Working Draft American National Standard

Project T10/1562-D

Revision 3
21 November 2002

Information technology - Serial Attached SCSI (SAS)

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 150801
PO Box 692000
Houston, TX 77269-2000
USA

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

Reference number
ISO/IEC XXXXX-XXX : 200x
ANSI INCITS.*:200x**

Printed Thursday, November 21, 2002 2:24 PM

Points of Contact

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

T10 Chair
John B. Lohmeyer
LSI Logic
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
USA

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

T10 Vice-Chair
George O. Penokie
IBM Corporation
MS: 2C6
3605 Highway 52 N
Rochester, MN 55901
USA

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

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

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 (SAS)

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) protocol, which defines transmission of SCSI protocol over a Serial ATA compatible physical layer and defines addressing of multiple target devices for the Serial ATA protocol.

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, no such claims have been made. 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 © 2002 by Information Technology Industry Council (ITI)
All rights reserved.

No part of this publication may by 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

Revision Information

1 Revision history

1.1 Revision 02-157r0 (25 April 2002)

First release as T10 proposal 02-157r0.

1.2 Revision 02-157r1 (12 May 2002)

Change bars from 02-157r0. Incorporates these changes (approved by May T10 plenary):

a) 02-183r0 Minutes from the SAS PHY study group 1-2 May 2002 (Editorial corrections)
b) 02-168r1 SAS ALIGN primitives (Bill Galloway)
c) 02-165r1 SAS SMP Report Manufacturer Info (Ron Roberts)
d) 02-167r1 SAS ERROR primitive (Bill Galloway)
e) Miscellaneous editorial and technical corrections from the SAS protocol study group 29-30 April 2002

1.3 Revision sas-r00 (12 May 2002)

Incorporates these changes:

a) Removes changes bars and introductory letter
b) Corrections to SL state machine from George Penokie (without change bars)
c) Corrected S12 and S13 to be BT+ and BT- in the device plug connector

1.4 Revision sas-r00a (27 May 2002)

Change bars are from sas-r00. Incorporates these changes (not yet approved by a T10 plenary):

a) 02-169r2 SAS XFER_RDY rules (Bill Galloway) [14 May 2002 SAS call]
b) 02-170r1 SAS wide link rules (Bill Galloway) [14 May 2002 SAS call]
c) Editorial changes from 02-199r0 Minutes of 21 May 2002 conference call [14 May 2002 SAS call]
d) Editorial changes from emails

1.5 Revision sas-r00b (14 June 2002)

Change bars are still from sas-r00. Incorporates these changes (not yet approved by a T10 plenary):

a) Editorial and minor technical changes from emails and from 6/6 SAS WG meeting
b) Require rotating through all ALIGNs [6 June 2002 SAS WG]
c) SMP remove DEVICE NAME from REPORT GENERAL function [6 June 2002 SAS WG]
d) 02-179r1 SAS SMP PHY MARGIN CONTROL function (Ron Roberts) [6 June 2002 SAS WG]
e) 02-211r1 SAS address frame field checking (Rob Elliott) [6 June 2002 SAS WG]
f) 02-212r1 SAS SSP RESPONSE IU format (Bill Galloway) [6 June 2002 SAS WG]
g) 02-214r1 SAS SSP hash transport layer checking (Bill Galloway) [6 June 2002 SAS WG]
h) 02-217r0 SAS OPEN_REJECT priorities (Jim Coomes) [6 June 2002 SAS WG]
i) 02-218r0 SAS Remove far-end retimed loopback (Jim Coomes) [6 June 2002 SAS WG]
j) 02-203r1 SAS request confirmation correlation across layers (John Worden) - only partially incorporated [6 June 2002 SAS WG]
k) 02-200r1 Minutes from SAS protocol WG meeting 5-7 June 2002 [6 June 2002 SAS WG]

1.6 Revision sas-r00c (15 June 2002)

Change bars are still from sas-r00. Incorporates these changes (not yet approved by a T10 plenary):

a) Changes to the SAS phy state machine (assumes 02-198 will be accepted)
b) More work on 02-203r1 incorporation (still not done)

