server.

10.2.1.9 Data-Out Received transport protocol service

An SSP target port uses the Data-Out Received transport protocol service indication to notify a device server of the result of transmitting an XFER_RDY frame (e.g., receiving write DATA frames in response).

Data-Out Received (IN (I_T_L_Q Nexus, Delivery Result))

Table 196 shows how the arguments to the Data-Out Received transport protocol service are determined.

Table 196 — Data-Out Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port to which the XFER_RDY frame was sent; b) T indicates the target port that sent the XFER_RDY frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the XFER_RDY frame header; and d) Q indicates the value of the TAG field in the XFER_RDY frame header.
Delivery Result	From the response to the XFER_RDY: a) DELIVERY SUCCESSFUL: The XFER_RDY frame was successfully transmitted and all the write DATA frames for the requested write data were received; or b) DELIVERY FAILURE: The XFER_RDY frame received a NAK or no response.

10.2.1.10 Terminate Data Transfer transport protocol service

A device server uses the Terminate Data Transfer transport protocol service request to request that an SSP target port terminate any Send Data-In () or Receive Data-Out () transport protocol services, if any, being processed using the specified nexus.

Terminate Data Transfer (IN (Nexus))

Table 197 shows how the arguments to the Terminate Data Transfer transport protocol service are used.

Table 197 — Terminate Data Transfer transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus, specifying the scope of the data transfer(s) to terminate.

10.2.1.11 Data Transfer Terminated transport protocol service

An SSP target port uses the Data Transfer Terminated transport protocol service indication to notify a device server that all data transfers for the indicated nexus have been terminated.

Data Transfer Terminated (IN (Nexus))

Table 198 shows how the arguments to the Data Transfer Terminated transport protocol service are determined.

Table 198 — Data Transfer Terminated transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus indicated by the preceding Terminate Data Transfer () call.

10.2.1.12 Send Task Management Request transport protocol service

An application client uses the Send Task Management Request transport protocol service request to request that an SSP initiator port transmit a TASK frame.

Send Task Management Request (IN (Nexus, Function Identifier, [Association], [Request Fence]))

Table 199 shows how the arguments to the Send Task Management Request transport protocol service are used.

Table 199 — Send Task Management Request transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the Function Identifier), where: a) I specifies the initiator port to send the TASK frame; b) T specifies the target port to which the TASK frame is sent; c) L specifies the LOGICAL UNIT NUMBER field in the TASK frame header; and d) Q (for an I_T_L_Q nexus) specifies the TAG OF TASK TO BE MANAGED field in the TASK frame header.
Function Identifier	Specifies the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK (Nexus argument specifies an I_T_L_Q Nexus); b) ABORT TASK SET (Nexus argument specifies an I_T_L Nexus); c) CLEAR ACA (Nexus argument specifies an I_T_L Nexus); d) CLEAR TASK SET (Nexus argument specifies an I_T_L Nexus); e) I_T NEXUS RESET (Nexus argument specifies an I_T Nexus); f) LOGICAL UNIT RESET (Nexus argument specifies an I_T_L Nexus); g) QUERY TASK (Nexus argument specifies an I_T_L_Q Nexus); h) QUERY TASK SET (Nexus argument specifies an I_T_L Nexus); and i) QUERY UNIT ATTENTION (Nexus argument specifies an I_T_L Nexus).
[Association]	Specifies the TAG field in the TASK frame header.
[Request Fence]	If included, specifies an I_T nexus, I_T_L nexus, or I_T_L_Q nexus for which the TASK frame is fenced.

The application client shall set the Request Fence argument to the Nexus argument.

10.2.1.13 Task Management Request Received transport protocol service

An SSP target port uses the Task Management Request Received transport protocol service indication to notify a task manager that it has received a TASK frame.

Task Management Request Received (IN (Nexus, Function Identifier, [Association]))

Table 200 shows how the arguments to the Task Management Request Received transport protocol service are determined.

Table 200 — Task Management Request Received transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the Function Identifier), where: a) I indicates the initiator port that sent the TASK frame; b) T indicates the target port that received the TASK frame; c) L indicated by the LOGICAL UNIT NUMBER field in the TASK frame header; and d) Q (for an I_T_L_Q nexus) indicated by the TAG OF TASK TO BE MANAGED field in the TASK frame header.
Function Identifier	Indicates the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK (Nexus argument indicates an I_T_L_Q Nexus); b) ABORT TASK SET (Nexus argument indicates an I_T_L Nexus); c) CLEAR ACA (Nexus argument indicates an I_T_L Nexus); d) CLEAR TASK SET (Nexus argument indicates an I_T_L Nexus); e) I_T NEXUS RESET (Nexus argument indicates an I_T Nexus); f) LOGICAL UNIT RESET (Nexus argument indicates an I_T_L Nexus); g) QUERY TASK (Nexus argument indicates an I_T_L_Q Nexus); h) QUERY TASK SET (Nexus argument indicates an I_T_L Nexus); and i) QUERY UNIT ATTENTION (Nexus argument indicates an I_T_L Nexus).
[Association]	Indicates the TAG field in the TASK frame header.

10.2.1.14 Task Management Function Executed transport protocol service

A task manager uses the Task Management Function Executed transport protocol service response to request that an SSP target port transmit a RESPONSE frame.

Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information], [Association], [Response Fence]))

A task manager shall only call Task Management Function Executed () after receiving Task Management Request Received ().

Table 201 shows how the arguments to the Task Management Function Executed transport protocol service are used.

Table 201 — Task Management Function Executed transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the function), where: a) I specifies the initiator port to which the RESPONSE frame is sent; b) T specifies the target port to send the RESPONSE frame; c) L specifies the LOGICAL UNIT NUMBER field in the RESPONSE frame header; and d) Q (for an I_T_L_Q nexus) indirectly specifies the TAG field in the RESPONSE frame header.
Service Response	Specifies the DATAPRES field and RESPONSE CODE field in the RESPONSE frame: a) FUNCTION COMPLETE: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION COMPLETE; b) FUNCTION SUCCEEDED: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION SUCCEEDED; c) FUNCTION REJECTED: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION NOT SUPPORTED; d) INCORRECT LOGICAL UNIT NUMBER: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to INCORRECT LOGICAL UNIT NUMBER; or e) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to: A) INVALID FRAME; B) TASK MANAGEMENT FUNCTION FAILED; or C) OVERLAPPED TAG ATTEMPTED.
[Additional Response Information]	Specifies the ADDITIONAL RESPONSE INFORMATION field in the RESPONSE frame.
[Association]	Specifies the TAG field in the RESPONSE frame header.
[Response Fence]	If included, specifies an I_T nexus, I_T_L nexus, or I_T_L_Q nexus for which the RESPONSE frame is fenced.

The device server shall set the Response Fence argument to the Nexus argument.

10.2.1.15 Received Task Management Function Executed transport protocol service

An SSP initiator port uses the Received Task Management Function Executed transport protocol service confirmation to notify an application client that it has received a response to a TASK frame (e.g., received a RESPONSE frame or a NAK).

Received Task Management Function Executed (IN (Nexus, Service Response, [Association]))

Table 202 shows how the arguments to the Received Task Management Function Executed transport protocol service are determined.

Table 202 — Received Task Management Function Executed transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the function), where: a) I indicates the initiator port that received the RESPONSE frame; b) T indicates the target port that sent the RESPONSE frame; c) L indicates the LOGICAL UNIT NUMBER field in the RESPONSE frame header or TASK frame header; and d) Q (for an I_T_L_Q nexus) indirectly indicates the TAG field in the RESPONSE frame header, or indicates the TAG OF TASK TO BE MANAGED field TASK frame header.
Service Response	Indicates the response to the TASK frame: a) FUNCTION COMPLETE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION COMPLETE; b) FUNCTION SUCCEEDED: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION SUCCEEDED; c) FUNCTION REJECTED: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED; d) INCORRECT LOGICAL UNIT NUMBER: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER; or e) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to: A) INVALID FRAME; B) TASK MANAGEMENT FUNCTION FAILED; or C) OVERLAPPED TAG ATTEMPTED; or a NAK was received for the TASK frame, or the length of the RESPONSE frame is incorrect.
[Additional Response Information]	Indicates the ADDITIONAL RESPONSE INFORMATION field in the RESPONSE frame.
[Association]	Indicates the TAG field in the RESPONSE frame header or the TASK frame header.

10.2.2 Application client error handling

If an SSP initiator port calls Command Complete Received () with a Service Response of:

- a) Service Delivery or Target Failure - XFER_RDY Information Unit Too Short;
- b) Service Delivery or Target Failure - XFER_RDY Information Unit Too Long;
- c) Service Delivery or Target Failure - XFER_RDY Incorrect Write Data Length;
- d) Service Delivery or Target Failure - XFER_RDY Requested Offset Error;
- e) Service Delivery or Target Failure - XFER_RDY Not Expected;
- f) Service Delivery or Target Failure - DATA Information Unit Too Short;
- g) Service Delivery or Target Failure - DATA Too Much Read Data;
- h) Service Delivery or Target Failure - DATA Data Offset Error;
- i) Service Delivery or Target Failure - DATA Not Expected;
- j) Service Delivery or Target Failure - NAK Received,

then the application client shall abort the command (e.g., by sending an ABORT TASK task management function).

After an application client calls Send SCSI Command (), if Command Complete Received () returns a Service Response of Service Delivery or Target Failure - ACK/NAK Timeout, the application client shall send a QUERY TASK task management function with Send Task Management Request () to determine whether the command was received successfully. If Received Task Management Function Executed () returns a Service Response of FUNCTION SUCCEEDED, the application client shall assume the command was delivered successfully. If Received Task Management Function Executed () returns a Service Response of FUNCTION COMPLETE, and Command Complete Received () has not yet been invoked a second time for the command in question (e.g., indicating a RESPONSE frame arrived for the command before the QUERY TASK was processed), the application client shall assume the command was not delivered successfully and may reuse the tag. The application client should call Send SCSI Command () again with identical arguments.

After a Received Task Management Function Executed () call with a Service Response of Service Delivery or Target Failure - ACK/NAK Timeout, an application client should call Send Task Management Request () with identical arguments, including the same tag.

After a Command Complete Received () or Received Task Management Function Executed () call returns a Service Response other than Service Delivery or Target Failure - ACK/NAK Timeout, an application client shall not reuse the tag until it determines the tag is no longer in use by the logical unit (e.g., the ACK for the RESPONSE frame was seen by the SSP target port). Examples of ways the application client may determine that a tag may be used are:

- a) receiving another frame in the same connection;
- b) receiving a DONE (NORMAL) or DONE (CREDIT TIMEOUT) in the same connection; or
- c) receiving a DONE (ACK/NAK TIMEOUT) in the same connection, then running a QUERY TASK task management function to confirm that the tag is no longer active in the logical unit.

10.2.3 Device server error handling

If the SCSI target device performs tag checking and an SSP target port calls SCSI Command Received () with a tag already in use by another SCSI command (i.e., an overlapped command) in any logical unit, the task router and device server(s) shall abort all task management functions received on that I_T nexus and shall respond to the overlapped command as defined in SAM-4.

If an SSP target port calls Data-Out Received () with a Delivery Result set to a value in table 203, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set as indicated in table 203.

Table 203 — Delivery Result to additional sense code mapping

Delivery Result	Additional sense code
DELIVERY FAILURE - DATA OFFSET ERROR	DATA OFFSET ERROR
DELIVERY FAILURE - TOO MUCH WRITE DATA	TOO MUCH WRITE DATA
DELIVERY FAILURE - INFORMATION UNIT TOO SHORT	INFORMATION UNIT TOO SHORT
DELIVERY FAILURE - ACK/NAK TIMEOUT	ACK/NAK TIMEOUT
DELIVERY FAILURE - NAK RECEIVED	NAK RECEIVED
DELIVERY FAILURE - INITIATOR RESPONSE TIMEOUT	INITIATOR RESPONSE TIMEOUT

10.2.4 Task router and task manager error handling

If the SCSI target device performs tag checking and:

- a) an SSP target port calls SCSI Command Received () with a tag already in use by a SCSI task management function in any logical unit; or
- b) an SSP target port calls Task Management Request Received () with a tag already in use by a SCSI command or SCSI task management function in any logical unit,

then the task router and task manager(s) shall:

- a) abort all commands received on that I_T nexus;
- b) abort all task management functions received on that I_T nexus; and
- c) call Task Management Function Executed () with the Service Response set to FUNCTION REJECTED - Overlapped Tag Attempted (i.e., requesting that the target port set the DATAPRES field to RESPONSE_DATA and the RESPONSE CODE field set to OVERLAPPED TAG ATTEMPTED).

10.2.5 SCSI transport protocol event notifications

Table 204 lists the SCSI transport protocol event notifications supported by this standard.

Table 204 — SCSI transport protocol events

Event notification	SAS SSP implementation
Transport Reset	Receipt of a hard reset sequence (see 4.4.2)
Nexus Loss	Receipt of specific OPEN_REJECTs for a specific time period (see 4.5).
Power Loss Expected	Receipt of a NOTIFY (POWER LOSS EXPECTED)(see 7.2.5.11.3)

10.2.6 SCSI commands

10.2.6.1 INQUIRY command

The vital product data returned by the INQUIRY command (see SPC-4) that shall be returned by a logical unit in a SAS device is described in 10.2.11.

10.2.6.2 MODE SELECT and MODE SENSE commands

SAS-specific mode pages accessed with the MODE SELECT and MODE SENSE commands (see SPC-4) are described in 10.2.7.

10.2.6.3 LOG SELECT and LOG SENSE commands

SAS-specific log pages accessed with the LOG SELECT and LOG SENSE commands (see SPC-4) are described in 10.2.8.

10.2.6.4 SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands

SAS-specific diagnostic pages accessed with the SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands (see SPC-4) are described in 10.2.9.

Zoning (see 4.9) is applied to SES-2 diagnostic pages as described in 10.2.9.

10.2.6.5 START STOP UNIT command

The power condition states controlled by the START STOP UNIT command (see SBC-3) for a SAS device are described in 10.2.10.

10.2.7 SCSI mode parameters

10.2.7.1 SCSI mode parameters overview

Table 205 defines mode pages supported by logical units in SCSI target devices in SAS domains (i.e., with SSP target ports) that support the MODE SELECT or MODE SENSE commands.

Table 205 — SSP target port mode pages

Mode page code	Subpage code	Description	Reference
02h	00h	Disconnect-Reconnect mode page	10.2.7.2
18h	00h	Protocol-Specific Logical Unit mode page	10.2.7.3
	01h - DFh	Reserved	
	E0h - FEh	Vendor specific	
	FFh	Return all subpages for this mode page code	SPC-4
19h	00h	Protocol-Specific Port mode page	10.2.7.4
	01h	Phy Control And Discover mode page	10.2.7.5
	02h	Shared Port Control mode page	10.2.7.6
	03h	SAS-2 Phy mode page	10.2.7.6
	04h - DFh	Reserved	
	E0h - FEh	Vendor specific	
	FFh	Return all subpages for this mode page code	SPC-4

10.2.7.2 Disconnect-Reconnect mode page

10.2.7.2.1 Disconnect-Reconnect mode page overview

The Disconnect-Reconnect mode page (see SPC-4) provides the application client the means to tune the performance of a service delivery subsystem. Table 206 defines the parameters which are applicable to SSP. If any field in the Disconnect-Reconnect mode page is not implemented, the value assumed for the functionality of the field shall be zero (i.e., as if the field in the mode page is implemented and the field is set to zero).

Editor's Note 49: Is a sentence like that needed for fields in other mode pages? Is a sentence like that needed for mode pages that are completely not implemented?

The application client sends the values in the fields to be used by the device server to control the SSP connections by means of a MODE SELECT command. The device server shall then communicate the field values to the SSP target port. The field values are communicated from the device server to the SSP target port in a vendor-specific manner.

SAS devices shall only use the parameter fields defined below in this subclause. If any other fields within the Disconnect-Reconnect mode page of the MODE SELECT command contain a non-zero value, the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 206 — Disconnect-Reconnect mode page for SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	Reserved							
3	Reserved							
4	(MSB)	BUS INACTIVITY TIME LIMIT						
5								(LSB)
6	Reserved							
7								
8	(MSB)	MAXIMUM CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to this mode page.

The PAGE CODE (PS) field shall be set to 02h.

The PAGE LENGTH field shall be set to 0Eh.

The BUS INACTIVITY TIME LIMIT field is defined in 10.2.7.2.2.

The MAXIMUM CONNECT TIME LIMIT field is defined in 10.2.7.2.3.

The MAXIMUM BURST SIZE field is defined in 10.2.7.2.4.

The FIRST BURST SIZE field is defined in 10.2.7.2.5.

10.2.7.2.2 BUS INACTIVITY TIME LIMIT field

The value in the BUS INACTIVITY TIME LIMIT field contains the maximum time in 100 μ s increments that an SSP target port is permitted to maintain a connection (see 4.1.12) without transferring a frame to the SSP initiator port. When this time is exceeded, the SSP target port shall prepare to close the connection (i.e., by requesting to have the link layer transmit DONE). This value may be rounded as defined in SPC-4. A value of 0000h in

this field specifies that there is no bus inactivity time limit. The bus inactivity time limit is enforced by the port layer (see 8.2.3).

10.2.7.2.3 MAXIMUM CONNECT TIME LIMIT field

The value in the MAXIMUM CONNECT TIME LIMIT field contains the maximum duration of a connection (see 4.1.12) in 100 μ s increments (e.g., a value of 0001h in this field means that the time is less than or equal to 100 μ s and a value of 0002h in this field means that the time is less than or equal to 200 μ s). When this time is exceeded, the SSP target port shall prepare to close the connection. If an SSP target port is transferring a frame when the maximum connection time limit is exceeded, the SSP target port shall complete transfer of the frame before preparing to close the connection. This value may be rounded as defined in SPC-4. A value of 0000h in this field specifies that there is no maximum connection time limit. The maximum connection time limit is enforced by the port layer (see 8.2.3).

10.2.7.2.4 MAXIMUM BURST SIZE field

For read data, the value in the MAXIMUM BURST SIZE field contains the maximum amount of data that is transferred during a connection by an SSP target port per I_T_L_Q nexus without transferring at least one frame for a different I_T_L_Q nexus. If the SSP target port:

- a) has read data to transfer for only one I_T_L_Q nexus, and
- b) has no requests to transfer write data for any I_T_L_Q nexus;

then the SSP target port shall prepare to close the connection after the amount of data specified by the MAXIMUM BURST SIZE field is transferred to the SSP initiator port.

For write data, the value shall specify the maximum amount of data that an SSP target port requests via a single XFER_RDY frame (see 9.2.2.3).

This value shall be specified in 512-byte increments (e.g., a value of one in this field means that the number of bytes transferred to the SSP initiator port for the nexus is less than or equal to 512 and a value of two in this field means that the number of bytes transferred to the SSP initiator port for the nexus is less than or equal to 1 024). A value of zero in this field shall specify that there is no maximum burst size.

In terms of the SCSI transport protocol services (see 10.2.1), the device server shall limit the Request Byte Count argument to the Receive Data-Out () protocol service and the Send Data-In () protocol service to the amount specified in this field.

10.2.7.2.5 FIRST BURST SIZE field

If the ENABLE FIRST BURST field in the COMMAND frame is set to zero, the FIRST BURST SIZE field is ignored.

If the ENABLE FIRST BURST field in the COMMAND frame is set to one, the value in the FIRST BURST SIZE field contains the maximum amount of write data in 512-byte increments that may be sent by the SSP initiator port to the SSP target port without having to receive an XFER_RDY frame (see 9.2.2.3) from the SSP target port (e.g., a value of one in this field means that the number of bytes transferred by the SSP initiator port is less than or equal to 512 and a value of two in this field means that the number of bytes transferred by the SSP initiator port is less than or equal to 1 024).

Specifying a non-zero value in the FIRST BURST SIZE field is equivalent to an implicit XFER_RDY frame for each command requiring write data where the WRITE DATA LENGTH field of the XFER_RDY frame is set to 512 times the value of the FIRST BURST SIZE field.

The rules for data transferred using the value in the FIRST BURST SIZE field are the same as those used for data transferred for an XFER_RDY frame (i.e., the number of bytes transferred using the value in the FIRST BURST SIZE field is as if that number of bytes was requested by an XFER_RDY frame).

If the amount of data to be transferred for the command is less than the amount of data specified by the FIRST BURST SIZE field, the SSP target port shall not transmit an XFER_RDY frame for the command. If the amount of data to be transferred for the command is greater than the amount of data specified by the FIRST BURST SIZE field, the SSP target port shall transmit an XFER_RDY frame after it has received all of the data specified by the FIRST BURST SIZE field from the SSP initiator port. All data for the command is not required to be transferred during the same connection in which the command is transferred.

A value of zero in this field shall specify that there is no first burst size (i.e., an SSP initiator port shall transmit no write DATA frames to the SSP target port before receiving an XFER_RDY frame).

The first burst size is handled by the SCSI transport protocol services (see 10.2.1) and the SSP transport layer (see 9.2.6).

10.2.7.3 Protocol-Specific Logical Unit mode page

The Protocol-Specific Logical Unit mode page (see SPC-4) contains parameters that affect SSP target port operation on behalf of the logical unit.

The mode page policy (see SPC-4) for this mode page shall be either shared or per target port. If the SAS target device has multiple SSP target ports, the mode page policy should be per target port.

Parameters in this mode page:

- a) shall affect all phys in the SSP target port if the mode page policy is per target port; and
- b) shall affect all SSP target ports in the SAS target device if the mode page policy is shared.

Table 207 defines the format of the page for SAS SSP.

Table 207 — Protocol-Specific Logical Unit mode page for SAS SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (18h)					
1	PAGE LENGTH (06h)							
2	Reserved			TRANSPORT LAYER RETRIES	PROTOCOL IDENTIFIER (6h)			
3	Reserved							
7								

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to this mode page.

The PAGE CODE field shall be set to 18h.

The PAGE LENGTH field shall be set to the number of bytes in the page after the PAGE LENGTH field (i.e., 06h).

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

A TRANSPORT LAYER RETRIES bit set to one specifies that, for commands received in COMMAND frames with the TLR CONTROL field set to 00b or 11b (see 9.2.1), the target port shall support transport layer retries for XFER_RDY and DATA frames for the logical unit as described in 9.2.4 (i.e., transport layer retries are enabled). A TRANSPORT LAYER RETRIES bit set to zero specifies that, for commands received in COMMAND frames with the TLR CONTROL field set to 00b or 11b (see 9.2.1), transport layer retries shall not be used (i.e., transport layer retries are disabled).

NOTE 87 - The TRANSPORT LAYER RETRIES bit may become obsolete in a future version of this standard.

10.2.7.4 Protocol-Specific Port mode page

The Protocol-Specific Port mode page (see SPC-4) contains parameters that affect SSP target port operation. If the mode page is implemented by one logical unit in a SCSI target device, it shall be implemented by all logical units in the SCSI target device that support the MODE SELECT or MODE SENSE commands.

The mode page policy (see SPC-4) for this mode page shall be either shared or per target port. If a SAS target device has multiple SSP target ports, the mode page policy should be per target port.

Parameters in this mode page:

- a) shall affect all phys in the SSP target port if the mode page policy is per target port; and
- b) shall affect all SSP target ports in the SAS target device if the mode page policy is shared.

Table 208 defines the format of the page for SAS SSP.

Table 208 — Protocol-Specific Port mode page for SAS SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (06h)							
2	Reserved	CONTINUE AWT	BROADCAST ASYNCHRONOUS EVENT	READY LED MEANING	PROTOCOL IDENTIFIER (6h)			
3	Reserved							
4	I_T NEXUS LOSS TIME							
5								
6	INITIATOR RESPONSE TIMEOUT							
7								

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to this mode page.

The PAGE CODE field shall be set to 19h.

The PAGE LENGTH field shall be set to the number of bytes in the page after the PAGE LENGTH field (i.e., 06h).

A CONTINUE AWT bit set to one specifies that the the SAS port shall not stop the Arbitration Wait Time timer and set the Arbitration Wait Time timer to zero when the SAS port receives an OPEN_REJECT (RETRY). A CONTINUE AWT bit set to zero specifies that the SAS port shall stop the Arbitration Wait Time timer and set the Arbitration Wait Time timer to zero when it receives an OPEN_REJECT (RETRY).

A BROADCAST ASYNCHRONOUS EVENT bit set to one specifies that the device server shall enable origination of Broadcast (Asynchronous Event) (see 4.1.13). A BROADCAST ASYNCHRONOUS EVENT bit set to zero specifies that the device server shall disable origination of Broadcast (Asynchronous Event).

The READY LED MEANING bit specifies the READY LED signal behavior (see 10.4.1). Regardless of the mode page policy (see SPC-4) for this mode page, the shared mode page policy shall be applied to the READY LED MEANING bit.

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The I_T NEXUS LOSS TIME field contains the time that the SSP target port shall retry connection requests to an SSP initiator port that are rejected with certain responses indicating that the SSP initiator port may no longer be present (see 8.2.2) before recognizing an I_T nexus loss (see 4.5). Table 209 defines the values of the I_T

NEXUS LOSS TIME field. If this mode page is not implemented, the I_T nexus loss time is vendor specific. This value is enforced by the port layer (see 8.2.2).

Table 209 — I_T NEXUS LOSS TIME field

Code	Description
0000h	Vendor-specific amount of time.
0001h to FFFEh	Time in milliseconds.
FFFFh	The SSP target port shall never recognize an I_T nexus loss (i.e., it shall retry the connection requests forever).

NOTE 88 - If this mode page is implemented, the default value of the I_T NEXUS LOSS TIME field should be non-zero. It is recommended that this value be 2 000 ms.

NOTE 89 - An SSP initiator port should retry connection requests for the time indicated by the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page for the SSP target port to which it is trying to establish a connection (see 4.5).

Editor's Note 50: That should be a "minimum" time

The INITIATOR RESPONSE TIMEOUT field contains the time in milliseconds that the SSP target port shall wait for the receipt of a frame (e.g., a write DATA frame) before aborting the command associated with that frame. An INITIATOR RESPONSE TIMEOUT field value of zero indicates that the SSP target port shall disable the initiator response timeout timer. If this mode page is not implemented, the logical unit shall not implement an initiator response timeout timer. This value is enforced by the transport layer (see 9.2.6.3).

10.2.7.5 Phy Control And Discover mode page

The Phy Control And Discover mode page contains parameters that affect SSP target phy operation. If the mode page is implemented by one logical unit in a SCSI target device, it shall be implemented by all logical units in the SCSI target device that support the MODE SELECT or MODE SENSE commands.

The mode page policy (see SPC-4) for this mode page shall be shared. Parameters in this mode page shall affect only the referenced phy.

Table 210 defines the format of this mode page.

Table 210 — Phy Control And Discover mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								
4	Reserved							
5	Reserved				PROTOCOL IDENTIFIER (6h)			
6	GENERATION CODE							
7	NUMBER OF PHYS							
SAS phy mode descriptor list								
8	SAS phy mode descriptor (first)(see table 211)							
55								
...	...							
n - 47	SAS phy mode descriptor (last)(see table 211)							
n								

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to one to access this mode page.

The PAGE CODE field shall be set to 19h.

The SUBPAGE CODE field shall be set to 01h.

The PAGE LENGTH field shall be set to the number of bytes in the page after the PAGE LENGTH field (i.e., 4 + (the value of the NUMBER OF PHYS field) × (the length in bytes of the SAS phy mode descriptor)).

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The GENERATION CODE field is a one-byte counter that shall be incremented by one by the device server every time the values in this mode page or the SAS-2 Phy mode page (see 10.2.7.6) field values are changed. A GENERATION CODE field set to 00h indicates the generation code is unknown. The device server shall wrap this field to 01h as the next increment after reaching its maximum value (i.e., FFh). The GENERATION CODE field is also contained in the Protocol-Specific Port log page and may be used to correlate phy settings across mode page and log page accesses.

NOTE 90 - Device servers compliant with previous versions of this standard set the GENERATION CODE field to 00h.

The NUMBER OF PHYS field contains the number of phys in the SAS target device and indicates the number of SAS phy mode descriptors that follow. This field shall not be changeable with the MODE SELECT command.

The SAS phy mode descriptor list contains a SAS phy mode descriptor for each phy in the SAS target device, not just the SAS target port, starting with the lowest numbered phy and ending with the highest numbered phy.

Table 211 defines the SAS phy mode descriptor.

Table 211 — SAS phy mode descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	Reserved							
3								
4	Reserved	ATTACHED DEVICE TYPE			ATTACHED REASON			
5	REASON				NEGOTIATED LOGICAL LINK RATE			
6	Reserved				ATTACHED SSP INITIATOR PORT	ATTACHED STP INITIATOR PORT	ATTACHED SMP INITIATOR PORT	Reserved
7	Reserved				ATTACHED SSP TARGET PORT	ATTACHED STP TARGET PORT	ATTACHED SMP TARGET PORT	Reserved
8	SAS ADDRESS							
15								
16	ATTACHED SAS ADDRESS							
23								
24	ATTACHED PHY IDENTIFIER							
25	Reserved							
31								
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
34	Reserved							
41								
42	Vendor specific							
43								
44	Reserved							
47								

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field are defined in the SMP PHY CONTROL function (see 10.4.3.27).

The fields in the SAS phy mode descriptor not defined in this subclause are defined in the SMP DISCOVER response (see 10.4.3.9). These fields shall not be changeable with the MODE SELECT command.

10.2.7.6 Shared Port Control mode page

The Shared Port Control mode page contains parameters that affect SSP target port operation. If the mode page is implemented by one logical unit in a SCSI target device, it shall be implemented by all logical units in the SCSI target device that support the MODE SELECT or MODE SENSE commands.

The mode page policy (see SPC-4) for this mode page shall be shared.

Table 212 defines the format of this mode page.

Table 212 — Shared Port Control mode page

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (02h)							
2	(MSB)	PAGE LENGTH (000Ch)						
3								(LSB)
4	Reserved							
5	Reserved				PROTOCOL IDENTIFIER (6h)			
6	(MSB)	POWER LOSS TIMEOUT						
7								(LSB)
8	Reserved							
15								

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to one to access this mode page.

The PAGE CODE field shall be set to 19h.

The SUBPAGE CODE field shall be set to 02h.

The PAGE LENGTH field shall be set to the number of bytes in the page after the PAGE LENGTH field (i.e., 000Ch).

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The POWER LOSS TIMEOUT field contains the maximum time, in one millisecond increments, that a target port shall respond to connection requests with OPEN_REJECT (RETRY) after receiving NOTIFY (POWER LOSS EXPECTED) (see 7.2.5.11.3). A POWER LOSS TIMEOUT field set to 0000h is vendor-specific. The power loss timeout shall be restarted on each NOTIFY (POWER LOSS EXPECTED) that is received.

Editor's Note 51: "is vendor-specific" should be "specifies that <...> is vendor-specific."

10.2.7.6 SAS-2 Phy mode page

The SAS-2 Phy mode page contains parameters that affect SSP target phy operation that were first defined in SAS-2. If the mode page is implemented by one logical unit in a SCSI target device, it shall be implemented by all logical units in the SCSI target device that support the MODE SELECT or MODE SENSE commands.

The mode page policy (see SPC-4) for this mode page shall be shared. Parameters in this mode page shall affect only the referenced phy.

Table 293 defines the format of this mode page.

Table 293 — SAS-2 Phy mode page

Byte\Bit	7	6	5	4	3	2	1	0	
0	PS	SPF (1b)	PAGE CODE (19h)						
1	SUBPAGE CODE (03h)								
2	(MSB)	PAGE LENGTH (n - 3)							
3									(LSB)
4	Reserved								
5	Reserved				PROTOCOL IDENTIFIER (6h)				
6	GENERATION CODE								
7	NUMBER OF PHYS								
SAS-2 phy mode descriptor list									
8	SAS-2 phy mode descriptor (first)(see table 294)								
27									
...	...								
n - 19	SAS-2 phy mode descriptor (last)(see table 294)								
n									

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to one to access this mode page.

The PAGE CODE field shall be set to 19h.

The SUBPAGE CODE field shall be set to 03h.

The PAGE LENGTH field shall be set to the number of bytes in the page after the PAGE LENGTH field (i.e., 4 + (the value of the NUMBER OF PHYS field) × (the length in bytes of the SAS phy mode descriptor)).

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The GENERATION CODE field is defined in the Phy Control and Discover mode page (see 10.2.7.5).

The NUMBER OF PHYS field contains the number of phys in the SAS target device and indicates the number of SAS-2 phy mode descriptors that follow. This field shall not be changeable with the MODE SELECT command.

The SAS-2 phy mode descriptor list contains a SAS-2 phy mode descriptor for each phy in the SAS target device, not just the SAS target port, starting with the lowest numbered phy and ending with the highest numbered phy.

Table 211 defines the SAS-2 phy mode descriptor.

Table 294 — SAS-2 phy mode descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	(MSB)							
3	DESCRIPTOR LENGTH (10h)							(LSB)
4	PROGRAMMED PHY CAPABILITIES							
7								
8	CURRENT PHY CAPABILITIES							
11								
12	ATTACHED PHY CAPABILITIES							
15								
16	Reserved							
17								
18	Reserved			NEGOTIATED SSC	NEGOTIATED PHYSICAL LINK RATE			
19	Reserved							HARDWARE MUXING SUPPORTED

The DESCRIPTOR LENGTH field contains the length in bytes that follow in the descriptor and shall be set to 10h.

The fields in the SAS-2 phy mode descriptor not defined in this subclause are defined in the SMP DISCOVER response (see 10.4.3.9). These fields shall not be changeable with the MODE SELECT command.

10.2.8 SCSI log parameters

10.2.8.1 Protocol-Specific Port log page

The Protocol-Specific Port log page for SAS SSP defined in table 295 is used to return information about phy events concerning the SAS target device's phy(s).

Table 295 — Protocol-Specific Port log page for SAS SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	DS	SPF (0b)	PAGE CODE (18h)					
1	SUBPAGE CODE (00h)							
2	(MSB)							
3	PAGE LENGTH (m - 3)							
	(LSB)							
Protocol-specific port log parameter list								
4	Protocol-specific port log parameter (first)(see table 296)							
...	...							
	Protocol-specific port log parameter (last)(see table 296)							
m								

The DISABLE SAVE (DS) bit is defined in SPC-4.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to this log page.

The PAGE CODE field shall be set to 18h.

The SUBPAGE CODE field shall be set to 00h.

The PAGE LENGTH field shall be set to the number of bytes in the log page after the PAGE LENGTH field.

Table 296 defines the format for the Protocol-Specific Port log parameter for SAS. The SAS log parameter is a list parameter (i.e., not a data counter) and only has cumulative (i.e., not threshold) values (see SPC-4).

Table 296 — Protocol-Specific Port log parameter for SAS

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (relative target port identifier) (LSB)							
2	Parameter control byte							
	DU	Obsolete	TSD	ETC	TMC		FORMAT AND LINKING	
3	PARAMETER LENGTH (y - 3)							
4	Reserved				PROTOCOL IDENTIFIER (6h)			
5	Reserved							
6	GENERATION CODE							
7	NUMBER OF PHYS							
SAS phy log descriptor list								
8	SAS phy log descriptor (first)(see table 298)							
8 + m								
...	...							
y - m	SAS phy log descriptor (last)(see table 298)							
y								

The PARAMETER CODE field contains the relative target port identifier (see SPC-4) of the SSP target port that the log parameter describes.

Table 297 defines the values of the fields in the parameter control byte for the log parameter.

Table 297 — Parameter control byte in the Protocol-Specific Port log parameter for SAS

Field	Value for LOG SENSE	Value for LOG SELECT	Description
DU	0	0 or 1	The DU bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TSD	0	0 or 1	The device server shall support implicitly saving the log parameter at vendor specific intervals.
ETC	0	0 or 1	The ETC bit is not defined for list parameters, so shall be set to zero when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
TMC	00b	any	The TMC field is not defined for list parameters, so shall be set to 00b when read with the LOG SENSE command and shall be ignored when written with the LOG SELECT command.
FORMAT AND LINKING	11b	11b	The log parameter is a binary format list parameter.

The PARAMETER LENGTH field is set to the number of bytes in the log parameter that follow the PARAMETER LENGTH field.

The PROTOCOL IDENTIFIER field is set to 6h.

The GENERATION CODE field is defined in the Phy Control and Discover mode page (see 10.2.7.5).

The NUMBER OF PHYS field contains the number of phys in the SAS target port (not in the entire SAS target device) and indicates the number of SAS phy log descriptors that follow.

The SAS phy log descriptor list contains SAS phy log descriptors.

Table 298 defines the SAS phy log descriptor.

Table 298 — SAS phy log descriptor (part 1 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	Reserved							
3	SAS PHY LOG DESCRIPTOR LENGTH (m - 3)							
4	Reserved	ATTACHED DEVICE TYPE			ATTACHED REASON			
5	REASON				NEGOTIATED LOGICAL LINK RATE			
6	Reserved				ATTACHED SSP INITIATOR PORT	ATTACHED STP INITIATOR PORT	ATTACHED SMP INITIATOR PORT	Reserved

Table 298 — SAS phy log descriptor (part 2 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
7	Reserved				ATTACHED SSP TARGET PORT	ATTACHED STP TARGET PORT	ATTACHED SMP TARGET PORT	Reserved
8	SAS ADDRESS							
15								
16	ATTACHED SAS ADDRESS							
23								
24	ATTACHED PHY IDENTIFIER							
25	Reserved							
31								
32	(MSB)	INVALID DWORD COUNT						(LSB)
35	RUNNING DISPARITY ERROR COUNT							
36								
39	LOSS OF DWORD SYNCHRONIZATION							
40								
43	PHY RESET PROBLEM							
44								
47	Reserved							
48								
50	NUMBER OF PHY EVENT DESCRIPTORS							
51								
Phy event descriptor list								
52	Phy event descriptor (first)(see table 358 in 10.4.3.13)							
63								
...	...							
m - 11	Phy event descriptor (last)(see table 358 in 10.4.3.13)							
m								

The SAS PHY LOG DESCRIPTOR LENGTH field indicates the number of bytes that follow in the SAS phy log descriptor. A SAS PHY LOG DESCRIPTOR LENGTH field set to zero indicates there are 44 additional bytes.

NOTE 91 - Logical units compliant with SAS and SAS-1.1 only support a 48 byte SAS phy log descriptor.

The INVALID DWORD COUNT field, RUNNING DISPARITY ERROR COUNT field, LOSS OF DWORD SYNCHRONIZATION field, and PHY RESET PROBLEM COUNT field are each defined in the SMP REPORT PHY ERROR LOG response (see 10.4.3.10).

For the INVALID DWORD COUNT field, RUNNING DISPARITY ERROR COUNT field, LOSS OF DWORD SYNCHRONIZATION COUNT field, and PHY RESET PROBLEM COUNT field, the phy may maintain any size counter but should maintain a 32-bit counter. If it reaches its maximum value, the counter shall stop and the device server shall set the field to FFFFFFFFh in the SAS phy log descriptor.

The NUMBER OF PHY EVENT DESCRIPTORS field indicates how many phy event descriptors follow.

Each phy event descriptor follows the format defined for the SMP REPORT PHY EVENT function in table 358 (see 10.4.3.13).

The fields in the SAS phy log descriptor not defined in this subclause are defined in the SMP DISCOVER response (see 10.4.3.9). These fields shall not be changeable with the LOG SELECT command.

10.2.9 SCSI diagnostic parameters

10.2.9.1 SCSI diagnostic parameters overview

Table 299 defines diagnostic pages supported by logical units in SCSI target devices in SAS domains (i.e., with SSP target ports) that support the SEND DIAGNOSTIC or RECEIVE DIAGNOSTIC RESULTS commands.

Table 299 — SSP target port diagnostic pages

Diagnostic page code	Description	Reference
3Fh	Protocol-Specific diagnostic page	10.2.9.2

An enclosure services process (see SES-2) describing elements in a SAS domain that are attached to a zoning expander device with zoning enabled (see 4.9) shall apply the zone permission table when providing access to those elements. Element types that may be subject to zoning include:

- a) Device element;
- b) Array Device element;
- c) Enclosure Services Controller Electronics element;
- d) SCC Controller Electronics element;
- e) SCSI Port/Transceiver element;
- f) SCSI Target Port element;
- g) SCSI Initiator Port element;
- h) SAS Expander element; and
- i) SAS Connector element.

Table 300 defines SCSI enclosure services diagnostic pages supported by logical units in SCSI target devices in SAS domains (e.g., with SSP target ports) that are affected by zoning.

Table 300 — Diagnostic pages affected by zoning

Diagnostic page code	Description	Reference
02h	Enclosure Control diagnostic page	SES-2 and 10.2.9.3
	Enclosure Status diagnostic page	SES-2 and 10.2.9.4
0Ah	Additional Element Status diagnostic page	SES-2 and 10.2.9.5

10.2.9.2 Protocol-Specific diagnostic page

The Protocol-Specific diagnostic page for SAS SSP provides a method for an application client to enable and disable phy test functions (see 4.10) for selected phys. The diagnostic page format is specified in SPC-4.

The Protocol-Specific diagnostic page is transmitted using the SEND DIAGNOSTIC command. If the device server receives a RECEIVE DIAGNOSTIC RESULTS command with the PAGE CODE field set to 3Fh, it shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB Table 301 defines the Protocol-Specific diagnostic page for SAS SSP.

Table 301 — Protocol-Specific diagnostic page for SAS SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (3Fh)							
1	Reserved				PROTOCOL IDENTIFIER (6h)			
2	(MSB) PAGE LENGTH (001Ch) (LSB)							
3								
4	PHY IDENTIFIER							
5	PHY TEST FUNCTION							
6	PHY TEST PATTERN							
7	Reserved				PHY TEST PATTERN PHYSICAL LINK RATE			
8	Reserved							
10								
11	PHY TEST PATTERN DWORDS CONTROL							
12	PHY TEST PATTERN DWORDS							
19								
20	Reserved							
31								

The PAGE CODE field shall be set to 3Fh.

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific diagnostic page.

The PAGE LENGTH field shall be set to the number of bytes in the diagnostic page after the PAGE LENGTH field (i.e., 001Ch)

The PHY IDENTIFIER field specifies the phy identifier (see 4.2.8) of the phy that is to perform or to stop performing a phy test function (i.e., the selected phy). If the PHY IDENTIFIER field specifies a phy that does not exist, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The PHY TEST FUNCTION field specifies the phy test function to be performed and is defined in table 302. If the PHY TEST FUNCTION field specifies a phy test function that is not supported, then the device server shall

terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 302 — PHY TEST FUNCTION field

Code	Name	Description
00h	STOP	If the selected phy is performing a phy test function, then the selected phy shall stop performing the phy test function and originate a link reset sequence. If the selected phy is not performing a phy test function, then this function has no effect on the selected phy. ^a
01h	TRANSMIT_PATTERNS	If the selected phy is not performing a phy test function, the selected phy shall be set to transmit the phy test pattern specified by the PHY TEST PATTERN field at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field and set to ignore its receiver. If the selected phy receives data while transmitting the pattern, then the selected phy shall ignore the received data. If the selected phy is performing a phy test function, the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to PHY TEST FUNCTION IN PROGRESS. ^a
02h - EFh	Reserved	
F0h - FFh	Vendor specific	
^a If there is no SSP target port available to receive a SEND DIAGNOSTIC command to stop a phy from performing a phy test function, then a power on may be required to cause the phy to stop performing the function and originate a phy reset sequence.		

If the PHY TEST FUNCTION field is set to 01h (i.e., TRANSMIT_PATTERNS), then the PHY TEST PATTERN field specifies the phy test pattern to be transmitted as defined by table 303. If the PHY TEST PATTERN field specifies a phy test pattern that is not supported by the specified SAS phy, then the device server shall terminate the

SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 303 — PHY TEST PATTERN field

Code	Name	Description
00h	Reserved	
01h	JTPAT	The selected phy shall continuously transmit the JTPAT for RD+ and RD- (see A.1).
02h	CJTPAT	The selected phy shall continuously transmit the CJTPAT (see A.2).
03h - 0Fh	Reserved	
10h	TRAIN	The selected phy shall continuously transmit the TRAIN pattern (see 6.7.4.2.3.4).
11h	TRAIN_DONE	The selected phy shall continuously transmit the TRAIN_DONE pattern (see 6.7.4.2.3.4).
12h	IDLE	The selected phy shall continuously transmit idle dwords (see 7.4).
13h - 3Fh	Reserved	
40h	TWO_DWORDS	The selected phy shall continuously transmit the dwords specified by the PHY TEST PATTERN DWORDS CONTROL field and the PHY TEST PATTERN DWORDS field without scrambling. This pattern is only for use for characterization of the transmitter device and the passive interconnect. Phys are not required to support all patterns that may be specified.
41h - EFh	Reserved	
F0h - FFh	Vendor specific	

Editor's Note 52: JTSPAT (from FC-PI-4) might also be useful to test a receiver with a wider spectrum than CJTPAT

The PHY TEST PATTERN PHYSICAL LINK RATE field specifies the physical link rate at which the phy test pattern shall be transmitted and is defined in table 304. If the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field is less than the hardware minimum physical link rate or greater than the hardware maximum physical link rate, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 304 — PHY TEST PATTERN PHYSICAL LINK RATE field

Code	Description
0h - 7h	Reserved
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future physical link rates

The PHY TEST PATTERN DWORDS CONTROL field and PHY TEST PATTERN DWORDS field are only used if the PHY TEST PATTERN field is set to 40h (i.e., TWO_DWORDS) (see table 303).

The PHY TEST PATTERN DWORDS CONTROL field defined in table 305 controls whether the bytes in the PHY TEST PATTERN DWORDS field are sent as control characters or data characters.

Table 305 — PHY TEST PATTERN DWORDS CONTROL field

Code	Description
00h	Each byte in the PHY TEST PATTERN DWORDS field shall be sent as a data character (i.e., Dxx.y)(see 6.3.3) without scrambling.
08h	The fifth byte in the PHY TEST PATTERN DWORDS field shall be sent as a control character (i.e., Kxx.y)(see 6.3.3); each other byte shall be sent as a data character without scrambling.
80h	The first byte in the PHY TEST PATTERN DWORDS field shall be sent as a control character; each other byte shall be sent as a data character without scrambling.
88h	The first and fifth bytes in the PHY TEST PATTERN DWORDS field shall each be sent as a control character; each other byte shall be sent as a data character without scrambling.
All others	Reserved

The PHY TEST PATTERN DWORDS field contains the two dwords that are sent during a TWO_DWORDS test pattern. Whether each byte in the dwords is sent as a control character or a data character is specified by the PHY TEST PATTERN DWORDS CONTROL field. A byte specifying a control character shall only specify a control character which is used in this standard (see table 77 in 6.3) and is supported by the phy (i.e., all phys support K28.5 (i.e., BCh), but only phys supporting STP support K28.3 (i.e., 7Ch) or K28.6 (i.e., DCh)).

The device server shall terminate a SEND DIAGNOSTIC command specifying any unsupported combination with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 306 lists some examples of TWO_DWORDS phy test patterns.

Table 306 — TWO_DWORDS phy test pattern examples

PHY TEST PATTERN DWORDS CONTROL field	PHY TEST PATTERN DWORDS field	Description
00h	4A4A4A4A 4A4A4A4Ah	D10.2 characters (see table 75 in 6.3.3). This high-frequency pattern contains 01b repeating and has the highest possible frequency. This pattern may be used for measuring skew and rise/fall times (see table 61 in 5.3.6.2, table 62 in 5.3.6.3, and table 64 in 5.3.7.2).
00h	B5B5B5B5 B5B5B5B5h	D21.5 characters (see table 75 in 6.3.3). This high-frequency pattern contains 10b repeating and has the highest possible frequency. This pattern may be used for measuring skew and rise/fall times (see table 61 in 5.3.6.2, table 62 in 5.3.6.3, and table 64 in 5.3.7.2).
00h	78787878 78787878h	D24.3 characters (see table 75 in 6.3.3). This mid-frequency pattern contains 0011b or 1100b repeating (depending on starting disparity) and has half the highest possible frequency.
00h	D926D926 D926D926h	Pairs of D25.6 and D6.1 characters (see table 75 in 6.3.3). This mid-frequency pattern contains 1001b repeating and has half the highest possible frequency.
00h	7E7E7E7E 7E7E7E7Eh	D30.3 characters (see table 75 in 6.3.3). This low-frequency pattern contains four bits of one polarity, three bits of the other polarity, and three bits of the first polarity (e.g., 1111000111b), followed by the inverse (e.g., 0000111000b).
00h	EBF4EBF4 EBF4EBF4h	Pairs of these D11.7 and D20.7 characters (see table 75 in 6.3.3). This pattern contains a single bit of one polarity after five bits of the other polarity (i.e., 0000010b and 1111101b). This pattern may be used as a lone bit pattern for measuring jitter in external cables (see table 54 in 5.2.6).
88h	BC4A4A7B BC4A4A7Bh	ALIGN (0) primitives (see table 104 in 7.2.3). This pattern appears during OOB bursts (see 6.6), the SATA speed negotiation sequence (see 6.7.2.2), and the SAS speed negotiation sequence (see 6.7.4.2).
88h	BC070707 BC070707h	ALIGN (1) primitives (see table 104 in 7.2.3). This pattern appears during the SAS speed negotiation sequences (see 6.7.4.2).
80h	BC4A4A7B 4A787E7Eh	Pairs of an ALIGN (0) primitive (see table 104 in 7.2.3) and a dword containing D10.2, D24.3, D30.3, and D30.3 characters (see table 75 in 6.3.3).

10.2.9.3 Enclosure Control diagnostic page

If the SELECT bit is set to one for any element that represents a device attached to an expander phy for which the SAS initiator port performing the SEND DIAGNOSTIC command does not have access according to the zone permission table, the enclosure services process shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

10.2.9.4 Enclosure Status diagnostic page

The enclosure services process shall set the ELEMENT STATUS CODE field to 8h (i.e., No Access Allowed) for each element that represents a device attached to an expander phy for which the SAS initiator port performing

the RECEIVE DIAGNOSTIC RESULTS command does not have access according to the zone permission table.

10.2.9.5 Additional Element Status diagnostic page

The enclosure services process shall set the INVALID bit to one in the Additional Element Status descriptor (see SES-2) for each element that represents a device attached to an expander phy for which the SAS initiator port performing the RECEIVE DIAGNOSTIC RESULTS command does not have access according to the zone permission table.

10.2.10 SCSI power conditions

10.2.10.1 SCSI power conditions overview

The logical unit power condition states from the Power Condition mode page (see SPC-4) and START STOP UNIT command (see SBC-3), if implemented, shall interact with the NOTIFY (ENABLE SPINUP) primitive (see 7.2.5.11) to control temporary consumption of additional power (e.g., spin-up of rotating media) as described in this subclause.

The logical unit uses NOTIFY (ENABLE SPINUP) to:

- a) initiate spin-up after power on; and
- b) delay spin-ups requested by START STOP UNIT commands.

Editor's Note 53: Since NOTIFY (POWER LOSS EXPECTED) handling was added to state machines, they should probably also handle NOTIFY (ENABLE SPINUP)

10.2.10.2 SA_PC (SCSI application layer power condition) state machine

10.2.10.2.1 SA_PC state machine overview

The SA_PC (SCSI application layer power condition) state machine describes how the SAS target device processes logical unit power condition state change requests and NOTIFY (ENABLE SPINUP) if it is a SCSI target device.

NOTE 92 - This state machine is an enhanced version of the logical unit power condition state machines described in SPC-4 and SBC-3.

This state machine consists of the following states:

- a) SA_PC_0:Powered_On (see 10.2.10.2.2)(initial state);
- b) SA_PC_1:Active (see 10.2.10.2.3);
- c) SA_PC_2:Idle (see 10.2.10.2.4);
- d) SA_PC_3:Standby (see 10.2.10.2.5);
- e) SA_PC_4:Stopped (see 10.2.10.2.6)(specific to SBC-3 logical units);
- f) SA_PC_5:Active_Wait (see 10.2.10.2.7)(specific to SAS devices); and
- g) SA_PC_6:Idle_Wait (see 10.2.10.2.8)(specific to SAS devices).

This state machine shall start in the SA_PC_0:Powered_On state after power on.

If the device server processes a START STOP UNIT command (see SBC-3) with the IMMED bit set to one, it may complete the command before completing the transition, if any, specified by the POWER CONDITION field and the START bit.

Figure 208 describes the SA_PC state machine.

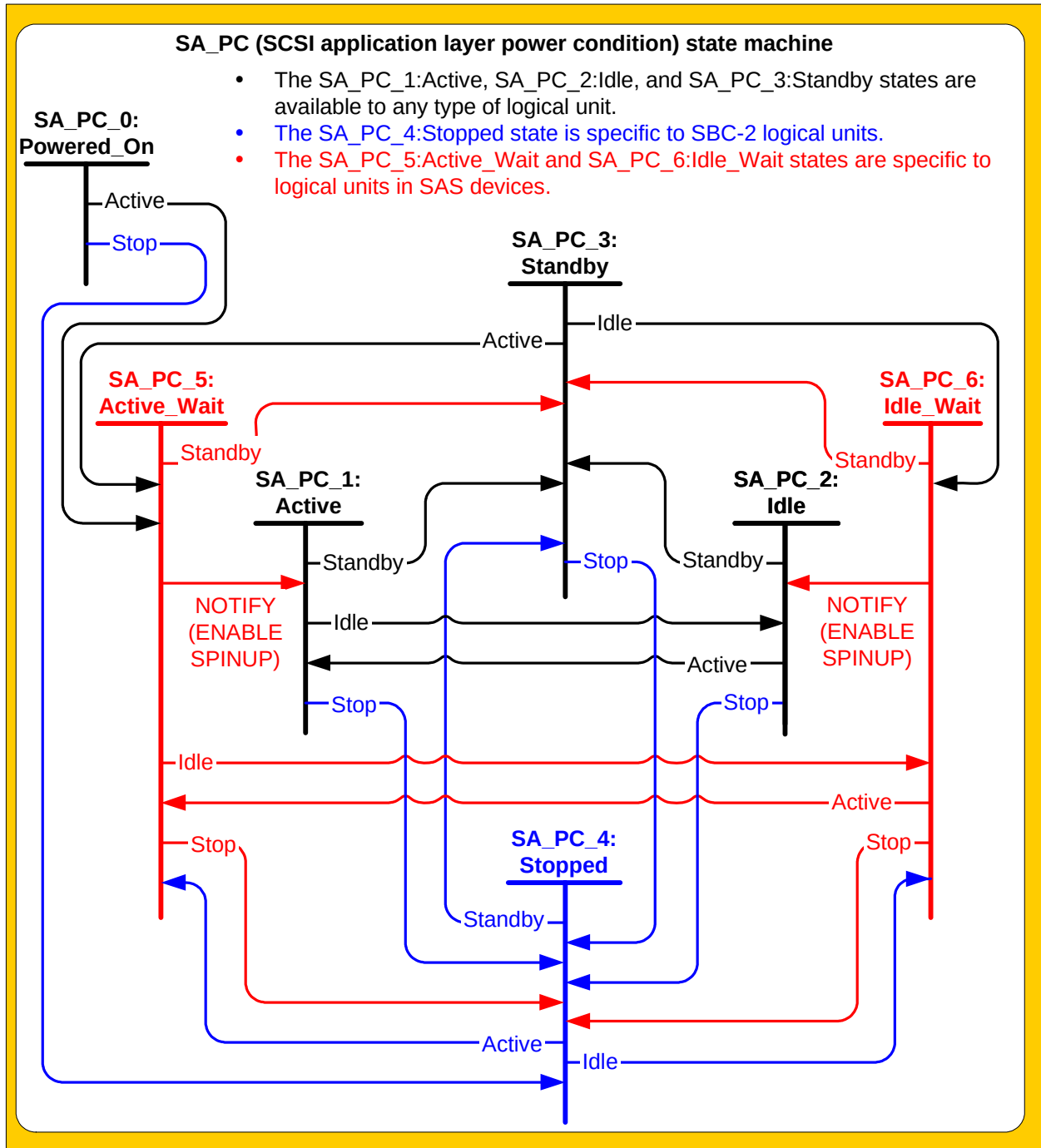


Figure 208 — SA_PC (SCSI application layer power condition) state machine for SAS

10.2.10.2.2 SA_PC_0:Powered_On state

10.2.10.2.2.1 State description

This state shall be entered upon power on. This state consumes zero time.

10.2.10.2.2.2 Transition SA_PC_0:Powered_On to SA_PC_4:Stopped

This transition shall occur if the SAS device has been configured to start in the SA_PC_4:Stopped state.

10.2.10.2.2.3 Transition SA_PC_0:Powered_On to SA_PC_5:Active_Wait

This transition shall occur if the SAS device has been configured to start in the SA_PC_5:Active_Wait state.

10.2.10.2.3 SA_PC_1:Active state**10.2.10.2.3.1 State description**

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-4 for more details about this state.

10.2.10.2.3.2 Transition SA_PC_1:Active to SA_PC_2:Idle

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is processed;
or
- c) the Power Condition mode page idle condition timer expires.

10.2.10.2.3.3 Transition SA_PC_1:Active to SA_PC_3:Standby

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is processed; or
- c) the Power Condition mode page standby condition timer expires.

10.2.10.2.3.4 Transition SA_PC_1:Active to SA_PC_4:Stopped

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to zero is processed.

10.2.10.2.4 SA_PC_2:Idle state**10.2.10.2.4.1 State description**

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-4 for more details about this state.

10.2.10.2.4.2 Transition SA_PC_2:Idle to SA_PC_1:Active

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to one is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is processed; or
- c) a command that requires the active power condition is processed.

10.2.10.2.4.3 Transition SA_PC_2:Idle to SA_PC_3:Standby

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is processed; or
- c) the Power Condition mode page standby condition timer expires.

10.2.10.2.4.4 Transition SA_PC_2:Idle to SA_PC_4:Stopped

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to zero is processed.

10.2.10.2.5 SA_PC_3:Standby state**10.2.10.2.5.1 State description**

While in this state, rotating media in block devices shall be stopped.

See SPC-4 for more details about this state.

10.2.10.2.5.2 Transition SA_PC_3:Standby to SA_PC_4:Stopped

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to zero is processed.

10.2.10.2.5.3 Transition SA_PC_3:Standby to SA_PC_5:Active_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to one is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is processed; or
- c) a command that requires the active power condition is processed.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_1:Active state.

10.2.10.2.5.4 Transition SA_PC_3:Standby to SA_PC_6:Idle_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is processed;
or
- c) a command that requires the idle power condition is processed.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.6 SA_PC_4:Stopped state**10.2.10.2.6.1 State description**

This state is only implemented in block devices.

While in this state, rotating media shall be stopped.

See SBC-3 for more details about this state.

10.2.10.2.6.2 Transition SA_PC_4:Stopped to SA_PC_3:Standby

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is processed; or
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is processed.

10.2.10.2.6.3 Transition SA_PC_4:Stopped to SA_PC_5:Active_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to one is processed; or

- b) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is processed.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_1:Active state.

10.2.10.2.6.4 Transition SA_PC_4:Stopped to SA_PC_6:Idle_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is processed; or
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is processed.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.7 SA_PC_5:Active_Wait state

10.2.10.2.7.1 State description

This state shall only be implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped. The device server shall be capable of processing commands and shall terminate each media access command or TEST UNIT READY command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED.

In response to a REQUEST SENSE command processed in this state, the device server shall return parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED and return GOOD status for the command.

10.2.10.2.7.2 Transition SA_PC_5:Active_Wait to SA_PC_1:Active

This transition shall occur if:

- a) a NOTIFY (ENABLE SPINUP) is detected; or
- b) the SAS device does not consume additional power as a result of the transition to SA_PC_1:Active.

10.2.10.2.7.3 Transition SA_PC_5:Active_Wait to SA_PC_3:Standby

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is processed; or
- c) the Power Condition mode page standby condition timer expires.

10.2.10.2.7.4 Transition SA_PC_5:Active_Wait to SA_PC_4:Stopped

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to zero is processed.

10.2.10.2.7.5 Transition SA_PC_5:Active_Wait to SA_PC_6:Idle_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is processed; or
- c) the Power Condition mode page idle condition timer expires.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.8 SA_PC_6:Idle_Wait state**10.2.10.2.8.1 State description**

This state shall only be implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped. The device server shall be capable of processing commands and shall terminate each media access command or TEST UNIT READY command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED.

In response to a REQUEST SENSE command processed in this state, the device server shall return parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED and return GOOD status for the command.

10.2.10.2.8.2 Transition SA_PC_6:Idle_Wait to SA_PC_2:Idle

This transition shall occur if:

- a) a NOTIFY (ENABLE SPINUP) is detected; or
- b) the SAS device does not consume additional power as a result of the transition to SA_PC_2:Idle.

10.2.10.2.8.3 Transition SA_PC_6:Idle_Wait to SA_PC_3:Standby

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is processed;
- b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is processed; or
- c) the Power Condition mode page standby condition timer expires.

10.2.10.2.8.4 Transition SA_PC_6:Idle_Wait to SA_PC_4:Stopped

This transition shall occur if:

- a) a START STOP UNIT command with the START bit set to zero is processed.

10.2.10.2.8.5 Transition SA_PC_6:Idle_Wait to SA_PC_5:Active_Wait

This transition shall occur if:

- a) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is processed; or
- b) a command that requires the active power condition is processed.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command until this state machine reaches the SA_PC_1:Active state.

10.2.11 SCSI vital product data (VPD)

10.2.11.1 SCSI vital product data (VPD) overview

Table 307 lists VPD pages for which this standard defines special requirements.

Table 307 — VPD pages with special requirements for SAS SSP

Page code	VPD Page Name	Reference	Support Requirements
83h	Device Identification VPD page	10.2.11.2 and SPC-4	Mandatory
90h	Protocol-Specific Logical Unit Information VPD page	10.2.11.3 and SPC-4	See ^a
^a Mandatory if the target port and logical unit support the TLR CONTROL field set to a non-zero value in the SSP frame header, otherwise optional.			

10.2.11.2 Device Identification VPD page

In the Device Identification VPD page (83h) returned by the INQUIRY command (see SPC-4), each logical unit in a SAS target device shall include the designation descriptors for the target port identifier (see 4.2.7) and the relative target port identifier (see SAM-4 and SPC-4) listed in table 308.

Table 308 — Device Identification VPD page designation descriptors for the SAS target port

Field in designation descriptor	Designation descriptor	
	Target port identifier	Relative target port identifier
DESIGNATOR TYPE	3h (i.e., NAA)	4h (i.e., relative target port identifier)
ASSOCIATION	01b (i.e., SCSI target port)	01b (i.e., SCSI target port)
CODE SET	1h (i.e., binary)	1h (i.e., binary)
DESIGNATOR LENGTH	8	4
PIV (protocol identifier valid)	1	1
PROTOCOL IDENTIFIER	6h (i.e., SAS)	6h (i.e., SAS)
DESIGNATOR	SAS address ^a in NAA IEEE Registered format (see 4.2.2)	Relative port identifier ^b as described in SAM-4 and SPC-4
^a The DESIGNATOR field contains the SAS address of the SSP target port through which the INQUIRY command was received.		
^b The DESIGNATOR field contains the relative port identifier of the SSP target port through which the INQUIRY command was received.		

In the Device Identification VPD page (83h) returned by the INQUIRY command (see SPC-4), each logical unit in a SAS target device shall include a designation descriptor for the SAS target device name (see 4.2.4)

using NAA format and may include a designation descriptor for the SAS target device name using the SCSI name string format as listed in table 309.

Table 309 — Device Identification VPD page designation descriptors for the SAS target device

Field in designation descriptor	Designation descriptor for SAS target device	
	NAA format (required)	SCSI name string format (optional)
DESIGNATOR TYPE	3h (i.e., NAA)	8h (i.e., SCSI name string)
ASSOCIATION	10b (i.e., SCSI target device)	10b (i.e., SCSI target device)
CODE SET	1h (i.e., binary)	3h (i.e., UTF-8)
DESIGNATOR LENGTH	8	24
PIV (protocol identifier valid)	1	0
PROTOCOL IDENTIFIER	6h (i.e., SAS)	0h ^a
DESIGNATOR	Device name of the SAS target device in NAA IEEE Registered format (see 4.2.2)	Device name of the SAS target device in SCSI name string format (e.g., "naa." followed by 16 hexadecimal digits followed by 4 ASCII null characters)
^a The PROTOCOL IDENTIFIER field is reserved when the PIV bit is set to zero.		

Logical units may include designation descriptors in addition to those required by this standard (e.g., SCSI target devices with SCSI target ports using other SCSI transport protocols may return additional SCSI target device names for those other SCSI transport protocols).

10.2.11.3 Protocol-Specific Logical Unit Information VPD page

The Protocol-Specific Logical Unit Information VPD page (see SPC-4) contains parameters for the logical unit that are protocol-specific based on the I_T nexus being used to access the logical unit.

Table 310 defines the Protocol-Specific Logical Unit Information VPD page for SAS target ports.

Table 310 — Logical Unit Information VPD page for SAS SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)	RELATIVE PORT IDENTIFIER						(LSB)
1								
2		Reserved			PROTOCOL IDENTIFIER (6h)			
3		Reserved						
5								
6	(MSB)	DESCRIPTOR LENGTH (04h)						(LSB)
7								
Per logical unit SCSI transport specific data								
8		Reserved						TLR CONTROL SUPPORTED
9								
11		Reserved						

The RELATIVE PORT IDENTIFIER field is defined in SPC-4.

The PROTOCOL IDENTIFIER field is defined in SPC-4 and shall be set to 6h.

The DESCRIPTOR LENGTH field is defined in SPC-4 and shall be set to 04h.

A TLR CONTROL SUPPORTED bit set to one indicates that the combination of the SCSI target port and logical unit support the TLR CONTROL field in the SSP frame header (see 9.1). A TLR CONTROL SUPPORTED bit set to zero indicates that the combination of the SCSI target port and logical unit do not support the TLR CONTROL field in the SSP frame header.

10.3 ATA application layer

No SAS-specific ATA features are defined.

10.4 Management application layer

10.4.1 READY LED signal behavior

A SAS target device uses the READY LED signal to activate an externally visible LED that indicates the state of readiness and activity of the SAS target device. The READY LED signal electrical characteristics are described in 5.4. All SAS target devices using the SAS Drive plug connector (see 5.2.3.2.1.1) shall support the READY LED signal.

The system is not required to generate any visual output when the READY LED signal is asserted. Additional vendor-specific flashing patterns may be used to signal vendor-specific conditions.

SAS target devices without SSP target ports may transmit the READY LED signal using vendor-specific patterns.

SAS target devices with SSP target ports shall follow the READY LED MEANING bit in the Protocol-Specific Port mode page (see 10.2.7.4) as described in table 311.

Table 311 — READY LED signal behavior

Power condition ^a (see 10.2.10) or activity	READY LED MEANING bit set to zero ^b	READY LED MEANING bit set to one
Active or Idle power condition	The SAS target device shall: a) when not processing a command, assert the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually on, but flashes on and off when commands are processed.)	The SAS target device shall: a) when not processing a command, negate the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually off, but flashes on and off when commands are processed.)
Standby or Stopped power condition	The SAS target device shall: a) when not processing a command, negate the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually off, but flashes on and off when commands are processed.) After a vendor-specific amount of time in this state, SAS target devices with rotating media may be removed with minimum risk of mechanical or electrical damage.	
Spinup/spindown	If the SAS target device has rotating media and is in the process of performing a spin-up or spin-down, then the SAS target device shall toggle the READY LED signal between the asserted and negated states with a 1 s ± 0,1 s cycle using a 50 % ± 10 % duty cycle (e.g., the LED is on for 0,5 s and off for 0,5 s).	
Formatting media	If the SAS target device is in the process of formatting media, then the SAS target device shall toggle the READY LED signal between the asserted and negated states in a vendor-specified manner (e.g., with each cylinder change on a disk drive).	
^a If the SAS target device has more than one logical unit and any logical unit is active or idle, its power condition should be used to control the READY LED signal. ^b If the target device has rotating media, a READY LED MEANING bit set to zero results in a READY LED signal behavior that provides an indication of the target device's readiness for removal. A target device with rotating media that is not in a state for safe removal shall either toggle the READY LED signal at a significant rate during spin-up, during spin-down, and while formatting media, or assert the READY LED signal continuously. When removal is safe from a mechanical standpoint, the READY LED signal shall be deasserted.		

10.4.2 Management protocol services

The management application client and management device server use a four-step process to perform management functions:

- 1) The management application client invokes Send SMP Function;
- 2) The SMP target port invokes SMP Function Received;
- 3) The management device server invokes Send SMP Function Response; and
- 4) The SMP initiator port invokes Received SMP Function Complete.

10.4.3 SMP functions

10.4.3.1 SMP function request frame format

An SMP request frame is sent by a management application client via an SMP initiator port to request an SMP function be performed by a management device server. Table 312 defines the SMP request frame format.

Table 312 — SMP request frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION							
2	Reserved							
3	REQUEST LENGTH ((n - 7) / 4)							
4	ADDITIONAL REQUEST BYTES							
m								
	Fill bytes, if needed							
n - 3	(MSB)							
	CRC							
n	(LSB)							

The SMP FRAME TYPE field is included in each frame format defined in this clause, although that field is parsed by the SMP transport layer (see 9.4). The SMP FRAME TYPE field is set to 40h.

The FUNCTION field specifies which SMP function is being requested and is defined in table 313. If the value in the FUNCTION field is not supported by the management device server, it shall return a function result of UNKNOWN SMP FUNCTION as described in table 315.

Table 313 — SMP functions (FUNCTION field) (part 1 of 3)

Code	SMP function	Description	Reference
00h	REPORT GENERAL	Return general information about the device	10.4.3.3
01h	REPORT MANUFACTURER INFORMATION	Return vendor and product identification	10.4.3.4
02h	READ GPIO REGISTER	See SFF-8485	
03h	REPORT SELF-CONFIGURATION STATUS	Return status of the discover process in a self-configuring expander device	10.4.3.5

Table 313 — SMP functions (FUNCTION field) (part 2 of 3)

Code	SMP function	Description	Reference
04h	REPORT ZONE PERMISSION TABLE	Return zone permission table values	10.4.3.6
05h	REPORT ZONE MANAGER PASSWORD	Return the zone manager password	10.4.3.7
06h	REPORT BROADCAST	Return information about Broadcast counters	10.4.3.8
07h - 0Fh	Reserved for general SMP input functions		
10h	DISCOVER	Return information about the specified phy	10.4.3.9
11h	REPORT PHY ERROR LOG	Return error logging information about the specified phy	10.4.3.10
12h	REPORT PHY SATA	Return information about a phy currently attached to a SATA phy	10.4.3.11
13h	REPORT ROUTE INFORMATION	Return phy-based expander route table information	10.4.3.12
14h	REPORT PHY EVENT	Return phy events for the specified phy	10.4.3.13
15h - 1Fh	Reserved for phy-based SMP input functions		
20h	DISCOVER LIST	Return information about the specified phys	10.4.3.14
21h	REPORT PHY EVENT LIST	Return phy events	10.4.3.15
22h	REPORT EXPANDER ROUTE TABLE LIST	Return contents of the expander-based expander route table	10.4.3.16
23h - 2Fh	Reserved for descriptor list-based SMP input functions		
30h - 3Fh	Reserved for SMP input functions		
40h - 7Fh	Vendor specific		
80h	CONFIGURE GENERAL	Configure the device	10.4.3.17
81h	ENABLE DISABLE ZONING	Enable or disable zoning	10.4.3.18
82h	WRITE GPIO REGISTER	See SFF-8485	
83h - 84h	Reserved for general SMP output functions		
85h	ZONED BROADCAST	Transmit the specified Broadcast on the expander ports in the specified zone group(s)	10.4.3.19
86h	ZONE LOCK	Lock a zoning expander device	10.4.3.20
87h	ZONE ACTIVATE	Set the zoning expander active values equal to the zoning expander shadow values	10.4.3.21
88h	ZONE UNLOCK	Unlock a zoning expander device	10.4.3.22
89h	CONFIGURE ZONE MANAGER PASSWORD	Configure the zone manager password	10.4.3.23
8Ah	CONFIGURE ZONE PHY INFORMATION	Configure zone phy information	10.4.3.24
8Bh	CONFIGURE ZONE PERMISSION TABLE	Configure the zone permission table	10.4.3.25
8Ch - 8Fh	Reserved for general SMP output functions		

Table 313 — SMP functions (FUNCTION field) (part 3 of 3)

Code	SMP function	Description	Reference
90h	CONFIGURE ROUTE INFORMATION	Change phy-based expander route table information	10.4.3.26
91h	PHY CONTROL	Request actions by the specified phy	10.4.3.27
92h	PHY TEST FUNCTION	Request a test function by the specified phy	10.4.3.28
93h	CONFIGURE PHY EVENT	Configure phy events for the specified phy	10.4.3.29
94h - 9Fh	Reserved for phy-based SMP output functions		
A0h - BFh	Reserved for SMP output functions		
C0h - FFh	Vendor specific		

The REQUEST LENGTH field specifies the number of dwords that follow, not including the CRC field. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h sometimes specifies a non-zero number of dwords; this is defined in the function description. If the request frame size including the CRC field is less than 8 bytes, or the REQUEST LENGTH field does not match the request frame size, the management device server shall return a function result of INVALID REQUEST FRAME LENGTH. The management device server shall consider any fields not included in the request frame to be set to zero.

The ADDITIONAL REQUEST BYTES field definition and length are based on the SMP function. The maximum size of the ADDITIONAL REQUEST BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL REQUEST BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is included in each request frame format defined in this clause, although that field is defined by the SMP transport layer (see 9.4.1) and parsed by the SMP link layer (see 7.18).

10.4.3.2 SMP function response frame format

An SMP response frame is sent by a management device server via an SMP target port in response to an SMP request frame. Table 314 defines the SMP response frame format.

Table 314 — SMP response frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION							
2	FUNCTION RESULT							
3	RESPONSE LENGTH ((n - 7) / 4)							
4	ADDITIONAL RESPONSE BYTES							
m								
	Fill bytes, if needed							
n - 3	(MSB)							
	CRC							
n								

The SMP FRAME TYPE field is included in each frame format defined in this clause, although that field is parsed by the SMP transport layer (see 9.4). The SMP FRAME TYPE field is set to 41h.

The FUNCTION field indicates the SMP function to which this frame is a response, and is defined in table 313 in 10.4.3.1.

The FUNCTION RESULT field is defined in table 315.

Table 315 — FUNCTION RESULT field (part 1 of 4)

Code	Name	SMP function(s)	Description
00h	SMP FUNCTION ACCEPTED	All	The management device server supports the SMP function.
01h	UNKNOWN SMP FUNCTION	Unknown	The management device server does not support the requested SMP function.
02h	SMP FUNCTION FAILED	All	The requested SMP function failed.
03h	INVALID REQUEST FRAME LENGTH	All	The SMP request frame length was invalid (see 10.4.3.1).
04h	INVALID EXPANDER CHANGE COUNT	CONFIGURE GENERAL, ENABLE DISABLE ZONING, ZONE LOCK, ZONE ACTIVATE, CONFIGURE ZONE MANAGER PASSWORD, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE, CONFIGURE ROUTE INFORMATION, PHY CONTROL, PHY TEST FUNCTION, CONFIGURE PHY EVENT	The management device server supports the SMP function, but the EXPECTED EXPANDER CHANGE COUNT field does not match the current expander change count.
05h	BUSY	ZONE UNLOCK, ENABLE DISABLE ZONING, CONFIGURE ZONE MANAGER PASSWORD, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE	For ZONE UNLOCK, the locked zoning expander device is processing the activate step. For the other functions, the management device server is currently saving zoning values.

Table 315 — FUNCTION RESULT field (part 2 of 4)

Code	Name	SMP function(s)	Description
06h	INCOMPLETE DESCRIPTOR LIST	ZONED BROADCAST, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE, CONFIGURE PHY EVENT INFORMATION	The request frame length results in the truncation of a multi-byte field or descriptor list (e.g., in the ZONED BROADCAST request, the request frame is not large enough to contain the number of broadcast source zone groups specified by the NUMBER OF BROADCAST SOURCE ZONE GROUPS field).
10h	PHY DOES NOT EXIST	DISCOVER, REPORT PHY ERROR LOG, REPORT PHY SATA, REPORT ROUTE INFORMATION, REPORT PHY EVENT, DISCOVER LIST, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ROUTE INFORMATION, PHY CONTROL, PHY TEST FUNCTION, CONFIGURE PHY EVENT	The phy specified by the PHY IDENTIFIER field or the STARTING PHY IDENTIFIER field in the SMP request frame does not exist (e.g., the value is not within the range of zero to the value of the NUMBER OF PHYs field reported in the SMP REPORT GENERAL response).
11h	INDEX DOES NOT EXIST	REPORT ROUTE INFORMATION, CONFIGURE ROUTE INFORMATION	The phy specified by the PHY IDENTIFIER field in the SMP request frame does not have the table routing attribute (see 4.6.7.1), or the expander route index specified by the EXPANDER ROUTE INDEX field does not exist (i.e., the value is not in the range of 0000h to the value of the EXPANDER ROUTE INDEXES field in the SMP REPORT GENERAL response). The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
12h	PHY DOES NOT SUPPORT SATA	REPORT PHY SATA, PHY CONTROL	See 10.4.3.11 and 10.4.3.27
13h	UNKNOWN PHY OPERATION	PHY CONTROL	See 10.4.3.27
14h	UNKNOWN PHY TEST FUNCTION	PHY TEST FUNCTION	See 10.4.3.28
15h	PHY TEST FUNCTION IN PROGRESS	PHY TEST FUNCTION	See 10.4.3.28

Table 315 — FUNCTION RESULT field (part 3 of 4)

Code	Name	SMP function(s)	Description
16h	PHY VACANT	DISCOVER, REPORT PHY ERROR LOG, REPORT PHY SATA, REPORT ROUTE INFORMATION, REPORT PHY EVENT, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ROUTE INFORMATION, PHY CONTROL, PHY TEST FUNCTION, CONFIGURE PHY EVENT	The management device server processing the SMP request frame does not have access to a specified phy (e.g., because of zoning or vendor-specific reasons), although the value is within the range of zero to the value of the NUMBER OF PHYS field reported in the SMP REPORT GENERAL response.
17h	UNKNOWN PHY EVENT SOURCE	CONFIGURE PHY EVENT	See 10.4.3.29.3
18h	UNKNOWN DESCRIPTOR TYPE	DISCOVER LIST	The descriptor type specified by the DESCRIPTOR TYPE field is not supported.
19h	UNKNOWN PHY FILTER	DISCOVER LIST	The phy filter specified by the PHY FILTER field is not supported.
1Ah	AFFILIATION VIOLATION	PHY CONTROL	The specified phy operation is not allowed due to the current state of affiliations.
20h	SMP ZONE VIOLATION	CONFIGURE GENERAL, ZONED BROADCAST, PHY CONTROL, PHY TEST FUNCTION, CONFIGURE PHY EVENT	Zoning is enabled and the SMP initiator port does not have access to a necessary zone group according to the zone permission table (see 4.9.3.2).
21h	NO MANAGEMENT ACCESS RIGHTS	REPORT ZONE MANAGER PASSWORD, ZONE LOCK, CONFIGURE ZONE MANAGER PASSWORD	For ZONE LOCK, any of the following are true: a) zoning is enabled, the ZONE LOCK bit is set to zero, the PHYSICAL PRESENCE bit is set to zero, the ZONE MANAGER PASSWORD field is not set to the current zone manager password, and the zone manager does not have access to zone group 2; b) zoning is enabled, the ZONE LOCK bit is set to one, and the request did not originate from the active zone manager; or c) zoning is disabled, the PHYSICAL PRESENCE bit is set to zero, the ZONE MANAGER PASSWORD field is not set to the current zone manager password. For REPORT ZONE MANAGER PASSWORD, see 10.4.3.7. For CONFIGURE ZONE MANAGER PASSWORD, see 10.4.3.23.

Table 315 — FUNCTION RESULT field (part 4 of 4)

Code	Name	SMP function(s)	Description
22h	UNKNOWN ENABLE DISABLE ZONING VALUE	ENABLE DISABLE ZONING	See 10.4.3.18
23h	ZONE LOCK VIOLATION	ENABLE DISABLE ZONING, ZONE LOCK, ZONE ACTIVATE, ZONE UNLOCK, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE	Zoning is enabled and: a) an unlocked zoning expander device receives an SMP zone configuration function request, a ZONE ACTIVATE request, or a ZONE UNLOCK request from an SMP initiator port that is not the active zone manager; or b) a locked zoning expander device receives an SMP ZONE LOCK request, an SMP zone configuration function request, a ZONE ACTIVATE request, or a ZONE UNLOCK request from an SMP initiator port that is not the active zone manager.
24h	NOT ACTIVATED	ZONE UNLOCK	The ACTIVATE REQUIRED bit in the request is set to one but the locked zoning expander device has not processed the activate step.
25h	ZONE GROUP OUT OF RANGE	CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE	The ZONE GROUP field or NUMBER OF ZONE GROUPS field contains a value that is not supported.
26h	NO PHYSICAL PRESENCE	CONFIGURE ZONE MANAGER PASSWORD	The new ZONE MANAGER PASSWORD field is set to DISABLED (see table 26 in 4.9.1) but physical presence is not asserted.
27h	SAVING NOT SUPPORTED	ENABLE DISABLE ZONING, CONFIGURE ZONE MANAGER PASSWORD, CONFIGURE ZONE PHY INFORMATION, CONFIGURE ZONE PERMISSION TABLE	The SAVE field is set to 01b or 11b and the management device server does not support saved values for the specified information.
28h	SOURCE ZONE GROUP DOES NOT EXIST	REPORT ZONE PERMISSION TABLE	See 10.4.3.6
All others	Reserved		

Table 316 defines the priority of the SMP function results defined in table 315.

Table 316 — Function result priority (part 1 of 5)

SMP function	SMP function result priority
REPORT GENERAL (see 10.4.3.3)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED

Table 316 — Function result priority (part 2 of 5)

SMP function	SMP function result priority
REPORT MANUFACTURER INFORMATION (see 10.4.3.4)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
READ GPIO REGISTER (see SFF-8485)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
REPORT SELF-CONFIGURATION STATUS (see 10.4.3.5)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
REPORT ZONE PERMISSION TABLE (see 10.4.3.6)	1) INVALID REQUEST FRAME LENGTH; 2) SOURCE ZONE GROUP DOES NOT EXIST; 3) SMP FUNCTION FAILED; and 4) SMP FUNCTION ACCEPTED
REPORT ZONE MANAGER PASSWORD (see 10.4.3.7)	1) INVALID REQUEST FRAME LENGTH; 2) NO MANAGEMENT ACCESS RIGHTS; 3) SMP FUNCTION FAILED; and 4) SMP FUNCTION ACCEPTED
REPORT BROADCAST (see 10.4.3.8)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
DISCOVER (see 10.4.3.9)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED
REPORT PHY ERROR LOG (see 10.4.3.10)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED
REPORT PHY SATA (see 10.4.3.11)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) PHY DOES NOT SUPPORT SATA; 5) SMP FUNCTION FAILED; and 6) SMP FUNCTION ACCEPTED
REPORT ROUTE INFORMATION (see 10.4.3.12)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) INDEX DOES NOT EXIST; 5) SMP FUNCTION FAILED; and 6) SMP FUNCTION ACCEPTED
REPORT PHY EVENT (see 10.4.3.13)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED

Table 316 — Function result priority (part 3 of 5)

SMP function	SMP function result priority
DISCOVER LIST (see 10.4.3.14)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) UNKNOWN DESCRIPTOR TYPE; 4) UNKNOWN PHY FILTER; 5) SMP FUNCTION FAILED; and 6) SMP FUNCTION ACCEPTED
REPORT PHY EVENT LIST (see 10.4.3.15)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
REPORT EXPANDER ROUTE TABLE LIST (see 10.4.3.16)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
CONFIGURE GENERAL (see 10.4.3.17)	1) INVALID REQUEST FRAME LENGTH; 2) SMP ZONE VIOLATION; 3) INVALID EXPANDER CHANGE COUNT; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED
WRITE GPIO REGISTER (see SFF-8485)	1) INVALID REQUEST FRAME LENGTH; 2) SMP FUNCTION FAILED; and 3) SMP FUNCTION ACCEPTED
ENABLE DISABLE ZONING (see 10.4.3.18)	1) INVALID REQUEST FRAME LENGTH; 2) ZONE LOCK VIOLATION; 3) UNKNOWN ENABLE DISABLE ZONING VALUE; 4) INVALID EXPANDER CHANGE COUNT; 5) SAVING NOT SUPPORTED; 6) SMP FUNCTION FAILED; and 7) SMP FUNCTION ACCEPTED
ZONED BROADCAST (see 10.4.3.19)	1) INVALID REQUEST FRAME LENGTH; 2) INCOMPLETE DESCRIPTOR LIST; 3) SMP ZONE VIOLATION; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED
ZONE LOCK (see 10.4.3.20)	1) INVALID REQUEST FRAME LENGTH; 2) ZONE LOCK VIOLATION; 3) NO MANAGEMENT ACCESS RIGHTS; 4) INVALID EXPANDER CHANGE COUNT; 5) SMP FUNCTION FAILED; and 6) SMP FUNCTION ACCEPTED
ZONE ACTIVATE (see 10.4.3.21)	1) INVALID REQUEST FRAME LENGTH; 2) ZONE LOCK VIOLATION; 3) INVALID EXPANDER CHANGE COUNT; 4) SMP FUNCTION FAILED; and 5) SMP FUNCTION ACCEPTED

Table 316 — Function result priority (part 4 of 5)

SMP function	SMP function result priority
ZONE UNLOCK (see 10.4.3.22)	1) INVALID REQUEST FRAME LENGTH; 2) ZONE LOCK VIOLATION; 3) NOT ACTIVATED; 4) BUSY; 5) SMP FUNCTION FAILED; and 6) SMP FUNCTION ACCEPTED
CONFIGURE ZONE MANAGER PASSWORD (see 10.4.3.23)	1) INVALID REQUEST FRAME LENGTH; 2) INVALID EXPANDER CHANGE COUNT; 3) NO MANAGEMENT ACCESS RIGHTS; 4) NO PHYSICAL PRESENCE; 5) SAVING NOT SUPPORTED; 6) SMP FUNCTION FAILED; and 7) SMP FUNCTION ACCEPTED
CONFIGURE ZONE PHY INFORMATION (see 10.4.3.24)	1) INVALID REQUEST FRAME LENGTH; 2) INCOMPLETE DESCRIPTOR LIST; 3) PHY DOES NOT EXIST; 4) PHY VACANT; 5) ZONE LOCK VIOLATION; 6) INVALID EXPANDER CHANGE COUNT; 7) SAVING NOT SUPPORTED; 8) ZONE GROUP OUT OF RANGE; 9) SMP FUNCTION FAILED; and 10) SMP FUNCTION ACCEPTED
CONFIGURE ZONE PERMISSION TABLE (see 10.4.3.25)	1) INVALID REQUEST FRAME LENGTH; 2) INCOMPLETE DESCRIPTOR LIST; 3) ZONE LOCK VIOLATION; 4) INVALID EXPANDER CHANGE COUNT; 5) SAVING NOT SUPPORTED; 6) ZONE GROUP OUT OF RANGE; 7) SMP FUNCTION FAILED; and 8) SMP FUNCTION ACCEPTED
CONFIGURE ROUTE INFORMATION (see 10.4.3.26)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) INVALID EXPANDER CHANGE COUNT; 5) INDEX DOES NOT EXIST; 6) SMP FUNCTION FAILED; and 7) SMP FUNCTION ACCEPTED
PHY CONTROL (see 10.4.3.27)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) SMP ZONE VIOLATION; 5) INVALID EXPANDER CHANGE COUNT; 6) UNKNOWN PHY OPERATION; 7) PHY DOES NOT SUPPORT SATA; 8) AFFILIATION VIOLATION; 9) SMP FUNCTION FAILED; and 10) SMP FUNCTION ACCEPTED

Table 316 — Function result priority (part 5 of 5)

SMP function	SMP function result priority
PHY TEST FUNCTION (see 10.4.3.28)	1) INVALID REQUEST FRAME LENGTH; 2) PHY DOES NOT EXIST; 3) PHY VACANT; 4) SMP ZONE VIOLATION; 5) INVALID EXPANDER CHANGE COUNT; 6) UNKNOWN PHY TEST FUNCTION; 7) PHY TEST FUNCTION IN PROGRESS; 8) SMP FUNCTION FAILED; and 9) SMP FUNCTION ACCEPTED
CONFIGURE PHY EVENT (see 10.4.3.29)	1) INVALID REQUEST FRAME LENGTH; 2) INCOMPLETE DESCRIPTOR LIST; 3) PHY DOES NOT EXIST; 4) PHY VACANT; 5) SMP ZONE VIOLATION; 6) INVALID EXPANDER CHANGE COUNT; 7) UNKNOWN PHY EVENT SOURCE; 8) SMP FUNCTION FAILED; and 9) SMP FUNCTION ACCEPTED

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h sometimes indicates a non-zero number of dwords; this is defined in the function description.

If the FUNCTION RESULT field is set to 00h, then the ADDITIONAL RESPONSE BYTES field definition depends on the SMP function requested. If the FUNCTION RESULT field is set to a value other than 00h, then the ADDITIONAL RESPONSE BYTES field may be present but shall be ignored. The maximum size of the ADDITIONAL RESPONSE BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL RESPONSE BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is included in each response frame format defined in this clause, although that field is defined by the SMP transport layer (see 9.4.1) and parsed by the SMP link layer (see 7.18).

10.4.3.3 REPORT GENERAL function

The REPORT GENERAL function returns general information about the SAS device (e.g., a SAS device contained in an expander device). This SMP function shall be implemented by all management device servers.

Table 317 defines the request format.

Table 317 — REPORT GENERAL request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (00h)							
2	Reserved							
3	REQUEST LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 00h.

The REQUEST LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.1.

Table 318 defines the response format.

Table 318 — REPORT GENERAL response (part 1 of 3)

Byte\Bit	7	6	5	4	3	2	1	0						
0	SMP FRAME TYPE (41h)													
1	FUNCTION (00h)													
2	FUNCTION RESULT													
3	RESPONSE LENGTH (10h)													
4	(MSB)	EXPANDER CHANGE COUNT												
5								(LSB)						
6	(MSB)	EXPANDER ROUTE INDEXES												
7								(LSB)						
8	Reserved													
9	NUMBER OF PHYS													
10	TABLE TO TABLE SUPPORTED	Reserved			CONFIGURES OTHERS	CONFIGURING	EXTERNALLY CONFIGURABLE ROUTE TABLE							
11	Reserved													
12	ENCLOSURE LOGICAL IDENTIFIER													
19														

Table 318 — REPORT GENERAL response (part 2 of 3)

Byte\Bit	7	6	5	4	3	2	1	0
20	Reserved							
29								
30	(MSB)	STP BUS INACTIVITY TIME LIMIT						(LSB)
31								
32	(MSB)	STP MAXIMUM CONNECT TIME LIMIT						(LSB)
33								
34	(MSB)	STP SMP I_T NEXUS LOSS TIME						(LSB)
35								
36	NUMBER OF ZONE GROUPS		Reserved	ZONE LOCKED	PHYSICAL PRESENCE SUPPORTED	PHYSICAL PRESENCE ASSERTED	ZONING SUPPORTED	ZONING ENABLED
37	Reserved			SAVING	SAVING ZONE MANAGER PASSWORD SUPPORTED	SAVING ZONE PHY INFORMATION SUPPORTED	SAVING ZONE PERMISSION TABLE SUPPORTED	SAVING ZONING ENABLED SUPPORTED
38	(MSB)	MAXIMUM NUMBER OF ROUTED SAS ADDRESSES						(LSB)
39								
40	ACTIVE ZONE MANAGER SAS ADDRESS						(LSB)	
47								
48	(MSB)	ZONE LOCK INACTIVITY TIME LIMIT						(LSB)
49								
50	Reserved							
51								
52	Reserved							
53	FIRST ENCLOSURE CONNECTOR ELEMENT INDEX							
54	NUMBER OF ENCLOSURE CONNECTOR ELEMENT INDEXES							
55	Reserved							
56	REDUCED FUNCTIONALITY	Reserved						
57	TIME TO REDUCED FUNCTIONALITY							
58	INITIAL TIME TO REDUCED FUNCTIONALITY							
59	MAXIMUM REDUCED FUNCTIONALITY TIME							

Table 318 — REPORT GENERAL response (part 3 of 3)

Byte\Bit	7	6	5	4	3	2	1	0
60	(MSB)	LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX						
61								(LSB)
62	(MSB)	MAXIMUM NUMBER OF STORED SELF-CONFIGURATION STATUS DESCRIPTORS						
63								(LSB)
64	(MSB)	LAST PHY EVENT LIST DESCRIPTOR INDEX						
65								(LSB)
66	(MSB)	MAXIMUM NUMBER OF STORED PHY EVENT LIST DESCRIPTORS						
67								(LSB)
68	(MSB)	CRC						
71								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 00h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 10h. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 6 dwords before the CRC field.

The EXPANDER CHANGE COUNT field counts the number of Broadcast (Change)s originated by an expander device (see 7.11). Management device servers in expander devices shall support this field. Management device servers in other device types (e.g., end devices) shall set this field to 0000h. This field shall be set to at least 0001h at power on. If the expander device has originated Broadcast (Change) for any reason described in 7.11 since transmitting any SMP response frame containing an EXPANDER CHANGE COUNT field, it shall increment this field at least once from the value in the previous REPORT GENERAL response. It shall not increment this field when forwarding a Broadcast (Change). This field shall wrap to at least 0001h after the maximum value (i.e., FFFFh) has been reached.

NOTE 93 - Application clients that use the EXPANDER CHANGE COUNT field should read it often enough to ensure that it does not increment a multiple of 65 535 times between reading the field in an expander device compliant with this standard or a multiple of 65 536 times between reading the field in an expander device compliant with previous versions of this standard.

NOTE 94 - Management device servers in expander devices compliant with previous versions of this standard may return an EXPANDER CHANGE COUNT field set to 0000h.

NOTE 95 - The originated Broadcast (Change) count is also reported in the REPORT BROADCAST response (see 10.4.3.8).

The EXPANDER ROUTE INDEXES field indicates the maximum number of expander route indexes per phy for the expander device (see 4.6.7.3). Management device servers in externally configurable expander devices containing phy-based expander route tables shall support this field. Management device servers in other device types (e.g., end devices, externally configurable expander devices with expander-based expander route tables, and self-configuring expander devices) shall set the EXPANDER ROUTE INDEXES field to zero. Not all phys in an externally configurable expander device are required to support the maximum number indicated by this field.

The NUMBER OF PHYS field indicates the number of phys in the device, including any virtual phys and any vacant phys.

A TABLE TO TABLE SUPPORTED bit set to one indicates the expander device is a self-configuring expander device that supports its table routing phys being attached to table routing phys in other expander devices. The TABLE TO TABLE SUPPORTED bit shall only be set to one if the EXTERNALLY CONFIGURABLE ROUTE TABLE bit is set to zero. A TABLE TO TABLE SUPPORTED bit set to zero indicates the expander device is not a self-configuring expander device that supports its table routing phys being attached to table routing phys in other expander devices.

A CONFIGURES OTHERS bit set to one indicates that the expander device is a self-configuring expander device that performs the configuration subprocess defined in 4.8. A CONFIGURES OTHERS bit set to zero indicates the expander device may or may not perform the configuration subprocess. Self-configuring expander devices compliant with this standard shall set the CONFIGURES OTHERS bit to one.

NOTE 96 - If the CONFIGURES OTHERS bit is set to zero, the expander device may configure all externally configurable expander devices in the SAS domain.

A CONFIGURING bit set to one indicates that either:

- a) the management device server is in a self-configuring expander device, the self-configuring expander device's management application client is currently performing the discover process (see 4.7), and it has identified at least one change to its expander routing table; or
- b) the zoning expander device is locked and the zoning expander shadow values differ from the zoning expander active values.

A CONFIGURING bit set to zero indicates that the management device server is not in a self-configuring expander device currently performing the discover process and changing its expander routing table. Changes in this bit from one to zero result in a Broadcast (Change) being originated (see 7.11). Management device servers in self-configuring expander devices shall support this bit. Management device servers in externally configurable expander devices and in other device types shall set the CONFIGURING bit to zero.

An EXTERNALLY CONFIGURABLE ROUTE TABLE bit set to one indicates that the management device server is in an externally configurable expander device that has a phy-based expander route table that is required to be configured with the SMP CONFIGURE ROUTE INFORMATION function (see 4.6.7.3). An EXTERNALLY CONFIGURABLE ROUTE TABLE bit set to zero indicates that the management device server is not in an externally configurable expander device (e.g., it is in an end device, in a self-configuring expander device, or in an expander device with no phys with table routing attributes).

The ENCLOSURE LOGICAL IDENTIFIER field identifies the enclosure, if any, in which the device is located, and is defined in SES-2. The ENCLOSURE LOGICAL IDENTIFIER field shall be set to the same value reported by the enclosure services process, if any, for the enclosure. An ENCLOSURE LOGICAL IDENTIFIER field set to zero indicates no enclosure information is available.

The STP BUS INACTIVITY TIME LIMIT field indicates the bus inactivity time limit for STP connections, which is set by the CONFIGURE GENERAL function (see 10.4.3.17).

The STP MAXIMUM CONNECT TIME LIMIT field indicates the maximum connect time limit for STP connections, which is set by the CONFIGURE GENERAL function (see 10.4.3.17).

The STP SMP I_T NEXUS LOSS TIME field indicates the time that an STP target port and an SMP initiator port retry certain connection requests which is set by the CONFIGURE GENERAL function (see 10.4.3.17).

Editor's Note 54: That should be a "minimum" time

The number of zone groups field indicates the number of zone groups (e.g., the number of entries in the zone group permission table) supported by the expander device and is defined in table 319.

Table 319 — NUMBER OF ZONE GROUPS field

Code	Description
00b	128 zone groups
01b	256 zone groups
All others	Reserved

A ZONE LOCKED bit set to one indicates that the zoning expander device is locked (see 4.9.6.2). A ZONE LOCKED bit set to zero indicates that the zoning expander device is not locked.

A PHYSICAL PRESENCE SUPPORTED bit set to one indicates that the expander device supports physical presence as a mechanism for allowing locking from phys in zone groups without access to zone group 2. A PHYSICAL PRESENCE SUPPORTED bit set to zero indicates that the expander device does not support physical presence as a mechanism for allowing locking.

A PHYSICAL PRESENCE ASSERTED bit set to one indicates that the expander device is currently detecting physical presence. A PHYSICAL PRESENCE ASSERTED bit set to zero indicates that the expander device is not currently detecting physical presence. The PHYSICAL PRESENCE ASSERTED bit shall be set to zero if the PHYSICAL PRESENCE SUPPORTED bit is set to zero.

A ZONING SUPPORTED bit set to one indicates that zoning is supported by the expander device (i.e., it is a zoning expander device). A ZONING SUPPORTED bit set to zero indicates that zoning is not supported by the expander device.

A ZONING ENABLED bit set to one indicates that zoning is enabled in the expander device. A ZONING ENABLED bit set to zero indicates that zoning is disabled in the expander device. The ZONING ENABLED bit shall be set to zero if the ZONING SUPPORTED bit is set to zero.

A SAVING bit set to one indicates that the management device server is currently saving zoning values to non-volatile storage and may return a function result of BUSY for zone management functions that access saved zoning values. A SAVING bit set to zero indicates that the management device server is not currently saving zoning values to non-volatile storage.

A SAVING ZONE MANAGER PASSWORD SUPPORTED bit set to one indicates that saving the zone manager password is supported. A SAVING ZONE MANAGER PASSWORD SUPPORTED bit set to zero indicates that saving the zone manager password is not supported.

A SAVING ZONE PHY INFORMATION SUPPORTED bit set to one indicates that saving the zone phy information is supported. A SAVING ZONE PHY INFORMATION SUPPORTED bit set to zero indicates that saving the zone phy information is not supported.

A SAVING ZONE PERMISSION TABLE SUPPORTED bit set to one indicates that saving the zone permission table is supported. A SAVING ZONE PERMISSION TABLE SUPPORTED bit set to zero indicates that saving the zone permission table is not supported.

A SAVING ZONING ENABLE SUPPORTED bit set to one indicates that saving the ZONING ENABLED bit is supported. A SAVING ZONING ENABLE SUPPORTED bit set to zero indicates that saving the ZONING ENABLED bit is not supported.

The MAXIMUM NUMBER OF ROUTED SAS ADDRESSES field indicates the number of routed SAS addresses in an expander-based expander route table (see 4.6.7.3 and 4.9.3.4). Management device servers in expander devices containing expander-based expander route tables shall support this field. Management device servers in other device types (e.g., end devices and expander devices with phy-based expander route tables) shall set this field to 0000h.

The ACTIVE ZONE MANAGER SAS ADDRESS field indicates the SAS address of the zone manager that last locked the zoning expander device. If the zoning expander device is currently being configured by a vendor-specific

sideband method then the ACTIVE ZONE MANAGER SAS ADDRESS field shall be set to zero. This field shall be set to zero at power on.

The ZONE LOCK INACTIVITY TIME LIMIT field indicates the minimum time between any SMP ZONE LOCK requests, SMP zone configuration function requests, or SMP ZONE ACTIVATE requests from the active zone manager that the locked expander device allows and is set in the SMP ZONE LOCK request (see 10.4.3.20).

The FIRST ENCLOSURE CONNECTOR ELEMENT INDEX field indicates the lowest CONNECTOR ELEMENT INDEX field of all the expander phys in all the expander devices in the enclosure that have CONNECTOR TYPE fields set to 20h through 2Fh (i.e., an internal connector to an end device) in their SMP DISCOVER responses.

The NUMBER OF ENCLOSURE CONNECTOR ELEMENT INDEXES field indicates the number of expander phys in all the expander devices in the enclosure that have CONNECTOR TYPE fields set to 20h through 2Fh (i.e., an internal connector to an end device) in their SMP DISCOVER responses.

NOTE 97 - The NUMBER OF ENCLOSURE CONNECTOR ELEMENT INDEXES field assumes that all internal connectors to end devices are assigned to a contiguous range of CONNECTOR ELEMENT INDEX field values.

A REDUCED FUNCTIONALITY bit set to one indicates that:

- a) the expander device is scheduled to reduce its functionality (see 4.6.8) in the time indicated in the TIME TO REDUCED FUNCTIONALITY field; or
- b) that the expander device is currently operating with reduced functionality (see 4.6.8).

A REDUCED FUNCTIONALITY bit set to zero indicates the expander device is not scheduled to reduce functionality and that the contents of the TIME TO REDUCED FUNCTIONALITY field shall be ignored.

If the REDUCED FUNCTIONALITY bit set to one, then the TIME TO REDUCED FUNCTIONALITY field indicates the time, in 100 ms increments, remaining until the expander device is scheduled to reduce functionality. The expander device starts the reduced functionality delay timer after originating a Broadcast (Expander) (see 4.6.8).

The INITIAL TIME TO REDUCED FUNCTIONALITY field indicates the minimum period of time, in 100 ms increments, that an expander device waits from originating a Broadcast (Expander) to reducing functionality. The expander device should set the default value for the INITIAL TIME TO REDUCED FUNCTIONALITY field to at least 2 000 ms (i.e., 14h).

The MAXIMUM REDUCED FUNCTIONALITY TIME field indicates the maximum time, in seconds, that the expander device responds with OPEN_REJECT (RETRY) to connection requests that map to an expander phy or an SMP target port that is not accessible during expander device reduced functionality. This timer starts after the reduced functionality delay timer expires.

The LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field is defined in the REPORT SELF-CONFIGURATION STATUS response (see 10.4.3.5).

The MAXIMUM NUMBER OF STORED SELF-CONFIGURATION STATUS DESCRIPTORS field indicates the maximum number of self-configuration status descriptors (see 10.4.3.5.4) that the management device server supports.

The LAST PHY EVENT LIST DESCRIPTOR INDEX field is defined in the REPORT PHY EVENT LIST response (see 10.4.3.15).

The MAXIMUM NUMBER OF STORED PHY EVENT LIST DESCRIPTORS field indicates the maximum number of phy event list descriptors (see 10.4.3.13.4) that the management device server supports.

The CRC field is defined in 10.4.3.2.

10.4.3.4 REPORT MANUFACTURER INFORMATION function

The REPORT MANUFACTURER INFORMATION function returns vendor and product identification. This SMP function may be implemented by any management device server.

Table 320 defines the request format.

Table 320 — REPORT MANUFACTURER INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (01h)							
2	Reserved							
3	REQUEST LENGTH (00h)							
4	(MSB)							
7	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 01h.

The REQUEST LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.1.

Table 321 defines the response format.

Table 321 — REPORT MANUFACTURER INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (01h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (0Eh)							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6		Reserved						
7								
8		Reserved						SAS-1.1 FORMAT
9		Reserved						
11								
12	(MSB)	VENDOR IDENTIFICATION						(LSB)
19								
20	(MSB)	PRODUCT IDENTIFICATION						(LSB)
35								
36	(MSB)	PRODUCT REVISION LEVEL						(LSB)
39								
40	(MSB)	COMPONENT VENDOR IDENTIFICATION						(LSB)
47								
48	(MSB)	COMPONENT ID						(LSB)
49								
50	COMPONENT REVISION LEVEL							
51	Reserved							
52	Vendor specific							
59								
60	(MSB)	CRC						(LSB)
63								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 01h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 0Eh. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 14 dwords before the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

A SAS-1.1 FORMAT bit set to one indicates that bytes 40 through 59 are as defined in this standard. A SAS-1.1 FORMAT bit set to zero indicates that bytes 40 through 59 are vendor-specific as defined in the original version of this standard.

ASCII data fields (e.g., the VENDOR IDENTIFICATION field, the PRODUCT IDENTIFICATION field, and PRODUCT REVISION LEVEL field, and the COMPONENT VENDOR IDENTIFICATION field) shall contain only graphic codes (i.e., code values 20h through 7Eh). Left-aligned fields shall place any unused bytes at the end of the field (i.e., at the highest offset) and the unused bytes shall be filled with space characters (i.e., 20h).

The VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the subsystem (e.g., the board or enclosure) containing the component. The data shall be left-aligned within the field. The vendor identification string shall be one assigned by INCITS for use in the standard INQUIRY data VENDOR IDENTIFICATION field. A list of assigned vendor identification strings is in SPC-4 and on the T10 web site (<http://www.t10.org>).

The PRODUCT IDENTIFICATION field contains sixteen bytes of ASCII data identifying the type of the subsystem (e.g., the board or enclosure model number) containing the component, as defined by the vendor of the subsystem. The data shall be left-aligned within the field. The PRODUCT IDENTIFICATION field should be changed whenever the subsystem design changes in a way noticeable to a user (e.g., a different stock-keeping unit (SKU)).

The PRODUCT REVISION LEVEL field contains four bytes of ASCII data identifying the revision level of the subsystem (e.g., the board or enclosure) containing the component, as defined by the vendor of the subsystem. The data shall be left-aligned within the field. The PRODUCT REVISION LEVEL field should be changed whenever the subsystem design changes (e.g., any component change, even including resistor values).

All components on a subsystem should have the same values for their VENDOR IDENTIFICATION fields, PRODUCT IDENTIFICATION fields, and PRODUCT REVISION LEVEL fields.

NOTE 98 - Application clients may use the VENDOR IDENTIFICATION field and PRODUCT IDENTIFICATION field to identify the subsystem (e.g., for a user interface). Application clients may use the VENDOR IDENTIFICATION field, PRODUCT IDENTIFICATION field, PRODUCT REVISION LEVEL field to perform workarounds for problems in a specific revision of a subsystem.

The COMPONENT VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the component (e.g., the expander device) containing the management device server. The data shall be left-aligned within the field. The component vendor identification string shall be one assigned by INCITS for use in the standard INQUIRY data VENDOR IDENTIFICATION field. A list of assigned vendor identification strings is in SPC-4 and on the T10 web site (<http://www.t10.org>).

The COMPONENT ID field contains a 16-bit identifier identifying the type of the component (e.g., the expander device model number) containing the management device server, as defined by the vendor of the component. The COMPONENT ID field should be changed whenever the component's programming interface (e.g., the management device server definition) changes.

The COMPONENT REVISION LEVEL field contains an 8-bit identifier identifying the revision level of the component (e.g., the expander device) containing the management device server, as defined by the vendor of the component. The COMPONENT REVISION LEVEL field should be changed whenever the component changes but its programming interface does not change.

NOTE 99 - Application clients may use the COMPONENT VENDOR IDENTIFICATION field and the COMPONENT ID field to interpret vendor-specific information (e.g., vendor-specific SMP functions) correctly for that component. Application clients may use the COMPONENT VENDOR IDENTIFICATION field, the COMPONENT ID field, and the COMPONENT REVISION LEVEL field to perform workarounds for problems in a specific revision of a component.

The vendor-specific bytes are defined by the vendor of the subsystem (e.g., the board or enclosure) containing the component.

The CRC field is defined in 10.4.3.2.

10.4.3.5 REPORT SELF-CONFIGURATION STATUS function

10.4.3.5.1 REPORT SELF-CONFIGURATION STATUS function overview

The REPORT SELF-CONFIGURATION STATUS function returns self-configuration expander device status. This SMP function shall be implemented by the management device server in self-configuring expander devices and shall not be implemented by any other management device servers.

10.4.3.5.2 REPORT SELF-CONFIGURATION STATUS request

Table 322 defines the request format.

Table 322 — REPORT SELF-CONFIGURATION STATUS request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (03h)							
2	Reserved							
3	REQUEST LENGTH (01h)							
4	Reserved							
5								
6	(MSB)	STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	CRC						
11								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 03h.

The REQUEST LENGTH field shall be set to 01h.

The STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field specifies the first self-configuration status descriptor that the management device server shall return in the SMP response frame. If the specified index does not contain a valid self-configuration status descriptor, then the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field in the response may differ from the specified index.

The CRC field is defined in 10.4.3.1.

10.4.3.5.3 REPORT SELF-CONFIGURATION STATUS response

Table 323 defines the response format.

Table 323 — REPORT SELF-CONFIGURATION STATUS response

Byte\Bit	7	6	5	4	3	2	1	0	
0	SMP FRAME TYPE (41h)								
1	FUNCTION (03h)								
2	FUNCTION RESULT								
3	RESPONSE LENGTH ((n - 7) / 4)								
4	(MSB)	EXPANDER CHANGE COUNT							
5								(LSB)	
6	(MSB)	STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX							
7								(LSB)	
8	(MSB)	TOTAL NUMBER OF SELF-CONFIGURATION STATUS DESCRIPTORS							
9								(LSB)	
10	(MSB)	LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX							
11								(LSB)	
12	SELF-CONFIGURATION STATUS DESCRIPTOR LENGTH								
13	Reserved								
18									
19	NUMBER OF SELF-CONFIGURATION STATUS DESCRIPTORS								
Self-configuration status descriptor list									
20	Self-configuration status descriptor (first)(see table 324 in 10.4.3.5.4)								
	...								
	Self-configuration status descriptor (last)(see table 324 in 10.4.3.5.4)								
n - 4									
n - 3	(MSB)	CRC							
n								(LSB)	

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 03h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3). If the SMP initiator port detects a change in the value of this field while retrieving multiple response frames, it should start again because the status information returned is incomplete and inconsistent.

The STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field indicates the index of the first self-configuration status descriptor being returned. If the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field in the SMP request is set to 0000h, then the management device server shall set the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field to 0000h, set the TOTAL NUMBER OF SELF-CONFIGURATION STATUS DESCRIPTORS field to 0000h, and return no descriptors. If the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field in the SMP request does not contain a valid descriptor, then the management device server shall set the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field to the next index, in ascending order wrapping from FFFFh to 0001h, that contains a valid descriptor. Otherwise, this field shall be set to the same value as the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field in the SMP request.

The SELF-CONFIGURATION DESCRIPTOR LENGTH field indicates the length, in dwords, of the REPORT SELF-CONFIGURATION STATUS descriptor (see table 324 in 10.4.3.5.4).

The TOTAL NUMBER OF SELF-CONFIGURATION STATUS DESCRIPTORS field indicates how many self-configuration status descriptors are available at this time from the management device server.

The LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field indicates the index of the last recorded self-configuration status descriptor.

The NUMBER OF SELF-CONFIGURATION STATUS DESCRIPTORS field indicates how many self-configuration status descriptors follow in this SMP response frame.

The self-configuration status descriptor list contains self-configuration status descriptors. The management device server shall return either all the self-configuration status descriptors that fit in one SMP response frame or all the self-configuration status descriptors until the index indicated in the LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field is reached. The self-configuration status descriptor list shall start with the self-configuration status descriptor specified by the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field in ascending order, wrapping from FFFFh to 0001h, based on the self-configuration status descriptor index. The self-configuration status descriptor list shall not contain any truncated self-configuration status descriptors. If the STARTING SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field is equal to the LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field, then the self-configuration status descriptor at that index shall be returned.

The CRC field is defined in 10.4.3.2.

10.4.3.5.4 Self-configuration status descriptor

Each self-configuration status descriptor follows the format defined in table 324.

Table 324 — Self-configuration status descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	STATUS TYPE							
1	Reserved							FINAL
2	Reserved							
3	PHY IDENTIFIER							
4	Reserved							
7								
8	(MSB)	SAS ADDRESS						
15								(LSB)

The STATUS TYPE field indicates the type of status being reported and is defined in table 325.

Table 325 — STATUS TYPE field (part 1 of 2)

Code	Description
00h	Reserved
01h	Error not related to a specific layer
02h	The expander device currently has a connection or is currently attempting to establish a connection with the SMP target port with the indicated SAS address.
03h	Expander route table is full. The expander device was not able to add the indicated SAS address to the expander route table.
04h	Expander device is out of resources (e.g., it discovered too many SAS addresses while performing the discover process through a subtractive port). This does not affect the expander route table.
05h - 1Fh	Reserved for status not related to specific layers
Status reported by the phy layer	
20h	Error reported by the phy layer
21h	All phys in the expander port containing the indicated phy lost dword synchronization
22h - 3Fh	Reserved for status reported by the phy layer
Status reported by the link layer	
40h	Error reported by the link layer
41h	Connection request failed: Open Timeout timer expired
42h	Connection request failed: Received an abandon-class OPEN_REJECT (e.g., BAD DESTINATION, PROTOCOL NOT SUPPORTED, ZONE VIOLATION, STP RESOURCES BUSY, WRONG DESTINATION)
43h	Connection request failed: Received a vendor-specific number of retry-class OPEN_REJECTS (e.g. RETRY, PATHWAY BLOCKED)

Table 325 — STATUS TYPE field (part 2 of 2)

Code	Description
44h	Connection request failed: I_T nexus loss occurred (e.g., OPEN_REJECT (NO DESTINATION) for longer than the time specified by the STP SMP I_T NEXUS LOSS TIME field in the CONFIGURE GENERAL function
45h	Connection request failed: Received BREAK
46h	Connection established: SMP response frame had a CRC error
47h - 5Fh	Reserved for status reported by the link layer
Status reported by the port layer	
60h	Error reported by the port layer
61h	During an SMP connection, there was no SMP response frame within the maximum SMP connection time
62h - 7Fh	Reserved for status reported by the port layer
Status reported by the transport layer	
80h	Error reported by the transport layer
81h - 9Fh	Reserved for status reported by the transport layer
Status reported by the application layer	
A0h	Error reported by the application layer
A1h	SMP response frame is too short
A2h	SMP response frame contains field(s) with unsupported values
A3h	SMP response frame contains results inconsistent with other SMP response frames (e.g., the DISCOVER response ATTACHED SAS ADDRESS field does not contain the SAS address the expander device expected)
A4h	The SAS ADDRESS field contains the SAS address of a self-configuring expander device that returned a REPORT GENERAL response with the CONFIGURING bit set to one. Accesses to SAS addresses two levels beyond this expander device may not succeed until the indicated expander device completes configuration. This is not necessarily an error.
A5h - BFh	Reserved for status reported by the application layer
Other status	
C0h - DFh	Reserved
E0h - FFh	Vendor-specific

A FINAL bit set to one indicates that the expander device is no longer attempting to establish connections to the SMP target port with the indicated SAS address as part of the discover process because of the error indicated by the descriptor. A FINAL bit set to zero indicates that the expander device is still attempting to access the SMP target port with the indicated SAS address as part of the discover process.