1.7 Revision sas-r00d (8 July 2002)

Change bars are still from sas-r00. Incorporates these changes (not yet approved by a T10 plenary):

a) Removed device name from REPORT GENERAL [5 June 2002 SAS WG]
b) 02-204r1 SAS SSP Dealing with extra ACK/NAK primitives (Mark Evans) [18 June 2002 SAS call]
c) Removed Protocol Violation and Failed Frame confirmations from SMP_SFR1 state [26 June 2002 SAS WG]
d) Changed “device name” to “SAS address” [26 June 2002 SAS WG]
e) 02-196r2 SAS SSP AEN (George Penokie) [26 June 2002 SAS WG]
f) 02-201r3 SAS SSP Transmission of a DONE primitive to close an SSP connection (Mark Evans) [26 June 2002 SAS WG]
g) 02-205r3 SAS connection behavior when receiving a BREAK primitive (Mark Evans) [26 June 2002 SAS WG]
h) Moved SPC-3 protocol-specific changes annex into 02-246r0
i) Moved SAM-3 protocol-specific changes into 02-245r0

1.8 Revision sas-r00e (11 July 2002)

Change bars are still from sas-r00. Incorporates these changes (not yet approved by a T10 plenary):

a) Changed “dword” to “data dword” where appropriate; clarified type of BREAK and AIP primitives
b) Changed “send” to “transmit” whenever referring to dwords on the wire (send is for state machine communication). Use “send” and “receive” for state machine signal passing (requests, confirmations, and parameters) rather than “indicate.”
c) 02-197r2 SAS Identification Sequence state diagrams (George Penokie) [18 June 2002 SAS call]
d) 02-219r3 SAS bit order for scrambling and CRC (Jim Coomes) [9 July 2002 SAS call]
e) Changed frame validity wording to “bytes between SOF and EOF less than 28, or bytes after SOF greater than 1052”
f) Marked all fill bytes as “vendor-specific”

1.9 Revision sas-r00f (16 July 2002)

Change bars are still from sas-r00. Incorporates these changes:

a) Broke “application layer” into SCSI layer, ATA layer, and management layer. Later review of SAM-3 letter ballot comments indicates SAM uses “application layer” and SAM-3 is not going to change to “SCSI layer”, so this may be best undone. [15 July 2002 SAS WG]
b) 02-198r6 SAS OOB timing [16 July 2002 SAS WG]
c) Removed 02-196r2 AEN on SAS (George Penokie) due to lack of support from the CAP WG for the REPORT AENS well-known LUN addition [18 July 2002 T10 Plenary]
d) 02-210r2 Changes for driver and receiver electrical characteristics (Russ Brown) (as incorporated into 02-289r0) [8 July 2002 and 15 July 2002 SAS PHY WG]
e) 02-215r1 SAS SSP Credit not ready addition to state machine (Bill Galloway) [16 July 2002 SAS WG]
f) 02-216r5 SAS SMP transport layer (George Penokie) [15 July 2002 SAS WG]
g) 02-230r2 SAS SSP initiator transport layer (Mark Evans) [15 July 2002 SAS WG]
h) 02-288r0 Minutes of SAS PHY WG - OOB Idle time definition [16 July 2002 SAS PHY WG]
i) 02-289r0 SAS Phy group revisions to section 5 [16 July 2002 SAS PHY WG]

1.10 Revision sas-r01 (24 July 2002)

This revision has no change bars, reflecting that all proposals were approved by the July T10 plenary. It incorporates these changes:

a) 02-256r2 SAS Pathway recovery modification (Tim Hoglund) [16 July 2002 SAS WG] (except port layer changes, which will be covered by 02-202)

1.11 Revision sas-r01a (28 August 2002)

This revision has no change bars. It incorporates these changes:

a) Converted to Adobe FrameMaker from Microsoft Word, spurred by continued destructive crashes by Word 2000 and Word XP (thanks to George Penokie). All Microsoft Word drawings converted to Microsoft Visio format, embedded as links in FrameMaker.
b) Minor editorial changes.