The PHY IDENTIFIER field indicates the phy (see 4.2.7) that was used to request a connection with the SMP target port with the indicated SAS address.

Editor's Note 55: Define that phy identifier FFh means "unknown phy". Some of the new functions depend on this; it should be a global rule.

The SAS ADDRESS field indicates the SAS address of the SMP target port to which the expander device established a connection or attempted to establish a connection.

10.4.3.6 REPORT ZONE PERMISSION TABLE function

10.4.3.6.1 REPORT ZONE PERMISSION TABLE function overview

The REPORT ZONE PERMISSION function returns a set of zone permission table entries. This function shall be supported by all zoning expander devices.

10.4.3.6.2 REPORT ZONE PERMISSION TABLE request

Table 326 defines the request format.

Table 326 — REPORT ZONE PERMISSION TABLE request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (04h)							
2	Reserved							
3	REQUEST LENGTH (01h)							
4	Reserved						REPORT TYPE	
5	Reserved							
6	STARTING SOURCE ZONE GROUP							
7	MAXIMUM NUMBER OF ZONE PERMISSION DESCRIPTORS							
8	(MSB)							
11	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 04h.

The REQUEST LENGTH field shall be set to 01h.

The REPORT TYPE field specifies the values that the management device server shall return and is defined in table 327.

Table 327 — REPORT TYPE field

Code	Description
00b	Current
01b	Shadow
10b	Saved
11b	Default

The STARTING SOURCE ZONE GROUP field specifies the first source zone group, (i.e., s) returned. If the value in this field exceeds the end of the zone permission table then the management device server shall return a function result of SOURCE ZONE GROUP DOES NOT EXIST in the response frame (see table 315 in 10.4.3.2).

The MAXIMUM NUMBER OF ZONE PERMISSION DESCRIPTORS field specifies the maximum number of complete zone permission descriptors that the management device server shall return.

The CRC field is defined in 10.4.3.1.

10.4.3.6.3 REPORT ZONE PERMISSION TABLE response

Table 328 defines the response format.

Table 328 — REPORT ZONE PERMISSION TABLE response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (04h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH ((n - 7) / 4)							
4	(MSB)							
5	EXPANDER CHANGE COUNT (LSB)							
6	ZONE LOCKED	Reserved					REPORT TYPE	
7	NUMBER OF ZONE GROUPS		Reserved					
8	Reserved							
13	Reserved							
14	STARTING SOURCE ZONE GROUP							
15	NUMBER OF ZONE PERMISSION DESCRIPTORS							
Zone permission descriptor list								
16	Zone permission descriptor (first)(see table 329 in 10.4.3.6.4)							
31	Zone permission descriptor (last)(see table 329 in 10.4.3.6.4)							
	...							
n - 20	Zone permission descriptor (last)(see table 329 in 10.4.3.6.4)							
n - 4	Zone permission descriptor (last)(see table 329 in 10.4.3.6.4)							
n - 3	(MSB)							
n	CRC (LSB)							

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 04h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3). If the SMP initiator port detects a change in the value of this field while retrieving multiple response frames, it should start again because the status information returned is incomplete and inconsistent.

The ZONE LOCKED bit is defined in the SMP REPORT GENERAL response.

The REPORT TYPE field indicates the value of the REPORT TYPE field in the request frame.

The NUMBER OF ZONE GROUPS field indicates the number of elements in the zone permission descriptor list and is defined in the REPORT GENERAL response (see table 319 in 10.4.3.3).

The STARTING SOURCE ZONE GROUP field indicates the first source zone group (i.e., s) being returned, and is set to the same value as the STARTING SOURCE ZONE GROUP field in the SMP request frame.

The NUMBER OF ZONE PERMISSION DESCRIPTORS field indicates how many zone permission descriptors follow in this SMP response frame.

The zone permission descriptor list contains a zone permission descriptor as defined in 10.4.3.6.4 for each source zone group in ascending order starting with the source zone group specified in the STARTING SOURCE ZONE GROUP field in the request.

The CRC field is defined in 10.4.3.1.

10.4.3.6.4 Zone permission descriptor

Table 329 defines the zone permission descriptor containing 128 entries.

Table 329 — Zone permission descriptor for a source zone group (i.e., s) with 128 zone groups

Byte\Bit	7	6	5	4	3	2	1	0
0	ZP[s, 127]	ZP[s, 126]	ZP[s, 125]	ZP[s, 124]	ZP[s, 123]	ZP[s, 122]	ZP[s, 121]	ZP[s, 120]
...	...							
15	ZP[s, 7] (0b)	ZP[s, 6] (0b)	ZP[s, 5] (0b)	ZP[s, 4] (0b)	ZP[s, 3]	ZP[s, 2]	ZP[s, 1] (1b)	ZP[s, 0] (0b)

Table 330 defines the zone permission descriptor containing 256 entries.

Table 330 — Zone permission descriptor for a source zone group (i.e., s) with 256 zone groups

Byte\Bit	7	6	5	4	3	2	1	0
0	ZP[s, 255]	ZP[s, 254]	ZP[s, 253]	ZP[s, 252]	ZP[s, 251]	ZP[s, 250]	ZP[s, 249]	ZP[s, 248]
...	...							
31	ZP[s, 7] (0b)	ZP[s, 6] (0b)	ZP[s, 5] (0b)	ZP[s, 4] (0b)	ZP[s, 3]	ZP[s, 2]	ZP[s, 1] (1b)	ZP[s, 0] (0b)

The zone permission descriptor contains all of the zone permission table entries for the source zone group (i.e., s).

Table 331 defines how the zone permission descriptor bits shall be set by the management device server.

Table 331 — Zone permission descriptor bit requirements

Source zone group (i.e., s)	Management device server requirement(s) ^a
0	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 2 through (z-1)] shall be set to zero.
1	ZP[s, 0 through (z-1)] shall be set to one.
4, 5, 6 or 7	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 4 through (z-1)] shall be set to zero.
2, 3, or 8 through (z-1) ^a	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 2 through 3] shall be set to zero or one as specified by the CONFIGURE ZONE PERMISSION TABLE function (see 10.4.3.25). ZP[s, 4 through 7] shall be set to zero. ZP[s, 8 through (z-1)] shall be set to zero or one as specified by the CONFIGURE ZONE PERMISSION TABLE function.
^a The number of zone groups (i.e., z) is reported in NUMBER OF ZONE GROUPS field.	

10.4.3.7 REPORT ZONE MANAGER PASSWORD function

The REPORT ZONE MANAGER PASSWORD function returns the zone manager password. This SMP function shall be implemented by all management device servers in zoning expander devices. This function shall only be processed if:

- a) the request is received from an SMP initiator port that has access to zone group 2 (see 4.9.3.2); or
- b) the request is received from any SMP initiator port while physical presence is asserted.

If physical presence is not asserted and the SMP initiator port does not have access to zone group 2, then the management device server shall return a function result of NO MANAGEMENT ACCESS RIGHTS in the response frame (see table 315 in 10.4.3.2).

Table 332 defines the request format.

Table 332 — REPORT ZONE MANAGER PASSWORD request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (05h)							
2	Reserved							
3	REQUEST LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 05h.

The REQUEST LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.1.

Table 333 defines the response format.

Table 333 — REPORT ZONE MANAGER PASSWORD response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (04h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (09h)							
4	(MSB)	EXPANDER CHANGE COUNT						
5								(LSB)
6		Reserved						
7								
8		ZONE MANAGER PASSWORD						
39								
40	(MSB)	CRC						
43								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 04h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 09h.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The CRC field is defined in 10.4.3.2.

10.4.3.8 REPORT BROADCAST function

10.4.3.8.1 REPORT BROADCAST function overview

The REPORT BROADCAST function returns information about Broadcasts (see 4.1.13) that were either originated from this expander device or SAS device, or received on a phy directly attached to an end device.

This SMP function may implemented by any management device server.

10.4.3.8.2 REPORT BROADCAST request

Table 334 defines the request format.

Table 334 — REPORT BROADCAST request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (06h)							
2	Reserved							
3	REQUEST LENGTH (01h)							
4	Reserved				BROADCAST TYPE			
5	Reserved							
7								
8	(MSB)	CRC						
11								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 06h.

The REQUEST LENGTH field shall be set to 01h.

The BROADCAST TYPE field, defined in the ZONED BROADCAST request (see table 377 in 10.4.3.19), specifies the type of Broadcast for which counts shall be returned in the response frame.

The CRC field is defined in 10.4.3.1.

10.4.3.8.3 REPORT BROADCAST response

Table 335 defines the response format.

Table 335 — REPORT BROADCAST response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (06h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6	Reserved				BROADCAST TYPE			
7	NUMBER OF BROADCAST DESCRIPTORS							
Broadcast descriptor list								
8	Broadcast descriptor (first)(see table 358 in 10.4.3.8.4)							
15								
...	...							
n - 11	Broadcast descriptor (last)(see table 358 in 10.4.3.8.4)							
n - 4								
n - 3	(MSB)	CRC						(LSB)
n								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 06h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The BROADCAST TYPE field indicates the value of the BROADCAST TYPE field in the request frame.

The NUMBER OF BROADCAST DESCRIPTORS field indicates how many broadcast descriptors follow.

NOTE 100 - The number of broadcast descriptors is limited to 126 by the SMP response frame size.

The broadcast descriptor list contains broadcast descriptors as defined in 10.4.3.8.4. Broadcast descriptors shall be returned for all Broadcasts of the type specified in the BROADCAST TYPE field, for which the count is non-zero. Broadcast descriptors shall be returned with the descriptor, if any, pertaining to no particular phy (i.e., PHY IDENTIFIER field set to FFh) first, followed by descriptors, if any, in ascending order sorted by the PHY IDENTIFIER field.

The CRC field is defined in 10.4.3.2.

10.4.3.8.4 REPORT BROADCAST response broadcast descriptor

Table 336 defines the broadcast descriptor.

Table 336 — Broadcast descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved				BROADCAST TYPE			
1	PHY IDENTIFIER							
2	Reserved				BROADCAST REASON			
3	Reserved							
4	(MSB)							
5	BROADCAST COUNT							
6	(LSB)							
15	Reserved							

The BROADCAST TYPE field, defined in the ZONED BROADCAST request (see table 378 in 10.4.3.19), indicates the type of Broadcast described by this broadcast descriptor.

The PHY IDENTIFIER field indicates the phy that caused the Broadcast described by this broadcast descriptor to be originated or the phy on which the Broadcast was received. A PHY IDENTIFIER field set to FFh indicates that no specific phy caused the Broadcast described by this broadcast descriptor.

The BROADCAST COUNT field counts the number of Broadcasts that were either:

- a) originated by the SAS device or expander device; or
- b) received by a phy attached to an end device.

If the SAS device or expander device has originated the Broadcast or received the Broadcast since transmitting a REPORT BROADCAST response, it shall increment this field at least once from the value in the previous REPORT BROADCAST response. It shall not increment this field when forwarding a Broadcast. This field shall wrap to at least 0001h after the maximum value (i.e., FFFFh) has been reached.

NOTE 101 - A management application client that uses the BROADCAST COUNT field should read and save all the BROADCAST COUNT field values after performing the discover process (see 4.7), and then read them after each receipt of each Broadcast to ensure that none of the counts increments a multiple of 65 535 times between reading them.

For Broadcasts that are received, the BROADCAST REASON field shall be set to Fh. For Broadcasts that are originated, the BROADCAST REASON field indicates the reason that the Broadcast described by this broadcast descriptor was originated and is defined in table 337.

Table 337 — BROADCAST REASON field

Code	BROADCAST TYPE field	Description
0h	0h (i.e., Broadcast (Change))	Unspecified ^{a b}
0h	4h (i.e., Broadcast (Expander))	Unspecified
1h		A phy event peak value detector has reached its threshold value.
2h		A phy event peak value detector has been cleared by the SMP CONFIGURE PHY EVENT function (see 10.4.3.29)
3h		The expander device is going to temporarily have reduced functionality (e.g., disable SMP access, reduced performance, disable phy to phy communication) for a period of time (see 4.6.8)
0h	8h (i.e., Broadcast (Zone Activate))	Unspecified
All others		Reserved
^a In an expander device, the Broadcast (Change) count is also reported in the REPORT GENERAL response (see 10.4.3.3) and in other SMP response frames containing an EXPANDER CHANGE COUNT field. ^b Broadcast (Change)s originated by this expander device or SAS device shall be counted, with the PHY IDENTIFIER field set to FFh.		

10.4.3.9 DISCOVER function

The DISCOVER function returns information about the specified phy. This SMP function provides information from the IDENTIFY address frame received by the phy and additional phy-specific information. This SMP function shall be implemented by all management device servers.

NOTE 102 - The DISCOVER LIST function (see 10.4.3.14) returns information about one or more phys.

Table 338 defines the request format.

Table 338 — DISCOVER request

Byte/Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (10h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	Reserved							
7								
8	Reserved							IGNORE ZONE GROUP
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)	CRC						
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 10h.

The REQUEST LENGTH field shall be set to 02h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 2 dwords before the CRC field.

An IGNORE ZONE GROUP bit set to one specifies that the management device server shall return information about the specified phy (i.e., the phy specified by the PHY IDENTIFIER field) regardless of the zone permission table.

An IGNORE ZONE GROUP bit set to zero specifies that the management device server shall:

- a) if the SMP initiator port has access to the specified phy based on the zone permission table, return the requested information; and
- b) if the SMP initiator port does not have access to the specified phy, return a function result of PHY VACANT in the response frame (see table 315 in 10.4.3.2).

If the management device server is not in a zoning expander device with zoning enabled, it shall ignore the IGNORE ZONE GROUP bit.

The PHY IDENTIFIER field specifies the phy (see 4.2.8) for which the information is being requested.

The CRC field is defined in 10.4.3.1.

Table 339 defines the response format.

Table 339 — DISCOVER response (part 1 of 3)

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (10h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (1Ah)							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	Reserved	ATTACHED DEVICE TYPE			ATTACHED REASON			
13	Reserved				NEGOTIATED LOGICAL LINK RATE			
14	Reserved				ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	ATTACHED SATA HOST
15	ATTACHED SATA PORT SELECTOR	Reserved			ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	ATTACHED SATA DEVICE
16	SAS ADDRESS							
23								
24	ATTACHED SAS ADDRESS							
31								
32	ATTACHED PHY IDENTIFIER							
33	Reserved					ATTACHED INSIDE ZPSDS PERSISTENT	ATTACHED REQUESTED INSIDE ZPSDS	ATTACHED BREAK_REPLY CAPABLE
34	Reserved							
39								
40	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
41	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
42	PHY CHANGE COUNT							
43	VIRTUAL PHY	Reserved			PARTIAL PATHWAY TIMEOUT VALUE			
44	Reserved				ROUTING ATTRIBUTE			
45	Reserved	CONNECTOR TYPE						

Table 339 — DISCOVER response (part 2 of 3)

Byte\Bit	7	6	5	4	3	2	1	0
46	CONNECTOR ELEMENT INDEX							
47	CONNECTOR PHYSICAL LINK							
48	Reserved							
49								
50	Vendor specific							
51								
52	ATTACHED DEVICE NAME							
59								
60	Reserved	REQUESTED INSIDE ZPSDS CHANGED BY EXPANDER	INSIDE ZPSDS PERSISTENT	REQUESTED INSIDE ZPSDS	Reserved	ZONE GROUP PERSISTENT	INSIDE ZPSDS	ZONING ENABLED
61	Reserved							
62								
63	ZONE GROUP							
64	SELF-CONFIGURATION STATUS							
65	SELF-CONFIGURATION LEVELS COMPLETED							
66	Reserved							
67								
68	SELF-CONFIGURATION SAS ADDRESS							
75								
76	PROGRAMMED PHY CAPABILITIES							
79								
80	CURRENT PHY CAPABILITIES							
83								
84	ATTACHED PHY CAPABILITIES							
87								
88	Reserved							
93								
94	REASON				NEGOTIATED PHYSICAL LINK RATE			
95	Reserved						NEGOTIATED SSC	HARDWARE MUXING SUPPORTED

Table 339 — DISCOVER response (part 3 of 3)

Byte\Bit	7	6	5	4	3	2	1	0
96	Reserved		DEFAULT INSIDE ZPSDS PERSISTENT	DEFAULT REQUESTED INSIDE ZPSDS	Reserved	DEFAULT ZONE GROUP PERSISTENT	Reserved	DEFAULT ZONING ENABLED
97	Reserved							
98	Reserved							
99	DEFAULT ZONE GROUP							
100	Reserved		SAVED INSIDE ZPSDS PERSISTENT	SAVED REQUESTED INSIDE ZPSDS	Reserved	SAVED ZONE GROUP PERSISTENT	Reserved	SAVED ZONING ENABLED
101	Reserved							
102	Reserved							
103	SAVED ZONE GROUP							
104	Reserved		SHADOW INSIDE ZPSDS PERSISTENT	SHADOW REQUESTED INSIDE ZPSDS	Reserved	SHADOW ZONE GROUP PERSISTENT	Reserved	
105	Reserved							
106	Reserved							
107	SHADOW ZONE GROUP							
108	(MSB)		CRC					
111			(LSB)					

Editor's Note 56: add an optional field indicating drive presence (if known)

Editor's Note 57: add a BAY NUMBER field next to CONNECTOR ELEMENT INDEX to continue moving key SES information about drive bays (slots) directly into SMP

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 10h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 1Ah. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 12 dwords before the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The PHY IDENTIFIER field indicates the phy for which information is being returned.

The ATTACHED DEVICE TYPE field indicates the device type attached to this phy and is defined in table 340.

Table 340 — ATTACHED DEVICE TYPE field

Code	Description
000b	No device attached
001b	SAS device or SATA device
010b	Expander device
011b	Expander device compliant with a previous version of this standard
All others	Reserved

If the phy is a physical phy, the ATTACHED DEVICE TYPE field shall only be set to a value other than 000b after:

- if a SAS device or expander device is attached, after the identification sequence is complete;
- if a SATA phy is attached and the STP/SATA bridge does not retrieve IDENTIFY (PACKET) DEVICE data, after the STP/SATA bridge receives the initial Register - Device to Host FIS; and
- if a SATA phy is attached and the STP/SATA bridge retrieves IDENTIFY (PACKET) DEVICE data, after the STP/SATA bridge receives IDENTIFY (PACKET) DEVICE data or it encounters a failure retrieving that data.

If the phy is a physical phy and a SAS phy or expander phy is attached, the ATTACHED REASON field indicates the value of the REASON field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence. If the phy is a physical phy and a SATA phy is attached, the ATTACHED REASON field shall be set to 0h after the initial Register - Device to Host FIS has been received. If the phy is a virtual phy, the ATTACHED REASON field shall be set to 0h.

The NEGOTIATED LOGICAL LINK RATE field is defined in table 341 and indicates the logical link rate being used by the phy. For physical phys, this is negotiated during the link reset sequence. For virtual phys, this field should be set to the maximum physical link rate supported by the expander device. This field may be different from the negotiated physical link rate when multiplexing is enabled.

Table 341 — NEGOTIATED LOGICAL LINK RATE field

NEGOTIATED PHYSICAL LINK RATE field	Multiplexing	NEGOTIATED LOGICAL LINK RATE field
9h (i.e., G2)	Disabled	9h (i.e., 3 Gbps)
	Enabled	8h (i.e., 1,5 Gbps)
Ah (i.e., G3)	Disabled	Ah (i.e., 6 Gbps)
	Enabled	9h (i.e., 3 Gbps)
All others	Any	Same as the NEGOTIATED LOGICAL LINK RATE field

NOTE 103 - In previous versions of this standard that did not define multiplexing, the NEGOTIATED LOGICAL LINK RATE field was called the NEGOTIATED PHYSICAL LINK RATE field and the NEGOTIATED PHYSICAL LINK RATE field in byte 94 did not exist.

Editor's Note 58: There is interest in getting rid of G1, G2, and G3 and using 1,5 Gbps, 3 Gbps, and 6 Gbps throughout. That would affect the names of some of the above code values.

Table 342 defines the ATTACHED SATA PORT SELECTOR bit and the ATTACHED SATA DEVICE bit.

Table 342 — ATTACHED SATA PORT SELECTOR and ATTACHED SATA DEVICE bits

ATTACHED SATA PORT SELECTOR bit value ^{a b d}	ATTACHED SATA DEVICE bit value ^{c d}	Description
0	0	Either: a) the phy is a virtual phy; or b) the phy is a physical phy, and neither a SATA port selector nor a SATA device is attached and ready on the selected phy.
0	1	The phy is a physical phy and the attached phy is a SATA device phy. No SATA port selector is present (i.e., the SP state machine did not detect COMWAKE in response to the initial COMINIT, but sequenced through the normal (non-SATA port selector) SATA device OOB sequence).
1	0	The phy is a physical phy, the attached phy is a SATA port selector host phy, and either: a) the attached phy is the inactive host phy, or b) the attached phy is the active host phy and a SATA device is either not present or not ready behind the SATA port selector (i.e., the SP state machine detected COMWAKE while waiting for COMINIT).
1	1	The phy is a physical phy, the attached phy is a SATA port selector's active host phy and a SATA device is present behind the SATA port selector (i.e., the SP state machine detected COMWAKE while waiting for COMINIT, timed out waiting for COMSAS, and exchanged COMWAKE with an attached SATA device).
^a The ATTACHED SATA PORT SELECTOR bit is invalid if the NEGOTIATED LOGICAL LINK RATE field is set to UNKNOWN (i.e., 0h), DISABLED (i.e., 1h), or RESET_IN_PROGRESS (i.e., 5h). ^b Whenever the ATTACHED SATA PORT SELECTOR bit changes, the phy shall originate a Broadcast (Change)(see 7.11). ^c For the purposes of the ATTACHED SATA DEVICE bit, a SATA port selector is not considered a SATA device. ^d The ATTACHED SATA PORT SELECTOR bit and the ATTACHED SATA DEVICE bit are updated as specified in the SP state machine (see 6.8).		

An ATTACHED SATA HOST bit set to one indicates a SATA host port is attached. An ATTACHED SATA HOST bit set to zero indicates a SATA host port is not attached.

NOTE 104 - Support for SATA hosts is outside the scope of this standard.

If a SAS phy reset sequence occurs (see 6.7.4)(i.e., one or more of the ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, the ATTACHED SMP INITIATOR PORT bit, the ATTACHED SSP TARGET PORT bit, the ATTACHED STP TARGET PORT bit, and/or the ATTACHED SMP TARGET PORT bit is set to one), then the ATTACHED SATA PORT SELECTOR bit, the ATTACHED SATA DEVICE bit, and the ATTACHED SATA HOST bit shall each be set to zero.

An ATTACHED SSP INITIATOR PORT bit set to one indicates the attached phy supports an SSP initiator port. An ATTACHED SSP INITIATOR PORT bit set to zero indicates the attached phy does not support an SSP initiator port. If the phy is a physical phy, the ATTACHED SSP INITIATOR PORT bit indicates the value of the SSP INITIATOR PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

An ATTACHED STP INITIATOR PORT bit set to one indicates the attached phy supports an STP initiator port. An ATTACHED STP INITIATOR PORT bit set to zero indicates the attached phy does not support an STP initiator port.

If the phy is a physical phy, the ATTACHED STP INITIATOR PORT bit indicates the value of the STP INITIATOR PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

An ATTACHED SMP INITIATOR PORT bit set to one indicates the attached phy supports an SMP initiator port. An ATTACHED SMP INITIATOR PORT bit set to zero indicates the attached phy does not support an SMP initiator port. If the phy is a physical phy, the ATTACHED SMP INITIATOR PORT bit indicates the value of the SMP INITIATOR PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

An ATTACHED SSP TARGET PORT bit set to one indicates the attached phy supports an SSP target port. An ATTACHED SSP TARGET PORT bit set to zero indicates the attached phy does not support an SSP target port. If the phy is a physical phy, the ATTACHED SSP TARGET PORT bit indicates the value of the SSP TARGET PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

An ATTACHED STP TARGET PORT bit set to one indicates the attached phy supports an STP target port. An ATTACHED STP TARGET PORT bit set to zero indicates the attached phy does not support an STP target port. If the phy is a physical phy, the ATTACHED STP TARGET PORT bit indicates the value of the STP TARGET PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

An ATTACHED SMP TARGET PORT bit set to one indicates the attached phy supports an SMP target port. An ATTACHED SMP TARGET PORT bit set to zero indicates the attached phy does not support an SMP target port. If the phy is a physical phy, the ATTACHED SMP TARGET PORT bit indicates the value of the SMP TARGET PORT bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

If the phy is a physical phy, the ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, and ATTACHED SMP TARGET PORT bit shall be updated at the end of the identification sequence.

If a SATA phy reset sequence occurs (see 6.7.3)(i.e., the ATTACHED SATA PORT SELECTOR bit is set to one, the ATTACHED SATA DEVICE bit is set to one, or the ATTACHED SATA HOST bit is set to one), then the ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, and ATTACHED SMP TARGET PORT bit shall each be set to zero.

If the phy is an expander phy, the SAS ADDRESS field contains the SAS address of the expander device (see 4.2.4). If the phy is a SAS phy, the SAS ADDRESS field contains the SAS address of the SAS port (see 4.2.7). If the phy is a physical phy, the SAS ADDRESS field contains the value of the SAS ADDRESS field transmitted in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED SAS ADDRESS field is defined as follows:

- a) if the attached port is an expander port, the ATTACHED SAS ADDRESS field contains the SAS address of the attached expander device (see 4.2.4);
- b) if the attached port is a SAS port, the ATTACHED SAS ADDRESS field contains SAS address of the attached SAS port (see 4.2.7); and
- c) if the attached port is a SATA device port, the ATTACHED SAS ADDRESS field contains the SAS address of the STP/SATA bridge (see 4.6.2).

For a physical phy, the ATTACHED SAS ADDRESS field contains the value of the SAS ADDRESS field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence, and shall be updated:

- a) after the identification sequence completes, if a SAS phy or expander phy is attached; or
- b) after the COMSAS Detect Timeout timer expires (see 6.8.3.9), if a SATA phy is attached.

An STP initiator port should not make a connection request to the attached SAS address until the ATTACHED DEVICE TYPE field is set to a value other than 000b (see table 340).

The ATTACHED PHY IDENTIFIER field is defined as follows:

- a) if the attached phy is a SAS phy, the ATTACHED PHY IDENTIFIER field contains the phy identifier of the attached SAS phy in the attached SAS device;
- b) if the attached phy is an expander phy, the ATTACHED PHY IDENTIFIER field contains the phy identifier (see 4.2.8) of the attached expander phy in the attached expander device; and
- c) if the attached phy is a SATA device phy, the ATTACHED PHY IDENTIFIER field contains 00h;

- d) if the attached phy is a SATA port selector phy and the expander device is able to determine the port of the SATA port selector to which it is attached, the ATTACHED PHY IDENTIFIER field contains 00h or 01h; and
- e) if the attached phy is a SATA port selector phy and the expander device is not able to determine the port of the SATA port selector to which it is attached, the ATTACHED PHY IDENTIFIER field contains 00h.

If the phy is a physical phy and the attached phy is a SAS phy or an expander phy, the ATTACHED PHY IDENTIFIER field contains the value of the PHY IDENTIFIER field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

For a physical phy, the ATTACHED PHY IDENTIFIER field shall be updated:

- a) after the identification sequence completes, if a SAS phy or expander phy is attached; or
- b) after the COMSAS Detect Timeout timer expires (see 6.8.3.9), if a SATA phy is attached.

If the phy is a physical phy, the ATTACHED INSIDE ZPSDS PERSISTENT bit indicates the value of the INSIDE ZPSDS PERSISTENT bit received in the IDENTIFY address frame (see 7.8.2) from the attached phy during the identification sequence. If the phy is a virtual phy, the ATTACHED INSIDE ZPSDS PERSISTENT bit shall be set to zero.

If the phy is a physical phy, the ATTACHED REQUESTED INSIDE ZPSDS bit indicates the value of the REQUESTED INSIDE ZPSDS bit received in the IDENTIFY address frame (see 7.8.2) from the attached phy during the identification sequence. If the phy is a virtual phy, the ATTACHED REQUESTED INSIDE ZPSDS bit shall be set to zero.

If the phy is a physical phy, the ATTACHED BREAK_REPLY CAPABLE bit indicates the value of the BREAK_REPLY CAPABLE bit received in the IDENTIFY address frame (see 7.8.2) during the identification sequence. If a phy reset sequence occurs (see 6.7) then the ATTACHED BREAK_REPLY CAPABLE bit shall be set to zero. If the phy is a virtual phy, the ATTACHED BREAK_REPLY CAPABLE bit shall be set to zero.

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate set by the PHY CONTROL function (see 10.4.3.27). The values are defined in table 343. The default value shall be the value of the HARDWARE MINIMUM PHYSICAL LINK RATE field.

The HARDWARE MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate supported by the phy. The values are defined in table 344.

The PROGRAMMED MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate set by the PHY CONTROL function (see 10.4.3.27). The values are defined in table 343. The default value shall be the value of the HARDWARE MAXIMUM PHYSICAL LINK RATE field.

Table 343 — PROGRAMMED MINIMUM PHYSICAL LINK RATE and PROGRAMMED MAXIMUM PHYSICAL LINK rate fields

Code	Description
0h	Not programmable
1h - 7h	Reserved
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future physical link rates

The **HARDWARE MAXIMUM PHYSICAL LINK RATE** field indicates the maximum physical link rate supported by the phy. The values are defined in table 344. If the phy is a virtual phy, this field should be set to the maximum physical link rate supported by the expander device.

Table 344 — **HARDWARE MINIMUM PHYSICAL LINK RATE and HARDWARE MAXIMUM PHYSICAL LINK RATE fields**

Code	Description
0h - 7h	Reserved
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future physical link rates

The **PHY CHANGE COUNT** field counts the number of Broadcast (Change)s originated by an expander phy. Expander devices shall support this field. Other device types shall not support this field. This field shall be set to zero at power on. The expander device shall increment this field at least once when:

- a) it originates a Broadcast (Change) for an expander phy-related reason described in 7.11 from the specified expander phy; or
- b) the zone phy information changes for the specified expander phy (e.g., when a locked expander device is unlocked (see 4.9.6.5)),

and shall not increment this field when forwarding a Broadcast (Change).

After incrementing the **PHY CHANGE COUNT** field, the expander device is not required to increment the **PHY CHANGE COUNT** field again unless a **DISCOVER** response is transmitted. The **PHY CHANGE COUNT** field shall wrap to zero after the maximum value (i.e., FFh) has been reached.

NOTE 105 - Application clients that use the **PHY CHANGE COUNT** field should read it often enough to ensure that it does not increment a multiple of 256 times between reading the field.

A **VIRTUAL PHY** bit set to one indicates the phy is a virtual phy and is part of an internal port and the attached device is contained within the expander device. A **VIRTUAL PHY** bit set to zero indicates the phy is a physical phy and the attached device is not contained within the expander device.

The **PARTIAL PATHWAY TIMEOUT VALUE** field indicates the partial pathway timeout value in microseconds (see 7.12.4.4) set by the **PHY CONTROL** function (see 10.4.3.27).

NOTE 106 - The recommended default value for **PARTIAL PATHWAY TIMEOUT VALUE** is 7 μ s.

The ROUTING ATTRIBUTE field indicates the routing attribute supported by the phy (see 4.6.7.1) and is defined in table 345.

Table 345 — ROUTING ATTRIBUTE field

Code	Name	Description
0h	Direct routing attribute	Direct routing method for attached end devices. Attached expander devices are not supported on this phy.
1h	Subtractive routing attribute	Either: a) subtractive routing method for attached expander devices; or b) direct routing method for attached end devices.
2h	Table routing attribute	Either: a) table routing method for attached expander devices; or b) direct routing method for attached end devices.
All others	Reserved	

The ROUTING ATTRIBUTE field shall not change based on the attached device type.

The CONNECTOR TYPE field indicates the type of connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2). A CONNECTOR TYPE field set to 00h indicates no connector information is available and that the CONNECTOR ELEMENT INDEX field and the CONNECTOR PHYSICAL LINK fields are invalid and shall be ignored.

The CONNECTOR ELEMENT INDEX indicates the element index of the SAS Connector element representing the connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2).

The CONNECTOR PHYSICAL LINK field indicates the physical link in the connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2).

The ATTACHED DEVICE NAME field is defined as follows:

- a) if the attached phy is an expander phy, the ATTACHED DEVICE NAME field contains the device name of the attached expander device (see 4.2.4);
- b) if the attached phy is a SAS phy, the ATTACHED DEVICE NAME field contains the device name of the attached SAS device (see 4.2.7); and
- c) if the attached phy is a SATA device phy, the ATTACHED DEVICE NAME field contains the world wide name of the SATA device (see 4.6.2) or 00000000 00000000h.

For physical phys, table 346 defines how the ATTACHED DEVICE NAME field is updated.

Table 346 — ATTACHED DEVICE NAME field

Condition	Update time	Value
A SAS phy or expander phy is attached	Completion of the identification sequence	The management device server shall set this field to the DEVICE NAME field in the incoming IDENTIFY address frame (i.e., the attached expander device name or attached SAS device name (see 4.2.4))
A SATA phy is attached	Expiration of the COMSAS Detect Timeout timer (see 6.6.3)	The management device server shall set this field to 00000000 00000000h
	Reception of IDENTIFY (PACKET) DEVICE data from the SATA device ^a	Either: a) if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is correct and words 108-111 (i.e., the World Wide Name field) are not set to zero, the management device server shall set this field to the world wide name indicated by words 108-111 according to table 12 in 4.2.5; b) if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is correct and words 108-111 (i.e., the World Wide Name) are set to zero, the management device server shall set this field to 00000000 00000000h; or c) if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is not correct, the management device server shall set this field to 00000000 00000000h.
	Processing a PHY CONTROL function SET ATTACHED DEVICE NAME phy operation	The management device server shall set this field to the value specified in the ATTACHED DEVICE NAME field in the PHY CONTROL request (see 10.4.3.27).
^a This row only applies if the expander device originates the IDENTIFY (PACKET) DEVICE command.		

A REQUESTED INSIDE ZPSDS CHANGED BY EXPANDER bit set to one indicates that the zoning expander device set the REQUESTED INSIDE ZPSDS bit to zero in the zone phy information at the completion of the last link reset sequence. A REQUESTED INSIDE ZPSDS CHANGED BY EXPANDER bit set to zero indicates that the zoning expander device did not set the REQUESTED INSIDE ZPSDS bit to zero in the zone phy information at the completion of the last link reset sequence.

NOTE 107 - The zone manager may use the REQUESTED INSIDE ZPSDS CHANGED BY EXPANDER bit to determine why the REQUESTED INSIDE ZPSDS bit has changed in the DISCOVER response from the value to which it last set the bit.

The INSIDE ZPSDS PERSISTENT bit indicates the value of the INSIDE ZPSDS PERSISTENT bit in the zone phy information (see 4.9.3.1).

The REQUESTED INSIDE ZPSDS bit indicates the value of the REQUESTED INSIDE ZPSDS bit in the zone phy information (see 4.9.3.1).

The ZONE GROUP PERSISTENT bit indicates the value of the ZONE GROUP PERSISTENT bit in the zone phy information (see 4.9.3.1).

The INSIDE ZPSDS bit indicates the value of the INSIDE ZPSDS bit in the zone phy information (see 4.9.3.1).

A ZONING ENABLED bit set to one indicates that zoning is enabled in the expander device. A ZONING ENABLED bit set to zero indicates that zoning is disabled in the expander device.

The ZONE GROUP field indicates the value of the ZONE GROUP field in the zone phy information (see 4.9.3.1).

The SELF-CONFIGURATION STATUS field indicates the status of a self-configuring expander device pertaining to the specified phy and is defined in table 347.

Table 347 — SELF-CONFIGURATION STATUS field

Code	Description
00h	No status available
01h - FFh	As defined for the STATUS TYPE field in the self-configuration status descriptor in the REPORT SELF-CONFIGURATION STATUS response (see table 323 in 10.4.3.5)

The SELF-CONFIGURATION LEVELS COMPLETED field indicates the number of levels of expander devices beyond the expander port containing the specified phy for which the self-configuring expander device's management application client has completed the discover process and is defined in table 348.

Table 348 — SELF-CONFIGURATION LEVELS COMPLETED field

Code	Description
00h	The management application client: a) has not begun the discover process through the expander port containing the specified phy; or b) has not completed the discover process through the expander port containing the specified phy.
01h	The management application client has completed discovery of the expander device attached to the expander port containing the specified phy (i.e., level 1).
02h	The management application client has completed discovery of the expander devices attached to the expander device attached to the expander port containing the specified phy (i.e., level 2).
...	...
FFh	The management application client has completed discovery of the expander devices attached at level 255.

NOTE 108 - The self-configuration levels completed field does not reflect the level of externally configurable expander devices that the configuration subprocess updates to enable the discover process to proceed to higher levels.

The SELF-CONFIGURATION SAS ADDRESS field indicates the SAS address of the SMP target port to which the self-configuring expander device established a connection or attempted to establish a connection using the specified phy and resulted in the status indicated by the SELF-CONFIGURATION STATUS field.

The PROGRAMMED PHY CAPABILITIES field indicates the SNW-3 phy capabilities bits that are going to be transmitted in the next link reset sequence containing SNW-3 as defined in table 90 in 6.7.4.2.3.3.

The CURRENT PHY CAPABILITIES field indicates the outgoing SNW-3 phy capabilities bits transmitted in the last link reset sequence as defined in table 90 in 6.7.4.2.3.3. If the last link reset sequence did not include SNW-3 or was a SATA link reset sequence, the current phy capabilities field shall be set to zero.

The ATTACHED PHY CAPABILITIES field indicates the incoming SNW-3 phy capabilities bits received in the last SNW-3 as defined in table 90 in 6.7.4.2.3.3. If the last link reset sequence did not include SNW-3 or was a SATA link reset sequence, the attached phy capabilities field shall be set to zero.

The REASON field indicates the reason for the last reset of the phy. If the phy is a physical phy, the REASON field indicates the value of the REASON field transmitted in the IDENTIFY address frame (see 7.8.2) during the identification sequence. If the phy is a physical phy and a SATA phy is attached, the REASON field indicates the reason for the link reset sequence (see 7.8.2).

A NEGOTIATED SSC field set to one indicates that SSC is enabled (see 5.3.8). A NEGOTIATED SSC field set to zero indicates that SSC is disabled. The NEGOTIATED SSC field is only valid when the NEGOTIATED PHYSICAL LINK RATE field is greater than or equal to 8h.

The NEGOTIATED PHYSICAL LINK RATE field is defined in table 349. If the phy is a physical phy, this field indicates the physical link rate negotiated during the link reset sequence. If the phy is a virtual phy, this field should be set to the maximum physical link rate supported by the expander device. The negotiated physical link rate may be less than the programmed minimum physical link rate or greater than the programmed maximum physical link rate if the programmed physical link rates have been changed since the last link reset sequence.

Table 349 — NEGOTIATED PHYSICAL LINK RATE field

SP state machine ResetStatus state machine variable	Code	Description
UNKNOWN	0h	Phy is enabled; unknown physical link rate. ^a
DISABLED	1h	Phy is disabled.
PHY_RESET_PROBLEM	2h	Phy is enabled; a phy reset problem occurred (see 6.7.4.2.4).
SPINUP_HOLD	3h	Phy is enabled; detected a SATA device and entered the SATA spinup hold state. The SMP PHY CONTROL function (see 10.4.3.27) phy operations of LINK RESET and HARD RESET may be used to release the phy.
PORT_SELECTOR	4h	Phy is enabled; detected a SATA port selector. The physical link rate has not been negotiated since the last time the phy's SP state machine entered the SP0:OOB_COMINIT state. The SATA spinup hold state has not been entered since the last time the phy's SP state machine entered the SP0:OOB_COMINIT state. The value in this field may change to 3h, 8h, 9h, or Ah if attached to the active phy of the SATA port selector. Presence of a SATA port selector is indicated by the ATTACHED SATA PORT SELECTOR bit (see table 342).
RESET_IN_PROGRESS	5h	Phy is enabled; the expander phy is performing an SMP PHY CONTROL function (see 10.4.3.27) phy operation of LINK RESET or HARD RESET. This value is returned if the specified phy contained a value of 8h, 9h, or Ah in this field when an SMP PHY CONTROL function phy operation of LINK RESET or HARD RESET phy operation is processed.
UNSUPPORTED_PHY_ATTACHED	6h	Phy is enabled; a phy is attached without any commonly supported settings.
Reserved	7h	Reserved
G1	8h	Phy is enabled; 1,5 Gbps physical link rate.
G2	9h	Phy is enabled; 3 Gbps physical link rate.
G3	Ah	Phy is enabled; 6 Gbps physical link rate.
Reserved	Bh - Fh	Phy is enabled; reserved for future logical or physical link rates.
^a This code may be used by an application client in its local data structures to indicate an unknown negotiated logical or physical link rate (e.g., before the discover process has queried the phy).		

A **HARDWARE MUXING SUPPORTED** bit set to one indicates the phy supports multiplexing (see 6.10). A **HARDWARE MUXING SUPPORTED** bit set to zero indicates the phy does not support multiplexing. This value is not adjusted based on the negotiated physical link rate.

The **DEFAULT INSIDE ZPSDS PERSISTENT** bit contains the default value of the **INSIDE ZPSDS PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **DEFAULT REQUESTED INSIDE ZPSDS** bit contains the default value of the **REQUESTED INSIDE ZPSDS** bit in the zone phy information (see 4.9.3.1).

The **DEFAULT ZONE GROUP PERSISTENT** bit contains the default value of the **ZONE GROUP PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **DEFAULT ZONING ENABLED** bit contains the default value of the **ZONING ENABLED** bit (see 4.9.3.1).

The **DEFAULT ZONE GROUP** field contains the default value of the **ZONE GROUP** field in the zone phy information (see 4.9.3.1).

The **SAVED INSIDE ZPSDS PERSISTENT** bit contains the default value of the **INSIDE ZPSDS PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **SAVED REQUESTED INSIDE ZPSDS** bit contains the default value of the **REQUESTED INSIDE ZPSDS** bit in the zone phy information (see 4.9.3.1).

The **SAVED ZONE GROUP PERSISTENT** bit contains the default value of the **ZONE GROUP PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **SAVED ZONING ENABLED** bit contains the default value of the **ZONING ENABLED** bit (see 4.9.3.1).

The **SAVED ZONE GROUP** field contains the default value of the **ZONE GROUP** field in the zone phy information (see 4.9.3.1).

The **SHADOW INSIDE ZPSDS PERSISTENT** bit contains the default value of the **INSIDE ZPSDS PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **SHADOW REQUESTED INSIDE ZPSDS** bit contains the default value of the **REQUESTED INSIDE ZPSDS** bit in the zone phy information (see 4.9.3.1).

The **SHADOW ZONE GROUP PERSISTENT** bit contains the default value of the **ZONE GROUP PERSISTENT** bit in the zone phy information (see 4.9.3.1).

The **SHADOW ZONING ENABLED** bit contains the default value of the **ZONING ENABLED** bit (see 4.9.3.1).

The **SHADOW ZONE GROUP** field contains the default value of the **ZONE GROUP** field in the zone phy information (see 4.9.3.1).

The **CRC** field is defined in 10.4.3.2.

10.4.3.10 REPORT PHY ERROR LOG function

The **REPORT PHY ERROR LOG** function returns error logging information about the specified phy. This SMP function may be implemented by any management device server.

Table 350 defines the request format.

Table 350 — REPORT PHY ERROR LOG request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (11h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)	CRC						(LSB)
15								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 11h.

The REQUEST LENGTH field shall be set to 02h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 2 dwords before the CRC field.

The PHY IDENTIFIER field specifies the phy (see 4.2.8) for which information shall be reported.

The CRC field is defined in 10.4.3.1.

Table 351 defines the response format.

Table 351 — REPORT PHY ERROR LOG response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (11h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (06h)							
4	(MSB)	EXPANDER CHANGE COUNT						
5								(LSB)
6		Reserved						
8								
9	PHY IDENTIFIER							
10		Reserved						
11								
12	(MSB)	INVALID DWORD COUNT						
15								(LSB)
16	(MSB)	RUNNING DISPARITY ERROR COUNT						
19								(LSB)
20	(MSB)	LOSS OF DWORD SYNCHRONIZATION COUNT						
23								(LSB)
24	(MSB)	PHY RESET PROBLEM COUNT						
27								(LSB)
28	(MSB)	CRC						
31								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 11h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 06h. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 6 dwords before the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The PHY IDENTIFIER field indicates the phy (see 4.2.8) for which information is being reported.

The INVALID DWORD COUNT field indicates the number of invalid dwords (see 3.1.112) that have been received outside of phy reset sequences (i.e., between when the SP state machine (see 6.8) sends a Phy Layer Ready

(SAS) confirmation or Phy Layer Ready (SATA) confirmation and when it sends a Phy Layer Not Ready confirmation to the link layer). The count shall stop at the maximum value. The INVALID DWORD COUNT field is set to a vendor-specific value after power on.

The RUNNING DISPARITY ERROR COUNT field indicates the number of dwords containing running disparity errors (see 6.2) that have been received outside of phy reset sequences. The count shall stop at the maximum value. The RUNNING DISPARITY ERROR COUNT field is set to a vendor-specific value after power on.

The LOSS OF DWORD SYNCHRONIZATION COUNT field indicates the number of times the phy has restarted the link reset sequence because it lost dword synchronization (see 6.9) (i.e., the SP state machine transitioned from SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)). The count shall stop at the maximum value. The LOSS OF DWORD SYNCHRONIZATION COUNT field is set to a vendor-specific value after power on.

The PHY RESET PROBLEM COUNT field indicates the number of times a phy reset problem (see 6.7.4.2.4) occurred. The count shall stop at the maximum value. The PHY RESET PROBLEM COUNT field is set to a vendor-specific value after power on.

The CRC field is defined in 10.4.3.2.

10.4.3.11 REPORT PHY SATA function

The REPORT PHY SATA function returns information about the SATA state for a specified phy. This SMP function shall be implemented by management device servers behind SMP target ports that share SAS addresses with STP target ports and by management device servers in expander devices with STP/SATA bridges. This SMP function shall not be implemented by any other type of management device server.

Table 352 defines the request format.

Table 352 — REPORT PHY SATA request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (12h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	Reserved							
8								
9	PHY IDENTIFIER							
10	AFFILIATION CONTEXT							
11	Reserved							
12	CRC							
15								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 12h.

The REQUEST LENGTH field shall be set to 02h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 2 dwords before the CRC field.

The PHY IDENTIFIER field specifies the phy (see 4.2.8) for which information shall be reported.

The AFFILIATION CONTEXT field specifies the relative identifier of the affiliation context for which information shall be reported (see 7.17.4).

The CRC field is defined in 10.4.3.1.

Table 353 defines the response format.

Table 353 — REPORT PHY SATA response (part 1 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (12h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (10h)							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11	Reserved					STP I_T NEXUS LOSS OCCURRED	AFFILIATIONS SUPPORTED	AFFILIATION VALID
12	Reserved							
15								
16	STP SAS ADDRESS							
23								
24	REGISTER DEVICE TO HOST FIS							
43								
44	Reserved							
47								
48	AFFILIATED STP INITIATOR SAS ADDRESS							
55								
56	STP I_T NEXUS LOSS SAS ADDRESS							
63								

Table 353 — REPORT PHY SATA response (part 2 of 2)

Byte/Bit	7	6	5	4	3	2	1	0
64	Reserved							
65	AFFILIATION CONTEXT							
66	CURRENT AFFILIATION CONTEXTS							
67	MAXIMUM AFFILIATION CONTEXTS							
68	(MSB)							
71	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 12h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 10h. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 13 dwords before the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The PHY IDENTIFIER field indicates the phy (see 4.2.8) for which information is being reported and is the same as the PHY IDENTIFIER field in the request frame.

An STP I_T NEXUS LOSS OCCURRED bit set to one indicates that the STP target port encountered an I_T nexus loss in the specified affiliation context for the STP initiator port whose SAS address is indicated in the STP I_T NEXUS LOSS SAS ADDRESS field. An STP I_T NEXUS LOSS OCCURRED bit set to zero indicates that:

- an I_T nexus loss has not occurred in the specified affiliation context;
- an I_T nexus loss has occurred in the specified affiliation context and been cleared by the SMP PHY CONTROL function CLEAR STP I_T NEXUS LOSS phy operation (see table 403 in 10.4.3.27); or
- the STP target port has successfully established a connection with the indicated STP initiator port in the specified affiliation context.

An AFFILIATIONS SUPPORTED bit set to one indicates that the specified affiliation context is supported by the STP target port containing the specified phy. An AFFILIATIONS SUPPORTED bit set to zero indicates that the specified affiliation context is not supported by the STP target port containing the specified phy.

An AFFILIATION VALID bit set to one indicates that the STP target port is currently maintaining an affiliation in the specified affiliation context and the AFFILIATED STP INITIATOR SAS ADDRESS field is valid. An AFFILIATION VALID bit set to zero indicates that the STP target port is not currently maintaining an affiliation in the specified affiliation context and the AFFILIATED STP INITIATOR SAS ADDRESS field is not valid.

The STP SAS ADDRESS field indicates the SAS address (see 4.2.2) of the STP target port that contains the specified phy.

The REGISTER DEVICE TO HOST FIS field indicates the contents of the initial Register - Device to Host FIS. For an STP/SATA bridge, this is delivered by the attached SATA device after a link reset sequence (see SATA-2). For a native STP target port in an end device, this is directly provided.

The FIS contents shall be stored with little-endian byte ordering (i.e., the first byte, byte 24, contains the FIS Type).

For an STP/SATA bridge, the first byte of the field (i.e., the FIS Type) shall be initialized to zero on power on and whenever the phy has restarted the link reset sequence after losing dword synchronization (see 6.9)(i.e., the SP state machine transitioned from SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)) to indicate the field is invalid and the attached SATA device has not delivered a Register – Device to Host FIS. The first byte of the field shall be set to 34h when the attached SATA device has delivered the initial Register

– Device to Host FIS. The remaining contents of the REGISTER DEVICE TO HOST FIS field shall remain constant until a link reset sequence causes the attached SATA device to deliver another initial Register – Device to Host FIS.

If the AFFILIATION VALID bit is set to one, the AFFILIATED STP INITIATOR SAS ADDRESS field indicates the SAS address (see 4.2.2) of the STP initiator port that has an affiliation in the specified affiliation context with the STP target port that contains the specified phy. If the AFFILIATION VALID bit is set to zero, the AFFILIATED STP INITIATOR SAS ADDRESS field may contain the SAS address of the STP initiator port that previously had an affiliation in the specified affiliation context with the STP target port that contains the specified phy.

The STP I_T NEXUS LOSS SAS ADDRESS field indicates the SAS address (see 4.2.2) of the last STP initiator port for which the STP target port experienced an I_T nexus loss (see 4.5) in the specified affiliation context.

The AFFILIATION CONTEXT field indicates the relative identifier of the affiliation context for which affiliation-related information (i.e., the AFFILIATIONS SUPPORTED bit, the AFFILIATION VALID bit, the AFFILIATED STP INITIATOR SAS ADDRESS field, the STP I_T NEXUS LOSS OCCURRED bit, and the STP I_T NEXUS LOSS SAS ADDRESS field) is being reported (see 7.17.4) and is the same as the AFFILIATION CONTEXT field in the request frame.

The CURRENT AFFILIATION CONTEXTS field indicates the current number of affiliations established by the STP target port.

The MAXIMUM AFFILIATION CONTEXTS field indicates the maximum number of affiliation contexts supported by the STP target port.

The CRC field is defined in 10.4.3.2.

10.4.3.12 REPORT ROUTE INFORMATION function

The REPORT ROUTE INFORMATION function returns an expander route entry from a phy-based expander route table within an expander device (see 4.6.7.3). This SMP function shall be supported by management device servers in expander devices if the EXPANDER ROUTE INDEXES field is set to a non-zero value in the SMP REPORT GENERAL response (see 10.4.3.3). This SMP function may be used as a diagnostic tool to resolve topology issues.

Table 354 defines the request format.

Table 354 — REPORT ROUTE INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (13h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	Reserved							
5								
6	(MSB)	EXPANDER ROUTE INDEX						
7								(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)	CRC						
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 13h.

The REQUEST LENGTH field shall be set to 02h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 2 dwords before the CRC field.

The EXPANDER ROUTE INDEX field specifies the expander route index for the expander route entry being requested (see 4.6.7.3).

The PHY IDENTIFIER field specifies the phy for which the expander route entry is being requested.

The CRC field is defined in 10.4.3.1.

Table 355 defines the response format.

Table 355 — REPORT ROUTE INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (13h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (09h)							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6	(MSB)	EXPANDER ROUTE INDEX						(LSB)
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11								
12	EXPANDER ROUTE ENTRY DISABLED	Reserved						
13	Reserved							
15								
16	ROUTED SAS ADDRESS							
23								
24	Reserved							
39								
40	(MSB)	CRC						(LSB)
43								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 13h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 09h. For compatibility with previous versions of this standard, a RESPONSE LENGTH field set to 00h indicates that there are 9 dwords before the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The EXPANDER ROUTE INDEX field indicates the expander route index for the expander route entry being returned (see 4.6.7.3).

The PHY IDENTIFIER field indicates the phy identifier for the expander route entry being returned.

The EXPANDER ROUTE ENTRY DISABLED bit indicates whether the ECM shall use the expander route entry to route connection requests (see 4.6.7.3). If the EXPANDER ROUTE ENTRY DISABLED bit is set to zero, then the ECM shall use the expander route entry to route connection requests. If the EXPANDER ROUTE ENTRY DISABLED bit is set to one, the ECM shall not use the expander route entry to route connection requests.

The ROUTED SAS ADDRESS field indicates the SAS address in the expander route entry (see 4.6.7.3).

The CRC field is defined in 10.4.3.2.

10.4.3.13 REPORT PHY EVENT function

10.4.3.13.1 REPORT PHY EVENT function overview

The REPORT PHY EVENT function returns phy events (see 4.11) concerning the specified phy. This SMP function may be implemented by any management device server.

NOTE 109 - The REPORT PHY EVENT LIST function (see 10.4.3.15) returns information about one or more phys.

10.4.3.13.2 REPORT PHY EVENT request

Table 356 defines the request format.

Table 356 — REPORT PHY EVENT request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (14h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	Reserved							
5	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)	CRC						
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 14h.

The REQUEST LENGTH field shall be set to 02h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) for which information shall be reported.

The CRC field is defined in 10.4.3.1.

10.4.3.13.3 REPORT PHY EVENT response

Table 351 defines the response format.

Table 357 — REPORT PHY EVENT response

Byte\Bit	7	6	5	4	3	2	1	0						
0	SMP FRAME TYPE (41h)													
1	FUNCTION (14h)													
2	FUNCTION RESULT													
3	RESPONSE LENGTH													
4	(MSB)	EXPANDER CHANGE COUNT												
5								(LSB)						
6	Reserved													
8														
9	PHY IDENTIFIER													
10	Reserved													
14														
15	NUMBER OF PHY EVENT DESCRIPTORS													
Phy event descriptor list														
16	Phy event descriptor (first)(see table 358 in 10.4.3.13.4)													
27														
...	...													
n - 15	Phy event descriptor (last)(see table 358 in 10.4.3.13.4)													
n - 4														
n - 3	(MSB)	CRC												
n								(LSB)						

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 14h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The PHY IDENTIFIER field indicates the phy (see 4.2.7) for which information is being reported.

The NUMBER OF PHY EVENT DESCRIPTORS field indicates how many phy event descriptors follow.

The phy event descriptor list contains phy event descriptors as defined in 10.4.3.13.4.

Editor's Note 59: require they be in ascending order based on phy event source value

The CRC field is defined in 10.4.3.2.

10.4.3.13.4 REPORT PHY EVENT response phy event descriptor

Table 358 defines the phy event descriptor.

Table 358 — Phy event descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
2								
3	PHY EVENT SOURCE							
4	(MSB)	PHY EVENT						
7								(LSB)
8	(MSB)	PEAK VALUE DETECTOR THRESHOLD						
11								(LSB)

The PHY EVENT SOURCE field, defined in table 37 in 4.11, indicates the type of phy event being reported in the PHY EVENT field.

The PHY EVENT field indicates the value (i.e., the count or peak value detected) of the phy event indicated by the PHY EVENT SOURCE field.

If the phy event source is a peak value detector, the PEAK VALUE DETECTOR THRESHOLD field indicates the value of the peak value detector that causes the expander device to originate a Broadcast (Expander)(see 7.2.5.5). If the phy event source is not a peak value detector, then the PEAK VALUE DETECTOR THRESHOLD field is reserved.

10.4.3.14 DISCOVER LIST function

10.4.3.14.1 DISCOVER LIST function overview

The DISCOVER LIST function returns information about the device (i.e., some fields from the REPORT GENERAL response (see 10.4.3.3)) and one or more phys (i.e., some fields from the DISCOVER response (see 10.4.3.9)). This SMP function shall be implemented by all management device servers. This function is intended to provide the necessary information in a single SMP response for a self-configuring expander device to perform the discover process and configure its own expander routing table.

10.4.3.14.2 DISCOVER LIST request

Table 359 defines the request format.

Table 359 — DISCOVER LIST request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (20h)							
2	Reserved							
3	REQUEST LENGTH (06h)							
4	Reserved							
7								
8	STARTING PHY IDENTIFIER							
9	MAXIMUM NUMBER OF DISCOVER LIST DESCRIPTORS							
10	IGNORE ZONE GROUP	Reserved			PHY FILTER			
11	Reserved				DESCRIPTOR TYPE			
12	Reserved							
15								
16	Vendor-specific							
27								
28	(MSB)	CRC						(LSB)
31								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 20h.

The REQUEST LENGTH field shall be set to 06h.

The STARTING PHY IDENTIFIER field specifies the phy identifier of the first phy for which the information is being requested.

The MAXIMUM NUMBER OF DISCOVER LIST DESCRIPTORS field specifies the maximum number of complete DISCOVER LIST descriptors that the management device server shall return.