1.12 Revision sas-r01b (6 September 2002)

This revision has change bars from sas-r01a. It incorporates these changes:

a) 02-231r2 CRC and scrambling sample code (Jim Coomes) [16 July 2002 SAS WG][already approved by T10 plenary]
b) 02-259r1 SAS command ID removal (Brian Day) [23 August 2002 SAS WG]
c) 02-279r4 SAS initiator-based configuration (Steve Fairchild) changes 1-13 only [3 September 2002 SAS call]
d) 02-322r0 Minutes from SAS protocol WG 21-23 August 2002 - motions [23 August 2002 SAS WG]
   A) that AEN be removed from SAS
   B) that SAS be modified to require targets to not send a response frame for a command until all data associated with any outstanding XFER_RDY for that command have been received
   C) that the open originator shall start rate matching on the next dword after the OPEN address frame has been transmitted
e) 02-336r0 Minutes of SAS protocol call - motions [3 September 2002 SAS call]
   A) that the discover function be numbered 10h instead of 00h and that the PHYSICAL LINK RATE in the IDENTIFY address frame be marked ‘ignored’

1.13 Revision sas-r01c (23 September 2002)

This revision has change bars from sas-r01a. It incorporates these changes:

a) 02-300r2 SAS OOB and DWS state machines update (Thomas Grieff) state machine changes [10 September 2002 SAS & SAS PHY WGs]
b) 02-300r2 SAS OOB and DWS state machines update (Thomas Grieff and George Penokie) editorial changes before state machines
c) 02-307r2 SAS SSP transport layer state machines (Mark Evans)[10 September 2002 SAS WG]
a) 02-335r2 SAS multiple connections on wide ports (George Penokie)[10 September 2002 SAS WG]
b) 02-338r1 SAS SCSI Phy Error Log log page (Mark Evans)[10 September 2002 SAS WG]
c) 02-347r1 SAS CRC bit order figure replacement (Jim Coomes)[10 September 2002 SAS WG]
d) 02-350r1 SAS idle at start of OOB signals (George Penokie)[10 September 2002 SAS WG]
e) 02-352r0 SAS Clarification of area of impedance dip (Alvin Cox) [10 September 2002 SAS PHY WG]
f) 02-354r0 SAS definition/note for skew (Alvin Cox) [10 September 2002 SAS PHY WG]
g) 02-355r1 SAS expander update (Tim Hoglund) [10 September 2002 SAS WG]
h) 02-342r0 Minutes of SAS protocol WG - 9-10 September 2002 [10 September 2002 SAS WG]
   A) change the Rate Change Delay from 480 UIs to 750,000 UIs
i) Rewrote IR state machine to include idle states and completed states and rework handling of enable/disable signals from phy layer
j) Reworked SL state machine handling of enable/disable signals from IR state machine
k) Added an SMP READ MARGIN SETTINGS function to fulfill the original goal of 02-179r1

1.14 Revision sas-r02 (23 September 2002)

This revision has no change bars, reflecting that all proposals were approved by the September T10 plenary.

1.15 Revision sas-r02a (7 October 2002)

This revision has change bars from sas-r02. It incorporates comments from the SAS Editor’s meeting on 3-4 October 2002, mostly through chapter 5 (Physical).

1.16 Revision sas-r02b (19 October 2002)

This revision has change bars from sas-r02. It incorporates these changes:

a) 02-276r3 SAS combined request-confirmation list (John Worden)[24 September 2002 SAS call] (partially incorporated)
b) 02-385r0 Minutes of SAS protocol teleconference 24 September 2002 [24 September 2002 SAS call]
   A) Make TASK IU be 28 bytes
c) 02-202r4 SAS Port Control state machine update (John Worden)[24 September 2002 SAS call]
d) 02-318r1 SAS data transfer rules (Ed Gardner)[24 September 2002 SAS call]
e) 02-359r3 SAS Expander configuration details (Steve Fairchild) [1 October 2002 SAS call]
f) 02-383r2 SAS change notification proposal (Jim Reif) [15 October 2002 SAS call]
g) 02-387r2 SAS Programmable PPTOV (Tim Hoglund) [15 October 2002 SAS call]
h) 02-393r1 SAS Port layer rewrite (Rob Elliott)[1 October 2002 SAS call]
i) 02-394r1 SPC-3 SAS Protocol-Specific log page (Rob Elliott) [15 October 2002 SAS call]
j) 02-408r0 Minutes of SAS protocol teleconference 15 October 2002 [15 October 2002 SAS call]
   A) Renumber DEVICE TYPE field values (Rob Elliott)
k) 02-410r0 Minutes of SAS PHY teleconference 14 October 2002 [14 October 2002 SAS PHY call]
   A) Accept REPORT PHY ERROR LOG fields as in sas-r02a
   B) Drop use of “loss of signal”
l) Significant editorial changes to the SP state machine in response to comments in the SAS Editor’s meeting
m) Other editorial changes from the SAS Editor’s meeting and emails