The IGNORE ZONE GROUP bit is defined in the SMP DISCOVER request (see 10.4.3.9).

The PHY FILTER field is defined in table 360 and specifies a filter limiting which phys that the management device server shall return in the DISCOVER LIST descriptor list in the DISCOVER response.

Table 360 — PHY FILTER field

Code	Description
0h	All phys.
1h	Phys with the ATTACHED DEVICE TYPE field (see 10.4.3.9) set to 010b or 011b (i.e., attached to expander devices).
2h	Phys with the ATTACHED DEVICE TYPE field (see 10.4.3.9) set to a value other than zero (i.e., attached to end devices or expander devices).
All others	Reserved

The DESCRIPTOR TYPE field is defined in table 361 and specifies the DISCOVER LIST descriptor format and length.

Table 361 — DESCRIPTOR TYPE field

Code	DISCOVER LIST descriptor format	Descriptor length
0h	DISCOVER response defined in table 339 (see 10.4.3.9), not including the CRC field.	The length of the DISCOVER response, not including the CRC field
1h	SHORT FORMAT descriptor defined in table 363 (see 10.4.3.14.4)	24 bytes
All others	Reserved	

The CRC field is defined in 10.4.3.1.

10.4.3.14.3 DISCOVER LIST response

Table 362 defines the response format.

Table 362 — DISCOVER LIST response (part 1 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (20h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH ((n - 7) / 4)							
4	(MSB)	EXPANDER CHANGE COUNT						
5								(LSB)
6	Reserved							
7								
8	STARTING PHY IDENTIFIER							
9	NUMBER OF DISCOVER LIST DESCRIPTORS							
10	Reserved				PHY FILTER			
11	Reserved				DESCRIPTOR TYPE			

Table 362 — DISCOVER LIST response (part 2 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
12	DESCRIPTOR LENGTH							
13	Reserved							
15								
16	ZONING SUPPORTED	ZONING ENABLED	Reserved				CONFIGURING	EXTERNALLY CONFIGURABLE ROUTE TABLE
17	Reserved							
18	(MSB)	LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX						(LSB)
19								
20	(MSB)	LAST PHY EVENT LIST DESCRIPTOR INDEX						(LSB)
21								
22	Reserved							
31								
32	Vendor specific							
47								
DISCOVER LIST descriptor list								
48	DISCOVER LIST descriptor (first)(see table 361 in 10.4.3.14.1, and table 339 in 10.4.3.9 or table 363 in 10.4.3.14.4)							
	...							
	DISCOVER LIST descriptor (last)(see table 361 in 10.4.3.14.1, and table 339 in 10.4.3.9 or table 363 in 10.4.3.14.4)							
n - 4								
n - 3	(MSB)							
n	CRC						(LSB)	

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 20h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The STARTING PHY IDENTIFIER field indicates the phy identifier of the first phy in the DISCOVER LIST descriptor list.

NOTE 110 - The STARTING PHY IDENTIFIER field may be different than the STARTING PHY IDENTIFIER field in the request frame (see 10.4.3.14.2) due to the filter specified by the PHY FILTER field in the request frame.

The NUMBER OF DISCOVER LIST DESCRIPTORS field indicates the number of DISCOVER LIST descriptors returned in the DISCOVER LIST descriptor list.

The DESCRIPTOR LENGTH field indicates the length of the DISCOVER LIST descriptor (see table 361 in 10.4.3.14.2).

The ZONING SUPPORTED bit is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The ZONING ENABLED bit is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The CONFIGURING bit is defined in the SMP DISCOVER response (see 10.4.3.9).

The EXTERNALLY CONFIGURABLE ROUTE TABLE bit is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The LAST SELF-CONFIGURATION STATUS DESCRIPTOR INDEX field is defined in the REPORT SELF-CONFIGURATION STATUS response (see 10.4.3.5).

The LAST PHY EVENT LIST DESCRIPTOR INDEX field is defined in the REPORT PHY EVENT LIST response (see 10.4.3.15).

The DISCOVER LIST descriptor list contains DISCOVER LIST descriptors for each phy:

- starting with the phy whose phy identifier is specified in the STARTING PHY IDENTIFIER field in the request (see 10.4.3.14.2);
- satisfying the filter specified in the PHY FILTER field in the request (see table 360 in 10.4.3.14.2); and
- that is able to be included in the response frame without being truncated.

Editor's Note 60: require they be in ascending order based on phy identifier

Each DISCOVER LIST descriptor shall use the format specified in the DESCRIPTOR TYPE field in the request (see table 361 in 10.4.3.14.2)

The management device server shall not include DISCOVER LIST descriptors for phys with phy identifiers greater than or equal to the NUMBER OF PHYs field reported in the SMP REPORT GENERAL response (see 10.4.3.3). The management device server shall not include partial DISCOVER LIST descriptors.

The CRC field is defined in 10.4.3.2.

10.4.3.14.4 DISCOVER LIST response SHORT FORMAT descriptor

Table 363 defines the SHORT FORMAT descriptor.

Table 363 — SHORT FORMAT descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	PHY IDENTIFIER							
1	FUNCTION RESULT							
2	Restricted	ATTACHED DEVICE TYPE			ATTACHED REASON			
3	Restricted for DISCOVER response byte 13			NEGOTIATED LOGICAL LINK RATE				
4	Restricted for DISCOVER response byte 14			ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	ATTACHED SATA HOST	
5	ATTACHED SATA PORT SELECTOR	Restricted for DISCOVER response byte 15			ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	ATTACHED SATA DEVICE
6	VIRTUAL PHY	Reserved			ROUTING ATTRIBUTE			
7	REASON			Reserved				
8	ZONE GROUP							

Table 363 — SHORT FORMAT descriptor

Byte\Bit	7	6	5	4	3	2	1	0
9	Restricted for DISCOVER response byte 60		INSIDE ZPSDS PERSISTENT	REQUESTED INSIDE ZPSDS	Reserved	ZONE GROUP PERSISTENT	INSIDE ZPSDS	Reserved
10	ATTACHED PHY IDENTIFIER							
11	PHY CHANGE COUNT							
12	ATTACHED SAS ADDRESS							
19								
20	Reserved							
23								

The PHY IDENTIFIER field indicates the phy for which physical configuration link information is being returned.

The FUNCTION RESULT field indicates the value that is returned in the FUNCTION RESULT field in the SMP DISCOVER response for the specified phy (e.g., SMP FUNCTION ACCEPTED, PHY VACANT, or PHY DOES NOT EXIST). If the FUNCTION RESULT field is set to PHY VACANT or PHY DOES NOT EXIST, the rest of the fields in the SHORT FORMAT descriptor shall be ignored.

The fields in the SHORT FORMAT descriptor not defined in this subclause are defined in the SMP DISCOVER response (see 10.4.3.9).

10.4.3.15 REPORT PHY EVENT LIST function

10.4.3.15.1 REPORT PHY EVENT LIST function overview

The REPORT PHY EVENT LIST function returns phy events (see 4.11). This SMP function may be implemented by any management device server.

10.4.3.15.2 REPORT PHY EVENT LIST request

Table 364 defines the request format.

Table 364 — REPORT PHY EVENT LIST request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (21h)							
2	Reserved							
3	REQUEST LENGTH (01h)							
4	Reserved							
5								
6	(MSB)	STARTING PHY EVENT LIST DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	CRC						
11								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 21h.

The REQUEST LENGTH field shall be set to 02h.

The STARTING PHY EVENT LIST DESCRIPTOR INDEX field specifies the first phy event list descriptor that the management device server shall return in the SMP response frame. A STARTING PHY EVENT LIST DESCRIPTOR INDEX field set to 0000h is reserved. The requested starting index and the indicated starting index in the response may differ.

The CRC field is defined in 10.4.3.1.

10.4.3.15.3 REPORT PHY EVENT LIST response

Table 365 defines the response format.

Table 365 — REPORT PHY EVENT LIST response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (21h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH ((n - 7) / 4)							
4	(MSB)	EXPANDER CHANGE COUNT						
5								(LSB)
6	(MSB)	STARTING PHY EVENT LIST DESCRIPTOR INDEX						
7								(LSB)
8	(MSB)	LAST PHY EVENT LIST DESCRIPTOR INDEX						
9								(LSB)
10	PHY EVENT LIST DESCRIPTOR LENGTH							
11	Reserved							
14								
15	NUMBER OF PHY EVENT LIST DESCRIPTORS							
Phy event list descriptor list								
16	Phy event list descriptor (first)(see table 366 in 10.4.3.15.4)							
...	...							
	Phy event list descriptor (last)(see table 366 in 10.4.3.15.4)							
n - 4								
n - 3	(MSB)	CRC						
n								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 21h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The STARTING PHY EVENT LIST DESCRIPTOR INDEX field indicates the index of the first phy event list descriptor being returned. If the STARTING PHY EVENT LIST DESCRIPTOR INDEX field in the SMP request is 0000h, then the

management device server shall set the STARTING PHY EVENT LIST DESCRIPTOR INDEX field to 0000h. If the STARTING PHY EVENT LIST DESCRIPTOR INDEX field specified in the SMP request does not contain a valid descriptor, then the device management server shall set the STARTING PHY EVENT LIST DESCRIPTOR INDEX field to the next index, in ascending order wrapping from FFFFh to 0001h, that contains a valid descriptor. Otherwise, this field shall be set to the same value as the STARTING PHY EVENT LIST DESCRIPTOR INDEX field in the SMP request frame.

The DESCRIPTOR LENGTH field indicates the length of the phy event list descriptor (see table 358 in 10.4.3.13.4).

The LAST PHY EVENT LIST DESCRIPTOR INDEX field indicates the last index of the last recorded phy event list descriptor.

The NUMBER OF PHY EVENT LIST DESCRIPTORS field indicates how many phy event list descriptors follow.

The phy event list descriptor list contains phy event list descriptors as defined in 10.4.3.13.4.

The CRC field is defined in 10.4.3.2.

10.4.3.15.4 REPORT PHY EVENT LIST response phy event list descriptor

Table 366 defines the phy event list descriptor.

Table 366 — Phy event listdescriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1								
2	PHY IDENTIFIER							
3	PHY EVENT SOURCE							
4	(MSB)	PHY EVENT						
7								(LSB)
8	(MSB)	PEAK VALUE DETECTOR THRESHOLD						
11								(LSB)

The PHY IDENTIFIER field indicates the phy for which information is being returned.

The PHY EVENT SOURCE field, defined in table 37 in 4.11, indicates the type of phy event being reported in the PHY EVENT field.

The PHY EVENT field indicates the value (i.e., the count or peak value detected) of the phy event indicated by the PHY EVENT SOURCE field.

If the phy event source is a peak value detector, the PEAK VALUE DETECTOR THRESHOLD field indicates the value of the peak value detector that causes the expander device to originate a Broadcast (Expander)(see 7.2.5.5). If the phy event source is not a peak value detector, then the PEAK VALUE DETECTOR THRESHOLD field is reserved.

10.4.3.16 REPORT EXPANDER ROUTE TABLE LIST function**10.4.3.16.1 REPORT EXPANDER ROUTE TABLE LIST function overview**

The REPORT EXPANDER ROUTE TABLE LIST function returns the contents of an expander-based expander route table (see 4.6.7.3 and 4.9.3.4). The list may be in any order. Self-configuring expander devices shall support this function.

10.4.3.16.2 REPORT EXPANDER ROUTE TABLE LIST request

Table 367 defines the request format.

Table 367 — REPORT EXPANDER ROUTE TABLE LIST request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (22h)							
2	Reserved							
3	REQUEST LENGTH (06h)							
4	Reserved							
7								
8	(MSB)	MAXIMUM NUMBER OF REPORT EXPANDER ROUTE TABLE DESCRIPTORS						(LSB)
9								
10	(MSB)	STARTING ROUTED SAS ADDRESS INDEX						(LSB)
11								
12	Reserved							
18								
19	STARTING PHY IDENTIFIER							
20	Reserved							
27								
28	(MSB)	CRC						(LSB)
31								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 22h.

The REQUEST LENGTH field shall be set to 06h.

The MAXIMUM NUMBER OF REPORT EXPANDER ROUTE TABLE DESCRIPTORS field specifies the maximum number of REPORT EXPANDER ROUTE TABLE descriptors that the management device server shall return.

The STARTING ROUTED SAS ADDRESS INDEX field specifies the index of the first routed SAS address to be returned in the descriptor list.

The STARTING PHY IDENTIFIER field specifies the first phy identifier of the phy identifier bit map returned in the REPORT EXPANDER ROUTE TABLE descriptor (see table 369 in 10.4.3.16.3). The value of this field shall be aligned to a boundary of 48 (e.g., 0, 48, 96) and not exceed the value of the NUMBER OF PHYS field reported in the REPORT GENERAL response (see 10.4.3.3).

The CRC field is defined in 10.4.3.1.

10.4.3.16.3 REPORT EXPANDER ROUTE TABLE LIST response

Table 368 defines the response format.

Table 368 — REPORT EXPANDER ROUTE TABLE LIST response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (22h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH ((n - 7) / 4)							
4	(MSB)	EXPANDER CHANGE COUNT						(LSB)
5								
6	(MSB)	EXPANDER ROUTE TABLE CHANGE COUNT						(LSB)
7								
8	Reserved						CONFIGURING	Reserved
9	Reserved							
10	(MSB)	NUMBER OF REPORT EXPANDER ROUTE TABLE DESCRIPTORS						(LSB)
11								
12	(MSB)	FIRST ROUTED SAS ADDRESS INDEX						(LSB)
13								
14	(MSB)	LAST ROUTED SAS ADDRESS INDEX						(LSB)
15								
16	Reserved							
18								
19	STARTING PHY IDENTIFIER							
20	Reserved							
31								
REPORT EXPANDER ROUTE TABLE descriptor list								
32	REPORT EXPANDER ROUTE TABLE descriptor (first)(see table 369 in 10.4.3.16.4)							
47								
...	...							
n - 20	REPORT EXPANDER ROUTE TABLE descriptor (last)(see table 369 in 10.4.3.16.4)							
n - 4								
n - 3	(MSB)	CRC						(LSB)
n								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 22h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field indicates the number of dwords that follow, not including the CRC field.

The EXPANDER CHANGE COUNT field is defined in the SMP REPORT GENERAL response (see 10.4.3.3).

The EXPANDER ROUTE TABLE CHANGE COUNT field counts the number of times the expander route table has been modified by the self-configuring expander device. Self-configuring expander devices shall support this field. This field shall be set to at least 0001h at power on. If the self-configuring expander device modified the expander route table since responding to a previous REPORT EXPANDER ROUTE TABLE LIST request, it shall increment this field at least once from the value in the previous REPORT EXPANDER ROUTE TABLE LIST response. This field shall wrap to at least 0001h after the maximum value (i.e., FFFFh) has been reached.

NOTE 111 - Application clients that use the EXPANDER ROUTE TABLE CHANGE COUNT field should read it often enough to ensure that it does not increment a multiple of 65 536 times between reading the field.

The CONFIGURING bit is defined in the REPORT GENERAL response (see 10.4.3.3).

The NUMBER OF REPORT EXPANDER ROUTE TABLE DESCRIPTORS field indicates the number of REPORT EXPANDER ROUTE TABLE descriptors contained in the REPORT EXPANDER ROUTE TABLE descriptor list.

The FIRST ROUTED SAS ADDRESS INDEX field indicates the index of the first REPORT EXPANDER ROUTE TABLE descriptor reported in the REPORT EXPANDER ROUTE TABLE descriptor list.

The LAST ROUTED SAS ADDRESS INDEX field indicates the index of the last REPORT EXPANDER ROUTE TABLE descriptor reported in the REPORT EXPANDER ROUTE TABLE descriptor list. The management application client may set the STARTING ROUTED SAS ADDRESS INDEX field in its next REPORT EXPANDER ROUTE TABLE LIST request to the value of this field plus one.

The REPORT EXPANDER ROUTE TABLE descriptor list contains REPORT EXPANDER ROUTE TABLE descriptors as defined in 10.4.3.16.4.

[Editor's Note 61: require they be in ascending order based on routed SAS address](#)

The CRC field is defined in 10.4.3.2.

10.4.3.16.4 REPORT EXPANDER ROUTE TABLE descriptor

Table 369 defines the REPORT EXPANDER ROUTE TABLE descriptor.

Table 369 — REPORT EXPANDER ROUTE TABLE descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	ROUTED SAS ADDRESS							
7								

Table 369 — REPORT EXPANDER ROUTE TABLE descriptor

Byte\Bit	7	6	5	4	3	2	1	0
8	(starting phy identifier + 47)							(starting phy identifier + 40)
13	(starting phy identifier + 7)							(starting phy identifier)
14	ZONE GROUP VALID	Reserved						
15	ZONE GROUP							

The ROUTED SAS ADDRESS field indicates the routed SAS address.

The PHY BIT MAP field indicates the phy(s) to which connection requests to the SAS address indicated by the ROUTED SAS ADDRESS field may be forwarded. This field is a bit map where each bit position indicates a corresponding phy (e.g., bit zero of byte 5 indicates the phy indicated by the starting phy identifier). A bit set to one indicates that connection requests to the SAS address indicated by the ROUTED SAS ADDRESS field may be forwarded to the corresponding phy. A bit set to zero indicates that connection requests to the SAS address indicated by the ROUTED SAS ADDRESS field are not forwarded to that corresponding phy. Bits representing phys beyond the value of the NUMBER OF PHYS field reported in the REPORT GENERAL response (see 10.4.3.3) shall be set to zero.

A ZONE GROUP VALID bit set to zero indicates that the ZONE GROUP field shall be ignored. A ZONE GROUP VALID bit set to one indicates that the ZONE GROUP field is valid.

The ZONE GROUP field is defined in 4.9.3.1.

10.4.3.17 CONFIGURE GENERAL function

The CONFIGURE GENERAL function requests actions by the device containing the management device server. This SMP function may be implemented by any management device server. In zoning expander devices, if zoning is enabled then this function shall only be processed from SMP initiator ports that have access to zone group 2 (see 4.9.3.2).

Table 370 defines the request format.

Table 370 — CONFIGURE GENERAL request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (80h)							
2	Reserved							
3	REQUEST LENGTH (04h)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						(LSB)
5								
6	Reserved							
7								
8	Reserved			UPDATE INITIAL TIME TO REDUCED FUNCTIONALITY	UPDATE STP SMP I_T NEXUS LOSS TIME	UPDATE STP MAXIMUM CONNECT TIME LIMIT	UPDATE STP BUS INACTIVITY TIME LIMIT	
9	Reserved							
10	(MSB)	STP BUS INACTIVITY TIME LIMIT						(LSB)
11								
12	(MSB)	STP MAXIMUM CONNECT TIME LIMIT						(LSB)
13								
14	(MSB)	STP SMP I_T NEXUS LOSS TIME						(LSB)
15								
16	INITIAL TIME TO REDUCED FUNCTIONALITY							
17	Reserved							
19								
20	(MSB)	CRC						(LSB)
23								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 80h.

The REQUEST LENGTH field shall be set to 04h.

If the management device server is not in an expander device or the EXPECTED EXPANDER CHANGE COUNT field is set to 0000h, the EXPECTED EXPANDER CHANGE COUNT field shall be ignored. If the management device server is in an expander device and the EXPECTED EXPANDER CHANGE COUNT field is not set to 0000h, then:

- a) if the EXPECTED EXPANDER CHANGE COUNT field contains the current expander change count (i.e., the value of the EXPANDER CHANGE COUNT field that would be returned by an SMP REPORT GENERAL response at this time), the management device server shall process the function; and
- b) If the EXPECTED EXPANDER CHANGE COUNT field does not contain the current expander change count, the management device server shall return a function result of INVALID EXPANDER CHANGE COUNT in the response frame (see table 315 in 10.4.3.2).

An UPDATE INITIAL TIME TO REDUCED FUNCTIONALITY bit set to one specifies that the INITIAL TIME TO REDUCED FUNCTIONALITY field shall be honored. An UPDATE INITIAL TIME TO REDUCED FUNCTIONALITY bit set to zero specifies that the INITIAL TIME TO REDUCED FUNCTIONALITY field shall be ignored.

An UPDATE STP BUS INACTIVITY TIME LIMIT bit set to one specifies that the STP BUS INACTIVITY TIME LIMIT field shall be honored. An UPDATE STP BUS INACTIVITY TIME LIMIT bit set to zero specifies that the STP BUS INACTIVITY TIME LIMIT field shall be ignored.

An UPDATE STP MAXIMUM CONNECT TIME LIMIT bit set to one specifies that the STP MAXIMUM CONNECT TIME LIMIT field shall be honored. An UPDATE STP MAXIMUM CONNECT TIME LIMIT bit set to zero specifies that the STP MAXIMUM CONNECT TIME LIMIT field shall be ignored.

An UPDATE STP SMP I_T NEXUS LOSS TIME bit set to one specifies that the STP SMP I_T NEXUS LOSS TIME field shall be honored. An UPDATE STP SMP I_T NEXUS LOSS TIME bit set to zero specifies that the STP SMP I_T NEXUS LOSS TIME field shall be ignored.

The STP BUS INACTIVITY TIME LIMIT field specifies the maximum time in 100 μ s increments that an STP target port is permitted to maintain a connection (see 4.1.12) while transmitting and receiving SATA_SYNC. When this time is exceeded, the STP target port shall close the connection. A value of 0000h in this field specifies that there is no bus inactivity time limit. This value is reported in the STP BUS INACTIVITY TIME LIMIT field in the SMP REPORT GENERAL response (see 10.4.3.3). The bus inactivity time limit is enforced by the port layer (see 8.2.3).

The STP MAXIMUM CONNECT TIME LIMIT field specifies the maximum duration of a connection (see 4.1.12) in 100 μ s increments (e.g., a value of 0001h in this field means that the time is less than or equal to 100 μ s and a value of 0002h in this field means that the time is less than or equal to 200 μ s). When this time is exceeded, the STP target port shall close the connection at the next opportunity. If the STP target port is transferring a frame when the maximum connection time limit is exceeded, the STP target port shall complete transfer of the frame before closing the connection. A value of 0000h in this field specifies that there is no maximum connection time limit. This value is reported in the STP MAXIMUM CONNECT TIME LIMIT field in the SMP REPORT GENERAL response (see 10.4.3.3). The maximum connection time limit is enforced by the port layer (see 8.2.3).

The STP SMP I_T NEXUS LOSS TIME field specifies the time that an STP target port or SMP initiator port shall retry connection requests that are rejected with responses indicating the destination port may no longer be present (see 8.2.2) before recognizing an I_T nexus loss (see 4.5). Table 371 defines the values of the STP SMP I_T NEXUS LOSS TIME field. This value is enforced by the port layer (see 8.2.2).

Table 371 — STP SMP I_T NEXUS LOSS TIME field

Code	Description
0000h	Vendor-specific amount of time.
0001h to FFFEh	Time in milliseconds.
FFFFh	The port shall never recognize an I_T nexus loss (i.e., it shall retry the connection requests forever).

NOTE 112 - The default value of the STP SMP I_T NEXUS LOSS TIME field should be non-zero. It is recommended that this value be 2 000 ms.

NOTE 113 - An STP initiator port should retry connection requests for the time indicated by the STP SMP L_T NEXUS LOSS TIME field in the SMP REPORT GENERAL response for the STP target port to which it is trying to establish a connection.

Editor's Note 62: That should be a "minimum" time

The INITIAL TIME TO REDUCED FUNCTIONALITY field specifies the minimum period of time, in 100 ms increments, that an expander device shall wait from originating a Broadcast (Expander) to reducing functionality (see 4.6.8). This value is reported in the INITIAL TIME TO REDUCED FUNCTIONALITY field in the SMP REPORT GENERAL response (see 10.4.3.3).

The CRC field is defined in 10.4.3.1.

Table 370 defines the response format.

Table 372 — CONFIGURE GENERAL response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (80h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 80h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.18 ENABLE DISABLE ZONING function

The ENABLE DISABLE ZONING function enables or disables zoning. This SMP function shall be supported by SMP target ports in zoning expander devices (see 4.9). Other SMP target ports shall not support this SMP function.

This function is an SMP zone configuration function (see 4.9.6.3).

Table 373 defines the request format.

Table 373 — ENABLE DISABLE ZONING request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (81h)							
2	Reserved							
3	REQUEST LENGTH (02h)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								(LSB)
6	Reserved						SAVE	
7	Reserved							
8	Reserved						ENABLE DISABLE ZONING	
9	Reserved							
11								
12	(MSB)	CRC						
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 81h.

The REQUEST LENGTH field shall be set to 02h.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the CONFIGURE GENERAL request (see 10.4.3.17).

The SAVE field specifies whether the management device server shall apply the specified changes to the current value and/or the saved value of the zoning enabled setting and is defined in table 374.

Table 374 — SAVE field

Code	Values updated	Return function result of SAVING NOT SUPPORTED if saving is not supported
00b	Current	no
01b	Saved ^a	yes
10b	Saved ^a , if saving is supported, and current.	no
11b	Saved ^a and current.	yes

^a The management device server shall return the function result without waiting for the save to complete, and set the SAVING bit to one in the REPORT GENERAL response until the save is complete.

The ENABLE DISABLE ZONING field is defined in table 375.

Table 375 — ENABLE DISABLE ZONING field

Code	Description
00b	No change
01b	Enable zoning
10b	Disable zoning
11b	Reserved

If the ENABLE DISABLE ZONING field is set to 11b (i.e., reserved), the management device server shall return a function result of UNKNOWN ENABLE DISABLE ZONING VALUE in the response frame (see table 315 in 10.4.3.2).

The CRC field is defined in 10.4.3.1.

Table 376 defines the response format.

Table 376 — ENABLE DISABLE ZONING response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (81h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 81h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.19 ZONED BROADCAST function

The ZONED BROADCAST function requests that the specified Broadcast (see 4.1.13) be forwarded as specified in 4.9.5. This SMP function shall be supported by management device servers in zoning expander devices (see 4.9). Other management device servers shall not support this SMP function. This SMP function shall only be processed from SMP initiator ports that have access to zone group 3 (see 4.9.3.2).

Table 377 defines the request format.

Table 377 — ZONED BROADCAST request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (85h)							
2	Reserved							
3	REQUEST LENGTH ((n - 7) / 4)							
4	Restricted							
5								
6	Reserved				BROADCAST TYPE			
7	NUMBER OF BROADCAST SOURCE ZONE GROUPS							
Broadcast source zone group list								
8	BROADCAST SOURCE ZONE GROUP (first)							
...	...							
	BROADCAST SOURCE ZONE GROUP (last)							
	PAD (if needed)							
n - 4								
n - 3	(MSB)							
n	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 85h.

The REQUEST LENGTH field specifies the number of dwords that follow, not including the CRC field.

The BROADCAST TYPE field specifies the type of Broadcast that shall be forwarded and is defined in Table 378.

Table 378 — BROADCAST TYPE field

Code	Description
0000b	Broadcast (Change)
0001b	Broadcast (Reserved Change 0)
0010b	Broadcast (Reserved Change 1)
0011b	Broadcast (SES)
0100b	Broadcast (Expander)
0101b	Broadcast (Asynchronous Event)
0110b	Broadcast (Reserved 3)
0111b	Broadcast (Reserved 4)
1000b	Broadcast (Zone Activate)
All others	Reserved for Broadcasts only supported by the ZONED BROADCAST function

The NUMBER OF BROADCAST SOURCE ZONE GROUPS field specifies the number of zone groups to which the specified Broadcast is to be forwarded.

The broadcast source zone group list contains BROADCAST SOURCE ZONE GROUP fields.

[Editor's Note 63: ascending order?](#)

Each BROADCAST SOURCE ZONE GROUP field specifies a source zone group for the Broadcast. The expander device forwards the Broadcast to each destination zone group accessible to that source zone group.

The PAD field contains zero, one, two, or three bytes set to 00h such that the total length of the SMP request is a multiple of four.

The CRC field is defined in 10.4.3.1.

Table 379 defines the response format.

Table 379 — ZONED BROADCAST response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (85h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	CRC							
7								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 85h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.20 ZONE LOCK function

The ZONE LOCK function locks a zoning expander device to provide exclusive access to SMP zone configuration functions (see 4.9.6.3) for one zone manager. All zoning expander devices shall support this function.

If:

- a) the ZONING ENABLED bit is set to one, the ZONE LOCKED bit is set to zero in the REPORT GENERAL response (see 10.4.3.3), and the SMP initiator port has access to zone group 2 (see 4.9.3.2);
- b) the ZONING ENABLED bit is set to one, the ZONE LOCKED bit is set to zero in the REPORT GENERAL response (see 10.4.3.3), and the PHYSICAL PRESENCE ASSERTED bit is set to one in the REPORT GENERAL response;
- c) the ZONING ENABLED bit is set to one, the ZONE LOCKED bit is set to zero in the REPORT GENERAL response (see 10.4.3.3), and the request contains the correct zone manager password;
- d) the ZONING ENABLED bit is set to zero and the PHYSICAL PRESENCE ASSERTED bit is set to one in the REPORT GENERAL response;
- e) the ZONING ENABLED bit is set to one, the ZONE LOCKED bit is set to one in the REPORT GENERAL response, and the request originated from the active zone manager; or
- f) the ZONING ENABLED bit is set to zero and the request contains the correct zone manager password,

then the management device server shall:

- a) set the ACTIVE ZONE MANAGER SAS ADDRESS field to the SAS address of the SMP initiator port in the ZONE LOCK response and the REPORT GENERAL response; and
- b) set the ZONE LOCKED bit to one in the REPORT GENERAL response.

When the management device server changes the ZONE LOCKED bit from zero to one, the locked zoning expander device sets the zoning expander shadow values equal to the zoning expander active values.

Table 380 defines the request format.

Table 380 — ZONE LOCK request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (86h)							
2	Reserved							
3	REQUEST LENGTH (09h)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								(LSB)
6	(MSB)	ZONE LOCK INACTIVITY TIME LIMIT						
7								(LSB)
8	ZONE MANAGER PASSWORD							
39								
40	(MSB)	CRC						
43								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 86h.

The REQUEST LENGTH field shall be set to 09h.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the CONFIGURE GENERAL request (see 10.4.3.17).

An ACTIVATE REQUIRED bit set to one specifies that the zoning expander device shall be unlocked only if the activate step has been completed. An ACTIVATE REQUIRED bit set to zero specifies that the zoning expander device shall be unlocked.

The ZONE LOCK INACTIVITY TIME LIMIT field specifies the minimum time that the locked expander device shall allow between any SMP zone configuration function requests or SMP ZONE LOCK requests from the active zone manager (i.e., the maximum time that a zone manager may allow to pass without accessing the locked expander device) and is reported in the SMP REPORT GENERAL response (see 10.4.3.20). This field specifies the number of 100 ms increments that a locked zoning expander device shall remain locked without processing any SMP zone configuration function or SMP ZONE LOCK function (e.g., a value of one in this field means that the time is less than or equal to 100 ms and a value of two in this field means that the time is less than or equal to 200 ms). A value of zero in this field specifies that there is no zone lock inactivity time limit (i.e., the zone lock inactivity timer is disabled).

The ZONE MANAGER PASSWORD field specifies a password used to allow permission to lock without physical presence being asserted.

The CRC field is defined in 10.4.3.1.

Table 381 defines the response format.

Table 381 — ZONE LOCK response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (86h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (03h)							
4	Reserved							
7								
8	ACTIVE ZONE MANAGER SAS ADDRESS							
15								
16	(MSB)	CRC						
19								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 86h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 03h.

The ACTIVE ZONE MANAGER SAS ADDRESS field is defined in the REPORT GENERAL response (see 10.4.3.3).

The CRC field is defined in 10.4.3.2.

10.4.3.21 ZONE ACTIVATE function

The ZONE ACTIVATE function causes the zoning expander device to set the zoning expander active values equal to the zoning expander shadow values (see 4.9.6.4). All zoning expander devices shall support this function. This function is an SMP zone configuration function (see 4.9.6.3).

Table 382 defines the request format.

Table 382 — ZONE ACTIVATE request

Byte\Bit	7	6	5	4	3	2	1	0					
0	SMP FRAME TYPE (40h)												
1	FUNCTION (87h)												
2	Reserved												
3	REQUEST LENGTH (01h)												
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT											
5								(LSB)					
6	Reserved												
7													
8	(MSB)	CRC											
11								(LSB)					

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 87h.

The REQUEST LENGTH field shall be set to 01h.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the CONFIGURE GENERAL request (see 10.4.3.17).

The CRC field is defined in 10.4.3.1.

Table 383 defines the response format.

Table 383 — ZONE ACTIVATE response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (87h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 87h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.22 ZONE UNLOCK function

The ZONE UNLOCK function unlocks a zoning expander device (see 4.9.6.5). All zoning expander devices shall support this function. This function is an SMP zone configuration function (see 4.9.6.3).

If a locked zoning expander device processes a ZONE UNLOCK request from the active zone manager then the management device server shall set the ZONE LOCKED bit to zero in the REPORT GENERAL response (see 10.4.3.3). If the CONFIGURING bit is set to one in the REPORT GENERAL response then the zoning expander device shall set the CONFIGURING bit to zero and originate a Broadcast (Change) from either:

- a) each zone group whose zone permission table entries or zone phy information has changed; or
- b) zone group 1.

Table 384 defines the request format.

Table 384 — ZONE UNLOCK request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (88h)							
2	Reserved							
3	REQUEST LENGTH (01h)							
4	Restricted							
5								
6	Reserved							ACTIVATE REQUIRED
7	Reserved							
8	(MSB)	CRC						
11								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 88h.

The REQUEST LENGTH field shall be set to 01h.

An ACTIVATE REQUIRED bit set to one specifies that the management device server shall unlock the zoning expander device only if the activate step has been completed. An ACTIVATE REQUIRED bit set to zero specifies that the management device server shall unlock the zoning expander device regardless of whether the activate step has been completed.

The CRC field is defined in 10.4.3.1.

Table 385 defines the response format.

Table 385 — ZONE UNLOCK response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (88h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)							
7	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 88h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.23 CONFIGURE ZONE MANAGER PASSWORD function

The CONFIGURE ZONE MANAGER PASSWORD function requests actions by the expander device containing the SMP target port. This SMP function shall be supported by SMP target ports in zoning expander devices. Other SMP target ports shall not support this SMP function. This SMP function shall only be processed if:

- a) the request is received from any SMP initiator port and specifies the correct zone manager password;
or
- b) the request is received from any SMP initiator port while physical presence is asserted.

Table 386 defines the request format.

Table 386 — CONFIGURE ZONE MANAGER PASSWORD request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (89h)							
2	Reserved							
3	REQUEST LENGTH (11h)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								(LSB)
6	Reserved						SAVE	
7	Reserved							
8	ZONE MANAGER PASSWORD							
39								
40	NEW ZONE MANAGER PASSWORD							
71								
72	(MSB)	CRC						
75								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 89h.

The REQUEST LENGTH field shall be set to 11h.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the CONFIGURE GENERAL request (see 10.4.3.17).

The SAVE field specifies whether the management device server shall apply the specified changes to the current value and/or the saved value of the zone manager password and is defined in table 387.

Table 387 — SAVE field

Code	Values updated	Return function result of SAVING NOT SUPPORTED if saving is not supported
00b	Current	no
01b	Saved ^a	yes
10b	Saved ^a , if saving is supported, and current.	no
11b	Saved ^a and current.	yes
^a The management device server shall return the function result without waiting for the save to complete, and set the SAVING bit to one in the REPORT GENERAL response until the save is complete.		

If physical presence is not asserted and the ZONE MANAGER PASSWORD field does not match the current zone manager password maintained by the management device server, then the management device server shall return a function result of NO MANAGEMENT ACCESS RIGHTS in the response frame (see table 315 in 10.4.3.2).

The NEW ZONE MANAGER PASSWORD field specifies a new value for the zone manager password maintained by the management device server. A NEW ZONE MANAGER PASSWORD field set to zero specifies that the zone manager password is disabled and all zone managers have access. A NEW ZONE MANAGER PASSWORD field set to DISABLED (see table 26 in 4.9.1) specifies that the zone manager password is disabled and shall only be allowed if physical presence is asserted.

Editor's Note 64: If the expander device chooses not to support the DISABLED value, what function result does it return?

The CRC field is defined in 10.4.3.1.

Table 388 defines the response format.

Table 388 — CONFIGURE ZONE MANAGER PASSWORD response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (89h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)							
7	CRC							
	(LSB)							

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 89h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.24 CONFIGURE ZONE PHY INFORMATION function

10.4.3.24.1 CONFIGURE ZONE PHY INFORMATION function overview

The CONFIGURE ZONE PHY INFORMATION function configures zone phy information for one or more phys in a locked zoning expander device. This function shall be supported by all zoning expander devices. This function is an SMP zone configuration function (see 4.9.6.3).

SMP zone configuration functions change the zoning expander shadow values. These do not become zoning expander active values until the activate step (see 4.9.6.4).

10.4.3.24.2 CONFIGURE ZONE PHY INFORMATION request

Table 389 defines the request format.

Table 389 — CONFIGURE ZONE PHY INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (8Ah)							
2	Reserved							
3	REQUEST LENGTH ((n - 7) / 4)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						(LSB)
5								
6	Reserved						SAVE	
7	NUMBER OF ZONE PHY CONFIGURATION DESCRIPTORS							
Zone phy configuration descriptor list								
8	Zone phy configuration descriptor (first)(see table 391 in 10.4.3.24.3)							
11								
...	...							
n - 7	Zone phy configuration descriptor (last)(see table 391 in 10.4.3.24.3)							
n - 4								
n - 3	(MSB)	CRC						(LSB)
n								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 8Ah.

The REQUEST LENGTH field specifies the number of dwords that follow, not including the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

The SAVE field specifies whether the management device server shall apply the specified changes to the current value and/or the saved value of the zone phy information and is defined in table 390.

Table 390 — SAVE field

Code	Values updated	Return function result of SAVING NOT SUPPORTED if saving is not supported
00b	Current	no
01b	Saved ^a	yes
10b	Saved ^a , if saving is supported, and current.	no
11b	Saved ^a and current.	yes
^a The management device server shall return the function result without waiting for the save to complete, and set the SAVING bit to one in the REPORT GENERAL response until the save is complete.		

The NUMBER OF ZONE PHY CONFIGURATION DESCRIPTORS field specifies the number of zone phy configuration descriptors in the request.

The zone phy configuration descriptor list contains zone phy configuration descriptors as defined in 10.4.3.24.3.

[Editor's Note 65: require they be in ascending order based on phy identifier](#)

The CRC field is defined in 10.4.3.1.

10.4.3.24.3 Zone phy configuration descriptor

Table 391 defines the zone phy configuration descriptor.

Table 391 — Zone phy configuration descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	PHY IDENTIFIER							
1	Reserved		INSIDE ZPSDS PERSISTENT	REQUESTED INSIDE ZPSDS	Reserved	ZONE GROUP PERSISTENT	Reserved	
2	Reserved							
3	ZONE GROUP							

The PHY IDENTIFIER field specifies the phy to which the zone phy configuration descriptor information shall be applied.

The INSIDE ZPSDS PERSISTENT bit specifies the value of the INSIDE ZPSDS PERSISTENT bit in the zone phy information (see 4.9.3.1).

The REQUESTED INSIDE ZPSDS bit specifies the value of the REQUESTED INSIDE ZPSDS bit in the zone phy information (see 4.9.3.1).

The ZONE GROUP PERSISTENT bit specifies the value of the ZONE GROUP PERSISTENT bit in the zone phy information (see 4.9.3.1).

The ZONE GROUP field specifies the value of the ZONE GROUP field in the zone phy information (see 4.9.3.1).

10.4.3.24.4 CONFIGURE ZONE PHY INFORMATION response

Table 392 defines the response format.

Table 392 — CONFIGURE ZONE PHY INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (8Ah)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)							
7	(LSB)							

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 8Ah.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.25 CONFIGURE ZONE PERMISSION TABLE function**10.4.3.25.1 CONFIGURE ZONE PERMISSION TABLE function overview**

The CONFIGURE ZONE PERMISSION TABLE function configures the zone permission table. This function shall be supported by all zoning expander devices. This function is an SMP zone configuration function (see 4.9.6.3).

SMP zone configuration functions change the zoning expander shadow values and do not become zoning expander active values until the zoning expander device processes the activate step (see 4.9.6.4).

Annex I describes examples of using multiple zone configuration descriptors.

10.4.3.25.2 CONFIGURE ZONE PERMISSION TABLE request

Table 393 defines the request format.

Table 393 — CONFIGURE ZONE PERMISSION TABLE request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (8Bh)							
2	Reserved							
3	REQUEST LENGTH ((n - 7) / 4)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								
6	STARTING SOURCE ZONE GROUP							
7	NUMBER OF ZONE PERMISSION CONFIGURATION DESCRIPTORS							
8	NUMBER OF ZONE GROUPS		Reserved				SAVE	
9	Reserved							
15								
Zone permission configuration descriptor list								
16	Zone permission configuration descriptor (first)(see table 396 in 10.4.3.25.3)							
31								
	...							
n - 20	Zone permission configuration descriptor (last)(see table 396 in 10.4.3.25.3)							
n - 4								
n - 3	(MSB)	CRC						
n								

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 8Bh.

The REQUEST LENGTH field specifies the number of dwords that follow, not including the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

The STARTING SOURCE GROUP field specifies the first source zone group (i.e., s) to be written with the first zone permission configuration descriptor.

The NUMBER OF ZONE PERMISSION CONFIGURATION DESCRIPTORS field specifies the number of zone permission configuration descriptors that follow.

The NUMBER OF ZONE GROUPS field specifies the number of elements in each zone permission configuration descriptor and is defined in table 394.

Table 394 — NUMBER OF ZONE GROUPS field

Code	Description
00b	128 zone groups
01b	256 zone groups
All others	Reserved

The SAVE field specifies whether the management device server shall apply the specified changes to the current value and/or the saved value of the zone permission table and is defined in table 395.

Table 395 — SAVE field

Code	Values updated	Return function result of SAVING NOT SUPPORTED if saving is not supported
00b	Current	no
01b	Saved ^a	yes
10b	Saved ^a , if saving is supported, and current.	no
11b	Saved ^a and current.	yes
^a The management device server shall return the function result without waiting for the save to complete, and set the SAVING bit to one in the REPORT GENERAL response until the save is complete.		

The zone permission configuration descriptor list contains zone permission configuration descriptors as defined in 10.4.3.25.3. The device server shall process the zone permission configuration descriptors in order (i.e., a subsequent zone permission configuration descriptor overrides a previous zone permission configuration descriptor).

The CRC field is defined in 10.4.3.2.

10.4.3.25.3 Zone permission configuration descriptor

Table 396 defines the zone permission configuration descriptor containing 128 zone groups.

Table 396 — Zone permission configuration descriptor for source zone group (i.e., s) for 128 zone groups

Byte\Bit	7	6	5	4	3	2	1	0
0	ZP[s, 127]	ZP[s, 126]	ZP[s, 125]	ZP[s, 124]	ZP[s, 123]	ZP[s, 122]	ZP[s, 121]	ZP[s, 120]
...	...							
15	ZP[s, 7] (ignored)	ZP[s, 6] (ignored)	ZP[s, 5] (ignored)	ZP[s, 4] (ignored)	ZP[s, 3]	ZP[s, 2]	ZP[s, 1] (ignored)	ZP[s, 0] (ignored)

Table 397 defines the zone permission configuration descriptor containing 256 zone groups.

Table 397 — Zone permission configuration descriptor for source zone group (i.e., s) for 256 zone groups

Byte\Bit	7	6	5	4	3	2	1	0
0	ZP[s, 255]	ZP[s, 254]	ZP[s, 253]	ZP[s, 252]	ZP[s, 251]	ZP[s, 250]	ZP[s, 249]	ZP[s, 248]
...	...							
31	ZP[s, 7] (ignored)	ZP[s, 6] (ignored)	ZP[s, 5] (ignored)	ZP[s, 4] (ignored)	ZP[s, 3]	ZP[s, 2]	ZP[s, 1] (ignored)	ZP[s, 0] (ignored)

The zone permission configuration descriptor contains all of the zone permission table entries for the source zone group (i.e., s). To preserve symmetry about the ZP[s, s] table axis, the management device server shall apply the same value to both the source and destination zone groups for the zone permission entries.

Table 398 defines how the zone permission descriptor bits shall be set by the management application client and processed by the management device server.

Table 398 — Zone permission configuration descriptor bit requirements

Source zone group (i.e., s)	Management application client requirement(s) ^a	Management device server requirement(s) ^a
0	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 2 through (z-1)] shall be set to zero.	ZP[s, 0 through (z-1)] shall be ignored.
1	ZP[s, 0 through (z-1)] shall be set to one.	ZP[s, 0 through (z-1)] shall be ignored.
4, 5, 6 or 7	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 4 through (z-1)] shall be set to zero.	ZP[s, 0 through (z-1)] shall be ignored.
2, 3, or 8 through (z-1) ^a	ZP[s, 0] shall be set to zero. ZP[s, 1] shall be set to one. ZP[s, 2 through 3] may be set to zero or one. ZP[s, 4 through 7] shall be set to zero. ZP[s, 8 through (z-1)] may be set to zero or one.	ZP[s, 0 through 1] shall be ignored. ZP[s, 2 through 3] shall be processed. ZP[s, 4 through 7] shall be ignored. ZP[s, 8 through (z-1)] shall be processed. For each source zone group t other than s, ZP[t, s] shall be set to ZP[s, t].
^a The number of zone groups (i.e., z) is specified in NUMBER OF ZONE GROUPS field.		

10.4.3.25.4 CONFIGURE ZONE PERMISSION TABLE response

Table 399 defines the response format.

Table 399 — CONFIGURE ZONE PERMISSION TABLE response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (8Bh)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 8Bh.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.26 CONFIGURE ROUTE INFORMATION function

The CONFIGURE ROUTE INFORMATION function sets an expander route entry within the expander route table of a configurable expander device. This SMP function shall be supported by management device servers in expander devices if the CONFIGURABLE ROUTE TABLE field is set to one in the SMP REPORT GENERAL response (see 10.4.3.3). Other management device servers shall not support this SMP function.

Table 400 defines the request format.

Table 400 — CONFIGURE ROUTE INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0						
0	SMP FRAME TYPE (40h)													
1	FUNCTION (90h)													
2	Reserved													
3	REQUEST LENGTH (09h)													
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT												
5								(LSB)						
6	(MSB)	EXPANDER ROUTE INDEX												
7								(LSB)						
8	Reserved													
9	PHY IDENTIFIER													
10	Reserved													
11														
12	DISABLE EXPANDER ROUTE ENTRY	Reserved												
13	Reserved													
15														
16	ROUTED SAS ADDRESS													
23														
24	Reserved													
39														
40	(MSB)	CRC												
43								(LSB)						

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 90h.

The REQUEST LENGTH field shall be set to 09h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 9 dwords before the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

The EXPANDER ROUTE INDEX field specifies the expander route index for the expander route entry being configured (see 4.6.7.3).

The PHY IDENTIFIER field specifies the phy for which the expander route entry is being configured (see 4.6.7.3).

The DISABLE EXPANDER ROUTE ENTRY bit specifies whether the ECM shall use the expander route entry to route connection requests (see 4.6.7.3). If the DISABLE EXPANDER ROUTE ENTRY bit is set to zero, then the ECM shall use the expander route entry to route connection requests. If the DISABLE EXPANDER ROUTE ENTRY bit is set to one, the ECM shall not use the expander route entry to route connection requests.

The ROUTED SAS ADDRESS field specifies the SAS address for the expander route entry being configured (see 4.6.7.3).

The CRC field is defined in 10.4.3.1.

Table 401 defines the response format.

Table 401 — CONFIGURE ROUTE INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (90h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	CRC							
7								

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 90h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.27 PHY CONTROL function

The PHY CONTROL function requests actions by the specified phy. This SMP function may be implemented by any management device server. In zoning expander devices, if zoning is enabled then this function shall only be processed from SMP initiator ports that have access to zone group 2 or the zone group of the specified phy (see 4.9.3.2).

Table 402 defines the request format.

Table 402 — PHY CONTROL request

Byte\Bit	7	6	5	4	3	2	1	0						
0	SMP FRAME TYPE (40h)													
1	FUNCTION (91h)													
2	Reserved													
3	REQUEST LENGTH (09h)													
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT												
5								(LSB)						
6	Reserved													
8														
9	PHY IDENTIFIER													
10	PHY OPERATION													
11	Reserved							UPDATE PARTIAL PATHWAY TIMEOUT VALUE						
12	Reserved													
23														
24	ATTACHED DEVICE NAME													
31														
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				Reserved									
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				Reserved									
34	Reserved													
35														
36	Reserved				PARTIAL PATHWAY TIMEOUT VALUE									
37	Reserved													
39														
40	(MSB)	CRC												
43								(LSB)						

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 91h.

The REQUEST LENGTH field shall be set to 09h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 9 dwords before the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

The PHY IDENTIFIER field specifies the phy (see 4.2.8) to which the SMP PHY CONTROL request applies.

Table 403 defines the PHY OPERATION field.

Table 403 — PHY OPERATION field (part 1 of 3)

Code	Operation	Description
00h	NOP	No operation.
01h	LINK RESET	If: a) a SAS phy is attached; b) a SATA phy is attached and there is no affiliation; or c) a SATA phy is attached and an affiliation exists for the STP initiator port with the same SAS address as the SMP initiator port that opened this SMP connection; then: a) if the specified phy is a physical phy, perform a link reset sequence (see 4.4) on the specified phy and enable the specified phy; and b) if the specified phy is a virtual phy, perform an internal reset and enable the specified phy. If a SATA phy is attached and an affiliation does not exist for the STP initiator port with the same SAS address as the SMP initiator port that opened this SMP connection, then the management device server shall return a function result of AFFILIATION VIOLATION in the response frame (see table 315 in 10.4.3.2). ^a See 7.11 for Broadcast (Change) requirements related to this phy operation in an expander device. Any affiliation (see 7.17.4) shall continue to be present. The phy shall bypass the SATA spinup hold state, if implemented (see 6.8.3.9). The management device server shall return the PHY CONTROL response without waiting for the LINK RESET phy operation to complete. While the LINK RESET phy operation is in progress, the management device server sets the NEGOTIATED PHYSICAL LINK RATE field and the NEGOTIATED PHYSICAL LINK RATE field to RESET_IN_PROGRESS in the SMP DISCOVER response (see 10.4.3.9).
^a Phys compliant with previous versions of this standard did not reject this phy operation due to affiliations. ^b Phys compliant with previous versions of this standard returned SMP FUNCTION REJECTED.		

Table 403 — PHY OPERATION field (part 2 of 3)

Code	Operation	Description
02h	HARD RESET	If the specified phy is a physical phy, perform a link reset sequence (see 4.4) on the specified phy and enable the specified phy. If the attached phy is a SAS phy or an expander phy, the link reset sequence shall include a hard reset sequence (see 4.4.2). If the attached phy is a SATA phy, the phy shall bypass the SATA spinup hold state. See 7.11 for Broadcast (Change) requirements related to this phy operation in an expander device. If the specified phy is a virtual phy, perform an internal reset and enable the specified phy. Any affiliation (see 7.17.4) shall be cleared. The management device server shall return the PHY CONTROL response without waiting for the HARD RESET phy operation to complete. While the HARD RESET phy operation is in progress, the management device server sets the NEGOTIATED PHYSICAL LINK RATE field and the NEGOTIATED PHYSICAL LINK RATE field to RESET_IN_PROGRESS in the SMP DISCOVER response (see 10.4.3.9).
03h	DISABLE	Disable the specified phy (i.e., stop transmitting valid dwords and receiving dwords on the specified phy). The LINK RESET and HARD RESET operations may be used to enable the phy. See 7.11 for Broadcast (Change) requirements related to this phy operation in an expander device.
04h	Reserved	
05h	CLEAR ERROR LOG	Clear the error log counters reported in the REPORT PHY ERROR LOG function (see 10.4.3.10) for the specified phy.
06h	CLEAR AFFILIATION	Clear an affiliation (see 7.17.4) from the STP initiator port with the same SAS address as the SMP initiator port that opened this SMP connection. If there is no such affiliation, the management device server shall return a function result of AFFILIATION VIOLATION^b in the response frame (see table 315 in 10.4.3.2).
07h	TRANSMIT SATA PORT SELECTION SIGNAL	This function shall only be supported by phys in an expander device. If the expander phy incorporates an STP/SATA bridge and supports SATA port selectors, the phy shall transmit the SATA port selection signal (see 6.6) which causes the SATA port selector to select the attached phy as the active host phy and make its other host phy inactive. See 7.11 for Broadcast (Change) requirements related to this phy operation in an expander device. Any affiliation (see 7.17.4) shall be cleared. If the expander phy does not support SATA port selectors, then the management device server shall return a function result of PHY DOES NOT SUPPORT SATA. If the expander phy supports SATA port selectors but is attached to a SAS phy or an expander phy, the management device server shall return a function result of SMP FUNCTION FAILED in the response frame (see table 315 in 10.4.3.2).
^a Phys compliant with previous versions of this standard did not reject this phy operation due to affiliations. ^b Phys compliant with previous versions of this standard returned SMP FUNCTION REJECTED.		

Table 403 — PHY OPERATION field (part 3 of 3)

Code	Operation	Description
08h	CLEAR STP I_T NEXUS LOSS	The STP I_T NEXUS LOSS OCCURRED bit in the REPORT PHY SATA function (see 10.4.3.11) shall be set to zero.
09h	SET ATTACHED DEVICE NAME	If the expander phy is attached to a SATA phy, set the ATTACHED DEVICE NAME field reported in the DISCOVER response (see 10.4.3.9) to the value of the ATTACHED DEVICE NAME field in the PHY CONTROL request.
All others	Reserved	
^a Phys compliant with previous versions of this standard did not reject this phy operation due to affiliations. ^b Phys compliant with previous versions of this standard returned SMP FUNCTION REJECTED.		

If the operation specified by the PHY OPERATION field is unknown, the management device sever shall return a function result of SMP FUNCTION FAILED in the response frame (see table 315 in 10.4.3.2) and not process any other fields in the request.

If the PHY IDENTIFIER field specifies the phy which is being used for the SMP connection and a phy operation of LINK RESET, HARD RESET, or DISABLE is requested, the management device server shall not perform the requested operation and shall return a function result of SMP FUNCTION FAILED in the response frame (see table 315 in 10.4.3.2).

An UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit set to one specifies that the PARTIAL PATHWAY TIMEOUT VALUE field shall be honored. An UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit set to zero specifies that the PARTIAL PATHWAY TIMEOUT VALUE field shall be ignored.

The ATTACHED DEVICE NAME field is used by the SET ATTACHED DEVICE NAME phy operation and is reserved for all other phy operations. If a management application client detects the ATTACHED DEVICE NAME field set to zero in the DISCOVER response when a SATA device is attached, it shall set the ATTACHED DEVICE NAME field based on the IDENTIFY (PACKET) DEVICE data retrieved by an ATA application client in the same SAS initiator device as follows:

- if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is correct and words 108-111 (i.e., the World Wide Name field) are not set to zero, set this field to the world wide name indicated by words 108-111 according to table 12 in 4.2.5;
- if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is correct and words 108-111 (i.e., the World Wide Name) are set to zero, set this field to 00000000 00000000h; and
- if IDENTIFY (PACKET) DEVICE data word 255 (i.e., the Integrity word) is not correct, set this field to 00000000 00000000h.

Editor's Note 66: a) and b) could be combined, but 07-216 will change b) to a hashed value of the Model Number and Serial Number fields

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field specifies the minimum physical link rate the phy shall support during a link reset sequence (see 4.4.1). Table 404 defines the values for this field. This value is reported in the DISCOVER response (see 10.4.3.9). If this field is changed along with a phy operation of LINK RESET or HARD RESET, that phy operation shall utilize the new value for this field.

The PROGRAMMED MAXIMUM PHYSICAL LINK RATE field specifies the maximum physical link rates the phy shall support during a link reset sequence (see 4.4.1). Table 404 defines the values for this field. This value is reported in the DISCOVER response (see 10.4.3.9). If this field is changed along with a phy operation of LINK RESET or HARD RESET, that phy operation shall utilize the new value for this field.

Table 404 — PROGRAMMED MINIMUM PHYSICAL LINK RATE and PROGRAMMED MAXIMUM PHYSICAL LINK RATE fields

Code	Description
0h	Do not change current value
1h - 7h	Reserved
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future physical link rates

If the PROGRAMMED MINIMUM PHYSICAL LINK RATE field or the PROGRAMMED MAXIMUM PHYSICAL LINK RATE field is set to an unsupported or reserved value, or the PROGRAMMED MINIMUM PHYSICAL LINK RATE field and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field are set to an invalid combination of values (e.g., the minimum is greater than the maximum), the management device server shall not change either of their values and may return a function result of SMP FUNCTION FAILED in the response frame (see table 315 in 10.4.3.2). If it returns a function result of SMP FUNCTION FAILED, it shall not perform the requested phy operation.

The PARTIAL PATHWAY TIMEOUT VALUE field specifies the amount of time in microseconds the expander phy shall wait after receiving an Arbitrating (Blocked On Partial) confirmation from the ECM before requesting that the ECM resolve pathway blockage (see 7.12.4.5). A PARTIAL PATHWAY TIMEOUT VALUE field value of zero (i.e., 0 μ s) specifies that partial pathway resolution shall be requested by the expander phy immediately upon reception of an Arbitrating (Blocked On Partial) confirmation from the ECM. This value is reported in the DISCOVER response (see 10.4.3.9). The PARTIAL PATHWAY TIMEOUT VALUE field is only honored when the UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit is set to one.

The CRC field is defined in 10.4.3.1.

Table 405 defines the response format.

Table 405 — PHY CONTROL response

Byte/Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (91h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 91h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.28 PHY TEST FUNCTION function

The PHY TEST FUNCTION function requests actions by the specified phy. This SMP function may be implemented by any management device server. In zoning expander devices, if zoning is enabled then this function shall only be processed from SMP initiator ports that have access to zone group 2 or the zone group of the specified phy (see 4.9.3.2).

Table 406 defines the request format.

Table 406 — PHY TEST FUNCTION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (92h)							
2	Reserved							
3	REQUEST LENGTH (09h)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								(LSB)
6	Reserved							
8								
9	PHY IDENTIFIER							
10	PHY TEST FUNCTION							
11	PHY TEST PATTERN							
12	Reserved							
14								
15	Reserved				PHY TEST PATTERN PHYSICAL LINK RATE			
16	Reserved							
18								
19	PHY TEST PATTERN DWORDS CONTROL							
20	PHY TEST PATTERN DWORDS							
27								
28	Reserved							
39								
40	(MSB)	CRC						
43								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 92h.

The REQUEST LENGTH field shall be set to 09h. For compatibility with previous versions of this standard, a REQUEST LENGTH field set to 00h specifies that there are 9 dwords before the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

The PHY IDENTIFIER field specifies the phy (see 4.2.8) to which the SMP PHY TEST PATTERN request applies.

If the PHY IDENTIFIER field specifies the phy which is being used for the SMP connection, the management device server shall not perform the requested operation and shall return a function result of SMP FUNCTION FAILED in the response frame (see table 315 in 10.4.3.2).

The PHY TEST FUNCTION field specifies the phy test function to be performed, and is defined in table 407. If the PHY TEST FUNCTION field specifies a phy test function that is not supported by the phy, the management device server shall return a function result of UNKNOWN PHY TEST FUNCTION in the response frame (see table 315 in 10.4.3.2).

Table 407 — PHY TEST FUNCTION field

Code	Name	Description
00h	STOP	If the selected phy is performing a phy test function, then the selected phy shall stop performing the phy test function and originate a link reset sequence. If the selected phy is not performing a phy test function, then this function has no effect on the selected phy.
01h	TRANSMIT_PATTERN	If the selected phy is not performing a phy test function, the selected phy shall be set to transmit the phy test pattern specified by the PHY TEST PATTERN field at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field and set to ignore its receiver. If the selected phy receives data while transmitting the pattern, then the selected phy shall ignore the received data. If the selected phy is performing a phy test function, the management device server shall return a function result of PHY TEST FUNCTION IN PROGRESS in the response frame (see table 315 in 10.4.3.2).
02h - EFh	Reserved	
F0h - FFh	Vendor specific	

If the PHY TEST FUNCTION field is set to 01h, the PHY TEST PATTERN field specifies the phy test pattern to be performed, and is the same as that defined in table 303 for the Protocol-Specific diagnostic page (see 10.2.9.2). The phy test pattern shall be sent at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field.

The PHY TEST PATTERN PHYSICAL LINK RATE field specifies the physical link rate at which the phy test function, if any, shall be performed. Table 408 defines the values for this field.

Table 408 — PHY TEST PATTERN PHYSICAL LINK RATE field

Code	Description
0h - 7h	Reserved
8h	1,5 Gbps
9h	3 Gbps
Ah	6 Gbps
Bh - Fh	Reserved for future physical link rates

The PHY TEST PATTERN DWORDS CONTROL field and the PHY TEST PATTERN DWORDS field are as defined in table 303 for the Protocol-Specific diagnostic page (see 10.2.9.2).

The CRC field is defined in 10.4.3.1.

Table 409 defines the response format.

Table 409 — PHY TEST FUNCTION response

Byte/Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (92h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 92h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

10.4.3.29 CONFIGURE PHY EVENT function

10.4.3.29.1 CONFIGURE PHY EVENT function overview

The CONFIGURE PHY EVENT function configures phy events (see 4.11) about the specified phy. This SMP function may be implemented by any management device server. In zoning expander devices, if zoning is enabled then this function shall only be processed from SMP initiator ports that have access to zone group 2 or the zone group of the specified phy (see 4.9.3.2).

10.4.3.29.2 CONFIGURE PHY EVENT request

Table 410 defines the request format.

Table 410 — CONFIGURE PHY EVENT request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (93h)							
2	Reserved							
3	REQUEST LENGTH ((n - 7) / 4)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5								(LSB)
6	Reserved							CLEAR PEAKS
7	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11	NUMBER OF PHY EVENT CONFIGURATION DESCRIPTORS							
Phy event configuration descriptor list								
12	Phy event configuration descriptor (first)(see table 411 in 10.4.3.29.3)							
	...							
	Phy event configuration descriptor (last)(see table 411 in 10.4.3.29.3)							
n - 4								
n - 3	(MSB)	CRC						
n								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 93h.

The REQUEST LENGTH field specifies the number of dwords that follow, not including the CRC field.

The EXPECTED EXPANDER CHANGE COUNT field is defined in the SMP CONFIGURE GENERAL request (see 10.4.3.17).

A CLEAR PEAKS field set to one specifies that all phy event peak value detectors shall be set to zero. A CLEAR PEAKS field set to zero specifies no change to the phy event peak value detectors.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) for which information shall be reported.

The NUMBER OF PHY EVENT CONFIGURATION DESCRIPTORS field specifies the number of phy event configuration descriptors, and shall be set to the same value as the NUMBER OF PHY EVENT DESCRIPTORS field in the SMP REPORT PHY EVENT function (see 10.4.3.13).

The phy event configuration descriptor list contains phy event configuration descriptors as defined in 10.4.3.29.3.

The CRC field is defined in 10.4.3.1.

10.4.3.29.3 CONFIGURE PHY EVENT request phy event configuration descriptor

Table 411 defines the phy event configuration descriptor.

Table 411 — Phy event configuration descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
2								
3	PHY EVENT SOURCE							
4	(MSB)	PEAK VALUE DETECTOR THRESHOLD						
7								(LSB)

The PHY EVENT SOURCE field, defined in table 37 in 4.11, specifies the type of event that shall be recorded by the corresponding phy event monitor.

If the phy event source is a peak value detector, the PEAK VALUE DETECTOR THRESHOLD field specifies the value of the peak value detector that causes the expander device to originate a Broadcast (Expander)(see 7.2.5.5). If the phy event source is not a peak value detector, the PEAK VALUE DETECTOR THRESHOLD field is reserved.

If the PHY EVENT SOURCE field contains a value that is not supported, the management device server shall return a function result of UNKNOWN PHY EVENT SOURCE in the response frame (see table 315 in 10.4.3.2).

10.4.3.29.4 CONFIGURE PHY EVENT response

Table 412 defines the response format.

Table 412 — CONFIGURE PHY EVENT response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (93h)							
2	FUNCTION RESULT							
3	RESPONSE LENGTH (00h)							
4	(MSB)	CRC						
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 93h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The RESPONSE LENGTH field shall be set to 00h.

The CRC field is defined in 10.4.3.2.

Annex A

(normative)

Jitter tolerance patterns

A.1 Jitter tolerance pattern (JTPAT)

The jitter tolerance pattern (JTPAT) consists of:

- 1) a long run of low transition density pattern;
- 2) a long run of high transition density pattern; and
- 3) another short run of low transition density pattern.

The transitions between the pattern segments stress the receiver. The JTPAT is designed to contain the phase shift in both polarities, from 0 to 1 and from 1 to 0. The critical pattern sections with the phase shifts are underlined in table A.1 and table A.2.

Table A.1 shows the JTPAT when there is positive running disparity (RD+) (see 6.2) at the beginning of the pattern. The 8b and 10b values of each character are shown.

Table A.1 — JTPAT for RD+

Dword(s)	Beginning RD	First character	Second character	Third character	Fourth character	Ending RD
0 to 40	RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD+
		1000011100b	0111100011b	1000011100b	0111100011b	
	The above dword of low transition density pattern is sent a total of 41 times					
41	RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D20.3 (74h)	RD-
		1000011100b	0111100011b	1000011100b	0010111100b	
	The above dword containing phase shift 11100001011b is sent 1 time					
42	RD-	D30.3 (7Eh)	D11.5 (ABh)	D21.5 (B5h)	D21.5 (B5h)	RD+
		0111100011b	1101001010b	1010101010b	1010101010b	
	The above dword containing phase shift 00011110100b is sent 1 time					
43 to 54	RD+	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	RD+
		1010101010b	1010101010b	1010101010b	1010101010b	
	The above dword of high transition density pattern is sent a total of 12 times					
55	RD+	D21.5 (B5h)	D30.2 (5Eh)	D10.2 (4Ah)	D30.3 (7Eh)	RD+
		1010101010b	1000010101b	0101010101b	0111100011b	
	The above dword containing phase shift 01010000b and 10101111b is sent 1 time					

If the same 8b characters specified in table A.1 are used when there is negative running disparity (RD-) at the beginning of the pattern, the resulting 10b pattern is different and does not provide the critical phase shifts. To achieve the same phase shift effects with RD-, a different 8b pattern is required. Table A.2 shows the JTPAT when there is negative running disparity (RD-) at the beginning of the pattern. The 8b and 10b values of each character are shown.

Table A.2 — JTPAT for RD-

Dword(s)	Beginning RD	First character	Second character	Third character	Fourth character	Ending RD
0 to 40	RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD-
		0111100011b	1000011100b	0111100011b	1000011100b	
	The above dword of low transition density pattern is sent a total of 41 times					
41	RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D11.3 (6Bh)	RD+
		0111100011b	1000011100b	0111100011b	1101000011b	
	The above dword containing phase shift 00011110100b is sent 1 time					
42	RD+	D30.3 (7Eh)	D20.2 (54h)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
		1000011100b	0010110101b	0101010101b	0101010101b	
	The above dword containing phase shift 11100001011b is sent 1 time					
43 to 54	RD-	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
		0101010101b	0101010101b	0101010101b	0101010101b	
	The above dword of high transition density pattern is sent a total of 12 times					
55	RD-	D10.2 (4Ah)	D30.5 (BEh)	D21.5 (B5h)	D30.3 (7Eh)	RD-
		0101010101b	0111101010b	1010101010b	1000011100b	
	The above dword containing phase shift 10101111b and 01010000b is sent 1 time					

Table A.3 shows a pattern containing both JTPAT for RD+ and JTPAT for RD-. The 10b pattern resulting from encoding the 8b pattern contains the desired bit sequences for the phase shifts with both starting running disparities.

Table A.3 — JTPAT for RD+ and RD-

Dword(s)	First character	Second character	Third character	Fourth character	Notes
0 to 40	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	This dword is sent a total of 41 times.
	...	...	...	...	
41	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D20.3(74h)	This dword is sent once.
42	D30.3(7Eh)	D11.5(ABh)	D21.5(B5h)	D21.5(B5h)	This dword is sent once.
43 to 54	D21.5(B5h)	D21.5(B5h)	D21.5(B5h)	D21.5(B5h)	This dword is sent a total of 12 times.
	...	...	...	...	
55	D21.5(B5h)	D30.2(5Eh)	D10.2(4Ah)	D30.3(7Eh)	This dword is sent once.
56 to 96	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	This dword is sent a total of 41 times.
	...	...	...	...	
97	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D11.3(6Bh)	This dword is sent once.
98	D30.3(7Eh)	D20.2(54h)	D10.2(4Ah)	D10.2(4Ah)	This dword is sent once.
99 to 110	D10.2(4Ah)	D10.2(4Ah)	D10.2(4Ah)	D10.2(4Ah)	This dword is sent a total of 12 times.
	...	...	...	...	
111	D10.2(4Ah)	D30.5(BEh)	D21.5(B5h)	D30.3(7Eh)	This dword is sent once.

A.2 Compliant jitter tolerance pattern (CJTPAT)

The compliant jitter tolerance pattern (CJTPAT) is the JTPAT for RD+ and RD- (see table A.3 in A.1) included as the payload in an SSP DATA frame or an SMP frame. The CJTPAT is:

- 1) SOF;
- 2) 6 data dwords containing either:
 - A) an SSP DATA frame header; or
 - B) an SMP frame header followed by 23 bytes;
- 3) 112 data dwords containing JTPAT for RD+ and RD-;
- 4) 1 data dword containing a CRC value; and
- 5) EOF.

Deletable primitives may be included in the transmission of the CJTPAT, but the number of deletable primitives transmitted should be as small as possible so that the percentage of the transfer that is the JTPAT is as high as possible.

Because the SOF, EOF, and CRC are the same in SSP and SMP, CJTPAT complies with:

- a) the SSP frame transmission format defined by the SSP link layer (see figure 177 in 7.16.3);
- b) the SSP frame format defined by the SSP transport layer (see table 155 in 9.2.1);
- c) the SMP frame transmission format defined by the SMP link layer (see figure 191 in 7.18.1); and
- d) the SMP frame format defined by the SMP transport layer (see table 183 in 9.4.1).

When a phy transmits a frame, it XORs the 8b data provided by the application layer with the output of a scrambler before transmission (see 7.6). It does not modify primitives and it re-initializes the scrambler at the beginning of each frame (e.g., at SOF). If the application layer XORs the desired 8b pattern with the expected output of the scrambler prior to submitting it to the transmitter, then the transmitter transmits the desired pattern. The 8b data dwords are scrambled by XORing the pattern with the expected scrambler dword output, taking into account the position of the 8b data dwords within the frame.

Figure A.1 shows how to pre-scramble CJTPAT into the phy transmitter so CJTPAT results on the physical link.

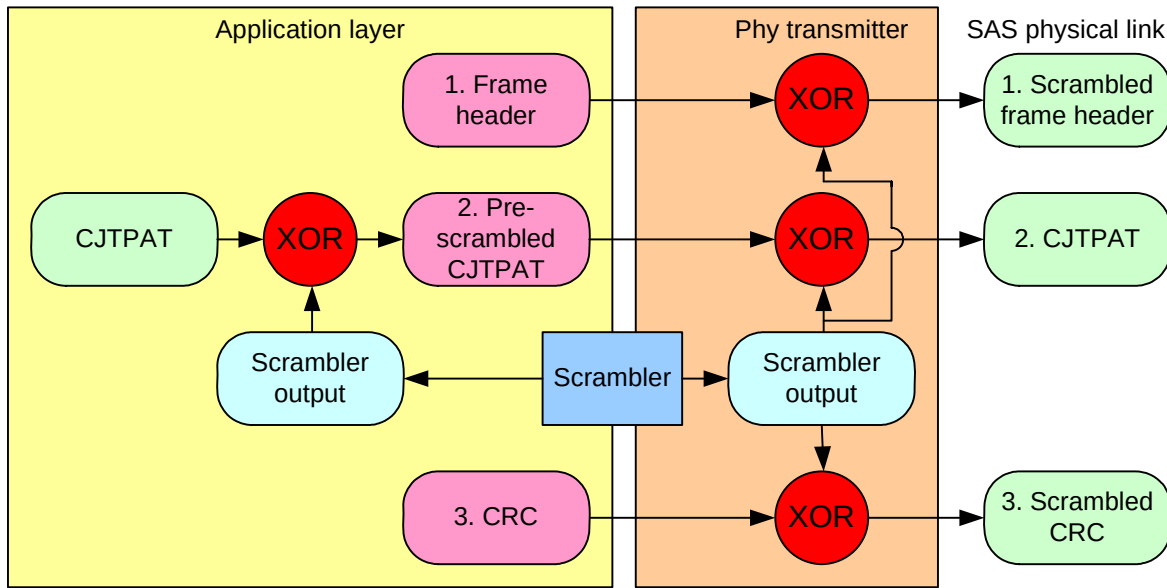


Figure A.1 — CJTPAT pre-scrambling

Table A.4 defines CJTPAT.

The “SSP frame contents” column in table A.4 shows the interpretation of the frame if viewed as an SSP DATA frame.

The “SMP frame contents” column in table A.4 shows the interpretation of the frame if viewed as an SMP frame.

The “Pre-scrambled CJTPAT” column in table A.4 shows the result of XORing CJTPAT with the expected scrambler output before presenting the frame to the phy transmitter. If the data in this column is supplied to the phy transmitter and it is scrambled again, the data in the “CJTPAT” column is transmitted onto the physical link. The frame header is not pre-scrambled, and the CRC is calculated over the frame header and the pre-scrambled CJTPAT.

The “Scrambler output” column in table A.4 shows the scrambler output for each data dword in the frame. The scrambler output is independent of the data pattern.

The “CJTPAT” column in table A.4 shows CJTPAT, transmitted on the physical link.

Table A.4 — CJTPAT (part 1 of 4)

Data dword number	SSP frame contents	SMP frame contents	Pre-scrambled CJTPAT	Scrambler output	CJTPAT
Not applicable	SOF		SOF	Not applicable	SOF
0	SSP frame header	SMP frame header and 3 frame-type dependent bytes	unknown	C2D2768Dh	unknown
1		Frame-type dependent bytes	unknown	1F26B368h	unknown
2			unknown	A508436Ch	unknown
3			unknown	3452D354h	unknown
4			unknown	8A559502h	unknown
5			unknown	BB1ABE1Bh	unknown
6	INFORMATION UNIT field (dwords 0 - 7)	Frame-type dependent bytes	8428C943h	FA56B73Dh	7E7E7E7Eh
7			2D887565h	53F60B1Bh	7E7E7E7Eh
8			8EFEE23Fh	F0809C41h	7E7E7E7Eh
9			0A01BD34h	747FC34Ah	7E7E7E7Eh
10			C0F82CEFh	BE865291h	7E7E7E7Eh
11			0411D9C8h	7A6FA7B6h	7E7E7E7Eh
12			4F1D98A8h	3163E6D6h	7E7E7E7Eh
13			8E488072h	F036FE0Ch	7E7E7E7Eh
14	INFORMATION UNIT field (dwords 8-15)	Frame-type dependent bytes	608D9457h	1EF3EA29h	7E7E7E7Eh
15			954A58EAh	EB342694h	7E7E7E7Eh
16			2DFB4569h	53853B17h	7E7E7E7Eh
17			9734A233h	E94ADC4Dh	7E7E7E7Eh
18			235E70F6h	5D200E88h	7E7E7E7Eh
19			177F93AEh	6901EDD0h	7E7E7E7Eh
20			84E046A0h	FA9E38DEh	7E7E7E7Eh
21			16A53579h	68DB4B07h	7E7E7E7Eh
22	INFORMATION UNIT field (dwords 16 - 23)	Frame-type dependent bytes	3B743D05h	450A437Bh	7E7E7E7Eh
23			E873A976h	960DD708h	7E7E7E7Eh
24			414B98E6h	3F35E698h	7E7E7E7Eh
25			8008E6DBh	FE7698A5h	7E7E7E7Eh
26			B670896Bh	C80EF715h	7E7E7E7Eh
27			181EEED1h	666090AFh	7E7E7E7Eh
28			848EABB5h	FAF0D5CBh	7E7E7E7Eh
29			55FC7EE1h	2B82009Fh	7E7E7E7Eh

Table A.4 — CJTPAT (part 2 of 4)

Data dword number	SSP frame contents	SMP frame contents	Pre-scrambled CJTPAT	Scrambler output	CJTPAT
30	INFORMATION UNIT field (dwords 24 - 31)	Frame-type dependent bytes	704F0AEFh	0E317491h	7E7E7E7Eh
31			088A1460h	76F46A1Eh	7E7E7E7Eh
32			8A131736h	F46D6948h	7E7E7E7Eh
33			05B3F4Edh	7BCD8A93h	7E7E7E7Eh
34			6B6DD300h	1513AD7Eh	7E7E7E7Eh
35			600C8090h	1E72FEEeh	7E7E7E7Eh
36			DE6AD445h	A014AA3Bh	7E7E7E7Eh
37			5DD4AA99h	23AAD4E7h	7E7E7E7Eh
38	INFORMATION UNIT field (dwords 32 - 39)	Frame-type dependent bytes	CEA2E019h	B0DC9E67h	7E7E7E7Eh
39			9EDB0D85h	E0A573FBh	7E7E7E7Eh
40			78B4EA31h	06CA944Fh	7E7E7E7Eh
41			1D9CEC6Ch	63E29212h	7E7E7E7Eh
42			3B061C13h	4578626Dh	7E7E7E7Eh
43			2D5872EDh	53260C93h	7E7E7E7Eh
44			40275C7Ch	3E592202h	7E7E7E7Eh
45			5510B41Dh	2B6ECA63h	7E7E7E7Eh
46	INFORMATION UNIT field (dwords 40 - 47)	Frame-type dependent bytes	1D146161h	636A1F1Fh	7E7E7E7Eh
47			4BCBD799h	35B5A9EDh	7E7E7E74h
48			34091548h	4AA2A0FDh	7EABB5B5h
49			C41A5423h	71AFE196h	B5B5B5B5h
50			5460CED7h	E1D57B62h	B5B5B5B5h
51			E015E33Fh	55A0568Ah	B5B5B5B5h
52			37643CDDh	82D18968h	B5B5B5B5h
53			96F9014Ah	234CB4FFh	B5B5B5B5h
54	INFORMATION UNIT field (dwords 48 - 55)	Frame-type dependent bytes	36FDABCAh	83481E7Fh	B5B5B5B5h
55			07AF5DCAh	B21AE87Fh	B5B5B5B5h
56			1C705F78h	A9C5EACDh	B5B5B5B5h
57			D7B41976h	6201ACC3h	B5B5B5B5h
58			43BC8C7Bh	F60939CEh	B5B5B5B5h
59			8CEAC3C8h	395F767Dh	B5B5B5B5h
60			9A10EDF4h	2FA55841h	B5B5B5B5h
61			36330004h	836D4A7Ah	B55E4A7Eh

Table A.4 — CJTPAT (part 3 of 4)

Data dword number	SSP frame contents	SMP frame contents	Pre-scrambled CJTPAT	Scrambler output	CJTPAT
62	INFORMATION UNIT field (dwords 56 - 63)	Frame-type dependent bytes	46F32604h	388D587Ah	7E7E7E7Eh
63			09438122h	773DFF5Ch	7E7E7E7Eh
64			425DE2CDh	3C239CB3h	7E7E7E7Eh
65			2833EFDEh	564D91A0h	7E7E7E7Eh
66			3D93759Fh	43ED0BE1h	7E7E7E7Eh
67			E60A57D9h	987429A7h	7E7E7E7Eh
68			9B53A5DCh	E52DDBA2h	7E7E7E7Eh
69			99F3B601h	E78DC87Fh	7E7E7E7Eh
70	INFORMATION UNIT field (dwords 64 - 71)	Frame-type dependent bytes	74C6B817h	0AB8C669h	7E7E7E7Eh
71			1AAEFDB7h	64D083C9h	7E7E7E7Eh
72			7B438744h	053DF93Ah	7E7E7E7Eh
73			9097A794h	EEE9D9Eah	7E7E7E7Eh
74			3AC345E9h	44BD3B97h	7E7E7E7Eh
75			719C35F2h	0FE24B8Ch	7E7E7E7Eh
76			8CF328EAh	F28D5694h	7E7E7E7Eh
77			1D6EC8A7h	6310B6D9h	7E7E7E7Eh
78	INFORMATION UNIT field (dwords 72 - 79)	Frame-type dependent bytes	69ECD0B0h	1792AECEh	7E7E7E7Eh
79			742850DFh	0A562EA1h	7E7E7E7Eh
80			CE36A117h	B048DF69h	7E7E7E7Eh
81			68645606h	161A2878h	7E7E7E7Eh
82			2B67B52Fh	5519CB51h	7E7E7E7Eh
83			678BC028h	19F5BE56h	7E7E7E7Eh
84			9181CAC8h	EFFFB4B6h	7E7E7E7Eh
85			CDFC100Ch	B3826E72h	7E7E7E7Eh
86	INFORMATION UNIT field (dwords 80 - 87)	Frame-type dependent bytes	9A0C53A4h	E4722DDAh	7E7E7E7Eh
87			1EC12F57h	60BF5129h	7E7E7E7Eh
88			5AF3EE8Bh	248D90F5h	7E7E7E7Eh
89			3378AC62h	4D06D21Ch	7E7E7E7Eh
90			00E86812h	7E96166Ch	7E7E7E7Eh
91			21D19DCAh	5FAFE3B4h	7E7E7E7Eh
92			2E12C62Bh	506CB855h	7E7E7E7Eh
93			258E4EE6h	5BF03098h	7E7E7E7Eh

Table A.4 — CJTPAT (part 4 of 4)

Data dword number	SSP frame contents	SMP frame contents	Pre-scrambled CJTPAT	Scrambler output	CJTPAT
94	INFORMATION UNIT field (dwords 88 - 95)	Frame-type dependent bytes	38AAC8CDh	46D4B6B3h	7E7E7E7Eh
95			7B65E06Fh	051B9E11h	7E7E7E7Eh
96			7F22BB28h	015CC556h	7E7E7E7Eh
97			9C6E4B91h	E21035EFh	7E7E7E7Eh
98			281E330Bh	56604D75h	7E7E7E7Eh
99			50081922h	2E76675Ch	7E7E7E7Eh
100			796A088Eh	071476F0h	7E7E7E7Eh
101			D18EF995h	AFF087EBh	7E7E7E7Eh
102	INFORMATION UNIT field (dwords 96 - 103)	Frame-type dependent bytes	651CA57Fh	1B62DB01h	7E7E7E7Eh
103			5D186107h	23661F6Ch	7E7E7E6Bh
104			8623FA6Dh	F877B027h	7E544A4Ah
105			BFA9C3E8h	F5E389A2h	4A4A4A4Ah
106			A48D7C5Bh	EEC73611h	4A4A4A4Ah
107			064EB1D9h	4C04FB93h	4A4A4A4Ah
108			A29D4578h	E8D70F32h	4A4A4A4Ah
109			F5BA761Eh	BFF03C54h	4A4A4A4Ah
110	INFORMATION UNIT field (dwords 104 - 111)	Frame-type dependent bytes	A90A764Bh	E3403C01h	4A4A4A4Ah
111			6AB08034h	20FACA7Eh	4A4A4A4Ah
112			D3080FC6h	9942458Ch	4A4A4A4Ah
113			7DA881C3h	37E2CB89h	4A4A4A4Ah
114			1050DDC9h	5A1A9783h	4A4A4A4Ah
115			8402E075h	CE48AA3Fh	4A4A4A4Ah
116			4C83ED2Bh	06C9A761h	4A4A4A4Ah
117			4C7E8BD5h	06C03EABh	4ABEB57Eh
118	CRC field ^a		depends on contents of first 6 data dwords	3D2D7984h	depends on contents of first 6 data dwords
Not applicable	EOF		<primitive>	<primitive>	<primitive>
^a The CRC field shall be set to a valid value for the frame.					

A phy or test equipment transmitting CJTPAT outside connections may transmit it with fixed content as defined in table A.5.

Table A.5 shows CJTPAT with fixed content:

- a) interpreted as an SSP frame, with the FRAME TYPE field in the frame header set to 01h (i.e., DATA), each other field in the frame header set to zero, the INFORMATION UNIT field containing JTPAT for RD+ and RD-, and the CRC field set to a fixed value; and

- b) interpreted as an SMP frame, with the SMP FRAME TYPE field in the frame header set to 01h (i.e., reserved), the frame-type dependent bytes containing JTPAT for RD+ and RD-, and the CRC field set to a fixed value.

Table A.5 — CJTPAT with fixed content

Data dword number	SSP frame contents	SMP frame contents	Pre-scrambled CJTPAT	Scrambler output	CJTPAT
Not applicable	SOF		SOF	Not applicable	SOF
0	FRAME TYPE field set to 01h (i.e., DATA frame) and 23 subsequent bytes each set to 00h	SMP FRAME TYPE field set to 01h (i.e., reserved) and 23 subsequent frame-type dependent bytes each set to 00h	01000000h	C2D2768Dh	C3D2768Dh
1			00000000h	1F26B368h	1F26B368h
2			00000000h	A508436Ch	A508436Ch
3			00000000h	3452D354h	3452D354h
4			00000000h	8A559502h	8A559502h
5			00000000h	BB1ABE1Bh	BB1ABE1Bh
6	INFORMATION UNIT field	Frame-type dependent bytes	See the INFORMATION UNIT field in table A.4		
117					
118	CRC field		79C211AAh	3D2D7984h	44EF682Eh
Not applicable	EOF		EOF	Not applicable	EOF

A.3 Considerations for a phy transmitting JTPAT and CJTPAT

A phy may be configured to transmit JTPAT for RD+ and RD- (see A.2) by:

- using the SMP PHY TEST FUNCTION function (see 10.4.3.28) or the Protocol-Specific diagnostic page (see 10.2.9.2) specifying the phy, with the PHY TEST FUNCTION field set to 01h (i.e., TRANSMIT_PATTERN), and the PHY TEST PATTERN field set to 01h (i.e., JTPAT); and
- vendor-specific mechanisms.

A phy may be configured to transmit CJTPAT (see A.2) by:

- using the SMP PHY TEST FUNCTION function (see 10.4.3.28) or the Protocol-Specific diagnostic page (see 10.2.9.2) specifying the phy, with the PHY TEST FUNCTION field set to 01h (i.e., TRANSMIT_PATTERN), and the PHY TEST PATTERN field set to 02h (i.e., CJTPAT);
- including CJTPAT as a data pattern while perform SCSI commands (e.g., the WRITE BUFFER command if the phy is in an initiator port, or the SCSI READ BUFFER command if the phy is in a target port). The frame length and scrambling need to be predictable to ensure the desired pattern is transmitted on the physical link; and
- vendor-specific mechanisms.

A.4 Considerations for a phy receiving JTPAT and CJTPAT

If a phy receives JTPAT (see A.1) inside or outside a connection, it considers the data dwords to be idle dwords and ignores them.

If a phy receives CJTPAT (see A.2) outside a connection, the SL receiver (see 7.14.2) considers the SOF and EOF to be unexpected dwords and ignores them, and considers the data dwords to be idle dwords and ignores them.

Phy-layer based phy event counters (e.g., invalid dword count, running disparity error count, loss of dword synchronization count, elasticity buffer overflow count, and received ERROR count) count events that occur while receiving idle dwords, so may be used to count events while receiving JTPAT or CTPAT.

If a phy receives CJTPAT inside an SSP connection, the phy expects it to have a valid frame header (e.g., valid frame type, source SAS address, etc.) and follow SSP frame transmission rules (e.g., RRDY credit, ACK or NAK exchange).

If a phy receives CJTPAT inside an SMP connection, the phy expects it to have a valid frame header (e.g., valid frame type) and follow SMP frame transmission rules (e.g., only one frame is transmitted in each direction per connection). Sending CJTPAT inside SMP connections is not recommended.

No standard mechanism is defined to configure a phy to expect to receive JTPAT or CJTPAT (e.g., to compare the incoming pattern to the expected pattern).

Annex B (informative)

Signal performance measurements

B.1 Signal performance measurements overview

This annex specifies the configuration requirements for making electrical performance measurements. These measurements consist of signal output, signal tolerance, and return loss. Standard loads are used in all cases so that independent specification of connection components and transportability of the measurement results are possible.

NOTE 114 - The methodology for making return loss measurements are specified in this annex although this standard does not define any return loss values. Statements in this annex that imply return loss values are specified should be ignored.

B.2 Simple physical link

B.2.1 Simple physical link overview

The physical link is considered to consist of the following component parts:

- the transmitter device;
- the interconnect; and
- the receiver device.

Each of those components is connected by a separable connector. On a TxRx connection, signals travel in opposite directions down the same nominal path.

Figure B.1 shows a physical link and the location of the connectors.

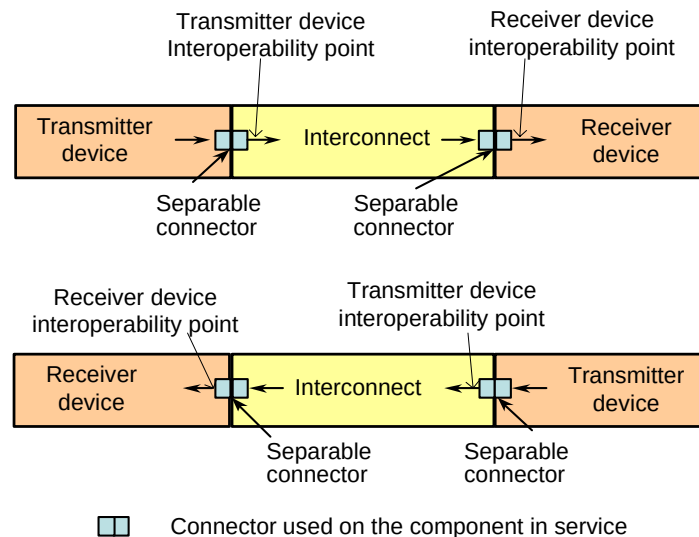


Figure B.1 — A simple physical link

Since connectors are always used in the mated condition, the only access to the signals is before the signal enters the mated connector (i.e., upstream) or after the signal exits the connector (i.e., downstream). Even if signals were able to be practically accessed at the point of mating within the connector, such access would disturb the connector to the point that the measurement of the signal would be compromised (e.g., attempting to access the unmated connector with probes does not provide valid results because the connector is not in the same condition when unmated as when mated and the probe contact points are not at the same location as the connector contact points).

In this annex, signal outputs are always measured downstream of the mated connector (see figure B.1) so that the contribution of the mated connector to the signal properties is included in the measurement. This approach assigns a portion of the connector losses to the upstream component, but it also makes the signal measurement conservative. If the connectors on both ends of the interconnect are the same, the additional loss at the downstream connector is offset by the reduced loss at the upstream connector. For transmitter devices, a slightly stronger transmitter is required to pass the signal through the downstream half of the connector that does not belong to the transmitter device. The signal coming into receiver devices is specified after the signal has passed through the connector.

Examination of the details of the measurement methods described in this annex shows that the mated connector issue may not be as severe as it appears.

B.2.2 Assumptions for the structure of the transmitter device and the receiver device

Figure B.2 shows the details of a transmitter device.

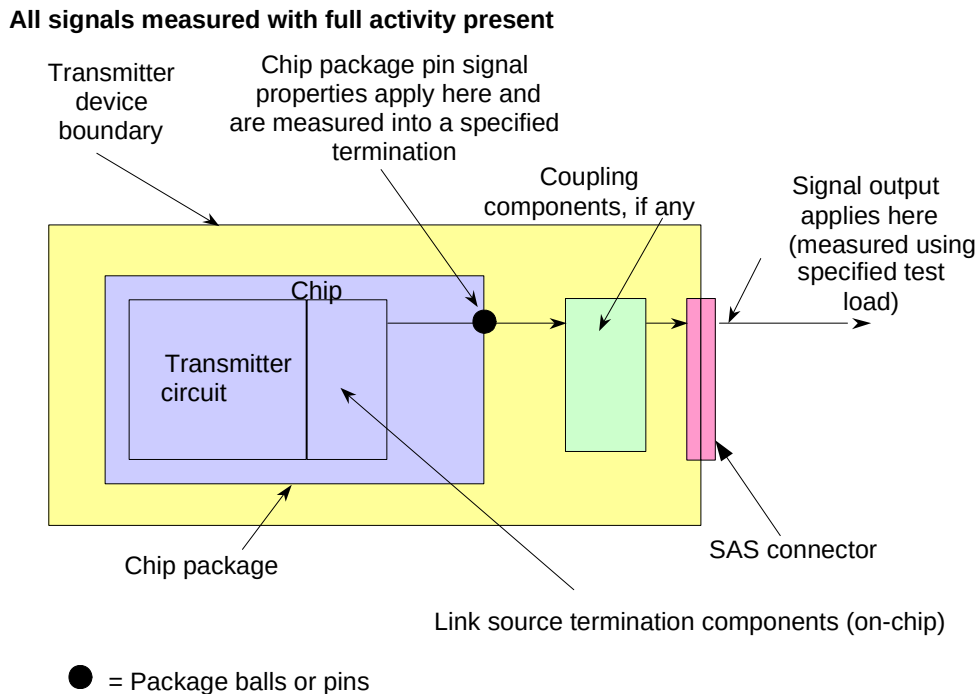


Figure B.2 — Transmitter device details

As figure B.2 shows, any of the following internal parts of this transmitter device may be labeled as the transmitter:

- the transmitter circuit in the chip;
- the chip itself; and
- the chip and its associated chip package.

The term transmitter is therefore not well defined and is not used in the terminology without a modifier.

The transmitter device contains:

- a connector (i.e., half a mated pair);
- coupling components, if any;
- PCB traces and vias;
- the chip package;
- ESD protection devices, if any;
- the source termination; and
- the transmitter circuit.

It is assumed that the source termination is contained within the chip package.

Interoperability points may be defined at the chip package pins in some standards (e.g., Ethernet XAUI). This standard does not define requirements at chip package pins.

Figure B.3 shows the details of a receiver device. It is similar to the transmitter device.

All signals measured with full activity present

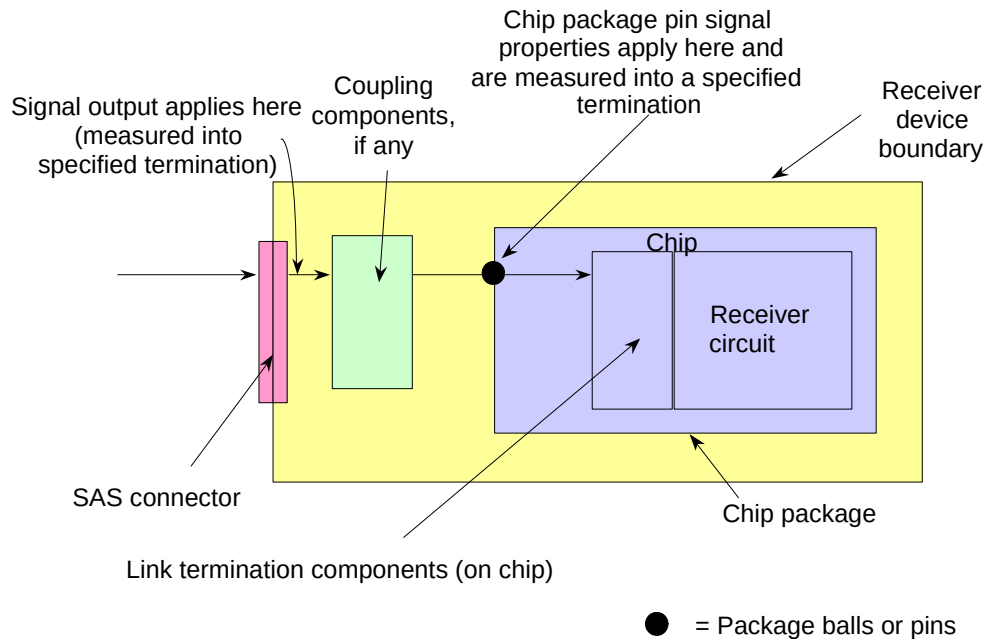


Figure B.3 — Receiver device details

As figure B.3 shows, any of the following internal parts of this receiver device may be labeled as the receiver:

- a) the receiver circuit in the chip;
- b) the chip itself; and
- c) the chip and its associated chip package.

The term receiver is therefore not well defined and is not used in the terminology without a modifier.

The receiver device contains:

- a) a connector (i.e., half a mated pair);
- b) coupling components, if any;
- c) PCB traces and vias;
- d) the chip package;
- e) ESD protection devices, if any;
- f) the physical link termination; and
- g) the receiver circuit.

It is assumed that the physical link termination is contained within the chip package.

B.2.3 Definition of receiver sensitivity and receiver device sensitivity

The term receiver sensitivity is problematic in common usage. This term is not used for interoperability in standards. A related term applicable to the receiver device input signal is receiver device sensitivity. While these two terms are related, they are significantly different because of the noise environment assumed. The description in this subclause is used to uniquely define these terms with the understanding that this standard discourages usage of either term.

Receiver sensitivity is defined as the minimum vertical inner eye opening at which the receiver chip delivers the required BER (see 5.3.3) with the horizontal eye opening at its minimum (i.e., maximum jitter is present) and all activity quiesced except for the receiver itself. The receiver in the receiver sensitivity refers to signal

properties at the chip package pin for the chip package that contains the receiver circuit. However, receiver sensitivity is not defined in this standard because there are no chip package pin specifications.

Receiver device sensitivity is defined as the minimum vertical inner eye opening measured at the signal output point for the input to the receiver device at which the receiver chip (i.e., the receiver circuit in the chip package on the board containing the receiver device interoperability point as shown in figure B.3) delivers the required BER (see 5.3.3) with:

- a) the minimum horizontal eye opening;
- b) all activity expected in the application for the receiver circuit present (i.e., not quiesced as for the receiver sensitivity definition); and
- c) the CJTPAT pattern being received (see Annex A).

Special test conditions are required to measure these sensitivities (see B.8). The terminology used in this standard is signal tolerance instead of receiver device sensitivity.

B.3 Measurement architecture requirements

B.3.1 General

Signal specifications are only meaningful if the signals are able to be measured with practical instrumentation. Another requirement is that different laboratories making measurements on the same signal get the same results within acceptable measurement error (i.e., the measurements need to be accessible, verifiable, and transportable). As of the publishing of this standard, there are no accepted standard for creating signals with traceable properties and with all the properties required for an effective signal specification architecture for high speed serial applications.

Some of the elements required for practical, verifiable, and transportable signal measurements are included in this standard.

Having signal specifications at interoperability points that do not depend on the actual properties of the other physical link components not under test requires that specified known test loads be used for the signal measurements. In service, the load presented to the interoperability point is that of the actual component and environment.

Interfacing with practical instruments also requires that specified impedance environments be used. This forces a signal measurement architecture where the impedance environment is 50 ohm single-ended (i.e., 100 ohm differential). It also forces the requirement for instrumentation-quality loads of the correct value.

Instrumentation-quality loads are readily available for simple transmission line termination. However, none are available for more complex loads that include specified propagation time, insertion loss properties, crosstalk properties, and jitter creation properties.

For signal tolerance measurements, the signal is calibrated before applying it to the interoperability point under test. This signal calibration is done by adjusting the properties of the specified signal source system as measured into a laboratory-quality test load until the desired signal tolerance specifications are met. The signal source system is then disconnected from the laboratory-quality test load and connected to the interoperability point under test. It is assumed that any changes to the signal from the calibration state to the measurement state are caused by the interoperability point under test and are therefore part of the performance sought by the measurement.

B.3.2 Relationship between signal compliance measurements at interoperability points and operation in systems

The signal measurements in this standard apply under specified test conditions that simulate some parts of the conditions existing in service (e.g., this simulation includes full-duplex traffic on all phys and under all applicable environmental conditions). Other features existing in service (e.g., non-ideal return loss in parts of the physical link that are not present when measuring signals in the specified test conditions) may be included in the specifications themselves. This methodology is required to specify signal performance requirements for each side of the interoperability point that do not depend on knowing the properties of the other side.

Measuring signals in an actual functioning system at an interoperability point does not verify compliance for the components on either side of the interoperability point, although it does verify that the specific combination of components in the system at the time of the measurement produces compliant signals. Interaction between components on either side of the interoperability point may allow the signal measured to be compliant, but this compliance may have resulted because one component does not meet the specification while the other exceeds the specification.

It is recommended that additional margin be allowed when performing signal compliance measurements to account for conditions existing in service that may not have been accounted for in the specified measurements and specifications.

B.4 De-embedding connectors in test fixtures

Connectors are part of the test fixtures (e.g. test loads) required for obtaining access to the interoperability points. This is intrinsic for most practical measurements because the connectors used on the service components are different from those used on the instrumentation.

The effects of the portions of the connector that is used on the test fixture need to be accounted for in order to not penalize the interoperability point under test by the performance of the test fixture connector. This accounting process is termed de-embedding.

Figure B.4 shows two cases that apply.

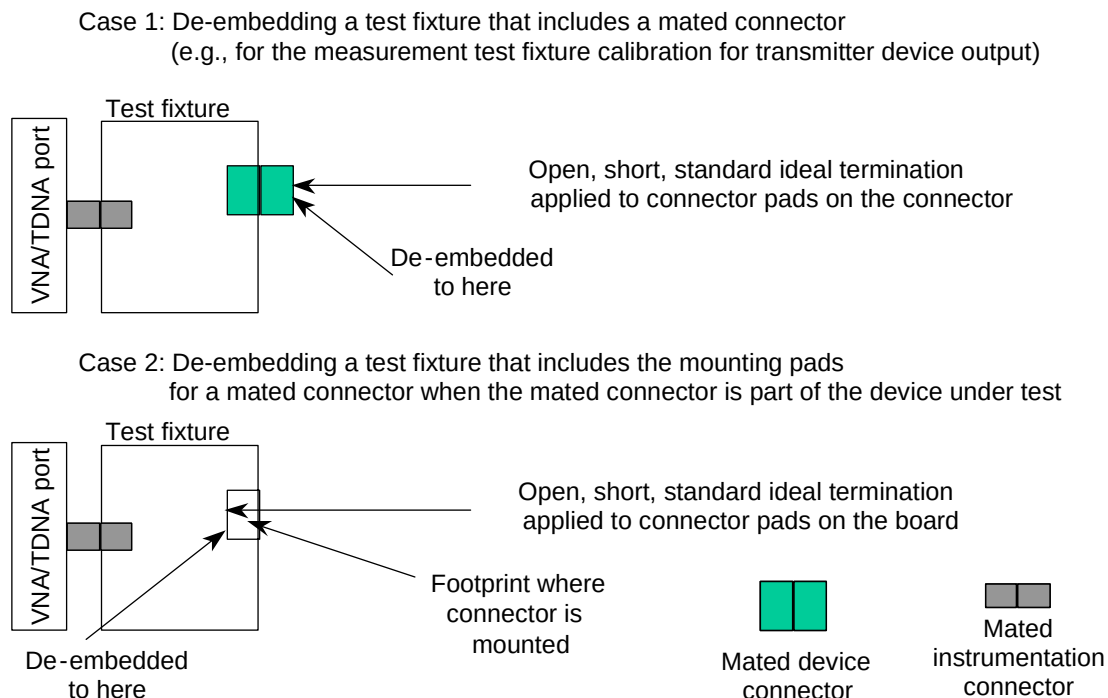


Figure B.4 — De-embedding of connectors in test fixtures

The de-embedding process assumes that the test fixture is linear and that S-parameter methodologies (see B.9) are used. Fundamentally, an S-parameter model for the test fixture with or without the connector in place is the result. Knowing this model for the test fixture, with or without the connector in place, allows simulation of the impact of the test fixture on the signal measurement.

B.5 Measurement conditions for signal output at the transmitter device

The measurement conditions required for a differential transmitter device signal output are shown in figure B.5. Figure B.5 applies to the following cases:

- the transmitter device is directly attached to the zero length test load (see 5.3.6.2); and
- the transmitter device is attached to the TCTF test load (see 5.3.6.3).

To simulate the properties of the interconnect assembly, instrumentation-quality test loads as shown in figure B.6 are used.

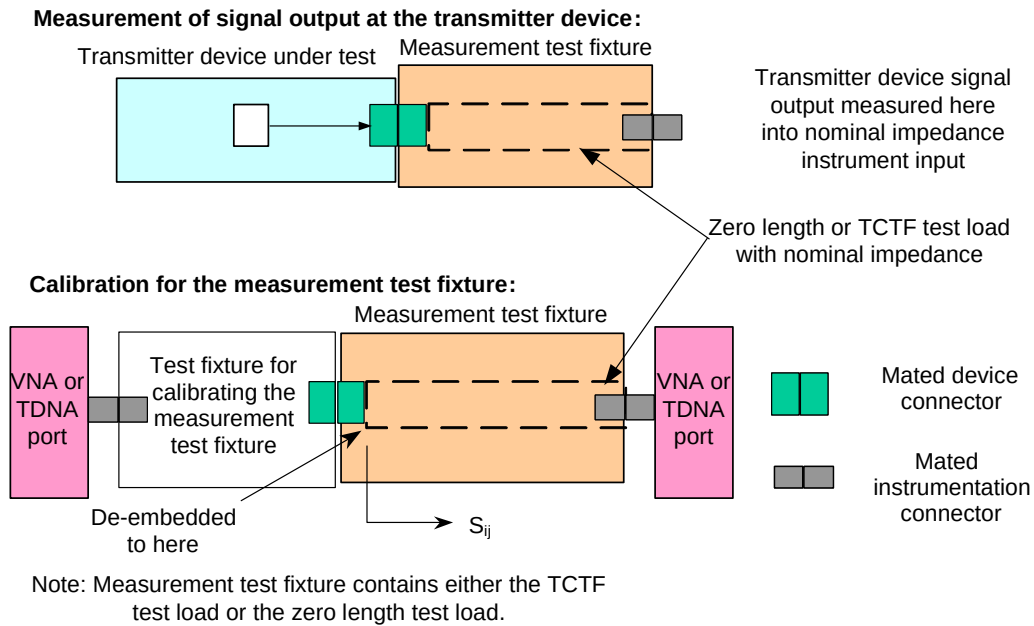


Figure B.5 — Measurement conditions for signal output at the transmitter device

An instrumentation-quality cable assembly connecting the measurement test fixture to the instrumentation port is usually required. This cable assembly is considered part of the instrumentation and is not specifically shown in figure B.5, figure B.6, figure B.7, figure B.8, figure B.9, figure B.10, and figure B.11.

A measurement test fixture may be constructed from an instrumentation-quality TCTF test load with instrumentation-quality connectors and a connector adapter as shown in figure B.6. This scheme may be useful when using multiple device connector types but adds extra components that may increase loss and delay. For best accuracy, this scheme is not recommended. Extra components make it more difficult for the transmitter device to meet the required output specifications.

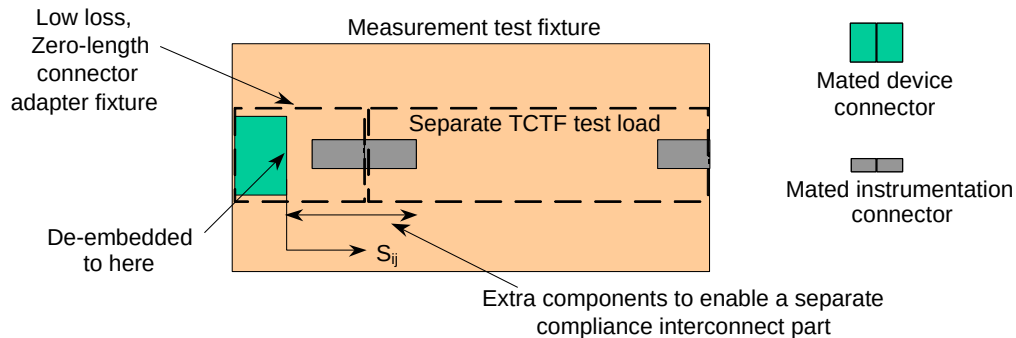


Figure B.6 — Transmitter device signal output measurement test fixture details

B.6 Measurement conditions for signal tolerance at the transmitter device

The measurement conditions required for the signal tolerance at the differential transmitter device interoperability point are shown in figure B.7. Figure B.7 shows the test signal is launched into the interconnect assembly (e.g., cable assembly or PCB) that is attached to the receiver device.

This standard does not include this performance requirement but is included here for completeness.

Measurement of signal tolerance at transmitter device (e.g., IT and CT):

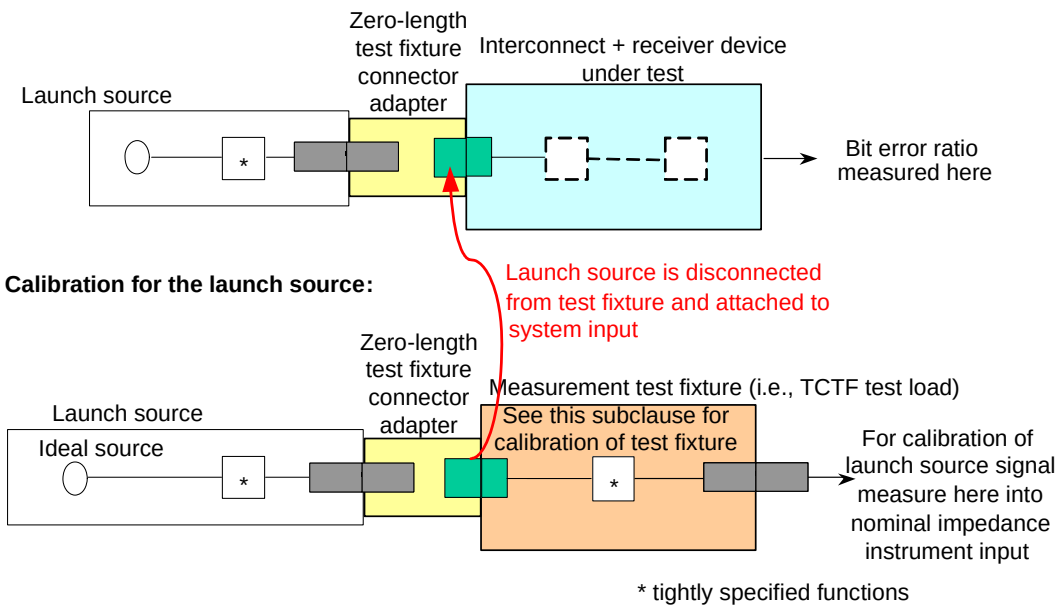


Figure B.7 — Measurement conditions for signal tolerance at the transmitter device

Figure B.8 shows calibration of the measurement test fixture.

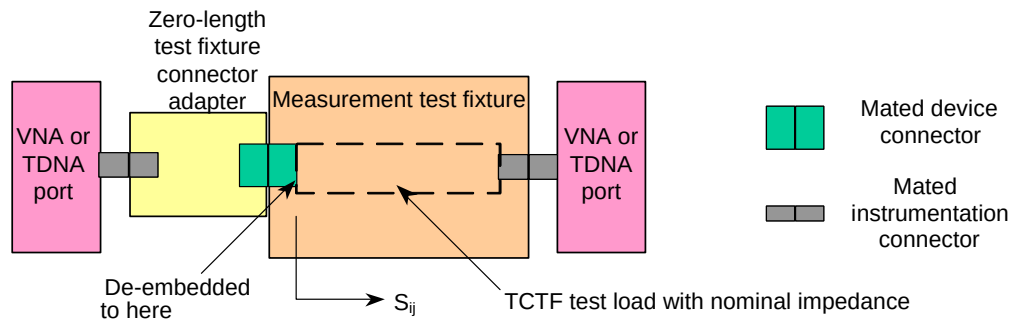


Figure B.8 — Calibration of test fixture for signal tolerance at the transmitter device

B.7 Measurement conditions for signal output at the receiver device

Figure B.9 shows the measurement conditions for the signal output at the receiver device.

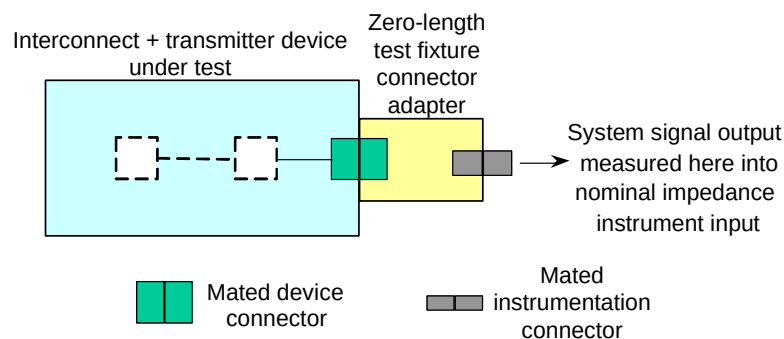


Figure B.9 — Measurement conditions for signal output at the receiver device

The interconnect may be the zero-length connector adaptor where the transmitter device is connected directly to the receiver device.

B.8 Measurement conditions for signal tolerance at the receiver device

Figure B.10 shows the measurement conditions required for the signal tolerance at the differential receiver device interoperability point (see 5.3.7.2).

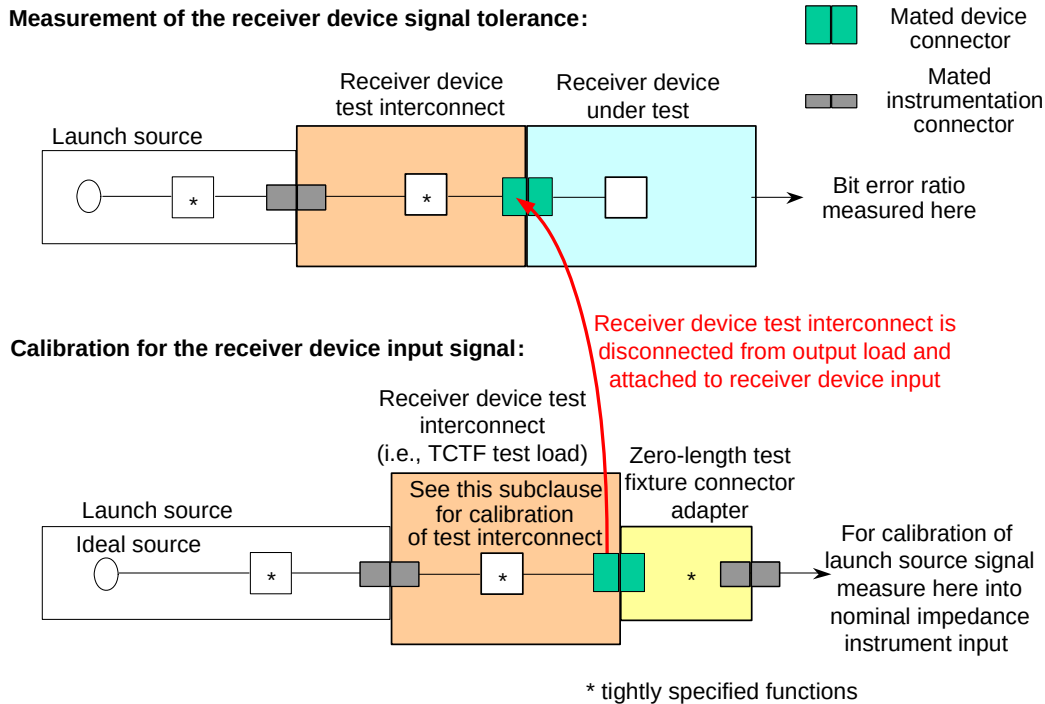


Figure B.10 — Measurement conditions for signal tolerance at the receiver device

Figure B.11 shows calibration of the measurement test fixture.