1.17 Revision sas-r02c (1 November 2002)

This revision has change bars from sas-r02. It incorporates these changes:

a) 02-291r2 SAS expander informative annex (Tim Hoglund) [29 October 2002 SAS call]
b) 02-354r1 SAS definition/note for skew (Alvin Cox) [23 September 2002 SAS PHY call - 02-354r0 was previously accepted; this revision has more editorial changes]
c) 02-360r3 SAS spinup (Rob Elliott) [24 October 2002 SAS WG]
d) 02-363r3 SAS STP initiators closing connections (Brian Day) [24 October 2002 SAS WG]
e) 02-390r3 SAS compliant jitter test pattern (Alvin Cox/Bernhard Laschinsky) [31 October 2002 SAS PHY call]
f) 02-391r1 SAS external connector text corrections and signal table modification (Alvin Cox) [3 October 2002 SAS PHY call]
g) 02-396r2 SAS Device Identification VPD page requirements (Rob Elliott) [21 October 2002 SAS call]
h) 02-397r0 SAS Protocol-Specific Port mode page (Rob Elliott) [24 October 2002 SAS WG]
i) 02-405r2 Alternate SAS Speed Negotiation state diagram (George Penokie) [21 October 2002 SAS call]
j) 02-409r3 SAS Expander CHANGE count proposal (Brad Besmer) [24 October 2002 SAS WG]
k) 02-418r2 SAS SATA_ERROR primitive definition (Tim Hoglund) [29 October 2002 SAS call]
l) 02-417r0 Minutes of SAS protocol teleconference 21 October 2002 [21 October 2002 SAS call]
   A) Make SAS address of all zeros invalid
m) 02-428r0 Minutes of SAS protocol WG 24-25 October 2002 [25 October 2002 SAS WG]
   A) Remove PHY MARGIN CONTROL pages
n) 02-427r1 SAS device names (George Penokie) [24 October 2002 SAS WG]
o) 02-433r1 SAS link rate clarification (Bob Sheffield) [24 October 2002 SAS WG]
p) 02-441r1 SAS Changes to REPORT PHY SATA (Steve Fairchild) [24 October 2002 SAS WG]
q) 02-422r2 SAS Bit ordering pictures (Jim Reif) [24 October 2002 SAS WG]
r) 02-444r0 SAS Interpretation of invalid SAS address (Rob Elliott) [29 October 2002 SAS call]
s) 02-445r0 SAS wide link error handling (Rob Elliott) [29 October 2002 SAS call]
t) 02-446r1 SAS READY LED characteristics (Alvin Cox) [31 October 2002 SAS PHY call]
u) 02-450r0 SAS SSP vendor-specific frame types (Rob Elliott) [29 October 2002 SAS call]
v) 02-451r1 SAS Bit and byte ordering (Rob Elliott) [29 October 2002 SAS call]
w) Changed a 0,35 to 0,55 in the physical section [28 October 2002 SAS PHY call]
x) 02-468r0 Minutes of SAS physical WG 31 October 2002 [31 October 2002 SAS PHY call]

A) combine Xr/Xt and Ir/It (meaning initiator or expander), then rename Dt/Dr to It/Ir (meaning internal) in the phy section

1.18 Revision sas-r02d (20 November 2002)

This revision has change bars from sas-r02. It incorporates these changes:

a) 02-349r4 Proposed additions to SAS driver and receiver electrical characteristics (Russ Brown) [5 November 2002 SAS PHY WG]
b) 02-353r1 SAS Output characteristics of the READY LED signal (Alvin Cox) [23 September 2002 SAS PHY call]
c) 02-372r0 Minutes of SAS protocol teleconference 17 September 2002 [17 September 2002 SAS call]
   A) Motion: STP initiators shall not generate BIST frames
d) 02-379r4 SAS test loads (Alvin Cox) [5 November 2002 SAS PHY WG]
e) 02-380r1 SAS OOB signal levels (Alvin Cox) [5 November 2002 SAS PHY WG]
f) 02-417r0 Minutes of SAS protocol teleconference 21 October 2002 [21 October 2002 SAS call]
   A) Fallback and Increment Speed states be removed from the SP state machine
g) 02-424r2 SAS Port layer additional tweaks (John Worden) [24 October 2002 SAS WG]
h) 02-430r3 SAS Simplified support for multiple STP initiators (Bob Sheffield) [5 November 2002 SAS WG]
i) 02-435r3 SAS STP buffering (Bob Sheffield) [5 November 2002 SAS WG]
j) 02-437r1 SAS Support for SATA II: Extensions to SATA 1.0 (Bob Sheffield) [5 November 2002 SAS WG]
k) 02-443r2 SAS Handling Link Rate Not Supported (Rob Elliott) [5 November 2002 SAS WG]
l) 02-449r3 SAS Simple relative offset (Bill Galloway) [5 November 2002 SAS WG]
m) 02-456r0 Minutes of SAS protocol WG 4-5 November 2002 [5 November 2002 SAS WG]
   A) in the REPORT PHY SATA function, change OPEN REJECT (RETRY) to OPEN REJECT (NO DESTINATION) and make other related changes throughout SAS
n) 02-459r1 Unknown Frame Types (Jim Coomes) [5 November 2002 SAS WG]
o) 02-461r1 SAS spare primitives (Bill Galloway) [5 November 2002 SAS WG]
p) 02-462r1 SAS zero length data frames (Bill Galloway) [5 November 2002 SAS WG]
q) 02-470r1 SAS SSP link layer rewrite (George Penokie) [5 November 2002 SAS WG]
r) 02-472r1 Make Protocol-Specific mode page optional (Rob Elliott) [5 November 2002 SAS WG]
s) 02-476r0 Making Report General Page Larger (Brad Besmer) [5 November 2002 SAS WG]
t) 02-477r1 Remove SMP PHY CONTROL function ENABLE (Steve Fairchild) [5 November 2002 SAS WG]
u) 02-478r0 SSP Transport layer state machine modification (Mark Evans) [5 November 2002 SAS WG]
v) 02-482r0 SAS internal cable configuration (Alvin Cox) [5 November 2002 SAS WG]
w) Compacted SMP function numbers, removing REPORT SATA and moving SMP CONFIGURE ROUTE INFORMATION to 90h (it uses PHY IDENTIFIER fields so belongs in that region).
x) Editorial changes from comments received on sas-r02, sas-r02a, sas-r02b, and sas-r02c.

1.19 Revision sas-r02c (21 November 2002)

This revision has no change bars, reflecting T10 plenary approval of all technical proposals. This revision is being released to T10 letter ballot on 22 November 2002.

No proposals are known to be pending.

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 5
3.1 Definitions 5
3.2 Symbols and abbreviations 12
3.3 Keywords 13
3.4 Editorial conventions 14
3.5 State machine conventions 16
3.5.1 State machine conventions overview 16
3.5.2 Transitions 16
3.5.3 Parameters, requests, indications, confirmations, and responses 17
3.6 Bit and byte ordering 17
3.7 Notation for procedures and functions 18

4 General 19
4.1 Architecture 19
4.1.1 Architecture overview 19
4.1.2 Physical links and phys 20
4.1.3 Ports (narrow ports and wide ports) 21
4.1.4 SAS devices 22
4.1.5 Initiator devices 23
4.1.6 Target devices 24
4.1.7 Target/initiator devices 24
4.1.8 Expander devices 25
4.1.8.1 Expander device overview 25
4.1.8.2 Edge expander device set 25
4.1.8.3 Configurable expander device 26
4.1.9 Domains 26
4.1.10 Expander device topologies 28
4.1.11 Connections 29
4.1.12 Pathways 31
4.2 Names and identifiers 32
4.2.1 Names and identifiers overview 32
4.2.2 SAS addresses 33
4.2.3 Hashed SAS address 34
4.2.4 Port names 34
4.2.5 Port identifiers 34
4.2.6 Phy identifier 34
4.3 State machines 35
4.3.1 State machine overview 35
4.3.2 Transmit data path 36
4.3.3 Signals between state machines 39
4.3.3.1 Signals between phy layer and other layers 39
4.3.3.2 Signals between link layer, port layer, and transport layer for SSP 40
4.3.3.3 Signals between link layer, port layer, and transport layer for SMP 42
4.3.3.4 Signals between link layer, port layer, and management application layer for all protocols 44
4.3.3.5 Transport layer to application layer for SSP 45