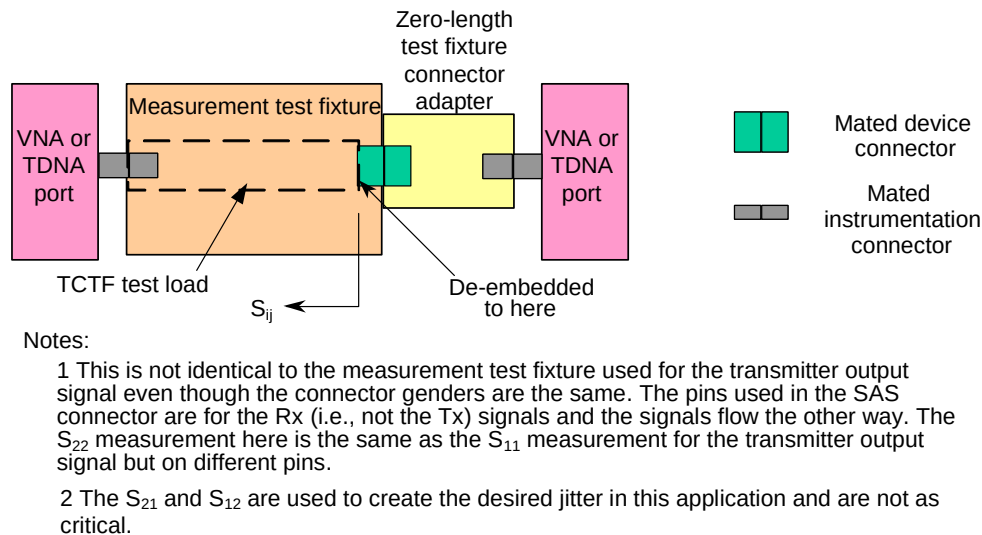


Figure B.11 — Calibration of test fixture for signal tolerance at the receiver device

B.9 S-parameter measurements

B.9.1 S-parameter overview

Properties of physical link elements that are linear may be represented by S-parameter (i.e., scattering parameter) spectra. There are two problematic areas when applying S-parameters to differential electrical physical links:

- a) naming conventions; and
- b) use of single-ended vector network methods on differential and common-mode systems.

This subclause explores both of these areas.

Measurement architecture for the most common conditions are described in some detail.

B.9.2 S-parameter naming conventions

There are two types of measurements performed with S-parameters:

- a) return loss from the same port of the element; and
- b) transfer function or insertion loss across the element.

Each s-parameter is a function of frequency returning complex numbers and is expressed with:

- a) a magnitude component, usually expressed in dB; and
- b) a phase component.

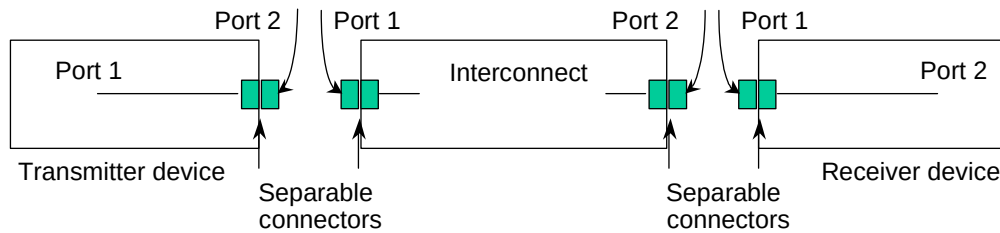
For a two-port linear element having ports i and j with the signals being either differential or common-mode, S_{ij} is the ratio of the signal coming out of the i th port (i.e., the response) to the signal coming into the j th port (i.e., the stimulus).

A port number convention is used where the downstream port is always port 2 and the upstream port is always port 1. The stream direction is determined by the direction of the primary signal launched from the transmitter device to the receiver device (e.g., in this standard, since each differential pair carries a signal in only one direction, the port nearest the transmitter device is port 1 and the port nearest the receiver device is port 2).

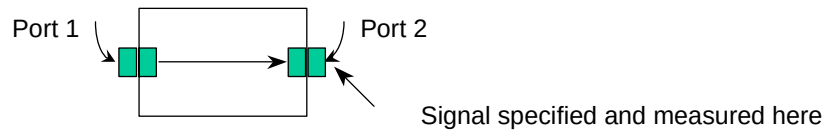
There are four combinations of ports for a two-ported system yielding the following S-parameters:

- a) S_{11} : return loss measured at port 1;
- b) S_{21} : transfer function or insertion loss measured at port 1;
- c) S_{22} : upstream return loss. Return loss measure at port 2 of the element. The measurement is the same kind of measurement that is done at port 1 to measure S_{11} ; and
- d) S_{12} : upstream insertion loss. Insertion loss measured at port 2 of the element. The measurement is the same kind of measurement that is done at port 1 to measure S_{21} .

Figure B.12 shows the port naming conventions for physical link elements, loads, and where those elements exit.



Port definitions for loads used for signal output testing and S-parameter measurements in multiline configurations



This load has ideal differential and common mode properties

Note: The transmitter device port 1 and receiver device port 2 are internal and are not defined. They are required to be an ideal source and an ideal sink, respectively.

Figure B.12 — S-parameter port naming conventions

B.9.3 Use of single-ended instrumentation in differential applications

There are four categories of S-parameters for a differential system:

- a) S_{DDij} : differential stimulus, differential response;
- b) S_{CDij} : differential stimulus, common-mode response (i.e., mode conversion causing emissions);
- c) S_{DCij} : common-mode stimulus, differential response (i.e., mode conversion causing susceptibility); and
- d) S_{CCij} : common-mode stimulus, common-mode response.

All the measurements specified in this standard relate to differential signal pairs, and all specified S-parameters are of the S_{DDij} form.

Figure B.13 shows the connections that are made to a four port VNA or TDNA for measuring S-parameters on a four single-ended port black box device.

VNA ports are all single-ended; the differential and common-mode properties for differential ports are calculated internal to the VNA or may mathematically derived. If using a TDNA, consult the details for the specific instrument. Four analyzer ports are required to measure the properties of two differential ports.

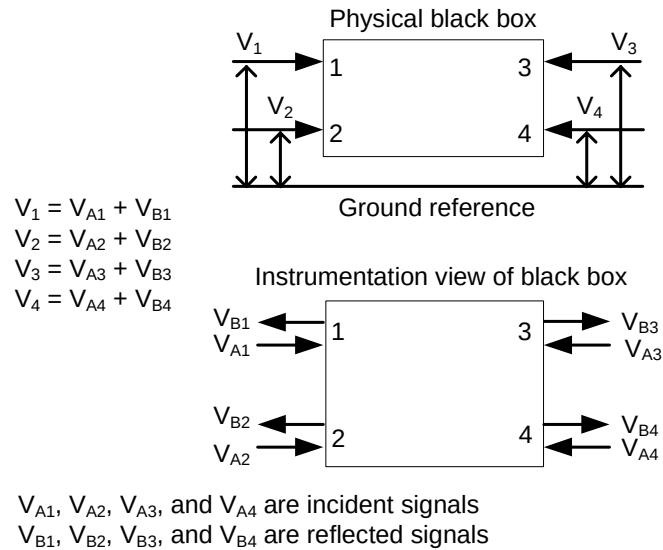


Figure B.13 — Four single-ended port or two differential port element

Figure B.14 shows the set of S-parameters for a single-ended system and for a differential system.

Single-ended

S-parameters:

		Stimulus			
Response		S_{11}	S_{12}	S_{13}	S_{14}
		S_{21}	S_{22}	S_{23}	S_{24}
		S_{31}	S_{32}	S_{33}	S_{34}
		S_{41}	S_{42}	S_{43}	S_{44}

Differential

S-parameters:

		Differential		Common-mode	
Response	Differential	S_{DD11}	S_{DD12}	S_{DC11}	S_{DC12}
		S_{DD21}	S_{DD22}	S_{DC21}	S_{DC22}
	Common-mode	S_{CD11}	S_{CD12}	S_{CC11}	S_{CC12}
		S_{CD21}	S_{CD22}	S_{CC21}	S_{CC22}

Figure B.14 — S-parameters for single-ended and differential systems

See SFF-8416 for details on connections required for differential S-parameter measurements.

B.9.4 Measurement configurations for physical link elements

B.9.4.1 Measurement configuration overview

Special test fixtures are required to make S-parameter measurements partly because the connectors used on real physical link elements are different from those used on instrumentation. The goal is for these test fixtures to be as invisible as possible.

See B.9.4 for the description of the measurement configurations. All of these measurements are return loss in this annex. A more complete set of S-parameters is used as part of the calibration process for test fixtures.

B.9.4.2 Transmitter device return loss

Figure B.15 shows the configuration to be used for the transmitter device return loss measurement.

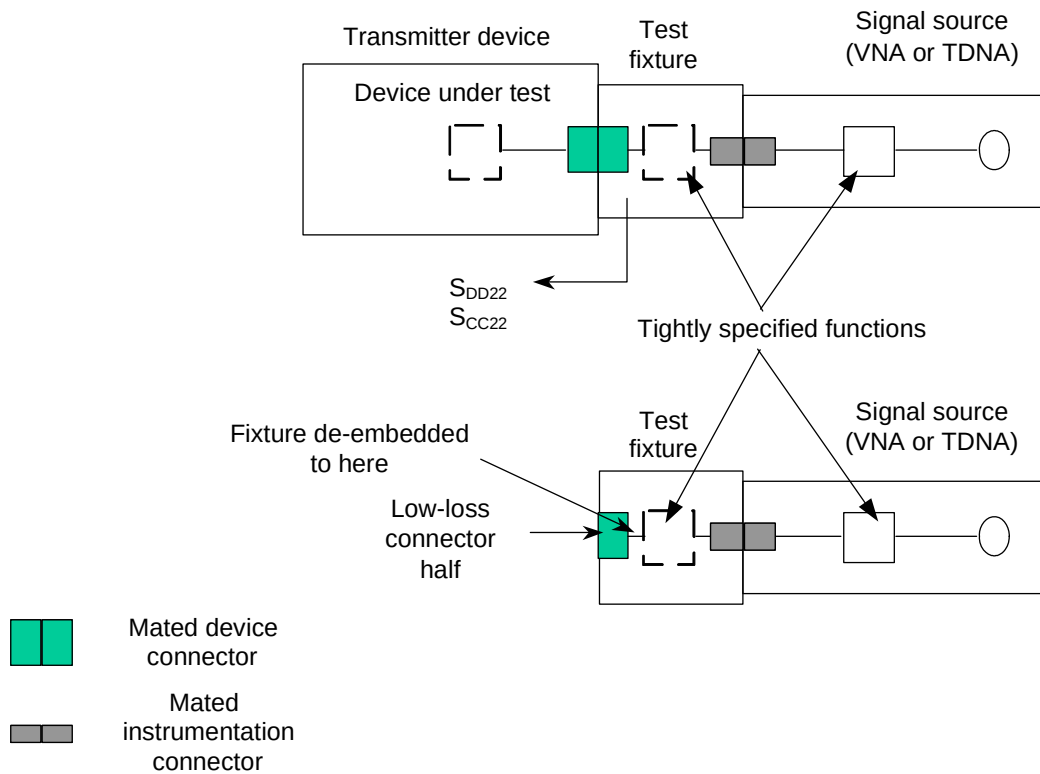


Figure B.15 — Measurement conditions for upstream return loss at the transmitter device connector

The test fixture in figure B.15 uses low loss connectors to avoid penalizing the transmitter device under test for the test fixture half of the connector. If the test fixture half of the device connector is poor then the transmitter device has to compensate for those losses.

The test fixture losses up to the mounting points for the device connector are de-embedded using the methods described in figure B.4.

B.9.4.3 Receiver device return loss

Figure B.16 shows the configuration to be used for the receiver device return loss measurement.

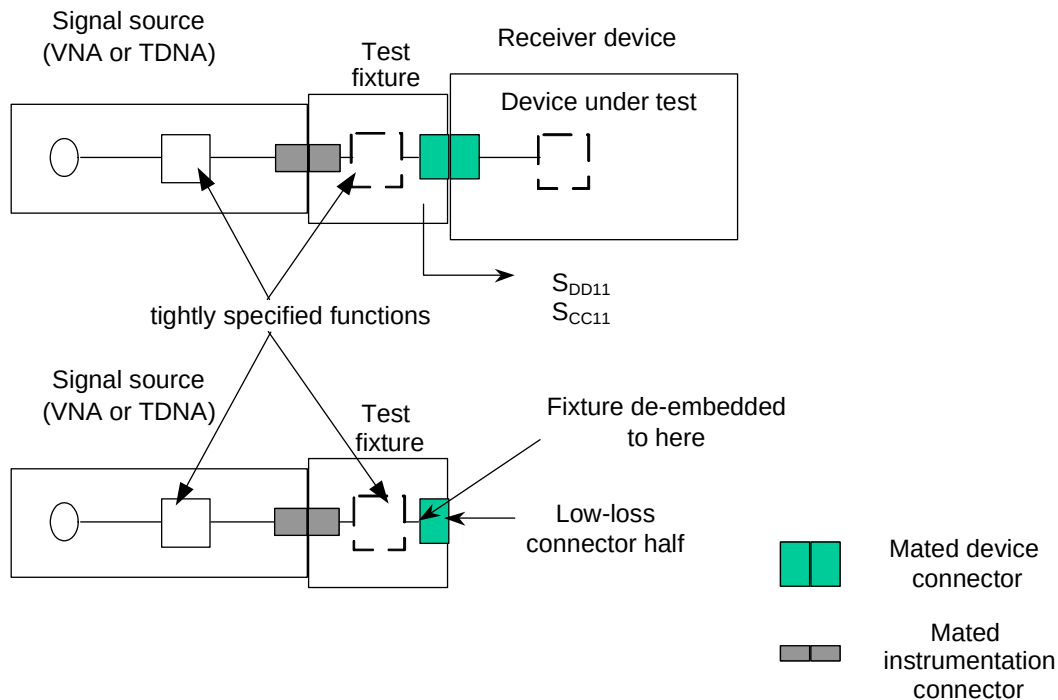


Figure B.16 — Measurement conditions for downstream return loss at the receiver device connector

The test fixture in figure B.16 uses low loss connectors to avoid penalizing the receiver device under test for the test fixture half of the connector. If the test fixture half of the device connector is poor then the receiver device has to compensate for those losses.

The test fixture losses up to the mounting points for the device connector are de-embedded using the methods described in figure B.4.

B.9.4.4 Return loss (S_{11}) at IT or CT

Figure B.17 shows the conditions for making the return loss measurement into the interconnect attached to the transmitter device.

This measurement, like the signal tolerance measurement at the transmitter device connector, requires both the interconnect and the receiver device to be in place and the combination is measured. If the receiver device is replaced by an ideal load then the return loss does not represent in service conditions. If the interconnect is very lossy then the effects of the load on the far end (i.e., where the receiver device is located) are not significant and an ideal load may be used. However, if the interconnect is not very lossy as in the zero length case, then the measured return loss may be dominated by the properties of the receiver device and not the properties of the interconnect.

For short physical links, this return loss performance may be the limiting factor for the entire physical link due to severe unattenuated reflections that create large DJ.

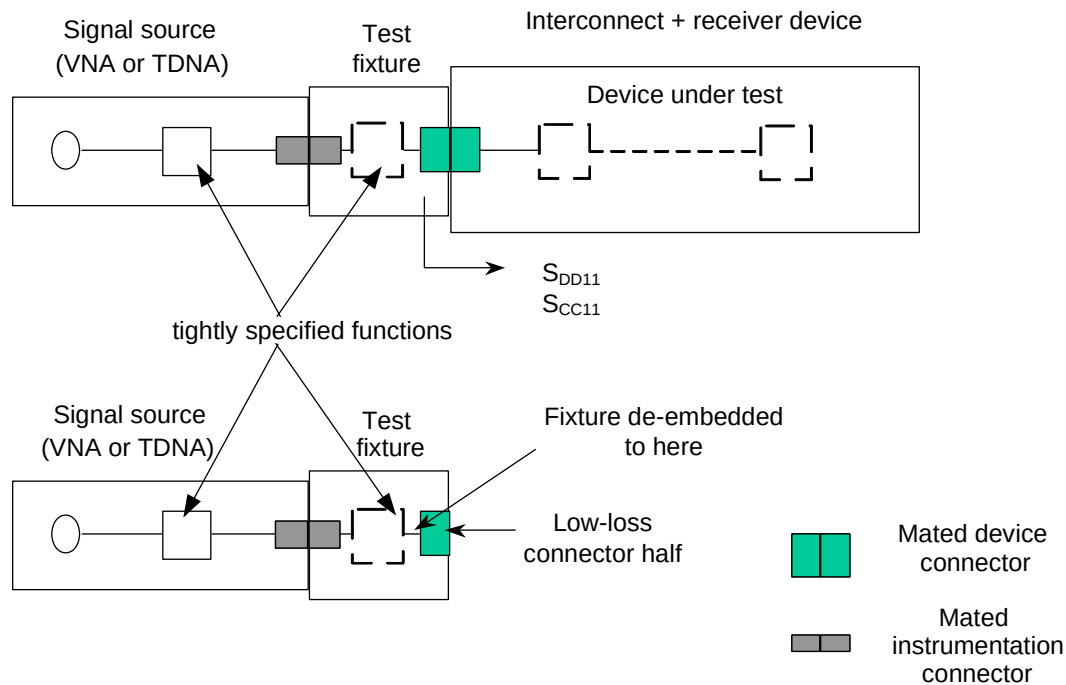


Figure B.17 — Measurement conditions for downstream return loss at IT or CT

B.9.4.5 Upstream return loss (S_{22}) at IR or CR

Figure B.18 shows the conditions for making the return loss measurement out of the interconnect attached to the receiver device.

This measurement is unique in that it requires both the interconnect and the transmitter device to be in place and the combination is measured. This may be considered a reverse direction signal tolerance measurement. If the transmitter device is replaced by an ideal load then the return loss does not represent in service conditions. If the interconnect is very lossy then the effects of the load on the far end (i.e., where the transmitter device is located) are not significant and an ideal load may be used. However, if the interconnect is not very lossy as in the zero length case, then the measured return loss may be dominated by the properties of the transmitter device and not the properties of the interconnect.

For short physical links, this return loss performance may be the limiting factor for the entire physical link due to severe unattenuated reflections that create large DJ.

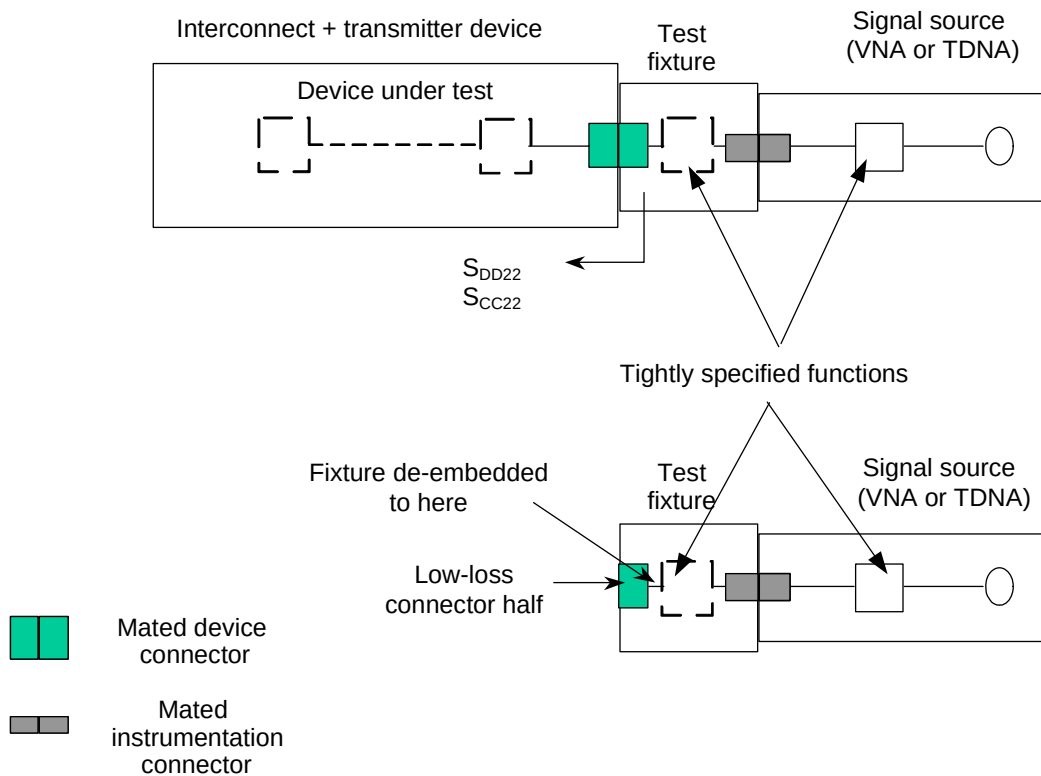


Figure B.18 — Measurement conditions for upstream return loss at IR or CR

B.9.5 Summary for S-parameter measurements

S-parameters are the preferred method of capturing the linear properties of physical link elements. Complex, but tractable, methods are required to use single-ended instruments for differential and common-mode applications. Careful attention to test configuration details is required.

A frequency domain spectrum output is required for all S-parameters and specifying pass/fail limits to such a spectrum may overconstrain the system because some peaks and properties are benign to the application.

Annex C

(informative)

SAS to SAS phy reset sequence examples

Editor's Note 67: With the new SNW-3 definition, some more examples may be needed.

Figure C.1 shows a speed negotiation between a phy A that supports only SNW-1 attached to a phy B that only supports SNW-1. Both phys run:

- 1) SNW-1, supported by both phys; and
- 2) SNW-2, supported by neither phy.

Both phys then select 1,5 Gbps for Final-SNW, which is used to establish the negotiated physical link rate.

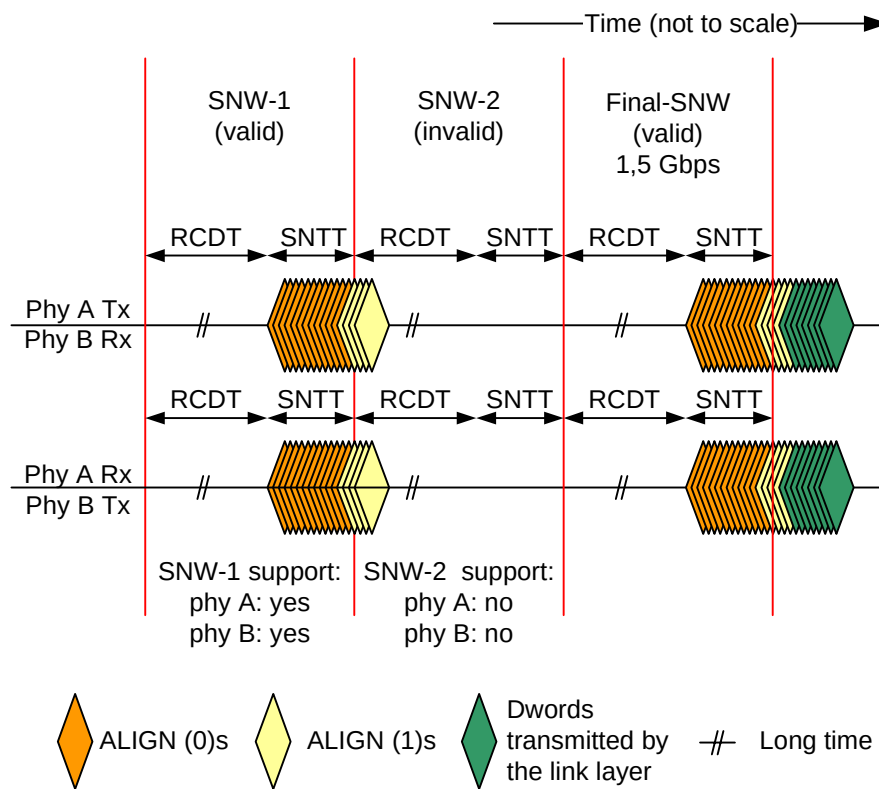


Figure C.1 — SAS speed negotiation sequence (phy A: SNW-1 only, phy B: SNW-1 only)

Figure C.2 shows a speed negotiation between a phy A that supports SNW-1 and SNW-2 attached to a phy B that supports SNW-1 and SNW-2. Both phys run:

- 1) SNW-1, supported by both phys;
- 2) SNW-2, supported by both phys; and
- 3) SNW-3, supported by neither phy.

Both phys then select 3 Gbps for Final-SNW, which is used to establish the negotiated physical link rate.

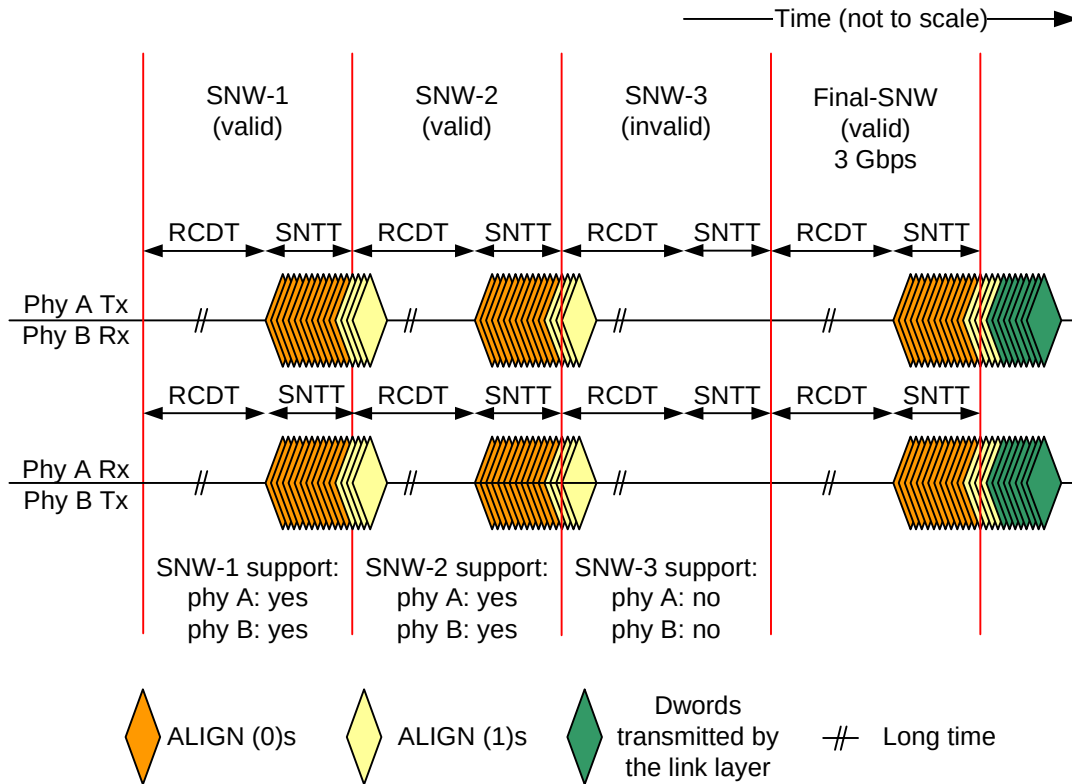


Figure C.2 — SAS speed negotiation sequence (phy A: SNW-1, SNW-2, phy B: SNW-1, SNW-2)

Figure C.3 shows a speed negotiation between a phy A that supports SNW-1 through SNW-3 attached to a phy B that only supports SNW-1 and SNW-2. Both phys run:

- 1) SNW-1, supported by both phys;
- 2) SNW-2, supported by both phys; and
- 3) SNW-3, supported by phy A but not by phy B.

Both phys then select 3 Gbps for Final-SNW, which is used to establish the negotiated physical link rate.

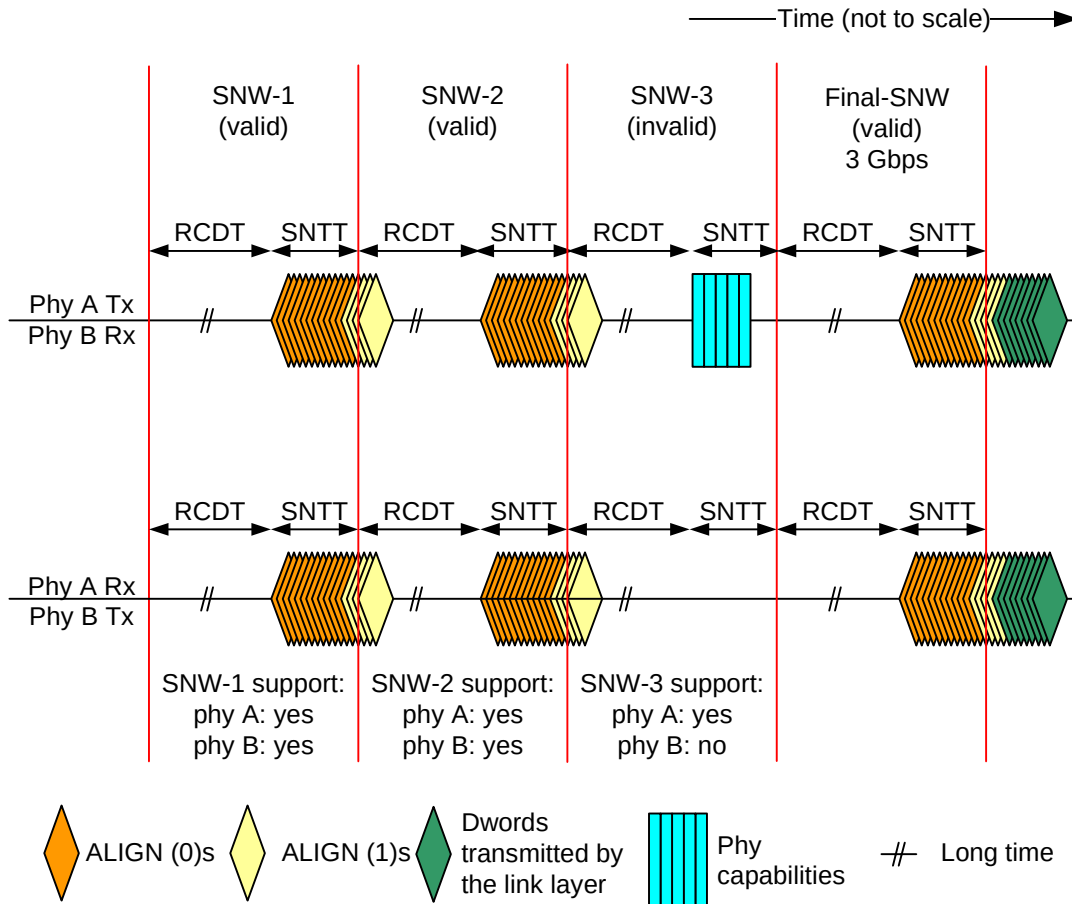


Figure C.3 — SAS speed negotiation sequence (phy A: SNW-1, SNW-2, SNW-3, phy B: SNW-1, SNW-2)

Figure C.4 shows a speed negotiation between a phy A that supports SNW-2 and SNW-3 attached to a phy B that only supports SNW-1 and SNW-2. Both phys run:

- 1) SNW-1, supported by phy B but not by phy A;
- 2) SNW-2, supported by both phys; and
- 3) SNW-3, supported by phy A but not by phy B.

Both phys then select 3 Gbps for Final-SNW, which is used to establish the negotiated physical link rate.

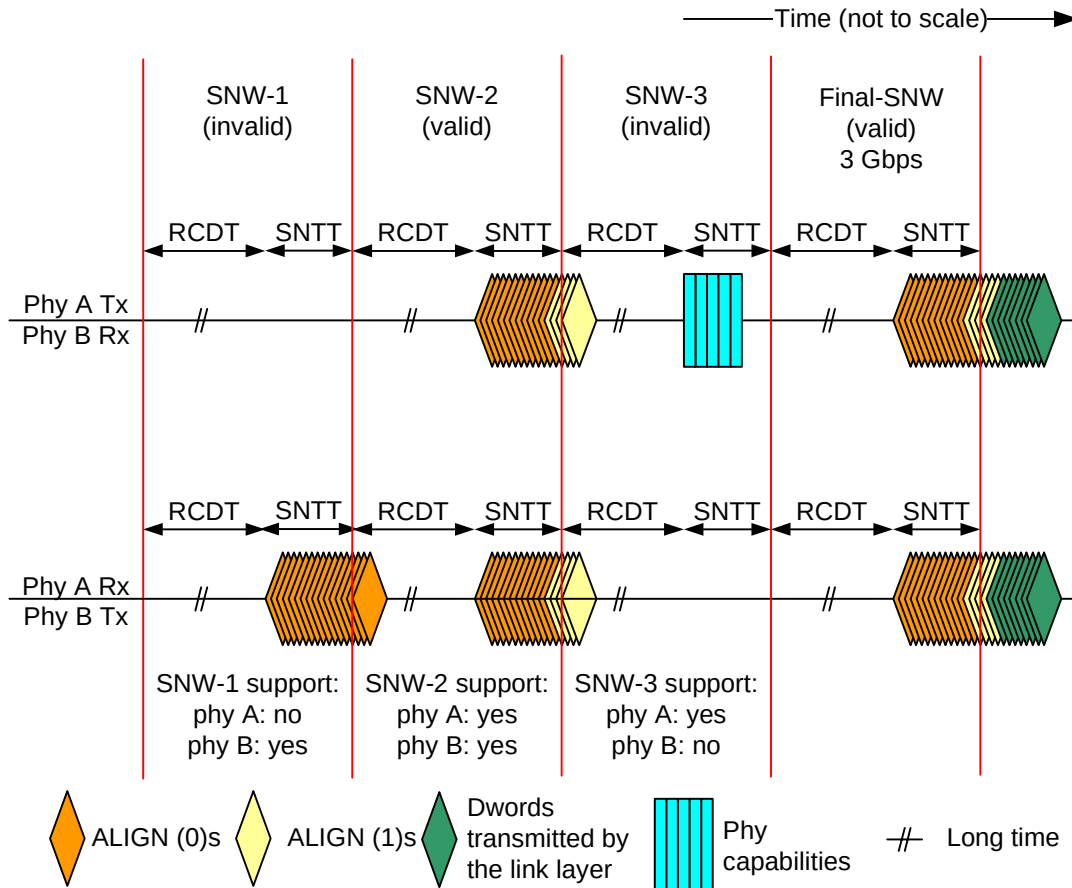


Figure C.4 — SAS speed negotiation sequence (phy A: SNW-2, SNW-3, phy B: SNW-1, SNW-2)

Figure C.5 shows a speed negotiation between a phy A that only supports SNW-1 attached to a phy B that only supports SNW-2. Both phys run:

- 1) SNW-1, supported by phy A but not by phy B; and
- 2) SNW-2, supported by phy B but not by phy A.

Phy B continues to run SNW-3, but phy A determines speed negotiation is unsuccessful and may attempt another phy reset sequence after a hot-plug timeout.

Phy B determines speed negotiation is not succeeding after SNW-3 and may retry the phy reset sequence after a hot-plug timeout.

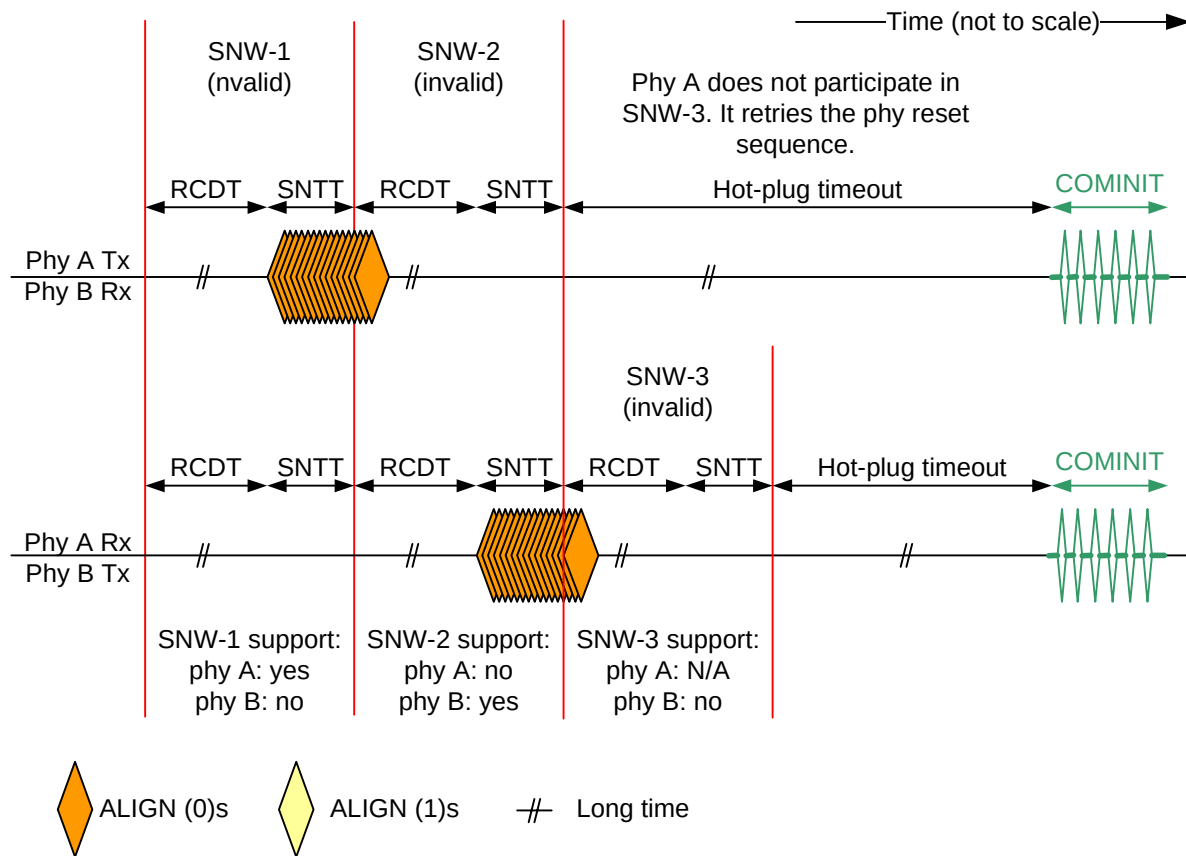


Figure C.5 — SAS speed negotiation sequence (phy A: SNW-1 only, phy B: SNW-2 only)

Annex D (informative)

CRC

D.1 CRC generator and checker implementation examples

Figure D.1 shows an example of a CRC generator.

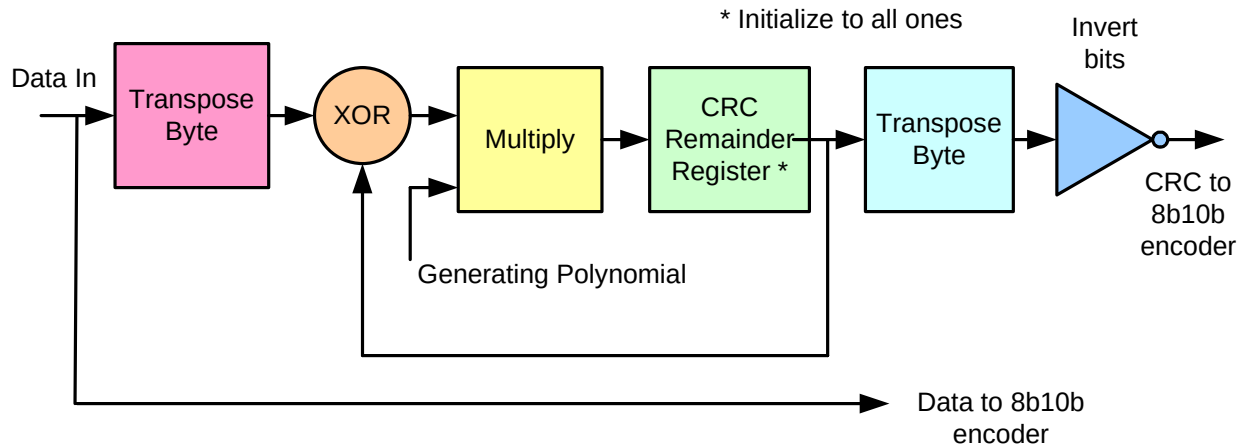


Figure D.1 — CRC generator example

Figure D.2 shows an example of a CRC checker.

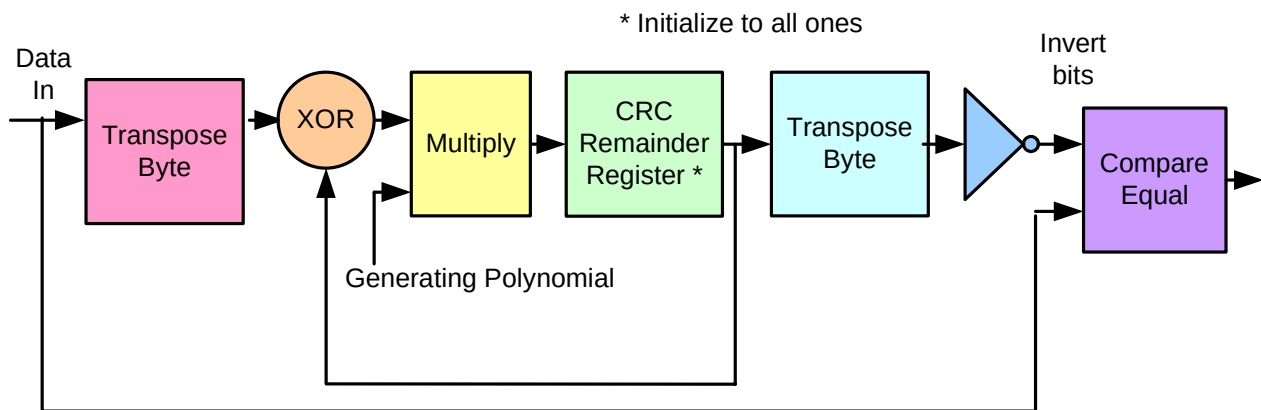


Figure D.2 — CRC checker example

D.2 CRC implementation in C

The following is an example C program that generates the value for the CRC field in frames. The inputs are the data dwords for the frame and the number of data dwords.

```
#include <stdio.h>

void main (void) {

    static unsigned long data_dwords[] = {
        0x06D0B992L, 0x00B5DF59L, 0x00000000L,
        0x00000000L, 0x1234FFFFL, 0x00000000L,
        0x00000000L, 0x00000000L, 0x00000000L,
        0x08000012L, 0x01000000L, 0x00000000L,
```

```

        0x00000000L}; /* example data dwords */

unsigned long calculate_crc(unsigned long *, unsigned long);
unsigned long crc;

crc = calculate_crc(data_dwords, 13);
printf ("Example CRC is %x\n", crc);
}

/* returns crc value */
unsigned long calculate_crc(unsigned long *frame, unsigned long length) {
long poly = 0x04C11DB7L;
unsigned long crc_gen, x;
union {
    unsigned long lword;
    unsigned char byte[4];
} b_access;
static unsigned char xpose[] = {
    0x0, 0x8, 0x4, 0xC, 0x2, 0xA, 0x6, 0xE,
    0x1, 0x9, 0x5, 0xD, 0x3, 0xB, 0x7, 0xF};
unsigned int i, j, fb;

crc_gen = ~0; /* seed generator with all ones */
for (i = 0; i < length; i++) {
    x = *frame++; /* get word */
    b_access.lword = x; /* transpose bits in byte */
    for (j = 0; j < 4; j++) {
        b_access.byte[j] = xpose[b_access.byte[j] >> 4] |
            xpose[b_access.byte[j] & 0xF] << 4;
    }
    x = b_access.lword;

    for (j = 0; j < 32; j++) { /* serial shift register implementation */
        fb = ((x & 0x80000000L) > 0) ^ ((crc_gen & 0x80000000L) > 0);
        x <<= 1;
        crc_gen <<= 1;
        if (fb)
            crc_gen ^= poly;
    }
}

b_access.lword = crc_gen; /* transpose bits in CRC */
for (j = 0; j < 4; j++) {
    b_access.byte[j] = xpose[b_access.byte[j] >> 4] |
        xpose[b_access.byte[j] & 0xF] << 4;
}
crc_gen = b_access.lword;

return ~crc_gen; /* invert output */
}

```

D.3 CRC implementation with XORs

These equations implement the multiply function shown in figure D.1 and figure D.2. The ^ symbol represents an XOR operation.

```

crc00 = d00^d06^d09^d10^d12^d16^d24^d25^d26^d28^d29^d30^d31;
crc01 = d00^d01^d06^d07^d09^d11^d12^d13^d16^d17^d24^d27^d28;

```

crc02 = d00^d01^d02^d06^d07^d08^d09^d13^d14^d16^d17^d18^d24^d26^d30^d31;
crc03 = d01^d02^d03^d07^d08^d09^d10^d14^d15^d17^d18^d19^d25^d27^d31;
crc04 = d00^d02^d03^d04^d06^d08^d11^d12^d15^d18^d19^d20^d24^d25^d29^d30^d31;
crc05 = d00^d01^d03^d04^d05^d06^d07^d10^d13^d19^d20^d21^d24^d28^d29;
crc06 = d01^d02^d04^d05^d06^d07^d08^d11^d14^d20^d21^d22^d25^d29^d30;
crc07 = d00^d02^d03^d05^d07^d08^d10^d15^d16^d21^d22^d23^d24^d25^d28^d29;
crc08 = d00^d01^d03^d04^d08^d10^d11^d12^d17^d22^d23^d28^d31;
crc09 = d01^d02^d04^d05^d09^d11^d12^d13^d18^d23^d24^d29;
crc10 = d00^d02^d03^d05^d09^d13^d14^d16^d19^d26^d28^d29^d31;
crc11 = d00^d01^d03^d04^d09^d12^d14^d15^d16^d17^d20^d24^d25^d26^d27^d28^d31;
crc12 = d00^d01^d02^d04^d05^d06^d09^d12^d13^d15^d17^d18^d21^d24^d27^d30^d31;
crc13 = d01^d02^d03^d05^d06^d07^d10^d13^d14^d16^d18^d19^d22^d25^d28^d31;
crc14 = d02^d03^d04^d06^d07^d08^d11^d14^d15^d17^d19^d20^d23^d26^d29;
crc15 = d03^d04^d05^d07^d08^d09^d12^d15^d16^d18^d20^d21^d24^d27^d30;
crc16 = d00^d04^d05^d08^d12^d13^d17^d19^d21^d22^d24^d26^d29^d30;
crc17 = d01^d05^d06^d09^d13^d14^d18^d20^d22^d23^d25^d27^d30^d31;
crc18 = d02^d06^d07^d10^d14^d15^d19^d21^d23^d24^d26^d28^d31;
crc19 = d03^d07^d08^d11^d15^d16^d20^d22^d24^d25^d27^d29;
crc20 = d04^d08^d09^d12^d16^d17^d21^d23^d25^d26^d28^d30;
crc21 = d05^d09^d10^d13^d17^d18^d22^d24^d26^d27^d29^d31;
crc22 = d00^d09^d11^d12^d14^d16^d18^d19^d23^d24^d26^d27^d29^d31;
crc23 = d00^d01^d06^d09^d13^d15^d16^d17^d19^d20^d26^d27^d29^d31;
crc24 = d01^d02^d07^d10^d14^d16^d17^d18^d20^d21^d27^d28^d30;
crc25 = d02^d03^d08^d11^d15^d17^d18^d19^d21^d22^d28^d29^d31;
crc26 = d00^d03^d04^d06^d10^d18^d19^d20^d22^d23^d24^d25^d26^d28^d31;
crc27 = d01^d04^d05^d07^d11^d19^d20^d21^d23^d24^d25^d26^d27^d29;
crc28 = d02^d05^d06^d08^d12^d20^d21^d22^d24^d25^d26^d27^d28^d30;
crc29 = d03^d06^d07^d09^d13^d21^d22^d23^d25^d26^d27^d28^d29^d31;
crc30 = d04^d07^d08^d10^d14^d22^d23^d24^d26^d27^d28^d29^d30;
crc31 = d05^d08^d09^d11^d15^d23^d24^d25^d27^d28^d29^d30^d31;

D.4 CRC examples

Table D.1 shows several CRC examples. Data is shown in dwords, from first to last.

Table D.1 — CRC examples

Frame contents	CRC	Frame contents	CRC
<SOF> 00010203h 04050607h 08090A0Bh 0C0D0E0Fh 10111213h 14151617h 18191A1Bh 1C1D1E1Fh <CRC> <EOF>	8A7E2691h	<SOF> 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000001h <CRC> <EOF>	3B650D6Eh
<SOF> 00000001h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h <CRC> <EOF>	898C0D7Ah	<SOF> 06D0B992h 00B5DF59h 00000000h 00000000h 1234FFFFh 00000000h 00000000h 00000000h 00000000h 00000000h 08000012h 01000000h 00000000h 00000000h <CRC> <EOF>	3F4F1C26h

Annex E (informative)

SAS address hashing

E.1 SAS address hashing overview

See 4.2.2 for a description of hashed SAS addresses and the algorithm used to create them.

E.2 Hash collision probability

The following are Monte-Carlo simulations evaluating the probability of collision in a system containing 128 addressable SAS ports. Four models were used for the models for the simulations:

- a) random model;
- b) sequential model;
- c) lots model; and
- d) three lots model.

The random model uses a system with 128 randomly chosen 64-bit integers as SAS addresses.

The sequential model uses a system with 128 sequentially-assigned SAS addresses starting from a random 64-bit base.

The lots model uses:

- a) Two sequentially assigned SAS addresses with unique company IDs and random vendor-specific identifiers;
- b) 125 randomly drawn SAS addresses from a 10 000-unit production lot. The vendor-specific identifiers within the lot were assigned by 10 SAS address-writers, randomly drawn from a pool of 4 096 possible SAS address-writers. Each SAS address-writer assigns vendor-specific identifiers sequentially within its own subset of the vendor-specific identifiers, starting from a randomly chosen base at the beginning of the production run; and
- c) One randomly chosen SAS address with another unique company ID, representing a replacement unit.

The three lots model uses:

- a) Two sequentially assigned SAS addresses with unique company IDs and random vendor-specific identifiers;
- b) 125 randomly drawn SAS addresses from three 10 000-unit lots. The vendor-specific identifiers within each lot were assigned by 10 SAS address-writers, randomly drawn from a pool of 4 096 possible SAS address-writers for that vendor. Each SAS address-writer assigns vendor-specific identifiers sequentially within its own subset of the vendor-specific identifiers, starting from a randomly chosen base at the beginning of the production run. Each of the three lots has a different company ID; and
- c) One randomly chosen SAS address with another unique company ID, representing a replacement unit.

Table E.1 lists the results of Monte-Carlo simulation.

Table E.1 — Monte-Carlo simulation results

SAS address model	Trials	Collisions	Average collisions per system
lots	2 000 000 000	45 063	0,000 022 531 5
three lots	2 000 000 000	662 503	0,000 331 251 5
random	10 000 000	4 882	0,000 488 2
sequential	10 000 000	0	0

E.3 Hash generation

One way to implement the hashing encoder in hardware is to use serial shift registers as shown in figure E.1. For error correction purposes, the number of data bits is limited to 39. For hashing purposes, the circuit shown serves as a divider. Because the period of this generator polynomial is 63, any binary sequence of length exceeding 63 is treated as a 63-bit sequence with $(\text{bit } 63) \times L + k$ added to $(\text{bit } k \text{ modulo } 2)$ for $k = 0, 1, \dots, 62$ and any integer L . Therefore, using this generator polynomial to hash a 64-bit address is equivalent to hashing a 63-bit sequence with bit 63 added modulo 2 to bit 0. With this wrapping, a binary sequence of any length is treated as an equivalent binary sequence of 63 bits, which, in turn, is treated as a degree-62 polynomial. After feeding this equivalent degree-62 polynomial into the circuit shown, the shift register contains the remainder from dividing the degree-62 input polynomial by the generator polynomial. This remainder is the hashed result.

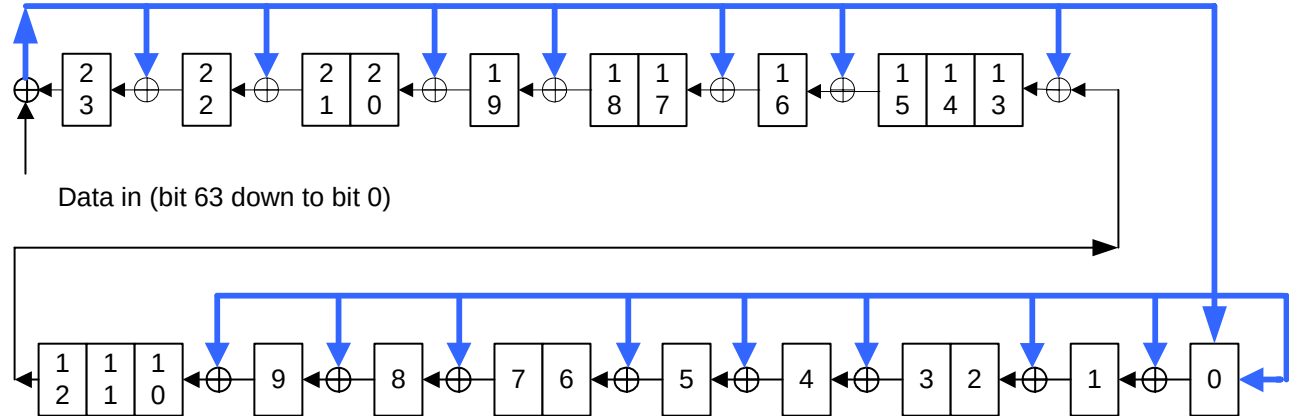


Figure E.1 — BCH(69, 39, 9) code generator

E.4 Hash implementation in C

The following is an example C program that generates a 24-bit hashed value from a 64-bit value.

```
typedef unsigned int uint32_t;
uint32_t hash(uint32_t upperbits, uint32_t lowerbits)
{
    const unsigned distance_9_poly = 0x01DB2777;
    uint32_t msb = 0x01000000;
    uint32_t moving_one, leading_bit;
    int i;
    unsigned regg;
    regg = 0;
    moving_one = 0x80000000;
    for (i = 31; i >= 0; i--) {
        leading_bit = 0;
        if (moving_one & upperbits) leading_bit = msb;
        regg <<= 1;
        regg ^= leading_bit;
        if (regg & msb) regg ^= distance_9_poly;
        moving_one >>= 1;
    }
    moving_one = 0x80000000;
    for (i = 31; i >= 0; i--) { // note lower limit of i = 0;
        leading_bit = 0;
        if (moving_one & lowerbits) leading_bit = msb;
        regg <<= 1;
        regg ^= leading_bit;
        if (regg & msb) regg ^= distance_9_poly;
    }
}
```

```

        moving_one >= 1;
    }
    return regg & 0x00FFFFFF;
}

```

E.5 Hash implementation with XORs

These equations generate the 24-bit hashed SAS address for the SSP frame header from a 64-bit SAS address. The ^ symbol represents an XOR.

```

hash00=d00^d01^d03^d05^d07^d09^d10^d11^d12^d15^d16^d17^d18^d19^d20^d21^d22^
d23^d24^d25^d28^d30^d31^d33^d34^d36^d38^d39^d63;
hash01=d00^d02^d03^d04^d05^d06^d07^d08^d09^d13^d15^d26^d28^d29^d30^d32^d33^
d35^d36^d37^d38^d40^d63;
hash02=d00^d04^d06^d08^d11^d12^d14^d15^d17^d18^d19^d20^d21^d22^d23^d24^d25^
d27^d28^d29^d37^d41^d63;
hash03=d01^d05^d07^d09^d12^d13^d15^d16^d18^d19^d20^d21^d22^d23^d24^d25^d26^
d28^d29^d30^d38^d42;
hash04=d00^d01^d02^d03^d05^d06^d07^d08^d09^d11^d12^d13^d14^d15^d18^d26^d27^
d28^d29^d33^d34^d36^d38^d43^d63;
hash05=d00^d02^d04^d05^d06^d08^d11^d13^d14^d17^d18^d20^d21^d22^d23^d24^d25^
d27^d29^d31^d33^d35^d36^d37^d38^d44^d63;
hash06=d00^d06^d10^d11^d14^d16^d17^d20^d26^d31^d32^d33^d37^d45^d63;
hash07=d01^d07^d11^d12^d15^d17^d18^d21^d27^d32^d33^d34^d38^d46;
hash08=d00^d01^d02^d03^d05^d07^d08^d09^d10^d11^d13^d15^d17^d20^d21^d23^d24^
d25^d30^d31^d35^d36^d38^d47^d63;
hash09=d00^d02^d04^d05^d06^d07^d08^d14^d15^d17^d19^d20^d23^d26^d28^d30^d32^
d33^d34^d37^d38^d48^d63;
hash10=d00^d06^d08^d10^d11^d12^d17^d19^d22^d23^d25^d27^d28^d29^d30^d35^d36^
d49^d63;
hash11=d01^d07^d09^d11^d12^d13^d18^d20^d23^d24^d26^d28^d29^d30^d31^d36^d37^
d50;
hash12=d02^d08^d10^d12^d13^d14^d19^d21^d24^d25^d27^d29^d30^d31^d32^d37^d38^
d51;
hash13=d00^d01^d05^d07^d10^d12^d13^d14^d16^d17^d18^d19^d21^d23^d24^d26^d32^
d34^d36^d52^d63;
hash14=d01^d02^d06^d08^d11^d13^d14^d15^d17^d18^d19^d20^d22^d24^d25^d27^d33^
d35^d37^d53;
hash15=d02^d03^d07^d09^d12^d14^d15^d16^d18^d19^d20^d21^d23^d25^d26^d28^d34^
d36^d38^d54;
hash16=d00^d01^d04^d05^d07^d08^d09^d11^d12^d13^d18^d23^d25^d26^d27^d28^d29^
d30^d31^d33^d34^d35^d36^d37^d38^d55^d63;
hash17=d00^d02^d03^d06^d07^d08^d11^d13^d14^d15^d16^d17^d18^d20^d21^d22^d23^
d25^d26^d27^d29^d32^d33^d35^d37^d56^d63;
hash18=d01^d03^d04^d07^d08^d09^d12^d14^d15^d16^d17^d18^d19^d21^d22^d23^d24^
d26^d27^d28^d30^d33^d34^d36^d38^d57;
hash19=d00^d01^d02^d03^d04^d07^d08^d11^d12^d13^d21^d27^d29^d30^d33^d35^d36^
d37^d38^d58^d63;
hash20=d00^d02^d04^d07^d08^d10^d11^d13^d14^d15^d16^d17^d18^d19^d20^d21^d23^
d24^d25^d33^d37^d59^d63;
hash21=d01^d03^d05^d08^d09^d11^d12^d14^d15^d16^d17^d18^d19^d20^d21^d22^d24^
d25^d26^d34^d38^d60;
hash22=d00^d01^d02^d03^d04^d05^d06^d07^d11^d13^d24^d26^d27^d28^d30^d31^d33^
d34^d35^d36^d38^d61^d63;
hash23=d00^d02^d04^d06^d08^d09^d10^d11^d14^d15^d16^d17^d18^d19^d20^d21^d22^
d23^d24^d27^d29^d30^d32^d33^d35^d37^d38^d62^d63;

```


E.6 Hash examples

Table E.2 shows examples using simple SAS addresses as input values. Two of the input values hash to the same value.