4.3.3.6 Transport layer to application layer for SMP 46
4.4 Resets 47
4.4.1 Reset overview 47
4.4.2 Hard reset 49
4.5 I_T nexus loss 49
4.6 Expander device model 49
4.6.1 Expander device model overview 49
4.6.2 Expander ports 50
4.6.3 Expander connection manager (ECM) 51
4.6.4 Expander connection router (ECR) 51
4.6.5 Broadcast primitive processor (BPP) 51
4.6.6 Expander device interface 51
4.6.7 Expander device interface detail 53
4.6.8 Expander connection manager interface 54
4.6.9 Expander connection router interface 55
4.6.10 Broadcast primitive processor interface 56
4.6.11 Expander device routing 56
4.6.11.1 Routing attributes and methods 56
4.6.11.2 Expander device connection request routing 57
4.6.11.3 Expander route table 58
4.6.11.4 Expander route index order 59
4.6.11.5 Discover process 64

5 Physical layer 65
5.1 SATA cables and connectors (informative) 65
5.2 SAS cables and connectors 65
5.3 Connectors 67
5.3.1 Connectors overview 67
5.3.2 SAS plug connector 67
5.3.2.1 SAS plug connector overview 67
5.3.3 SAS internal cable receptacle connector 67
5.3.4 SAS backplane receptacle connector 67
5.3.5 SAS internal connector pin assignments 68
5.3.6 SAS external cable plug connector 68
5.3.7 SAS external receptacle connector 69
5.3.8 SAS external connector pin assignments 69
5.4 Cables 70
5.4.1 SAS internal cables 70
5.4.2 SAS external cables 71
5.5 Backplanes 71
5.6 READY LED pin 71
5.7 Driver and receiver electrical characteristics 72
5.7.1 Compliance points 72
5.7.2 General interface specification 72
5.7.3 Eye masks 74
5.7.3.1 Eye masks overview 74
5.7.3.2 Delivered (receive) eye mask at IR, CR, and XR 75
5.7.3.3 Jitter tolerance masks 75
5.7.4 Transmitted signal characteristics 76
5.7.5 Received signal characteristics 77
5.7.6 Jitter 79
5.7.7 Jitter tolerance 79
5.7.8 Jitter compliance test pattern (CJTPAT) 80
5.7.9 Impedance specifications 81
5.7.10 Electrical TxRx connections 82
5.7.11 Transmitter characteristics 82
5.7.12 Receiver characteristics 84

5.7.13 Spread spectrum clocking ... 85
5.8 Non-tracking clock architecture ... 85

6 Phy layer ... 86
6.1 Phy layer overview ... 86
6.2 Encoding (8b10b) ... 86
6.2.1 Encoding overview ... 86
6.2.2 8b10b coding introduction ... 86
6.2.3 8b10b encoding notation conventions ... 86
6.3 Character encoding and decoding ... 87
6.3.1 Introduction ... 87
6.3.2 Transmission order ... 88
6.3.3 Valid and invalid transmission characters ... 88
6.3.3.1 Definitions ... 88
6.3.3.2 Generating transmission characters ... 92
6.3.3.3 Validity of received transmission characters ... 92
6.4 Bit order ... 92
6.5 Out of band (OOB) signals ... 94
6.6 Phy reset sequences ... 98
6.6.1 Overview ... 98
6.6.2 SATA phy reset sequence (informative) ... 99
6.6.2.1 SATA OOB sequence (informative) ... 99
6.6.2.2 SATA speed negotiation sequence (informative) ... 99
6.6.3 SAS to SATA phy reset sequence ... 99
6.6.4 SAS to SAS phy reset sequence ... 100
6.6.4.1 SAS OOB sequence ... 100
6.6.4.2 SAS speed negotiation sequence ... 103
6.6.5 Phy reset sequence after device is attached ... 106
6.7 SAS phy (SP) transmitter and receiver ... 107
6.8 SAS phy (SP) state machine ... 108
6.8.1 Overview ... 108
6.8.2 OOB sequence states ... 109
6.8.2.1 SP1:OOB_COMINIT state ... 109
6.8.2.1.1 State description ... 109
6.8.2.1.2 Transition SP1:OOB_COMINIT to SP2:OOB_AwaitCOMX ... 110
6.8.2.1.3 Transition SP1:OOB_COMINIT to SP3:OOB_AwaitCOMINIT_Sent ... 110
6.8.2.1.4 Transition SP1:OOB_COMINIT to SP4:OOB_COMSAS ... 110
6.8.2.2 SP2:OOB_AwaitCOMX state ... 110
6.8.2.2.1 State description ... 110
6.8.2.2.2 Transition SP2:OOB_AwaitCOMX to SP1:OOB_COMINIT ... 110
6.8.2.2.3 Transition SP2:OOB_AwaitCOMX to SP4:OOB_COMSAS ... 110
6.8.2.3 SP3:OOB_AwaitCOMINIT_Sent state ... 110
6.8.2.3.1 State description ... 110
6.8.2.3.2 Transition SP3:OOB_AwaitCOMINIT_Sent to SP4:COMSAS ... 110
6.8.2.4 SP4:OOB_COMSAS state ... 110
6.8.2.4.1 State description ... 110
6.8.2.4.2 Transition SP4:OOB_COMSAS to SP5:OOB_AwaitCOMSAS_Sent ... 110
6.8.2.4.3 Transition SP4:OOB_COMSAS to SP6:OOB_AwaitNoCOMSAS ... 111
6.8.2.4.4 Transition SP4:OOB_COMSAS to SP7:OOB_AwaitCOMSAS ... 111
6.8.2.5 SP5:OOB_AwaitCOMSAS_Sent state ... 111
6.8.2.5.1 State description ... 111
6.8.2.5.2 Transition SP5:OOB_AwaitCOMSAS_Sent to SP6:OOB_AwaitNoCOMSAS ... 111
6.8.2.6 SP6:OOB_AwaitNoCOMSAS state ... 111
6.8.2.6.1 State description ... 111
6.8.2.6.2 Transition SP6:OOB_AwaitNoCOMSAS to SP8:SAS_Start ... 111
6.8.2.7 SP7:OOB_AwaitCOMSAS state ... 111
6.8.2.7.1 State description ... 111