Table E.2 — Hash results for simple SAS addresses

64-bit input value	24-bit hashed value
00000000_00000000h	000000h
00000000_00000001h	DB2777h
FFFFFFFF_FFFFFFFFh	DB2777h

Table E.3 shows examples using realistic SAS addresses as input values.

Table E.3 — Hash results for realistic SAS addresses

64-bit input value	24-bit hashed value
50010753_4F0CFC88h	D0B992h
50010B92_B3CBF639h	B5DF59h
5002037E_157FEC63h	B064F7h
50004CF6_FBCE3889h	88FF12h
50020374_C4657EC7h	F36570h
50010D92_A016E450h	9F9571h
50002A58_850ACC66h	64B6B9h
50008C7B_EE7910DEh	8D6135h
500508BD_C22CAC94h	86ECF1h
500805F3_334B0AD3h	752AB2h
500A0B8A_FAA6A820h	5543A7h
500805E6_BCC55C68h	463DEDh

Table E.4 shows examples using a walking ones pattern to generate the input values.

Table E.4 — Hash results for a walking ones pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
0000000000000001h	DB2777h	0000000100000000h	8232C2h
0000000000000002h	6D6999h	0000000200000000h	DF42F3h
0000000000000004h	DAD332h	0000000400000000h	65A291h
0000000000000008h	6E8113h	0000000800000000h	CB4522h
0000000000000010h	DD0226h	0000001000000000h	4DAD33h
0000000000000020h	61233Bh	0000002000000000h	9B5A66h
0000000000000040h	C24676h	0000004000000000h	ED93BBh
0000000000000080h	5FAB9Bh	0000008000000000h	000001h
0000000000000100h	BF5736h	0000010000000000h	000002h
0000000000000200h	A5891Bh	0000020000000000h	000004h
0000000000000400h	903541h	0000040000000000h	000008h
0000000000000800h	FB4DF5h	0000080000000000h	000010h
0000000000001000h	2DBC9Dh	0000100000000000h	000020h
0000000000002000h	5B793Ah	0000200000000000h	000040h
0000000000004000h	B6F274h	0000400000000000h	000080h
0000000000008000h	B6C39Fh	0000800000000000h	000100h
0000000000010000h	B6A049h	0001000000000000h	000200h
0000000000020000h	B667E5h	0002000000000000h	000400h
0000000000040000h	B7E8BDh	0004000000000000h	000800h
0000000000080000h	B4F60Dh	0008000000000000h	001000h
0000000000100000h	B2CB6Dh	0010000000000000h	002000h
0000000000200000h	BEB1ADh	0020000000000000h	004000h
0000000000400000h	A6442Dh	0040000000000000h	008000h
0000000000800000h	97AF2Dh	0080000000000000h	010000h
0000000001000000h	F4792Dh	0100000000000000h	020000h
0000000002000000h	33D52Dh	0200000000000000h	040000h
0000000004000000h	67AA5Ah	0400000000000000h	080000h
0000000008000000h	CF54B4h	0800000000000000h	100000h
0000000010000000h	458E1Fh	1000000000000000h	200000h
0000000020000000h	8B1C3Eh	2000000000000000h	400000h
0000000040000000h	CD1F0Bh	4000000000000000h	800000h
0000000080000000h	411961h	8000000000000000h	DB2777h

Table E.5 shows examples using a walking zeros pattern to generate the input values.

Table E.5 — Hash results for a walking zeros pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
FFFFFFFFFFFFFFFFFEh	000000h	FFFFFFFFFFFFFFFFFh	5915B5h
FFFFFFFFFFFFFFFFFDh	B64EEh	FFFFFFFFDFFFFFFFFh	046584h
FFFFFFFFFFFFFFFFFBh	01F445h	FFFFFFFFBFFFFFFFFh	BE85E6h
FFFFFFFFFFFFFFFFF7h	B5A664h	FFFFFFF7FFFFFFFFh	106255h
FFFFFFFFFFFFFFFFEFh	062551h	FFFFFFFFEFFFFFFFFh	968A44h
FFFFFFFFFFFFFFFFDFh	BA044Ch	FFFFFFFFDFFFFFFFFh	407D11h
FFFFFFFFFFFFFFFFBFh	196101h	FFFFFFFFBFFFFFFFFh	36B4CCh
FFFFFFFFFFFFFFFFF7h	848CECh	FFFFFFF7FFFFFFFFh	DB2776h
FFFFFFFFFFFFFFFFEFh	647041h	FFFFFFFFEFFFFFFFFh	DB2775h
FFFFFFFFFFFFFFFFDFh	7EAE6Ch	FFFFFFFFDFFFFFFFFh	DB2773h
FFFFFFFFFFFFFFFFBFh	4B1236h	FFFFFFFFBFFFFFFFFh	DB277Fh
FFFFFFFFFFFFFFFFF7h	206A82h	FFFFFFF7FFFFFFFFh	DB2767h
FFFFFFFFFFFFFFFFEFh	F69BEAh	FFFFFFFFEFFFFFFFFh	DB2757h
FFFFFFFFFFFFFFFFDFh	805E4Dh	FFFFFFFFDFFFFFFFFh	DB2737h
FFFFFFFFFFFFFFFFBFh	6DD503h	FFFFFFFFBFFFFFFFFh	DB27F7h
FFFFFFFFFFFFFFFFF7h	6DE4E8h	FFFFF7FFFFFFFFh	DB2677h
FFFFFFFFFFFFFFFFEFh	6D873Eh	FFFFFFFFEFFFFFFFFh	DB2577h
FFFFFFFFFFFFFFFFDFh	6D4092h	FFFFFFFFDFFFFFFFFh	DB2377h
FFFFFFFFFFFFFFFFBFh	6CCFCAh	FFFBFFFFFFFFh	DB2F77h
FFFFFFFFFFFFFFFFF7h	6FD17Ah	FFF7FFFFFFFFh	DB3777h
FFFFFFFFFFFFFFFFEFh	69EC1Ah	FFEFFFFFFFFFh	DB0777h
FFFFFFFFFFFFFFFFDFh	6596DAh	FFDFFFFFFFFh	DB6777h
FFFFFFFFFFFFFFFFBFh	7D635Ah	FFBFFFFFFFFh	DBA777h
FFFFFFFFFFFFFFFFF7h	4C885Ah	FF7FFFFFFFFh	DA2777h
FFFFFFFFFFFFFFFFEFh	2F5E5Ah	FEFFFFFFFFh	D92777h
FFFFFFFFFFFFFFFFDFh	E8F25Ah	FDFFFFFFFFh	DF2777h
FFFFFFFFFFFFFFFFBFh	BC8D2Dh	FBFFFFFFFFh	D32777h
FFFFFFFFFFFFFFFFF7h	1473C3h	F7FFFFFFFFh	CB2777h
FFFFFFFFFFFFFFFFEFh	9EA968h	EFFFFFFFFFh	FB2777h
FFFFFFFFFFFFFFFFDFh	503B49h	DFFFFFFFFFh	9B2777h
FFFFFFFFFFFFFFFFBFh	16387Ch	BFFFFFFFFh	5B2777h
FFFFFFFFFFFFFFFFF7h	9A3E16h	7FFFFFFFFh	000000h

Annex F (informative)

Scrambling

F.1 Scrambler implementation example

Figure F.1 shows an example of a data scrambler. This example generates the value to XOR with the dword input with two 16 bit parallel multipliers. 16 bits wide is the maximum width for the multiplier as the generating polynomial is 16 bits.

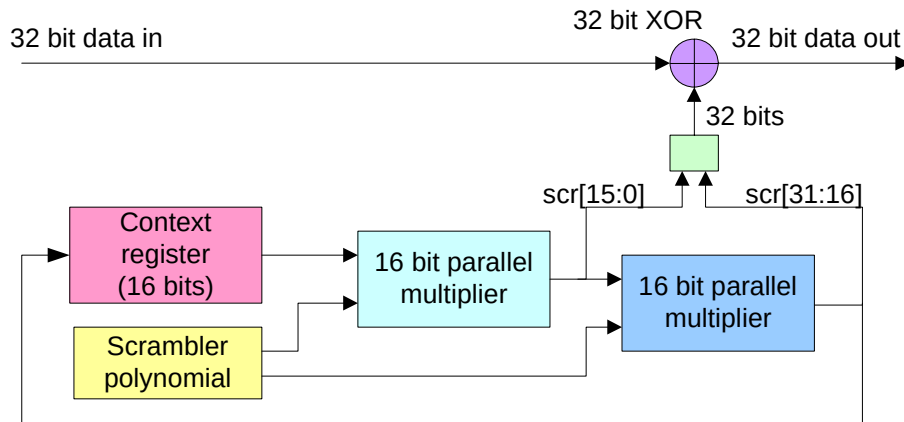


Figure F.1 — Scrambler

The generator polynomial is:

$$G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$$

For all implementations, the context register is initialized to produce a first dword output of C2D2768Dh for a dword input of all zeros.

F.2 Scrambler implementation in C

The following is an example C program that generates the scrambled data dwords for transmission. The inputs are the data dword to scramble and control indication to reinitialize the residual value (e.g., following an SOF).

```

#include <stdio.h>

unsigned long scramble(int reset, unsigned long dword);
void main(void)
{
    int i;

    for (i = 0; i < 12; i++)
        printf(" %08X \n", scramble(i==0, 0)); /* scramble all 0s */
}

#define poly 0xA011
unsigned long scramble(int reset, unsigned long dword) {
    static unsigned short scramble;
    int i;

    if (reset)
        scramble = 0xFFFF;

```

```

    for (i = 0; i < 32; i++) /* serial shift register implementation */
    {
        dword ^= (scramble & 0x8000)? (1 << i):0;
        scramble = (scramble << 1) ^ ((scramble & 0x8000)? poly:0);
    }
    return dword;
}

```

F.3 Scrambler implementation with XORs

These equations generate the scrambled dwords to XOR with dwords before transmission and dword reception to recover the original data. The ^ symbol represents an XOR operation. The initialized value for d[15:0] is F0F6h (i.e., 0xF0F6) in this example.

```

scr0=d15^d13^d4^d0;
scr1=d15^d14^d13^d5^d4^d1^d0;
scr2=d14^d13^d6^d5^d4^d2^d1^d0;
scr3=d15^d14^d7^d6^d5^d3^d2^d1;
scr4=d13^d8^d7^d6^d3^d2^d0;
scr5=d14^d9^d8^d7^d4^d3^d1;
scr6=d15^d10^d9^d8^d5^d4^d2;
scr7=d15^d13^d11^d10^d9^d6^d5^d4^d3^d0;
scr8=d15^d14^d13^d12^d11^d10^d7^d6^d5^d1^d0;
scr9=d14^d12^d11^d8^d7^d6^d4^d2^d1^d0;
scr10=d15^d13^d12^d9^d8^d7^d5^d3^d2^d1;
scr11=d15^d14^d10^d9^d8^d6^d3^d2^d0;
scr12=d13^d11^d10^d9^d7^d3^d1^d0;
scr13=d14^d12^d11^d10^d8^d4^d2^d1;
scr14=d15^d13^d12^d11^d9^d5^d3^d2;
scr15=d15^d14^d12^d10^d6^d3^d0;
scr16=d11^d7^d1^d0;
scr17=d12^d8^d2^d1;
scr18=d13^d9^d3^d2;
scr19=d14^d10^d4^d3;
scr20=d15^d11^d5^d4;
scr21=d15^d13^d12^d6^d5^d4^d0;
scr22=d15^d14^d7^d6^d5^d4^d1^d0;
scr23=d13^d8^d7^d6^d5^d4^d2^d1^d0;
scr24=d14^d9^d8^d7^d6^d5^d3^d2^d1;
scr25=d15^d10^d9^d8^d7^d6^d4^d3^d2;
scr26=d15^d13^d11^d10^d9^d8^d7^d5^d3^d0;
scr27=d15^d14^d13^d12^d11^d10^d9^d8^d6^d1^d0;
scr28=d14^d12^d11^d10^d9^d7^d4^d2^d1^d0;
scr29=d15^d13^d12^d11^d10^d8^d5^d3^d2^d1;
scr30=d15^d14^d12^d11^d9^d6^d3^d2^d0;
scr31=d12^d10^d7^d3^d1^d0;

```

F.4 Scrambler examples

Table F.1 shows several scrambler examples. Data is shown in dwords, from first to last.

Table F.1 — Scrambler examples

Frame contents	Scrambled output
<SOF> 06D0B992h 00B5DF59h 00000000h 00000000h 1234FFFFh 00000000h 00000000h 00000000h 00000000h 08000012h 01000000h 00000000h 00000000h 3F4F1C26h ^a <EOF>	<SOF> C402CF1Fh 1F936C31h A508436Ch 3452D354h 98616AFDh BB1ABE1Bh FA56B73Dh 53F60B1Bh F0809C41h 7C7FC358h BF865291h 7A6FA7B6h 3163E6D6h CF79E22Ah ^a <EOF>
<SOF> 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h B00F2BCCh ^a <EOF>	<SOF> C2D2768Dh 1F26B368h A508436Ch 3452D354h 8A559502h BB1ABE1Bh FA56B73Dh 53F60B1Bh F0809C41h 747FC34Ah BE865291h 7A6FA7B6h 3163E6D6h 4039D5C0h ^a <EOF>
^a The last dword represents a CRC dword.	

Annex G

(informative)

ATA architectural notes

G.1 STP differences from Serial ATA (SATA)

Some of the differences of STP compared with SATA are:

- a) STP adds addressing of multiple SATA devices. Each SATA device is assigned a SAS address by its attached expander device with STP/SATA bridge functionality. The STP initiator port understands addressing more than one STP target port;
- b) STP allows multiple STP initiator ports to share access to a SATA device behind an STP/SATA bridge using affiliations (see 7.17.4);
- c) interface power management is not supported;
- d) far-end analog loopback testing is not supported;
- e) far-end retimed loopback testing is not supported;
- f) near-end analog loopback testing is not supported;
- g) use of SATA_CONT is required; and
- h) BIST Activate frames are not supported.

G.2 STP differences from Serial ATA II

The following features of Serial ATA II are specifically excluded from SAS STP or handled differently in a SAS domain:

- a) extended differential voltages;
- b) enclosure services;
- c) staggered spin-up (see 6.11);
- d) device activity indication;
- e) presence detect; and
- f) power management improvements.

G.3 Affiliation policies

G.3.1 Affiliation policies overview

SATA is based on a model that assumes a SATA device is controlled by a single SATA host, and does not address the notion of multiple SATA hosts having the ability to access any given SATA device.

With STP/SATA bridges, SATA devices are cast into an environment where multiple STP initiator ports, by sharing the SATA host port of the STP/SATA bridge, have access to the same SATA device. The SATA protocol used inside STP connections does not account for the possibility that more than one STP initiator port might be vying for access to the SATA device. Affiliations provide a way to ensure contention for a SATA device does not result in incoherent access to the SATA device when commands from different STP initiator ports collide at the SATA device.

To prevent a SATA device from confusing commands from one STP initiator port with commands from another, an STP initiator port needs a means to maintain exclusive access to a SATA or STP device for the duration of the processing of a command.

For example, consider the case where an STP initiator port establishes a connection to send a command (e.g., a read), and then closes the connection while the SATA device (e.g., a disk drive) retrieves the data (e.g., performs a seek operation to the track containing the data). If, after the connection is closed, another STP initiator port is allowed to establish a connection and send another command, the SATA device would no longer have a means to determine which STP initiator port should receive the data when the device requests the connection to send the data for the first command. This is because, unlike SCSI target devices, SATA devices have no notion of multiple SATA hosts.

The consequences are worse for write commands since the result could be wrong data written to media, with the original data being overwritten and permanently lost.

Affiliations provide a means for an STP initiator port to establish atomic access to a SATA device across the processing of a command or series of commands to the SATA device, without requiring the STP initiator port to maintain a connection open to the STP target port for the duration of command processing.

G.3.2 Affiliation policy for static STP initiator port to STP target port mapping

Affiliations should not be used to enforce policies establishing fixed associations between STP initiator ports and STP target ports.

G.3.3 Affiliation policy with SATA queued commands and multiple STP initiator ports

When sharing an affiliation context, STP initiator ports using queued commands when other STP initiator ports may be accessing the same STP target port should, at vendor-specific intervals, allow commands to complete and release the affiliation to allow other STP initiator ports access to the STP target port.

G.3.4 Applicability of affiliation for STP target ports

Affiliation may or may not be necessary for STP target ports depending on whether the STP target port tracks the STP initiator port's SAS address on each command received. If the STP target port has the means to manage and track commands from each STP initiator port independently, then affiliations are not necessary because the STP target port is capable of associating each information transfer with the appropriate STP initiator port, and is capable of establishing a connection to the appropriate STP initiator port when sending information back for a command.

An STP target port capable of tracking commands may support a limited number of STP initiator ports (i.e., more than one, but less than one per command) and use multiple affiliations in order to manage that restriction.

An STP target port that behaves the same as a SATA device, in that it maintains only a single ATA task file register context to be shared among all STP initiator ports, supports a single affiliation in order to provide a way for STP initiator ports to maintain exclusive access to the STP target port while commands remain outstanding. In this model, an STP target port is capable of establishing connections to an STP initiator port, but is only capable of remembering the SAS address of the last STP initiator port to establish a connection, and therefore is only capable of requesting a connection back to that same STP initiator port.

See 10.4.3.11 for an explanation of how an STP target port reports support for affiliations.

G.4 SATA port selector considerations

Not all the protocol elements for STP initiator ports to manage a SATA port selector (see SATA-2) in a SAS domain are defined in this standard. Additional coordination between STP initiator ports may be needed to avoid conflicting usage of the SATA port selector between STP initiator ports (e.g., between two SAS domains). Such additional coordination is outside the scope of this standard.

G.5 SATA device not transmitting initial Register Device-to-Host FIS

Some SATA devices do not return the initial Register Device-to-Host FIS after a link reset sequence if they did not detect the COMINIT during the link reset sequence (e.g., if the SATA device originated the link reset sequence). While waiting for the initial Register Device-to-Host FIS, an STP/SATA bridge responds as follows:

- a) In the SMP DISCOVER response (see 10.4.3.9):
 - A) the ATTACHED DEVICE TYPE field is set to 000b;
 - B) the NEGOTIATED LOGICAL LINK RATE field and the NEGOTIATED PHYSICAL LINK RATE field are set to a value indicating the phy is enabled at a valid link rate (e.g., G1 (i.e., 8h), G2 (i.e., 9h), or G3 (i.e., Ah));
 - C) the ATTACHED SATA DEVICE bit is set to one; and
 - D) the ATTACHED SAS ADDRESS field is set to the SAS address of the STP target port of the STP/SATA bridge;

and

- b) returns OPEN_REJECT (NO DESTINATION) for connection requests to the SAS address of the STP target port.

If an STP initiator port detects this situation for a vendor-specific amount of time, an SMP application client should send an SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET to originate a new link reset sequence. The SATA device is expected to detect the COMINIT during this link reset sequence and provide the initial Register Device-to-Host FIS.

Annex H

(informative)

Minimum deletable primitive insertion rate summary

Table H.1 shows all the possible combinations of deletable primitive insertion rates for physical link rate tolerance management (see 7.3) and rate matching (see 7.13).

Table H.1 — Minimum deletable primitive insertion rate examples

Physical link rate	Connection rate	Deletable primitive insertion rate (per specified number of dwords)
6 Gbps	6 Gbps	4 per 512 (physical link rate tolerance management)
	3 Gbps	4 per 512 (physical link rate tolerance management) + 1 per 2 (rate matching)
	1,5 Gbps	4 per 512 (physical link rate tolerance management) + 3 per 4 (rate matching)
3 Gbps	3 Gbps	2 per 256 (clock skew management)
	1,5 Gbps	2 per 256 (clock skew management) + 1 per 2 (rate matching)
1,5 Gbps	1,5 Gbps	1 per 128 (physical link rate tolerance management)

Annex I

(informative)

Zone permission configuration descriptor examples

This annex provides examples of using multiple zone permission configuration descriptors in the SMP CONFIGURE ZONE PERMISSION TABLE function (see 10.4.3.25) if the number of zone groups is 128.

Table I.1 shows an example initial value of the zone permission table.

Table I.1 — Zone permission table example initial value

Zone group	0 ^a	1 ^a	2 to 3	4 to 7 ^a	8	9	10	11	12 to 127
0 ^a	0	1	0	0	0	0	0	0	0
1 ^a	1	1	1	1	1	1	1	1	1
2 to 3	0	1	0	0	0	0	0	0	0
4 to 7 ^a	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	0	0	0
9	0	1	0	0	0	0	0	0	0
10	0	1	0	0	0	0	0	0	0
11	0	1	0	0	0	0	0	0	0
12 to 127	0	1	0	0	0	0	0	0	0
^a Zone permission table entries for this zone group are not changeable.									

Table I.2 shows an example SMP CONFIGURE ZONE PERMISSION TABLE request where the STARTING ZONE GROUP field is set to 10 (i.e., 0Ah) and the zone permission configuration descriptor list contains two zone permission configuration descriptors.

Table I.2 — CONFIGURE ZONE PERMISSION TABLE request example

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (8Bh)							
2	Reserved							
3	REQUEST LENGTH (0Bh)							
4	(MSB)	EXPECTED EXPANDER CHANGE COUNT						
5							(LSB)	
6	STARTING SOURCE ZONE GROUP (0Ah)							
7	NUMBER OF ZONE PERMISSION CONFIGURATION DESCRIPTORS (02h)							
Zone permission configuration descriptor (first)(source zone group 10)								
8	FFh							
...	(each byte set to FFh)							
23	0Eh							
Zone permission configuration descriptor (second)(source zone group 11)								
24	00h							
...	(each byte set to 00h)							
47	00h							
48	(MSB)	CRC						
51							(LSB)	

Table I.3 shows the zone permission table after processing the first zone permission configuration descriptor (i.e., source zone group 10).

Table I.3 — Zone permission table after processing the first zone permission configuration descriptor

Zone group	0 ^a	1 ^a	2 to 3	4 to 7 ^a	8	9	10 ^b	11	12 to 127
0 ^a	0	1	0	0	0	0	0	0	0
1 ^a	1	1	1	1	1	1	1	1	1
2 to 3	0	1	0	0	0	0	1	0	0
4 to 7 ^a	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	1	0	0
9	0	1	0	0	0	0	1	0	0
10 ^b	0	1	1	0	1	1	1	1	1
11	0	1	0	0	0	0	1	0	0
12 to 127	0	1	0	0	0	0	1	0	0
^a Zone permission table entries for this zone group are not changeable.									
^b Changeable entries in this zone group are changed by the descriptor.									

Table I.4 shows the zone permission table after processing the second zone permission configuration descriptor (i.e., source zone group 11).

Table I.4 — Zone permission table after processing the second zone permission configuration descriptor

Zone group	0 ^a	1 ^a	2 to 3	4 to 7 ^a	8	9	10	11 ^b	12 to 127
0 ^a	0	1	0	0	0	0	0	0	0
1 ^a	1	1	1	1	1	1	1	1	1
2 to 3	0	1	0	0	0	0	1	0	0
4 to 7 ^a	0	1	0	0	0	0	0	0	0
8	0	1	0	0	0	0	1	0	0
9	0	1	0	0	0	0	1	0	0
10	0	1	1	0	1	1	1	0	1
11 ^b	0	1	0	0	0	0	0	0	0
12 to 127	0	1	0	0	0	0	1	0	0
^a Zone permission table entries for this zone group are not changeable.									
^b Changeable entries in this zone group are changed by the descriptor.									

Annex J

(informative)

Expander device handling of connections

J.1 Expander device handling of connections overview

This annex provides examples of how expander devices process connection requests.

Figure J.1 shows the topology used by examples in this annex.

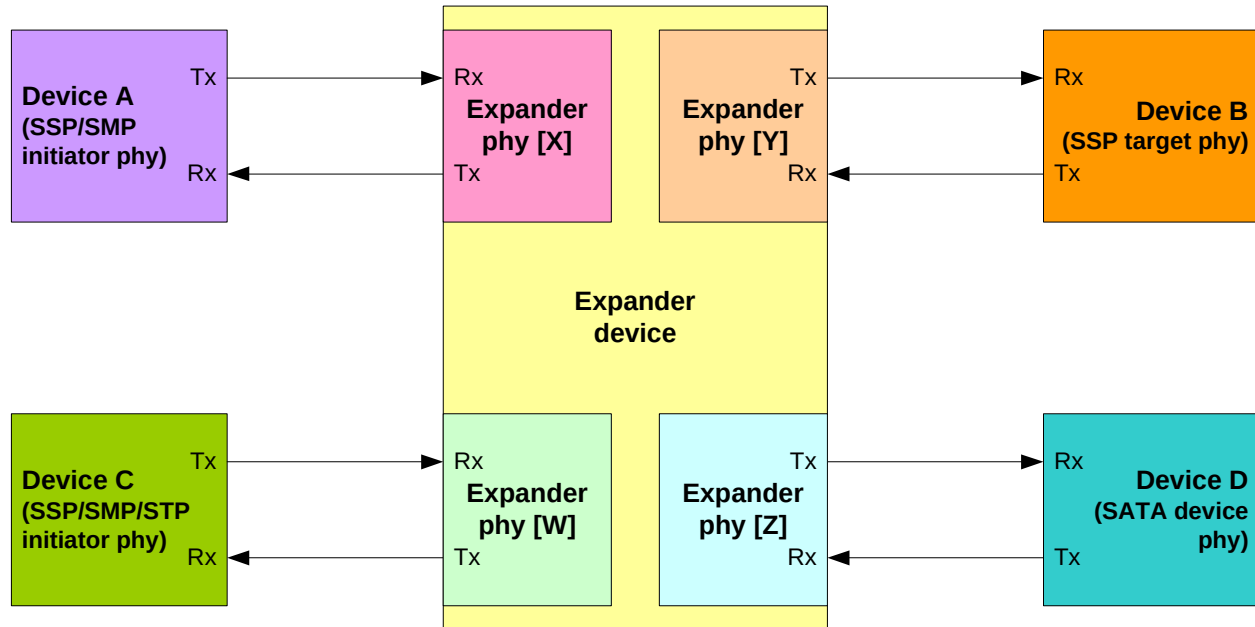


Figure J.1 — Example topology

Table J.1 defines the column headers used within the figures contained within this annex.

Table J.1 — Column descriptions for connection examples

Column header	Description
Phy [W] Rx	Expander phy [W] Receive from device C
Phy [W] Tx	Expander phy [W] Transmit to device C
Phy [W] XL state	Expander phy [W] XL state machine state (see 7.15)
Phy [W] XL req/rsp	Expander phy [W] XL requests and responses (see 4.6.6)
Phy [W] XL cnf/ind	Expander phy [W] XL confirmations and indications (see 4.6.6)
Phy [X] Rx	Expander phy [X] Receive from device A
Phy [X] Tx	Expander phy [X] Transmit to device A
Phy [X] XL state	Expander phy [X] XL state machine state (see 7.15)
Phy [X] XL req/rsp	Expander phy [X] XL requests and responses (see 4.6.6)
Phy [X] XL cnf/ind	Expander phy [X] XL confirmations and indications (see 4.6.6)
Phy [Y] XL cnf/ind	Expander phy [Y] XL confirmations and indications (see 4.6.6)
Phy [Y] XL req/rsp	Expander phy [Y] XL requests and responses (see 4.6.6)
Phy [Y] XL state	Expander phy [Y] XL state machine state (see 7.15)
Phy [Y] Tx	Expander phy [Y] Transmit to device B
Phy [Y] Rx	Expander phy [Y] Receive from device B
Phy [Z] XL cnf/ind	Expander phy [Y] XL confirmations and indications (see 4.6.6)
Phy [Z] XL req/rsp	Expander phy [Y] XL requests and responses (see 4.6.6)
Phy [Z] XL state	Expander phy [Y] XL state machine state (see 7.15)
Phy [Z] Tx	Expander phy [Y] Transmit to device D
Phy [Z] Rx	Expander phy [Y] Receive from device D

J.2 Connection request - OPEN_ACCEPT

Figure J.2 shows the establishment of a successful connection between two end devices.

Expander phy [X]					Expander phy [Y]									
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx					
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords					
SOAF														
OPEN (A to B)														
EOAF														
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)										
				Arb Won										
		XL2: Request_Open	Forward Open			Forward Open		XL5: Forward_Open	SOAF					
									OPEN (A to B)					
									EOAF					
		XL3: Open_Cnf_Wait	Forward Dword (idle dwords)			Forward Dword (idle dwords)		XL6: Open_Rsp_Wait	idle dwords (forwarded or generated)					
											Arb Status (Waiting On Device)		Arb Status (Waiting On Device)	
AIP (WAITING ON DEVICE)				Arb Status (Waiting On Device)		Arb Status (Waiting On Device)		idle dwords (forwarded or generated)	OPEN_ACCEPT					
idle dwords														
OPEN_ACCEPT														
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)		Forward Dword (connection dwords)		XL7:Connected	idle dwords (forwarded)	connection dwords					

J.3 Connection request - OPEN_REJECT by end device

Figure J.3 shows failure to establish a connection due to rejection of the connection request by an end device.

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN (A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
				Arb Won					
		XL2: Request_Open	Forward Open		Forward Open				
		XL3: Open_Cnf_Wait	Forward Dword (idle dwords)		Forward Dword (idle dwords)				
	AIP (WAITING ON DEVICE)			Arb Status (Waiting On Device)		Arb Status (Waiting On Device)			
	idle dwords								
	OPEN_REJECT								
	idle dwords								
		XL0:Idle							
		XL0:Idle							

Figure J.3 — Connection request - OPEN_REJECT by end device

J.4 Connection request - OPEN_REJECT by expander device

Figure J.4 shows failure to establish a connection due to rejection of the connection request by an expander device.

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN (A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
	Arb Reject								
	OPEN_REJECT	XL4:Open_Reject							
idle dwords	XL0:Idle								

Figure J.4 — Connection request - OPEN_REJECT by expander device

J.5 Connection request - arbitration lost

Figure J.5 shows two end devices attempting to establish a connection at the same time. This example assumes that the OPEN (A to B) address frame has higher priority than the OPEN (B to A) address frame and therefore device A wins arbitration and device B loses arbitration.

Expander phy [X]					Expander phy [Y]						
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx		
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords		
SOAF									SOAF		
OPEN (A to B)									OPEN (B to A)		
EOAF									EOAF		
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)	Arbitrating (Normal)	Request Path	XL1: Request_Path	AIP (NORMAL) and/or idle dwords	idle dwords		
				Arb Won	Arb Lost						
		XL2: Request_Open	Forward Open		Forward Open		XL0:Idle	idle dwords			
					XL3: Open_Cnf_Wait		Forward Dword (idle dwords)			XL5: Forward_Open	SOAF
										OPEN (A to B)	
			EOAF								
	AIP (WAITING ON DEVICE)				Arb Status (Waiting On Device)	Arb Status (Waiting On Device)	XL6: Open_Rsp_Wait	idle dwords (forwarded or generated)			
	idle dwords			Open Accept		XL7:Connected	idle dwords (forwarded)	connection dwords			
OPEN_ACCEPT	Forward Dword (connection dwords)			Forward Dword (connection dwords)							
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)		Forward Dword (connection dwords)			connection dwords			

Figure J.5 — Connection request - arbitration lost

J.6 Connection request - backoff and retry

Figure J.6 shows a higher priority OPEN address frame (B to C) received by a phy which has previously forwarded an OPEN address frame (A to B) whose source (A) differs from the winning destination (C). In this case expander phy [X] is required to back off and retry path arbitration (see 7.15.9).

Expander phy [X]					Expander phy [Y]										
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx						
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords						
SOAF															
OPEN (A to B)															
EOAF															
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)											
				Arb Won											
		XL2: Request_Open	Forward Open		Forward Open		XL5: Forward_Open	SOAF							
					XL3: Open_Cnf_Wait			Forward Dword (idle dwords)		Forward Dword (idle dwords)	Arb Status (Waiting On Device)	OPEN (A to B)			
												EOAF			
	AIP (WAITING ON DEVICE)					XL6: Open_Rsp_Wait	idle dwords (forwarded or generated)								
									idle dwords						SOAF
	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path		Arbitrating (Normal)			Request Path	XL1: Request_Path	idle dwords					
					ArbWon										
											Forward Open	XL2: Request_Open			

Figure J.6 — Connection request - backoff and retry

J.7 Connection request - backoff and reverse path

Figure J.7 shows a higher priority OPEN address frame (B to A) received by a phy which has previously forwarded an OPEN address frame (A to B) whose source (A) matches the winning destination (A). In this case expander phy [Y] forwards the higher priority OPEN to expander phy [X] (see 7.15.9).

Expander phy [X]					Expander phy [Y]								
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx				
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords				
SOAF													
OPEN (A to B)													
EOAF													
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)									
				ArbWon									
		XL2: Request_Open	Forward Open		Forward Open		XL5: Forward_Open	SOAF					
								XL3: Open_Cnf_Wait		Forward Dword (idle dwords)	Forward Dword (idle dwords)	XL6: Open_Rsp_Wait	OPEN (A to B)
		EOAF											
		idle dwords (forwarded or generated)											
												AIP (WAITING ON DEVICE)	
		idle dwords			Backoff Reverse Path		Backoff Reverse Path	OPEN (B to A)					
	EOAF												
	SOAF	XL5: Forward_Open		Forward Open		Forward Open	AIP (NORMAL)						
	OPEN (B to A)			Forward Dword (idle dwords)		Forward Dword (idle dwords)							
	EOAF	XL6: Open_Rsp_Wait							XL3: Open_Cnf_Wait				
	idle dwords (forwarded or generated)			Arb Status (Waiting on Device)									
				AIP (WAITING ON DEVICE)									

Figure J.7 — Connection request - backoff and reverse path

J.8 Connection close - single step

Figure J.8 shows an end device initiating the closing of a connection by transmitting CLOSE, followed by another end device responding with CLOSE at a later time.

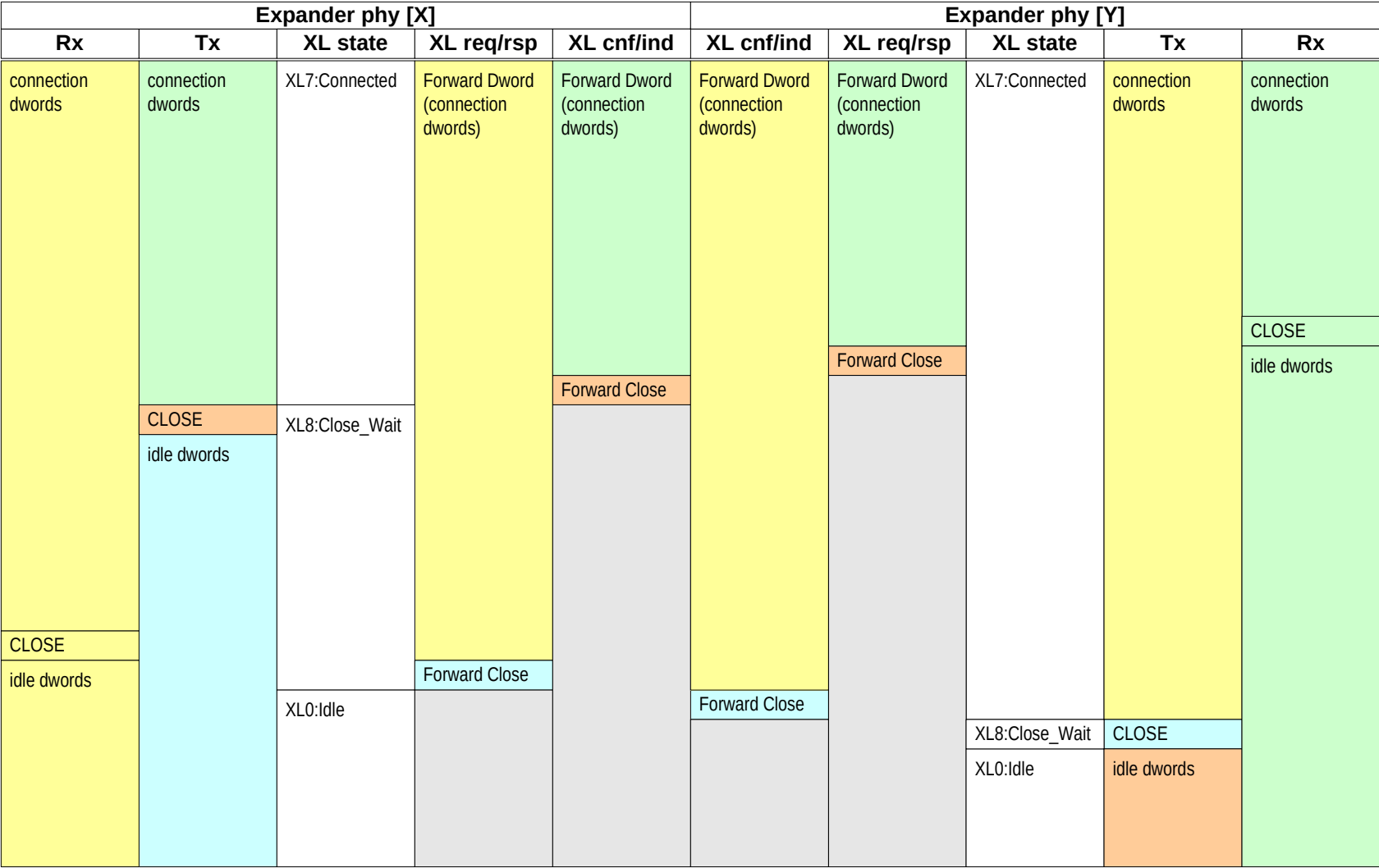


Figure J.8 — Connection close - single step

J.9 Connection close - simultaneous

Figure J.9 shows two end devices simultaneously transmitting CLOSE to each other.

Expander phy [X]					Expander phy [Y]						
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx		
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	XL7:Connected	connection dwords	connection dwords		
CLOSE											CLOSE
idle dwords										Forward Close	
	CLOSE	XL8:Close_Wait		Forward Close	Forward Close		XL8:Close_Wait	CLOSE			
	idle dwords	XL0:Idle					XL0:Idle	idle dwords			

Figure J.9 — Connection close - simultaneous

J.10 BREAK handling during path arbitration when the BREAK_REPLY method is disabled

Figure J.10 shows an expander device responding to the reception of a BREAK during path arbitration when the BREAK_REPLY method of responding to BREAK primitive sequences is disabled (see 7.12.5).

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN(A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
BREAK									
idle dwords	BREAK	XL9:Break							
	idle dwords	XL0:Idle							

Figure J.10 — BREAK handling during path arbitration when the BREAK_REPLY method is disabled

J.11 BREAK handling during connection when the BREAK_REPLY method is disabled

Figure J.11 shows an expander device responding to the reception of a BREAK during a connection when the BREAK_REPLY method of responding to BREAK primitive sequences is disabled (see 7.12.5).

Expander phy [X]					Expander phy [Y]					
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx	
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	XL7:Connected	connection dwords	connection dwords	
									BREAK	
						Forward Break			idle dwords	
	BREAK	idle dwords	XL10: Break_Wait				XL9:Break	BREAK		
								XL0:Idle		idle dwords

Figure J.11 — BREAK handling during a connection when the BREAK_REPLY method is disabled

J.12 BREAK handling during path arbitration when the BREAK_REPLY method is enabled

Figure J.10 shows an expander device responding to the reception of a BREAK during path arbitration when the BREAK_REPLY method of responding to BREAK primitive sequences is enabled (see 7.12.5).

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN(A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
BREAK									
idle dwords	BREAK_REPLY	XL9:Break							
	idle dwords	XL0:Idle							

Figure J.12 — BREAK handling during path arbitration when the BREAK_REPLY method is enabled

J.13 BREAK handling during connection when BREAK_REPLY method is enabled

Figure J.11 shows an expander device responding to the reception of a BREAK during a connection when the BREAK_REPLY method of responding to BREAK primitive sequences is enabled (see 7.12.5).

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	XL7:Connected	connection dwords	connection dwords
				BREAK					
				idle dwords					
	BREAK	XL10: Break_Wait		Forward Break			XL9:Break	BREAK_REPLY	idle dwords
	idle dwords						XL0:Idle	idle dwords	
BREAK_REPLY									
idle dwords									

Figure J.13 — BREAK handling during a connection when the BREAK_REPLY method is enabled

J.14 STP connection - originated by STP initiator port

Figure J.14 shows an STP initiator port originating a connection to an STP target port in an STP/SATA bridge.

Expander phy [W] - STP target port in an STP/SATA bridge					Expander phy [Z] - SATA host port in an STP/SATA bridge						
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx		
idle dwords	idle dwords	XL0:Idle						SYNC/CONT	SATA device dwords		
SOAF											
OPEN (C to D)											
EOAF											
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)							
				Arb Won							
		XL2: Request_Open	Forward Open		Forward Open						
						XL3: Open_Cnf_Wait	Forward Dword (idle dwords)				Forward Dword (idle dwords)
	idle dwords	Open Accept									
			OPEN_ACCEPT	Forward Dword (SATA device dwords ¹)							
	STP connection dwords	SATA device dwords ¹	XL7:Connected		Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords ¹)					
		SATA device dwords		Forward Dword (SATA device dwords)		Forward Dword (SATA device dwords)	STP connection dwords				

¹ STP/SATA bridge duplicates the dword stream which is being received from the SATA device before forwarding dwords - this ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

Figure J.14 — STP connection - originated by STP initiator port

J.15 STP connection - originated by STP target port in an STP/SATA bridge

Figure J.15 shows an STP target port in an STP/SATA bridge originating a connection on behalf of a SATA device which is requesting to transmit a frame.

Expander phy [W] - STP target port in an STP/SATA bridge					Expander phy [Z] - SATA host port in an STP/SATA bridge					
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx	
idle dwords	idle dwords	XL0:Idle						SYNC/CONT	SYNC/CONT	
									Request Path	X_RDY/CONT
					Arbitrating (Normal)					
						Arb Won				
	SOAF	XL5: Forward_Open		Forward Open		Forward Open		Forward Dword (idle dwords)		
	OPEN (D to C)			Forward Dword (idle dwords)						
	EOAF									
	idle dwords (forwarded or generated)	XL6: Open_Rsp_Wait		Arb Status (Waiting On Device)		Arb Status (Waiting On Device)				
				Open Accept	Open Accept	Forward Dword (SATA device dwords ^a)			Forward Dword (SATA device dwords)	
	OPEN_ACCEPT	SATA device dwords ^a		XL7:Connected	Transmit Dword (STP connection dwords)			Forward Dword (SATA device dwords)	Forward Dword (SATA device dwords)	STP connection dwords

^a STP/SATA bridge duplicates the dword stream which is being received from the SATA device before forwarding dwords. This ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

Figure J.15 — STP connection - originated by STP target port in an STP/SATA bridge

Working Draft Serial Attached SCSI - 2 (SAS-2)

Figure J.16 — STP connection close - originated by STP initiator port

J.17 STP connection close - originated by STP target port in an STP/SATA bridge

Figure J.17 shows an STP target port in an STP/SATA bridge closing an STP connection.

Expander phy [W] - STP target port in an STP/SATA bridge					Expander phy [Z] - SATA host port in an STP/SATA bridge				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
STP connection dwords	SATA device dwords	XL7:Connected	Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords)	Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords)		STP connection dwords	SATA device dwords
			Forward Dword (SYNC/CONT)	Forward Dword (SYNC/CONT)		Forward Dword (SYNC/CONT)			SYNC/CONT
	SYNC/CONT			Forward Dword (SYNC/CONT)	Forward Dword (SYNC/CONT)	Forward Close		SYNC/CONT	
SYNC/CONT	CLOSE		XL8:Close_Wait						
	idle dwords	XL0:Idle							
							CLOSE		
idle dwords									

Figure J.17 — STP connection close - originated by STP target port in an STP/SATA bridge

Working Draft Serial Attached SCSI - 2 (SAS-2)

718

Figure J.18 — XL1:Request_Path to XL5:Forward_Open transition

J.19 Pathway blocked and pathway recovery example

Figure J.19 shows a topology used to illustrate pathway recovery. Exp[1] and Exp[2] are expander devices. A, B, and C are end devices. A attempts to open a connection to B while B attempts to open a connection to A.

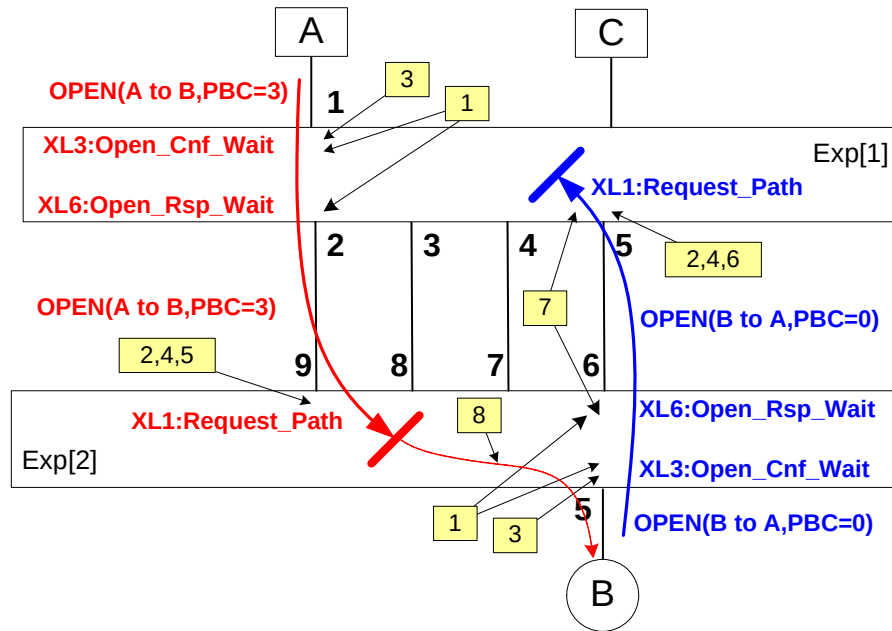


Figure J.19 — Partial pathway recovery

The sequence of events used to identify pathway blockage and to perform pathway recovery are as follows:

- 1) Exp[1].Phy[1,2] and Exp[2].Phy[5,6] each send Phy Status (Partial Pathway) responses to the ECM to indicate that they contain partial pathways;
- 2) Exp[1].Phy[5] and Exp[2].Phy[9] each receive Arbitrating (Waiting On Partial) confirmations from the ECM which cause them to transmit AIP (WAITING ON PARTIAL);
- 3) AIP (WAITING ON PARTIAL) is received by Exp[1].Phy[2] and Exp[2].Phy[6] then forwarded to Exp[1].Phy[1] and Exp[2].Phy[5] as an Arb Status (Waiting On Partial) confirmation. Exp[1].Phy[1] and Exp[2].Phy[5] each send Phy Status (Blocked On Partial) responses to ECM as confirmation that they are blocked waiting on a partial pathway in another expander device;
- 4) Exp[1].Phy[5] and Exp[2].Phy[9] each receive Arbitrating (Blocked On Partial) confirmations from the ECM while all destination phys send Phy Status (Blocked On Partial) responses which cause them to run their Partial Pathway Timeout timers;
- 5) The Partial Pathway Timeout timer expires in Exp[2].Phy[9]. This causes a request to the ECM to resolve pathway blockage. The pathway recovery priority for this phy is not lower than all phys within the destination port which are also blocked (i.e., the Request Path from Exp[2].Phy[9] has higher priority than the Request Path from Exp[2].Phy[5], which is receiving Arbitrating (Blocked On Partial)). The ECM does not provide an Arb Reject (Pathway Blocked) confirmation to Exp[2].Phy[9], so this expander phy waits for pathway resolution to occur elsewhere in the topology;
- 6) the Partial Pathway Timeout timer expires in Exp[1].Phy[5]. This causes a request to the ECM to resolve pathway blockage. The pathway recovery priority for this expander phy is lower than all expander phys within the destination port which are also blocked (i.e., the Request Path from Exp[1].Phy[5] is lower priority than the Request Path from Exp[1].Phy[5], which is receiving Arbitrating (Blocked On Partial)). The ECM provides an Arb Reject (Pathway Blocked) confirmation to Exp[1].Phy[5] which instructs this expander phy to reject the connection request using OPEN_REJECT (PATHWAY BLOCKED);

NOTE 115 - The Partial Pathway Timeout timer in Exp[1].Phy[5] might expire before, after, or at the same time the Partial Pathway Timeout timer expires in Exp[2].Phy[9].

- 7) OPEN_REJECT (PATHWAY BLOCKED) tears down partial pathway all the way to the originating end device (Device B);
- 8) Exp[2].Phy[9] receives Arb Won and the partial pathway is extended through Exp[2].Phy[5]; and
- 9) OPEN (A to B) is delivered to device B.

Annex K

(informative)

Primitive encoding

Table K.1 describes a set of the K28.5-based primitive encodings whose 40-bit values (after 8b10b encoding with either starting running disparity) have a Hamming distance (i.e., the number of bits different in two patterns) of at least 8. All the primitive encodings in 7.2 except for TRAIN and TRAIN_DONE were selected from this list. Unassigned encodings may be used by future versions of this standard.

Table K.1 — Primitives with Hamming distance of 8 (part 1 of 3)

Character				Assignment
1 st	2 nd	3 rd	4 th	
K28.5	D01.3	D01.3	D01.3	ALIGN (2)
K28.5	D01.4	D01.4	D01.4	ACK
K28.5	D01.4	D02.0	D31.4	RRDY (RESERVED 0)
K28.5	D01.4	D04.7	D24.0	NAK (RESERVED 1)
K28.5	D01.4	D07.3	D30.0	CREDIT_BLOCKED
K28.5	D01.4	D16.7	D07.3	NAK (RESERVED 2)
K28.5	D01.4	D24.0	D16.7	RRDY (NORMAL)
K28.5	D01.4	D27.4	D04.7	NAK (CRC ERROR)
K28.5	D01.4	D30.0	D02.0	RRDY (RESERVED 1)
K28.5	D01.4	D31.4	D29.7	NAK (RESERVED 0)
K28.5	D02.0	D01.4	D29.7	ERROR
K28.5	D02.0	D02.0	D02.0	HARD_RESET
K28.5	D02.0	D04.7	D01.4	CLOSE (RESERVED 1)
K28.5	D02.0	D07.3	D04.7	CLOSE (CLEAR AFFILIATION)
K28.5	D02.0	D16.7	D31.4	MUX (LOGICAL LINK 0)
K28.5	D02.0	D24.0	D07.3	BREAK
K28.5	D02.0	D29.7	D16.7	BREAK_REPLY
K28.5	D02.0	D30.0	D27.4	CLOSE (NORMAL)
K28.5	D02.0	D31.4	D30.0	CLOSE (RESERVED 0)
K28.5	D04.7	D01.4	D24.0	BROADCAST (EXPANDER)
K28.5	D04.7	D02.0	D01.4	BROADCAST (CHANGE)
K28.5	D04.7	D04.7	D04.7	BROADCAST (ASYNCHRONOUS EVENT)
K28.5	D04.7	D07.3	D29.7	BROADCAST (SES)
K28.5	D04.7	D16.7	D02.0	BROADCAST (RESERVED 3)
K28.5	D04.7	D24.0	D31.4	BROADCAST (RESERVED CHANGE 0)
K28.5	D04.7	D27.4	D07.3	BROADCAST (RESERVED CHANGE 1)
K28.5	D04.7	D29.7	D30.0	BROADCAST (RESERVED 4)
K28.5	D04.7	D31.4	D27.4	MUX (LOGICAL LINK 1)
K28.5	D07.0	D07.0	D07.0	ALIGN (1)
K28.5	D07.3	D01.4	D31.4	
K28.5	D07.3	D02.0	D04.7	

Table K.1 — Primitives with Hamming distance of 8 (part 2 of 3)

Character				Assignment
1 st	2 nd	3 rd	4 th	
K28.5	D07.3	D04.7	D30.0	
K28.5	D07.3	D07.3	D07.3	
K28.5	D07.3	D24.0	D29.7	
K28.5	D07.3	D27.4	D16.7	
K28.5	D07.3	D29.7	D27.4	
K28.5	D07.3	D30.0	D24.0	
K28.5	D07.3	D31.4	D02.0	
K28.5	D10.2	D10.2	D27.3	ALIGN (0)
K28.5	D16.7	D01.4	D02.0	
K28.5	D16.7	D02.0	D07.3	
K28.5	D16.7	D04.7	D31.4	
K28.5	D16.7	D16.7	D16.7	OPEN_ACCEPT
K28.5	D16.7	D24.0	D27.4	
K28.5	D16.7	D27.4	D30.0	
K28.5	D16.7	D29.7	D24.0	
K28.5	D16.7	D30.0	D04.7	
K28.5	D16.7	D31.4	D01.4	
K28.5	D24.0	D01.4	D16.7	
K28.5	D24.0	D02.0	D29.7	
K28.5	D24.0	D04.7	D07.3	SOF
K28.5	D24.0	D07.3	D31.4	EOAF
K28.5	D24.0	D16.7	D27.4	EOF
K28.5	D24.0	D24.0	D24.0	
K28.5	D24.0	D27.4	D02.0	
K28.5	D24.0	D29.7	D04.7	
K28.5	D24.0	D30.0	D01.4	SOAF
K28.5	D27.3	D27.3	D27.3	ALIGN (3)
K28.5	D27.4	D01.4	D07.3	AIP (RESERVED WAITING ON PARTIAL)
K28.5	D27.4	D04.7	D02.0	
K28.5	D27.4	D07.3	D24.0	AIP (WAITING ON CONNECTION)
K28.5	D27.4	D16.7	D30.0	AIP (RESERVED 1)
K28.5	D27.4	D24.0	D04.7	AIP (WAITING ON PARTIAL)
K28.5	D27.4	D27.4	D27.4	AIP (NORMAL)
K28.5	D27.4	D29.7	D01.4	AIP (RESERVED 2)
K28.5	D27.4	D30.0	D29.7	AIP (WAITING ON DEVICE)
K28.5	D27.4	D31.4	D16.7	AIP (RESERVED 0)
K28.5	D29.7	D02.0	D30.0	OPEN_REJECT (RESERVED CONTINUE 0)
K28.5	D29.7	D04.7	D27.4	OPEN_REJECT (RESERVED STOP 1)
K28.5	D29.7	D07.3	D16.7	OPEN_REJECT (RESERVED INITIALIZE 1)

Table K.1 — Primitives with Hamming distance of 8 (part 3 of 3)

Character				Assignment
1 st	2 nd	3 rd	4 th	
K28.5	D29.7	D16.7	D04.7	OPEN_REJECT (PATHWAY BLOCKED)
K28.5	D29.7	D24.0	D01.4	OPEN_REJECT (RESERVED CONTINUE 1)
K28.5	D29.7	D27.4	D24.0	OPEN_REJECT (RETRY)
K28.5	D29.7	D29.7	D29.7	OPEN_REJECT (NO DESTINATION)
K28.5	D29.7	D30.0	D31.4	OPEN_REJECT (RESERVED INITIALIZE 0)
K28.5	D29.7	D31.4	D07.3	OPEN_REJECT (RESERVED STOP 0)
K28.5	D30.0	D01.4	D04.7	DONE (ACK/NAK TIMEOUT)
K28.5	D30.0	D02.0	D16.7	
K28.5	D30.0	D07.3	D27.4	DONE (CREDIT TIMEOUT)
K28.5	D30.0	D16.7	D01.4	DONE (RESERVED 0)
K28.5	D30.0	D24.0	D02.0	
K28.5	D30.0	D27.4	D29.7	DONE (RESERVED TIMEOUT 0)
K28.5	D30.0	D29.7	D31.4	DONE (RESERVED 1)
K28.5	D30.0	D30.0	D30.0	DONE (NORMAL)
K28.5	D30.0	D31.4	D24.0	DONE (RESERVED TIMEOUT 1)
K28.5	D31.3	D01.3	D07.0	NOTIFY (RESERVED 1)
K28.5	D31.3	D07.0	D01.3	NOTIFY (POWER LOSS EXPECTED)
K28.5	D31.3	D10.2	D10.2	NOTIFY (RESERVED 2)
K28.5	D31.3	D31.3	D31.3	NOTIFY (ENABLE SPINUP)
K28.5	D31.4	D01.4	D30.0	OPEN_REJECT (RESERVED ABANDON 3)
K28.5	D31.4	D02.0	D27.4	OPEN_REJECT (ZONE VIOLATION)
K28.5	D31.4	D04.7	D29.7	OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)
K28.5	D31.4	D07.3	D02.0	OPEN_REJECT (RESERVED ABANDON 2)
K28.5	D31.4	D16.7	D24.0	OPEN_REJECT (WRONG DESTINATION)
K28.5	D31.4	D27.4	D01.4	OPEN_REJECT (STP RESOURCES BUSY)
K28.5	D31.4	D29.7	D07.3	OPEN_REJECT (PROTOCOL NOT SUPPORTED)
K28.5	D31.4	D30.0	D16.7	OPEN_REJECT (RESERVED ABANDON 1)
K28.5	D31.4	D31.4	D31.4	OPEN_REJECT (BAD DESTINATION)

Table K.2 describes the K28.5-based primitive encodings that do not have Hamming distances of 8 from the other primitives.

Table K.2 — Primitives without Hamming distance of 8

Character				Assignment
1 st	2 nd	3 rd	4 th	
K28.5	D30.3	D30.3	D30.3	TRAIN
K28.5	D30.3	D30.3	D10.2	TRAIN_DONE

Annex L

(informative)

Discover process example implementation

L.1 Discover process example implementation overview

This annex includes a C program implementing the discover process. Table L.1 describes the source files.

Table L.1 — C program files

Filename	Description
SASDiscoverSimulation.h	header file
SASDiscoverSimulation.cpp	C source file

L.2 Header file

The following is the C header file for the discover process.

Editor's Note 68: Changes to the .h file are necessary to remove edge/fanout and edge expander device set handling. Delete "edge" and "fanout" from commands. Replace "edge expander device set" with a term for all the expanders beyond an S-S link. Otherwise, code should work as is.

```
// SASDiscoverSimulation.h
// updated 2005/05/29

// assume the maximum number of phys in an expander device is 128
#define MAXIMUM_EXPANDER_PHYS          128

// assume the maximum number of indexes per phy is 128
#define MAXIMUM_EXPANDER_INDEXES      128

// limit to 8 initiators for this example
#define MAXIMUM_INITIATORS            8

// defines for address frame types
#define ADDRESS_IDENTIFY_FRAME        0x00
#define ADDRESS_OPEN_FRAME            0x01

// defines for SMP frame types
#define SMP_REQUEST_FRAME              0x40
#define SMP_RESPONSE_FRAME            0x41

// defines for SMP request functions
#define REPORT_GENERAL                  0x00
#define REPORT_MANUFACTURER_INFORMATION 0x01
#define DISCOVER                       0x10
#define REPORT_PHY_ERROR_LOG           0x11
#define REPORT_PHY_SATA                0x12
#define REPORT_ROUTE_INFORMATION       0x13
#define CONFIGURE_ROUTE_INFORMATION    0x90
#define PHY_CONTROL                    0x91
#define PHY_TEST                       0x92
```

```

// defines for the protocol bits
#define SATA 0x01
#define SMP 0x02
#define STP 0x04
#define SSP 0x08

// defines for open responses, arbitrary values, not defined in the spec
#define OPEN_ACCEPT 0
#define OPEN_REJECT_BAD_DESTINATION 1
#define OPEN_REJECT_RATE_NOT_SUPPORTED 2
#define OPEN_REJECT_NO_DESTINATION 3
#define OPEN_REJECT_PATHWAY_BLOCKED 4
#define OPEN_REJECT_PROTOCOL_NOT_SUPPORTED 5
#define OPEN_REJECT_RESERVE_ABANDON 6
#define OPEN_REJECT_RESERVE_CONTINUE 7
#define OPEN_REJECT_RESERVE_INITIALIZE 8
#define OPEN_REJECT_RESERVE_STOP 9
#define OPEN_REJECT_RETRY 10
#define OPEN_REJECT_STP_RESOURCES_BUSY 11
#define OPEN_REJECT_WRONG_DESTINATION 12
#define OPEN_REJECT_WAITING_ON_BREAK 13

// definitions for discovery algorithm use
enum
{
    SAS_SIMPLE_LEVEL_DESCENT = 0,
    SAS_UNIQUE_LEVEL_DESCENT
};

enum
{
    SAS_10_COMPATIBLE = 0,
    SAS_11_COMPATIBLE
};

// definitions for SMP function results
enum SMPFunctionResult
{
    SMP_REQUEST_ACCEPTED = 0, // from original example

    SMP_FUNCTION_ACCEPTED = 0,
    SMP_UNKNOWN_FUNCTION,
    SMP_FUNCTION_FAILED,
    SMP_INVALID_REQUEST_FRAME_LENGTH,
    SMP_PHY_DOES_NOT_EXIST = 0x10,
    SMP_INDEX_DOES_NOT_EXIST,
    SMP_PHY_DOES_NOT_SUPPORT_SATA,
    SMP_UNKNOWN_PHY_OPERATION,
    SMP_UNKNOWN_PHY_TEST_FUNCTION,
    SMP_UNKNOWN_PHY_TEST_FUNCTION_IN_PROGRESS,
    SMP_PHY_VACANT
};

// DeviceTypes
enum DeviceTypes
{
    NO_DEVICE = 0,

```

```

    END_DEVICE,
    EDGE_EXPANDER_DEVICE,
    FANOUT_EXPANDER_DEVICE,

    END = END_DEVICE,           // from original example
    EDGE = EDGE_EXPANDER_DEVICE, // from original example
    FANOUT = FANOUT_EXPANDER_DEVICE // from original example
};

// RoutingAttribute
enum RoutingAttribute
{
    DIRECT = 0,
    SUBTRACTIVE,
    TABLE,

    // this attribute is a psuedo attribute, used to reflect the function
    // result of SMP_PHY_VACANT in a fabricated discover response
    PHY_NOT_USED = 15
};

// ConnectorType
enum ConnectorType
{
    UNKNOWN_CONNECTOR = 0,
    SFF_8470_EXTERNAL_WIDE,
    SFF_8484_INTERNAL_WIDE = 16,
    SFF_8482_BACKPLANE = 32,
    SATA_HOST_PLUG,
    SAS_DEVICE_PLUG,
    SATA_DEVICE_PLUG
};

// RouteFlag
enum DisableRouteEntry
{
    ENABLED = 0,
    DISABLED
};

// PhyLinkRate(s)
enum PhysicalLinkRate
{
    RATE_UNKNOWN = 0,
    PHY_DISABLED,
    PHY_FAILED,
    SPINUP_HOLD_OOB,
    PORT_SELECTOR_DETECTED,

    // this is a psuedo link rate, used to reflect the function
    // result of SMP_PHY_VACANT in a fabricated discover response
    PHY_DOES_NOT_EXIST,

    GBPS_1_5 = 8,
    GBPS_3_0
};

```



```

// PhyOperation
enum PhyOperation
{
    NOP = 0,
    LINK_RESET,
    HARD_RESET,
    DISABLE,
    CLEAR_ERROR_LOG = 5,
    CLEAR_AFFILIATION,
    TRANSMIT_SATA_PORT_SELECTION_SIGNAL
};

// provide the simple type definitions
typedef unsigned char byte;
typedef unsigned short word;
typedef unsigned long dword;
typedef unsigned _int64 quadword;

// the structures assume a char bitfield is valid, this is compiler
// dependent defines would be more portable, but less descriptive

// the Identify frame is exchanged following 00B, for this
// code it contains the identity information for the attached device
// and the initiator application client
struct Identify
{
    // byte 0
    byte AddressFrameType:4;           // ADDRESS_IDENTIFY_FRAME
    byte DeviceType:3;                 // END_DEVICE
    //
    byte RestrictedByte0Bit7:1;

    // byte 1
    byte RestrictedByte1;

    // byte 2
    union
    {
        struct
        {
            byte RestrictedByte2Bit0:1;
            byte SMPInitiatorPort:1;
            byte STPInitiatorPort:1;
            byte SSPInitiatorPort:1;
            byte ReservedByte2Bit4_7:4;
        };
        byte InitiatorBits;
    };

    // byte 3
    union
    {
        struct
        {
            byte RestrictedByte3Bit0:1;
            byte SMPTargetPort:1;
            byte STPTargetPort:1;
        };
    };
};

```

```

        byte SSPTargetPort:1;
        byte ReservedByte3Bit4_7:4;
    };
    byte TargetBits;
};

// byte 4-11
byte RestrictedByte4_11[8];

// byte 12-19
quadword SASAddress;

// byte 20
byte PhyIdentifier;

// byte 21-27
byte ReservedByte21_27[4];

// byte 28-31
dword CRC;
}; // struct Identify

// the Open address frame is used to send open requests
struct OpenAddress
{
    // byte 0
    byte AddressFrameType:4;           // ADDRESS_OPEN_FRAME
    byte Protocol:3;                   // SMP
                                        // STP
                                        // SSP

    byte InitiatorPort:1;

    // byte 1
    byte ConnectionRate:4;             // GBPS_1_5
                                        // GBPS_3_0

    byte Features:4;

    // byte 2-3
    word InitiatorConnectionTag;

    // byte 4-11
    quadword DestinationSASAddress;

    // byte 12-19
    quadword SourceSASAddress;

    // byte 20
    byte CompatibleFeatures;

    // byte 21
    byte PathwayBlockedCount;

    // byte 22-23
    word ArbitrationWaitTime;

    // byte 24-27
    byte MoreCompatibleFeatures[4];
};

```

```

        // byte 28-31
        dword CRC[4];
    }; // struct OpenAddress

    // request specific bytes for a general input function
    struct SMPRequestGeneralInput
    {
        // byte 4-7
        dword CRC;
    };

    // request specific bytes for a phy input function
    struct SMPRequestPhyInput
    {
        // byte 4-7
        byte IgnoredByte4_7[4];

        // byte 8
        byte ReservedByte8;

        // byte 9
        byte PhyIdentifier;

        // byte 10
        byte IgnoredByte10;

        // byte 11
        byte ReservedByte11;

        // byte 12-15
        dword CRC;
    }; // struct SMPRequestPhyInput

    // the ConfigureRouteInformation structure is used to provide the
    // expander route entry for the expander route table, it is intended
    // to be referenced by the SMPRequestConfigureRouteInformation struct
    struct ConfigureRouteInformation
    {
        // byte 12
        byte IgnoredByte12Bit0_6:7;
        byte DisableRouteEntry:1; // if a routing error is detected
                                // then the route is disabled by
                                // setting this bit

        // byte 13-15
        byte IgnoredByte13_15[3];

        // byte 16-23
        quadword RoutedSASAddress; // identical to the AttachedSASAddress
                                // found through discovery

        // byte 24-35
        byte IgnoredByte24_35[12];

        // byte 36-39
        byte ReservedByte36_39[4];
    };

```

```

}; // struct ConfigureRouteInformation

// request specific bytes for SMP ConfigureRouteInformation function
struct SMPRequestConfigureRouteInformation
{
    // byte 4-5
    byte ReservedByte4_5[2];

    // byte 6-7
    word ExpanderRouteIndex;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10-11
    byte ReservedByte10_11[2];

    // byte 12-39
    struct ConfigureRouteInformation Configure;

    // byte 40-43
    dword CRC;
}; // struct SMPRequestConfigureRouteInformation

// the PhyControlInformation structure is used to provide the
// expander phy control values, it is intended
// to be referenced by the SMPRequestPhyControl struct
struct PhyControlInformation
{
    // byte 12-31
    byte IgnoredByte12_31[20];

    // byte 32
    byte IgnoredByte32Bit0_3:4;
    byte ProgrammedMinimumPhysicalLinkRate:4;

    // byte 33
    byte IgnoredByte33Bit0_3:4;
    byte ProgrammedMaximumPhysicalLinkRate:4;

    // byte 34-35
    byte IgnoredByte34_35[2];

    // byte 36
    byte PartialPathwayTimeoutValue:4;
    byte ReservedByte36Bit4_7:4;

    // byte 37-39
    byte ReservedByte37_39[3];
}; // struct PhyControlInformation

// request specific bytes for SMP Phy Control function
struct SMPRequestPhyControl
{

```

```
// byte 4-7
byte IgnoredByte4_7[4];

// byte 8
byte ReservedByte8;

// byte 9
byte PhyIdentifier;

// byte 10
byte PhyOperation;

// byte 11
byte UpdatePartialPathwayTimeoutValue:1;
byte ReservedByte11Bit1_7:7;

// byte 12-39
struct PhyControlInformation Control;

// byte 40-43
dword CRC;
}; // struct SMPRequestPhyControl

// request specific bytes for SMP Phy Test function
struct SMPRequestPhyTest
{
    // byte 4-7
    byte IgnoredByte4_7[4];

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte PhyTestFunction;

    // byte 11
    byte PhyTestPattern;

    // byte 12-14
    byte ReservedByte12_14[3];

    // byte 15
    byte PhyTestPatternPhysicalLinkRate:4;
    byte ReservedByte15Bit4_7:4;

    // byte 16-39
    byte ReservedByte16_39[24];

    // byte 40-43
    dword CRC;
}; // struct SMPRequestPhyTest

// generic structure referencing an SMP Request, must be initialized
// before being used
```

```

struct SMPRequest
{
    // byte 0
    byte SMPFrameType;                // always SMP_REQUEST_FRAME

    // byte 1
    byte Function;                    // REPORT_GENERAL
                                        // REPORT_MANUFACTURER_INFORMATION
                                        // DISCOVER
                                        // REPORT_PHY_ERROR_LOG
                                        // REPORT_PHY_SATA
                                        // REPORT_ROUTE_INFORMATION
                                        // CONFIGURE_ROUTE_INFORMATION
                                        // PHY_CONTROL
                                        // PHY_TEST

    // byte 2-3
    byte ReservedByte2_3[2];

    // bytes 4-n
    union
    {
        struct SMPRequestGeneralInput ReportGeneral;
        struct SMPRequestGeneralInput ReportManufacturerInformation;
        struct SMPRequestPhyInput Discover;
        struct SMPRequestPhyInput ReportPhyErrorLog;
        struct SMPRequestPhyInput ReportPhySATA;
        struct SMPRequestPhyInput ReportRouteInformation;
        struct SMPRequestConfigureRouteInformation ConfigureRouteInformation;
        struct SMPRequestPhyControl PhyControl;
        struct SMPRequestPhyTest PhyTest;
    } Request;
}; // struct SMPRequest

// request specific bytes for SMP Report General response, intended to be
// referenced by SMPResponse
struct SMPResponseReportGeneral
{
    // byte 4-5
    word ExpanderChangeCount;

    // byte 6-7
    word ExpanderRouteIndexes;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte NumberOfPhys;

    // byte 10
    byte ConfigurableRouteTable:1;
    byte Configuring:1;
    byte ReservedByte10Bit2_7:6;

    // byte 11
    byte ReservedByte11;

```

```

    // byte 12-19
    byte EnclosureLogicalIdentifier[8];

    // byte 20-27
    byte ReservedByte20_27[8];

    // byte 28-31
    dword CRC;
}; // struct SMPResponseReportGeneral

struct SAS11FormatReportManufacturerInformation
{
    // byte 40-47
    byte ComponentVendorIdentification[8];

    // byte 48-49
    byte ComponentID[2];

    // byte 50
    byte ComponentRevisionID;

    // byte 51
    byte Reserved;

    // byte 52-59
    byte VendorSpecific[8];
}; // struct SAS11FormatReportManufacturerInformation

// request specific bytes for SMP Report Manufacturer Information response,
// intended to be referenced by SMPResponse
struct SMPResponseReportManufacturerInformation
{
    // byte 4-7
    byte IgnoredByte4_7[4];

    // byte 8
    byte SAS11Format:1;
    byte ReservedByte8_Bit1_7:7;

    // byte 9-10
    byte IgnoredByte9_10[2];

    // byte 11
    byte ReservedByte11;

    // byte 12-19
    byte VendorIdentification[8];

    // byte 20-35
    byte ProductIdentification[16];

    // byte 36-39
    byte ProductRevisionLevel[4];

    union
    {

```

```

    struct SAS11FormatReportManufacturerInformation SAS11;

    // byte 40-59
    byte VendorSpecific[20];
};

// byte 60-63
dword CRC;
}; // struct SMPResponseReportManufacturerInformation

// the Discover structure is used to retrieve expander port information
// it is intended to be referenced by the SMPResponseDiscover structure
struct Discover
{
    // byte 12
    byte ReservedByte12Bit0_3:4;
    byte AttachedDeviceType:3;
    byte IgnoredByte12Bit7:1;

    // byte 13
    byte NegotiatedPhysicalLinkRate:4;
    byte ReservedByte13Bit4_7:4;

    // byte 14
    union
    {
        struct
        {
            byte AttachedSATAHost:1;
            byte AttachedSMPInitiator:1;
            byte AttachedSTPInitiator:1;
            byte AttachedSSPInitiator:1;
            byte ReservedByte14Bit4_7:4;
        };
        byte InitiatorBits;
    };

    // byte 15
    union
    {
        struct
        {
            byte AttachedSATADevice:1;
            byte AttachedSMPTarget:1;
            byte AttachedSTPTarget:1;
            byte AttachedSSPTarget:1;
            byte ReservedByte15Bit4_6:3;
            byte AttachedSATAPortSelector:1;
        };
        byte TargetBits;
    };

    // byte 16-23
    quadword SASAddress;

    // byte 24-31
    quadword AttachedSASAddress;
};

```



```

    // byte 32
    byte AttachedPhyIdentifier;

    // byte 33-39
    byte ReservedByte33_39[7];

    // byte 40
    byte HardwareMinimumPhysicalLinkRate:4;
    byte ProgrammedMinimumPhysicalLinkRate:4;

    // byte 41
    byte HardwareMaximumPhysicalLinkRate:4;
    byte ProgrammedMaximumPhysicalLinkRate:4;

    // byte 42
    byte PhyChangeCount;

    // byte 43
    byte PartialPathwayTimeoutValue:4;
    byte IgnoredByte36Bit4_6:3;
    byte VirtualPhy:1;

    // byte 44
    byte RoutingAttribute:4;
    byte ReservedByte44Bit4_7:4;

    // byte 45
    byte ConnectorType:7;
    byte ReservedByte45Bit7:1;

    // byte 46
    byte ConnectorElementIndex;

    // byte 47
    byte ConnectorPhysicalLink;

    // byte 48-49
    byte ReservedByte48_49[2];

    // byte 50-51
    byte VendorSpecific[2];

    // byte 52-55
    dword CRC;
}; // struct Discover

// response specific bytes for SMP Discover, intended to be referenced by
// SMPResponse
struct SMPResponseDiscover
{
    // byte 4-7
    byte IgnoredByte4_7;

    // byte 8
    byte ReservedByte8;

```

```

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte ReservedByte11;

    union                                // original example used Results instead
                                        // of Result, this allows both
    {
        // byte 12-55
        struct Discover Results;
        struct Discover Result;
    };
}; // struct SMPResponseDiscover

// response specific bytes for SMP Report Phy Error Log, intended to be
// referenced by SMPResponse
struct SMPResponseReportPhyErrorLog
{
    // byte 4-7
    byte IgnoredByte4_7;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte ReservedByte11;

    // byte 12-15
    dword InvalidDwordCount;

    // byte 16-19
    dword DisparityErrorCount;

    // byte 20-23
    dword LossOfDwordSynchronizationCount;

    // byte 24-27
    dword PhyResetProblemCount;

    // byte 28-31
    dword CRC;
}; // struct SMPResponseReportPhyErrorLog

// this structure describes the Register Device to Host FIS defined in the
// SATA specification
struct RegisterDeviceToHostFIS
{

```

```
// byte 24
byte FISType;

// byte 25
byte ReservedByte25Bit0_5:6;
byte Interrupt:1;
byte ReservedByte25Bit7:1;

// byte 26
byte Status;

// byte 27
byte Error;

// byte 28
byte SectorNumber;

// byte 29
byte CylLow;

// byte 30
byte CylHigh;

// byte 31
byte DevHead;

// byte 32
byte SectorNumberExp;

// byte 33
byte CylLowExp;

// byte 34
byte CylHighExp;

// byte 35
byte ReservedByte35;

// byte 36
byte SectorCount;

// byte 37
byte SectorCountExp;

// byte 38-43
byte ReservedByte38_43[6];
}; // struct RegisterDeviceToHostFIS

// response specific bytes for SMP Report Phy SATA, intended to be
// referenced by SMPResponse
struct SMPResponseReportPhySATA
{
    // byte 4-7
    byte IgnoredByte4_7;

    // byte 8
    byte ReservedByte8;
```

```

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte AffiliationValid:1;
    byte AffiliationsSupported:1;
    byte ReservedByte11Bit2_7:6;

    // byte 12-15
    byte ReservedByte12_15[4];

    // byte 16-32
    quadword STPSASAddress;

    // byte 24-43
    struct RegisterDeviceToHostFIS FIS;

    // byte 44-47
    byte ReservedByte44_47[4];

    // byte 48-55
    quadword AffiliatedSTPInitiatorSASAddress;

    // byte 56-59
    dword CRC;
}; // struct SMPResponseReportPhySATA

struct ReportRouteInformation
{
    // byte 12
    byte IgnoredByte12Bit0_6:7;
    byte ExpanderRouteEntryDisabled:1;

    // byte 13-15
    byte IgnoredByte13_15[3];

    // byte 16-23
    quadword RoutedSASAddress;

    // byte 24-35
    byte IgnoredByte24_35[12];

    // byte 36-39
    byte ReservedByte36_39[4];
}; // struct ReportRouteInformation

// response specific bytes for SMP Report Route Information, intended to be
// referenced by SMPResponse
struct SMPResponseReportRouteInformation
{
    // byte 4-5
    byte IgnoredByte4_5;

```

```

    // byte 6-7
    word ExpanderRouteIndex;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte ReservedByte11;

    // byte 12-39
    struct ReportRouteInformation Result;

    // byte 40-43
    dword CRC;
}; // struct SMPResponseReportRouteInformation

// response specific bytes for SMP Configure Route Information,
// intended to be referenced by SMPResponse
struct SMPResponseConfigureRouteInformation
{
    // byte 4-7
    dword CRC;
};

// response specific bytes for SMP Phy Control,
// intended to be referenced by SMPResponse
struct SMPResponsePhyControl
{
    // byte 4-7
    dword CRC;
};

// response specific bytes for SMP Phy Test,
// intended to be referenced by SMPResponse
struct SMPResponsePhyTest
{
    // byte 4-7
    dword CRC;
};

// generic structure referencing an SMP Response, must be initialized
// before being used
struct SMPResponse
{
    // byte 0
    byte SMPFrameType; // always 41h for SMP responses

    // byte 1
    byte Function;

    // byte 2

```

```

    byte FunctionResult;

    // byte 3
    byte ReservedByte3;

    // bytes 4-n
    union
    {
        struct SMPResponseReportGeneral ReportGeneral;
        struct SMPResponseReportManufacturerInformation
            ReportManufacturerInformation;
        struct SMPResponseDiscover Discover;
        struct SMPResponseReportPhyErrorLog ReportPhyErrorLog;
        struct SMPResponseReportPhySATA ReportPhySATA;
        struct SMPResponseReportRouteInformation ReportRouteInformation;
        struct SMPResponseConfigureRouteInformation ConfigureRouteInformation;
        struct SMPResponsePhyControl PhyControl;
        struct SMPResponsePhyTest PhyTest;
    } Response;
}; // struct SMPResponse

// this structure is how this simulation obtains its knowledge about the
// initiator port that is doing the discover, it is not defined as part of
// the standard...
struct ApplicationClientKnowledge
{
    quadword SASAddress;
    byte NumberOfPhys;
    byte InitiatorBits;
    byte TargetBits;
};

// the RouteTableEntry structure is used to contain the internal copy of
// the expander route table
struct RouteTableEntry
{
    byte ExpanderRouteEntryDisabled;
    quadword RoutedSASAddress;
};

// the TopologyTable structure is the summary of the information gathered
// during the discover process, the table presented here is not concerned
// about memory resources consumed, production code would be more concerned
// about specifying necessary elements explicitly
struct TopologyTable
{
    // pointer to a simple list of expanders in topology
    // a walk thru this link will encounter all expanders in
    // discover order
    struct TopologyTable *Next;

    // simple reference to this device, primarily to keep identification of
    // this structure simple, otherwise, the only place the address is
    // located is within the Phy element
    quadword SASAddress;

    // information from REPORT_GENERAL

```

```

    struct SMPResponseReportGeneral Device;

    // information from DISCOVER
    struct SMPResponseDiscover Phy[MAXIMUM_EXPANDER_PHYS];

    // list of route indexes for each phy
    word RouteIndex[MAXIMUM_EXPANDER_PHYS];

    // internal copy of the route table for the expander
    struct RouteTableEntry
        RouteTable[MAXIMUM_EXPANDER_PHYS][MAXIMUM_EXPANDER_INDEXES];

    //
    // in production code there would also be links to the necessary device
    // information like end device; vendor, model, serial number, etc.
    // the gathering of that type of information is not done here...
    //
}; // struct TopologyTable

```

L.3 Source file

The following is the C source file for the discover process.

[Editor's Note 69: Changes to the .cpp file are necessary to remove edge/fanout and edge expander device set handling](#)

```

// SASDiscoverSimulation.cpp
// updated 2005/05/29
//
// This is a simple simulation and code implementation of the initiator
// based expander discovery and configuration.

// There is no attempt to handle phy errors, arbitration issues, etc.
// Production level implementation need to handle errors appropriately.

// Structure names used are equivalent to those referenced in the standard.

// Basic assumptions:
// 1. BROADCAST (CHANGE) primitives initiate rediscovery/reconfiguration
// 2. Table locations for SASAddresses are deterministic for a specific
//    topology only. When the topology changes, the location of a SASAddress
//    in an ASIC table cannot be assumed.
// 3. A complete discovery level occurs before the configuration of the
//    level begins. Multiple passes are required as the levels of expanders
//    encountered between the initiator and the end devices increases.
// 4. Configuration of a single expander occurs before proceeding to
//    subsequent attached expanders.
// 5. The Attached structure is filled in following OOB and is available
//    from the initialization routines.
// 6. The Iam structure is provide by the application client.

#include <malloc.h>
#include <memory.h>
#include <stdlib.h>

```

```

// include the SAS structures
#include "SASDiscoverSimulation.h"

// this defines the type of algorithm used for discover
int DiscoverAlgorithm = SAS_SIMPLE_LEVEL_DESCENT;
int SASCompatibility = SAS_11_COMPATIBLE;

// loaded by the application client, in this simulation it is provided
// in a text file, SASDeviceSetExample.ini
extern struct ApplicationClientKnowledge Iam[MAXIMUM_INITIATORS];

// obtained following OOB from the attached phy, in this simulation
// it is provided in a text file, SASDeviceSetExample.ini
extern struct Identify Attached[MAXIMUM_INITIATORS];

// buffers used to request and return SMP data
extern struct SMPRequest SMPRequestFrame;
extern struct SMPResponse SMPResponseFrame;

// resulting discover information ends up in this table
extern struct TopologyTable *SASDomain[MAXIMUM_INITIATORS];

// this is the function used to send an SMPRequest and get a response back
extern byte SMPRequest(byte PhyIdentifier,
                      quadword Source,
                      quadword Destination,
                      struct SMPRequest *SMPRequestFrame,
                      struct SMPResponse *SMPResponseFrame,
                      byte *OpenStatus,
                      byte Function,
                      ...);

// this function is used to output error information, it mimics fprintf
// functionality to an open trace file
extern int TracePrint(char *String, ...);

// this function gets the report general and discover information for
// a specific expander, the discover process begins at the subtractive
// boundary and progress downstream
static
struct TopologyTable *DiscoverExpander(byte PhyIdentifier,
                                       quadword SourceSASAddress,
                                       quadword DestinationSASAddress)
{
    struct TopologyTable *expander = 0;
    byte phyCount = 0;
    int error = 1;

    byte openStatus = OPEN_ACCEPT;

    // get the report general information for the expander
    SMPRequest(PhyIdentifier,
              SourceSASAddress,
              DestinationSASAddress,
              &SMPRequestFrame,

```



```

        &SMPResponseFrame,
        &openStatus,
        REPORT_GENERAL);

// don't worry about too much in the 'else' case for this example,
// production code needs to handle
if((openStatus == OPEN_ACCEPT) &&
    (SMPResponseFrame.FunctionResult == SMP_FUNCTION_ACCEPTED))
{
    if(SMPResponseFrame.Response.ReportGeneral.NumberOfPhys <=
        MAXIMUM_EXPANDER_PHYS)
    {
        // allocate space to retrieve the expander information
        expander = (struct TopologyTable *)
            calloc(1,
                sizeof(struct TopologyTable));

        // make sure we only do this if the allocation is successful
        if(expander)
        {
            // save the address of this expander
            expander->SASAddress = DestinationSASAddress;

            // copy the result into the topology table
            memcpy((void *)&(expander->Device),
                (void *)&SMPResponseFrame.Response.ReportGeneral,
                sizeof(struct SMPResponseReportGeneral));

            // now walk through all the phys of the expander
            for(phyCount = 0;
                (phyCount < expander->Device.NumberOfPhys);
                phyCount++)
            {
                // get the discover information for each phy
                SMPRequest(PhyIdentifier,
                    SourceSASAddress,
                    DestinationSASAddress,
                    &SMPRequestFrame,
                    &SMPResponseFrame,
                    &openStatus,
                    DISCOVER,
                    phyCount);

                // don't worry about the 'else' case for this example,
                // production code needs to handle
                if((openStatus == OPEN_ACCEPT) &&
                    (SMPResponseFrame.FunctionResult == SMP_FUNCTION_ACCEPTED))
                {
                    // clear the error flag
                    error = 0;

                    // copy the result into the topology table
                    memcpy((void *)&(expander->Phy[phyCount]),
                        (void *)&SMPResponseFrame.Response.Discover,
                        sizeof(struct SMPResponseDiscover));
                }
            }
            else if((openStatus == OPEN_ACCEPT) &&

```

```

        (SMPResponseFrame.FunctionResult == SMP_PHY_VACANT))
    {
        struct Discover *discover;

        discover = &SMPResponseFrame.Response.Discover.Result;

        // clear the error flag
        error = 0;

        // set the routing attribute and link rate to indicate that
        // the phy is not being used, this keeps it from being
        // included in the routing table information, these values
        // are not defined in the spec at this time, but are listed
        // as reserved values
        discover->NegotiatedPhysicalLinkRate = PHY_DOES_NOT_EXIST;
        discover->RoutingAttribute = PHY_NOT_USED;

        // copy the result into the topology table
        memcpy((void *)&(expander->Phy[phyCount]),
              (void *)&SMPResponseFrame.Response.Discover,
              sizeof(struct SMPResponseDiscover));
    }
    else
    {
        // if we had a problem on this link, then don't bother
        // to do anything else, production code needs to be more
        // robust...
        // for this simulation example, the addresses are
        // described as strings, so we can print them out...
        // not true for production code...
        TracePrint("\n"
                  "discover error, %02Xh at %s\n",
                  SMPResponseFrame.FunctionResult,
                  (char *)&DestinationSASAddress);

        // something happened so just bailout on this expander
        error = 1;

        // release the memory we allocated for this...
        free(expander);
        expander = 0;
        break;
    }
}
}
}
// the assumptions we made were exceeded, need to bump simulation
// limits...
else
{
    TracePrint("\n"
              "report general error"
              ", NumberOfPhys %d exceeded limit %d on %s\n",
              expander->Device.NumberOfPhys,
              MAXIMUM_EXPANDER_PHYS,
              (char *)&DestinationSASAddress);
}
}

```

```

    }
    else
    {
        // if we had a problem getting report general for this expander,
        // something is wrong, can't go any further down this path...
        // production code needs to be more robust...
        // for this simulation example, the addresses are
        // described as strings, so we can print them out...
        // not true for production code...
        TracePrint("\n"
            "report general error, open %02Xh result %02Xh at %s\n",
            openStatus,
            SMPResponseFrame.FunctionResult,
            (char *)&DestinationSASAddress);
    }

    // the expander pointer is the error return, a null indicates something
    // bad happened...
    return(expander);
} // DiscoverExpander

// this routine searches upstream for the subtractive boundary that defines
// the edge expander device set
static
struct TopologyTable *FindBoundary(byte PhyIdentifier,
                                   quadword SourceSASAddress,
                                   struct TopologyTable *Expander,
                                   struct TopologyTable **DeviceSet)
{
    struct TopologyTable *expander = Expander;
    struct TopologyTable *nextExpander;

    struct Discover *discover;

    byte phyCount;

    int error = 0;
    int foundSubtractivePort = 0;

    quadword subtractiveSASAddress;
    byte attachedPhyIdentifier;

    // make sure the device set link is initialized
    *DeviceSet = 0;

    // outer loop searches for subtractive phys and finds the SAS addresses
    // connected to them, validates that the subtractive phys all resolve
    // to the same expander address, then moves upstream searching for the
    // edge expander device set boundary
    do
    {
        // initialize the subtractive address, a zero value is not valid
        subtractiveSASAddress = 0;
        attachedPhyIdentifier = 0;

        // walk through all the phys of this expander
        for(phyCount = 0;

```

```

        (phyCount < expander->Device.NumberOfPhys);
        phyCount++)
    {
        // this is just a pointer helper
        discover = &(expander->Phy[phyCount].Result);

        // look for phys with edge or fanout devices attached...
        if((discover->RoutingAttribute == SUBTRACTIVE) &&
            ((discover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
             (discover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
        {
            // make sure all the subtractive phys point to the same address
            // when we are connected to an expander device
            if(!subtractiveSASAddress)
            {
                subtractiveSASAddress = discover->AttachedSASAddress;
                attachedPhyIdentifier = discover->AttachedPhyIdentifier;
                foundSubtractivePort = 1;
            }
            // the addresses don't match... problem...
            else if(subtractiveSASAddress !=
                    discover->AttachedSASAddress)
            {
                // production code needs to deal with this better, for this
                // example, the SASAddresses are assumed to strings
                // so just print out the error information
                TracePrint("\n"
                           "topology error, diverging subtractive phys"
                           ", '%s' != '%s' \n",
                           (char *)&subtractiveSASAddress,
                           (char *)&discover->AttachedSASAddress);
                error = 1;
                break;
            }
        }
    }

    // if no error, then decide if we need to go upstream or stop
    if(!error)
    {
        // if we have a subtractive address then go upstream to see
        // if it is part of the edge expander device set
        if(subtractiveSASAddress)
        {
            // get the discover information
            nextExpander = DiscoverExpander(PhyIdentifier,
                                           SourceSASAddress,
                                           subtractiveSASAddress);

            // if we successfully got the information from the next
            // expander then proceed upstream...
            if(nextExpander)
            {
                struct Discover *discover;

                // this is just a pointer helper
                discover = &(nextExpander->Phy[attachedPhyIdentifier].Result);
            }
        }
    }
}

```

```

        // check to see if we are connected to the subtractive
        // port of the next expander, if we are then we have two
        // expander device sets connected together, stop here
        // and save the address of next expander in device set,
        // the return is expander
        if(discover->RoutingAttribute == SUBTRACTIVE)
        {
            *DeviceSet = nextExpander;
            break;
        }
        // go ahead and continue upstream looking for the boundary
        else
        {

            // release the memory we allocated for this
            free(expander);

            // move upstream to the next expander
            expander = nextExpander;
        }
    }
    // if there are no more upstream expanders stop here...
    else
    {
        break;
    }
}
// if we did not get a subtractive address this time around, stop
else
{
    // if we did find a subtractive port on a previous pass,
    // then return with expander pointing to the last device
    // with the subtractive port
    if(foundSubtractivePort)
    {
        break;
    }
    // if we never found a subtractive port, then return with a
    // null indicating there are no subtractive phys, don't free
    // the memory, because it is still in use by the calling routine
    else
    {
        expander = 0;
    }
}
}
// if there was an error make sure we return a null expander pointer
else
{
    // to get here, we had to see more than one subtractive phy that
    // connect to different SAS addresses, this is a topology error
    // do cleanup on any memory allocated if necessary
    if((expander != Expander) &&
        (expander != *DeviceSet))
    {
        // release the memory we allocated for this and make sure

```

```

        // we return a null
        free(expander);
        expander = 0;
    }
}
}
while(!error &&
       expander &&
       subtractiveSASAddress);

// on return expander contains the subtractive boundary expander
// or a null indicating there were no subtractive phys,
// or a null indicating there was an error
return(expander);
} // FindBoundary

// find the table structure associated with a specific SAS address
static
struct TopologyTable *FindExpander(struct TopologyTable *Expander,
                                   quadword SASAddress)
{
    // walk the list of expanders, when we find the one that matches, stop
    while(Expander)
    {
        // do the SASAddresses match
        if(SASAddress == Expander->SASAddress)
        {
            break;
        }

        Expander = Expander->Next;
    }

    return(Expander);
} // FindExpander

// this routine searches the subtractive phys for the upstream
// expander address
static
int UpstreamExpander(struct TopologyTable *Expander,
                    quadword *SASAddress,
                    byte *PhyIdentifier)
{
    struct Discover *discover;

    byte phyCount;

    int found = 0;

    // walk through all the phys of this expander, searching for subtractive
    // phys return the SASAddress and PhyIdentifier for the first subtractive
    // phy encountered, they are all be the same if they have anything
    // attached
    for(phyCount = 0;
        (phyCount < Expander->Device.NumberOfPhys);
        phyCount++)
    {

```

```

    // this is just a pointer helper
    discover = &(Expander->Phy[phyCount].Result);

    // look for phys with edge or fanout devices attached...
    if((discover->RoutingAttribute == SUBTRACTIVE) &&
        ((discover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
         (discover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
    {
        *SASAddress = discover->AttachedSASAddress;
        *PhyIdentifier = discover->AttachedPhyIdentifier;
        found = 1;
        break;
    }
}

return(found);
} // UpstreamExpander

// this routine determines whether a SAS address is directly attached to
// an expander
static
int DirectAttached(struct TopologyTable *Expander,
                  quadword SASAddress)
{
    int direct = 0;
    byte phyCount;

    for(phyCount = 0;
        phyCount < Expander->Device.NumberOfPhys;
        phyCount++)
    {
        // did we find the address attached locally
        if(SASAddress ==
            Expander->Phy[phyCount].Result.AttachedSASAddress)
        {
            direct = 1;
            break;
        }
    }

    return(direct);
} // DirectAttached

// this route determines whether a SAS address is already in the route table
static
int AlreadyInTable(struct TopologyTable *Expander,
                  quadword *SASAddress,
                  byte PhyIdentifier)
{
    int inTable = 0;
    int routeIndex;

    for(routeIndex = 0;
        routeIndex < Expander->Device.ExpanderRouteIndexes;
        routeIndex++)
    {
        struct RouteTableEntry *entry =

```

```

        &Expander->RouteTable[PhyIdentifier][routeIndex];

    if(entry->RoutedSASAddress ==
        SASAddress)
    {
        inTable = 1;
        break;
    }
}

return(inTable);
} // AlreadyInTable

// this routine determines whether the SAS address, can be optimized out
// of the route table
static
int QualifiedAddress(struct TopologyTable *Expander,
                    byte PhyIdentifier,
                    quadword SASAddress,
                    byte RoutingAttribute,
                    byte *DisableRouteEntry)
{
    int qualified = 1;
    word routeIndex;

    if(DiscoverAlgorithm == SAS_UNIQUE_LEVEL_DESCENT)
    {
        if(((RoutingAttribute == SUBTRACTIVE) ||
            (RoutingAttribute == TABLE)) &&
            ((SASAddress == 0) ||
            (SASAddress == Expander->SASAddress) ||
            !memcmp(SASAddress, Expander->SASAddress, 8) ||
            DirectAttached(Expander, SASAddress) ||
            AlreadyInTable(Expander, SASAddress, PhyIdentifier)))
        {
            // if we made it here then we are disqualifying the address,
            // rules 2, 3, 4, and 5
            qualified = 0;
        }

        // if qualified, but the address is zero, then make sure it is
        // disabled
        if(qualified &&
            (SASAddress == 0))
        {
            *DisableRouteEntry = DISABLE;
        }
    }

    return(qualified);
} // QualifiedAddress

// this function is the configuration cycle from the current expander to
// the hub expander
static
int ConfigureExpander(byte PhyIdentifier,

```



```

        quadword SourceSASAddress,
        struct TopologyTable *HubExpander,
        struct TopologyTable *ThisExpander)
{
    struct TopologyTable *thisExpander = ThisExpander;
    struct TopologyTable *expander = ThisExpander;
    struct TopologyTable *configureExpander;

    struct Discover *discover;

    quadword upstreamSASAddress = 0;
    byte upstreamPhyIdentifier = 0;

    byte phyIndex;
    word routeIndex;
    byte openStatus = OPEN_ACCEPT;

    int error = 0;

    do
    {
        // move upstream from here to find the expander table to configure with
        // information from "thisExpander"
        if(!UpstreamExpander(thisExpander,
                             &upstreamSASAddress,
                             &upstreamPhyIdentifier))
        {
            break;
        }

        if(upstreamSASAddress)
        {
            // get the expander associated with the upstream address
            configureExpander = FindExpander(HubExpander,
                                             upstreamSASAddress);

            // if we found an upstream expander, then program it's route
            // table
            if(configureExpander)
            {
                byte disableRouteEntry = ENABLED;

                for(phyIndex = 0;
                    phyIndex < configureExpander->Device.NumberOfPhys;
                    phyIndex++)
                {
                    if(configureExpander->Phy[phyIndex].Result.AttachedSASAddress
                       == thisExpander->SASAddress)
                    {
                        // loop through all the phys of the attached expander
                        for(routeIndex = 0;
                            ((routeIndex <
                               expander->Device.NumberOfPhys) &&
                             (configureExpander->RouteIndex[phyIndex] <
                              configureExpander->Device.ExpanderRouteIndexes));
                            routeIndex++)
                        {

```

```

discover = &(amp;expander->Phy[routeIndex].Result);

// assume the route entry is enabled
disableRouteEntry = ENABLED;

// make sure if the attached device type is 0, that the
// SAS address is 0, to simplify the qualified address
// check
if(discover->AttachedDeviceType == 0)
{
    discover->AttachedSASAddress = 0;
}

// check to see if the address needs to be configured
// in the route table, this decision is based on the
// optimization flag
if(QualifiedAddress(configureExpander,
                    phyIndex,
                    discover->AttachedSASAddress,
                    discover->RoutingAttribute,
                    &disableRouteEntry))
{
    word index = configureExpander->RouteIndex[phyIndex];

    struct RouteTableEntry *entry =
        &configureExpander->RouteTable[phyIndex][index];

    // configure the route indexes for the expander
    // with the attached address information
    SMPRequest(PhyIdentifier,
              SourceSASAddress,
              configureExpander->SASAddress,
              &SMPRequestFrame,
              &SMPResponseFrame,
              &openStatus,
              CONFIGURE_ROUTE_INFORMATION,
              index,
              phyIndex,
              disableRouteEntry,
              discover->AttachedSASAddress);

    if((openStatus != OPEN_ACCEPT) ||
        (SMPResponseFrame.FunctionResult !=
         SMP_FUNCTION_ACCEPTED))
    {
        error = 1;
        break;
    }

    // add the address to the internal copy of the
    // route table, if successfully configured
    entry->RoutedSASAddress =
        discover->AttachedSASAddress;

    // increment the route index for this phy

```

```

        configureExpander->RouteIndex[phyIndex]++;
    }
}

// move upstream
thisExpander = configureExpander;

} while(!error &&
        thisExpander &&
        upstreamSASAddress);

return(error);
} // ConfigureExpander

// this determines if the expander has a table routed phy attached to the
// SAS address and phyIndex provided
static
int CheckForTableAttribute(struct TopologyTable *Expander,
                          byte PhyIndex,
                          quadword SASAddress)
{
    int table = 0;
    byte phyCount;

    for(phyCount = 0;
        phyCount < Expander->Device.NumberOfPhys;
        phyCount++)
    {
        // did we find the address attached locally
        if((SASAddress ==
            Expander->Phy[phyCount].Result.AttachedSASAddress) &&
            (phyCount == PhyIndex) &&
            (Expander->Phy[phyCount].Result.RoutingAttribute == TABLE))
        {
            table = 1;
            break;
        }
    }

    return(table);
} // CheckForTableAttribute

// this discovers then configures as necessary the expanders it finds
// within the SAS domain that are "downstream"
static
struct TopologyTable *DiscoverAndConfigure(byte PhyIdentifier,
                                          quadword SourceSASAddress,
                                          struct TopologyTable *HubExpander,
                                          struct TopologyTable **DeviceSet)
{
    struct TopologyTable *currentExpander = HubExpander;
    struct TopologyTable *nextExpander;

```

```

struct Discover *currentDiscover;

quadword sasAddress;
byte phyIndex;

int error = 0;

// this is a level descent traversal with a configuration stage
// at each transition to a new level, if a configuration is required
// by the expander

// the discover process moves forward through the topology, but the
// configuration process stays anchored at the hub of the
// topology, meaning the fanout expander, or the top most subtractive
// edge expander device
// this ensures that as each new expander is added to the
// topology table list, it is in the configuration chain

while(!error &&
      currentExpander)
{
    // start at phy 0
    phyIndex = 0;

    // walk through all the phys of the current expander looking for
    // new expanders to add to the topology table
    do
    {
        // this is just a pointer helper
        currentDiscover = &(amp;currentExpander->Phy[phyIndex].Result);

        // look for phys with edge or fanout devices attached...
        if((currentDiscover->RoutingAttribute == TABLE) &&
           ((currentDiscover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
            (currentDiscover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE))
        )
        {
            struct TopologyTable *thisExpander = currentExpander;
            struct TopologyTable *previousExpander = currentExpander;

            // check to see if we already have the address information
            // in our expander list
            while(thisExpander)
            {
                // if we do, then stop here
                if(currentDiscover->AttachedSASAddress ==
                   thisExpander->SASAddress)
                {
                    break;
                }

                // setup the pointer references
                previousExpander = thisExpander;
                thisExpander = thisExpander->Next;
            }

            // if we did not have the expander in our list, then get

```

```

// the information
if(!thisExpander)
{
    // discover all the details about the attached expander
    // and insert into the master list
    thisExpander =
        DiscoverExpander(PhyIdentifier,
                        SourceSASAddress,
                        currentDiscover->AttachedSASAddress);

    // if we got the discover information, then add it to the
    // list
    if(thisExpander)
    {
        if(CheckForTableAttribute(thisExpander,
                                phyIndex,
                                currentDiscover->SASAddress))
        {
            // output an error message
            TracePrint("\n"
                    "error table phys connected together\n");
        }
        else
        {
            previousExpander->Next = thisExpander;

            // go through the configure cycle progressively ascending
            // to each expander starting at "thisExpander"
            ConfigureExpander(PhyIdentifier,
                            SourceSASAddress,
                            HubExpander,
                            thisExpander);
        }
    }
}

// look for subtractive phys with edge or fanout devices attached...
else
if(DeviceSet &&
    (currentDiscover->RoutingAttribute == SUBTRACTIVE) &&
    ((currentDiscover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
    (currentDiscover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
{
    if(*DeviceSet == 0)
    {
        struct TopologyTable *thisExpander = currentExpander;
        struct TopologyTable *previousExpander = currentExpander;

        // check to see if we already have the address information
        // in our expander list
        while(thisExpander)
        {
            // if we do, then stop here
            if(!memcmp(&currentDiscover->AttachedSASAddress,
                    &thisExpander->SASAddress,

```

```

        8))
    {
        break;
    }

    // setup the pointer references
    previousExpander = thisExpander;
    thisExpander = thisExpander->Next;
}

// if we did not have the expander in our list, then get
// the information
if(!thisExpander)
{
    // discover all the details about the attached expander
    // and insert into the master list
    thisExpander =
        DiscoverExpander(PhyIdentifier,
                        SourceSASAddress,
                        currentDiscover->AttachedSASAddress);

    // if we got the discover information, then set it as the
    // other device set
    if(thisExpander)
    {
        *DeviceSet = thisExpander;
    }
}
}

// move to the next phy on this expander
phyIndex++;

} while(phyIndex <
        currentExpander->Device.NumberOfPhys);

// cycle to the next expander to discover
currentExpander = currentExpander->Next;
}

// return the top of expander list
return(HubExpander);
} // DiscoverAndConfigure

// this routine appends the leaf to the tree domain
static
void ConcatenateSASDomains(struct TopologyTable *Tree,
                          struct TopologyTable *Leaf)
{
    while(Tree)
    {
        if(Tree->Next == 0)
        {
            Tree->Next = Leaf;
            break;
        }
    }
}

```

```

        Tree = Tree->Next;
    }
} // ConcatenateSASDomains

// validate the route table entries for all expanders
static
int ValidateRouteTables(byte PhyIdentifier,
                        quadword SourceSASAddress,
                        struct TopologyTable *Expander,
                        int SASCompatibility)
{
    struct ReportRouteInformation *route;

    // buffers used to request and return SMP data
    struct SMPRequest request = { 0 };
    struct SMPResponse response = { 0 };

    byte phyIndex;
    word routeIndex;

    byte openStatus = OPEN_ACCEPT;

    int valid = 1;

    if(SASCompatibility == SAS_10_COMPATIBLE)
    {
        // this is just a pointer helper
        route = &(response.Response.ReportRouteInformation.Result);

        // walk the list of expanders
        while(valid &&
              Expander)
        {
            if(Expander->Device.ConfigurableRouteTable)
            {
                struct RouteTableEntry *entry;

                word expanderRouteIndexes;

                _swab( (char *)&Expander->Device.ExpanderRouteIndexes,
                      (char *)&expanderRouteIndexes,
                      sizeof(word) );

                for(phyIndex = 0;
                    (valid &&
                     (phyIndex < Expander->Device.NumberOfPhys));
                    phyIndex++)
                {
                    // loop through all the phys of the expander
                    for(routeIndex = 0;
                        (valid &&
                         ((routeIndex <
                           Expander->RouteIndex[phyIndex]) &&
                          (routeIndex <
                           expanderRouteIndexes))));
                        routeIndex++)

```

```

    {
        openStatus = OPEN_ACCEPT;

        // report the route indexes for the expander
        SMPRequest(PhyIdentifier,
                   SourceSASAddress,
                   Expander->SASAddress,
                   &request,
                   &response,
                   &openStatus,
                   REPORT_ROUTE_INFORMATION,
                   routeIndex,
                   phyIndex);

        if((openStatus != OPEN_ACCEPT) ||
           (response.FunctionResult !=
            SMP_FUNCTION_ACCEPTED))
        {
            break;
        }

        entry = &(Expander->RouteTable[phyIndex][routeIndex]);

        if((memcmp(&entry->RoutedSASAddress,
                   &route->RoutedSASAddress,
                   8)) ||
           (entry->ExpanderRouteEntryDisabled !=
            route->ExpanderRouteEntryDisabled))
        {
            valid = 0;
        }
    }
}

    Expander = Expander->Next;
}

```

```

    return(valid);
} // ValidateRouteTables

```

```

// validate that the change count for the hub expander is still the same
// as when we started

```

```

static
int ChangeCount(byte PhyIdentifier,
                quadword SourceSASAddress,
                struct TopologyTable *Expander)
{
    // buffers used to request and return SMP data
    struct SMPRequest request = { 0 };
    struct SMPResponse response = { 0 };

    int change = 0;

    byte openStatus = OPEN_ACCEPT;

```



```

// get the report general information for the expander
SMPRequest(PhyIdentifier,
           SourceSASAddress,
           Expander->SASAddress,
           &request,
           &response,
           &openStatus,
           REPORT_GENERAL);

// don't worry about too much in the 'else' case for this example,
// production code needs to handle
if((openStatus == OPEN_ACCEPT) &&
    (response.FunctionResult == SMP_FUNCTION_ACCEPTED))
{
    if(response.Response.ReportGeneral.ExpanderChangeCount !=
        Expander->Device.ExpanderChangeCount)
    {
        change = 1;
    }
}

return(change);
} // ChangeCount

// delete all the topology information and start over
void DeleteSASDomain(struct TopologyTable **Expander)
{
    struct TopologyTable *expander = *Expander;
    struct TopologyTable *nextExpander = 0;

    // walk the list of expanders
    while(expander)
    {
        nextExpander = expander->Next;

        free(expander);

        expander = nextExpander;
    }

    *Expander = 0;
} // DeleteSASDomain

// the application client for the initiator device makes a call to
// this function to begin the discover process...
// to simplify the setup for the simulation, the DiscoverProcess gets
// the Initiator number to allow multiple initiators...
void DiscoverProcess(byte Initiator,
                   byte PhyIdentifier)
{
    // check to see if an expander is attached
    if((Attached[Initiator].DeviceType == EDGE_EXPANDER_DEVICE) ||
        (Attached[Initiator].DeviceType == FANOUT_EXPANDER_DEVICE))
    {
        struct TopologyTable *connectedExpander;

        // get some local variables to keep things simple

```

```

quadword sourceSASAddress = Iam[Initiator].SASAddress;
quadword destinationSASAddress = Attached[Initiator].SASAddress;

// expander is attached, so begin by getting the information about
// the connected expander
connectedExpander = DiscoverExpander(PhyIdentifier,
                                     sourceSASAddress,
                                     destinationSASAddress);

// make sure we get the information from the expander
if(connectedExpander)
{
    struct TopologyTable *thisDeviceSet;
    struct TopologyTable *attachedDeviceSet = 0;

    int redoDiscover = 0;
    int changed = 0;

    do
    {
        // go upstream on the subtractive phys until we discover that we
        // are attached to another subtractive phy or a fanout expander
        // then begin the discover process from that point, this works
        // because any new address that we find naturally moves
        // upstream due to the subtractive addressing method
        // if during the discover cycle, it is determined that there
        // are two device sets connected, then a second discover
        // and configuration cycle is required for the other device set
        thisDeviceSet = FindBoundary(PhyIdentifier,
                                    sourceSASAddress,
                                    connectedExpander,
                                    &attachedDeviceSet);

        // set the root for the domain as the subtractive boundary
        if(thisDeviceSet)
        {
            // output a little information about the subtractive boundary
            TracePrint("subtractive boundary at %s\n",
                      (char *)&thisDeviceSet->SASAddress);

            SASDomain[Initiator] = thisDeviceSet;
        }
        // if there was no subtractive boundary, then the root is the
        // expander connected to the initiator
        else
        {
            // output a little information about the subtractive boundary
            TracePrint("connected expander at %s\n",
                      (char *)&connectedExpander->SASAddress);

            SASDomain[Initiator] = connectedExpander;
        }

        // begin the discover and configuration cycle
        DiscoverAndConfigure(PhyIdentifier,
                             sourceSASAddress,
                             SASDomain[Initiator],

```

```

        &attachedDeviceSet);

// if two device sets are connected, then the attached device set
// has to be discovered and configured
if(attachedDeviceSet)
{
    // output a little information about the attached device set
    TracePrint("attached device set at %s\n",
        (char *)&attachedDeviceSet->SASAddress);

    // discover and configure the attached device set
    DiscoverAndConfigure(PhyIdentifier,
        sourceSASAddress,
        attachedDeviceSet,
        0);

    // put the domains together
    ConcatenateSASDomains(SASDomain[Initiator],
        attachedDeviceSet);
}

// if the change count of the top most expander is different
// from when we started, then the topology was not stable
// so do the discover again
changed = ChangeCount(PhyIdentifier,
    sourceSASAddress,
    SASDomain[Initiator]);

// if we used the route table optimization,
// check the route tables for each expander phy, if they are
// incorrect then change back to the original discover algorithm
// and redo discover, continue to use the original algorithm
// for any new discover
if(!changed &&
    (DiscoverAlgorithm == SAS_UNIQUE_LEVEL_DESCENT) &&
    !ValidateRouteTables(PhyIdentifier,
        sourceSASAddress,
        SASDomain[Initiator],
        SASCompatibility))
{
    redoDiscover = 1;

    // if the change count of the top most expander is the same
    // as when we started the validation of the route tables
    // then the topology was stable, so change the algorithm
    // before the rediscover
    if(!ChangeCount(PhyIdentifier,
        sourceSASAddress,
        SASDomain[Initiator]))
    {
        DiscoverAlgorithm = SAS_SIMPLE_LEVEL_DESCENT;
    }

    // delete everything allocated and start over
    DeleteSASDomain(&SASDomain[Initiator]);
}

```

```
        } while(redoDiscover ||
              changed);
    }
} // DiscoverProcess
```

Annex M (informative)

SAS icons

A SAS icon should be included on or near all connectors used by devices compliant with this standard.

NOTE 116 - Contact the SCSI Trade Association at <http://www.scsita.org> for versions of the SAS icons in various graphics formats.

Figure M.1 shows the primary SAS icon.

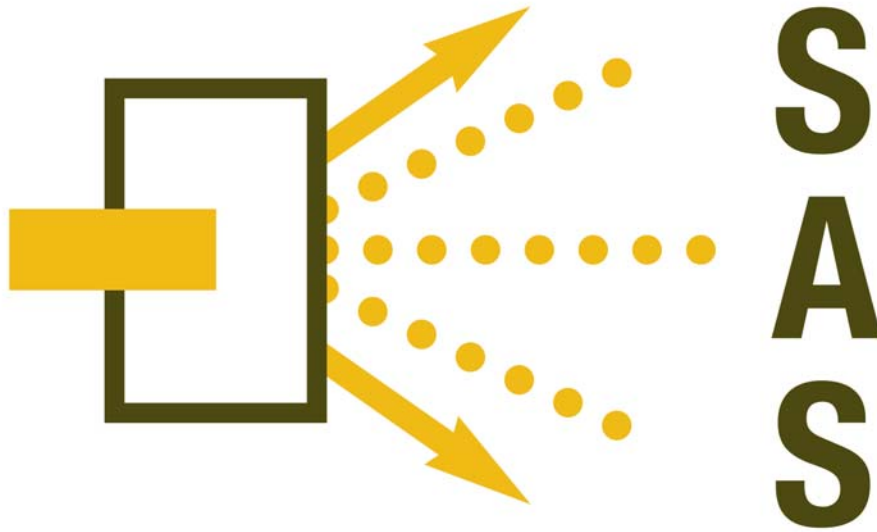


Figure M.1 — SAS primary icon

Figure M.2 shows an alternate SAS icon that may be used instead of the primary SAS icon when the area for the icon is small.

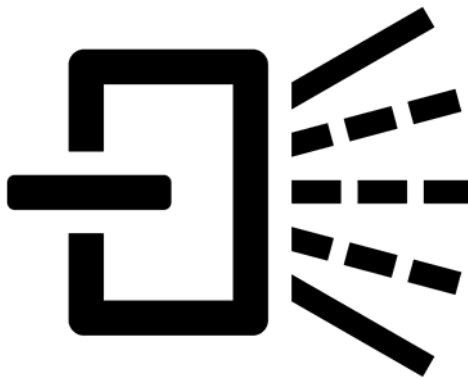


Figure M.2 — SAS alternate icon

Figure M.3 shows an alternate SAS icon with the SAS abbreviation letters alongside.

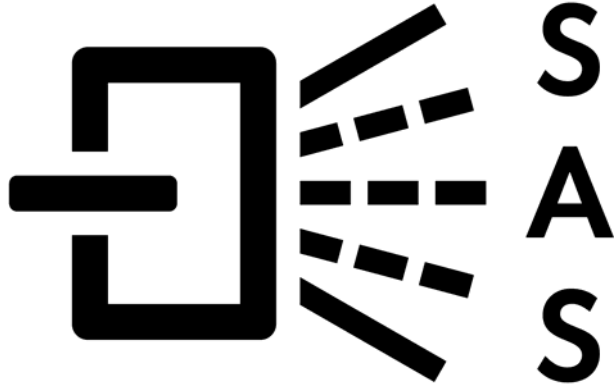


Figure M.3 — SAS alternate icon with SAS letters