6.8.2.7.2 Transition SP7:OOB_AwaitCOMSAS to SP1:OOB_COMINIT 111
6.8.2.7.3 Transition SP7:OOB_AwaitCOMSAS to SP6:OOB_AwaitNoCOMSAS 111
6.8.2.7.4 Transition SP7:OOB_AwaitCOMSAS to SP16:SATA_COMWAKE 112
6.8.2.7.5 Transition SP7:OOB_AwaitCOMSAS to SAS_AwaitNoCOMX 112
6.8.3 SAS speed negotiation states 113
6.8.3.1 SP8:SAS_Start state 114
6.8.3.1.1 State description 114
6.8.3.1.2 Transition SP8:SAS_Start to SP10:SAS_AwaitALIGN 114
6.8.3.1.3 Transition SP8:SAS_Start to SP9:SAS_RateNotSupported 114
6.8.3.2 SP9:SAS_RateNotSupported state 114
6.8.3.2.1 State description 114
6.8.3.2.2 Transition SP9:SAS_RateNotSupported to SP14:SAS_Fail 114
6.8.3.3 SP10:SAS_AwaitALIGN state 114
6.8.3.3.1 State description 114
6.8.3.3.2 Transition SP10:SAS_AwaitALIGN to SP11:SAS_AwaitALIGN1 115
6.8.3.3.3 Transition SP10:SAS_AwaitALIGN to SP12:SAS_AwaitSNW 115
6.8.3.3.4 Transition SP10:SAS_AwaitALIGN to SP14:SAS_Fail 115
6.8.3.4 SP11:SAS_AwaitALIGN1 state 115
6.8.3.4.1 State description 115
6.8.3.4.2 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_Fail 115
6.8.3.4.3 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_ AwaitSNW 115
6.8.3.5 SP12:SAS_AwaitSNW state 115
6.8.3.5.1 State description 115
6.8.3.5.2 Transition SP12:SAS_AwaitALIGN1 to SP13:SAS_Pass 115
6.8.3.6 SP13:SAS_Pass state 115
6.8.3.6.1 State description 115
6.8.3.6.2 Transition SP13:SAS_Pass to SP8:SAS_Start 115
6.8.3.6.3 Transition SP13:SAS_Pass to SP15:SAS_PHY_Ready 116
6.8.3.7 SP14:SAS_Fail state 116
6.8.3.7.1 State description 116
6.8.3.7.2 Transition SP14:SAS_Fail to SP2:OOB_AwaitCOMX 116
6.8.3.7.3 Transition SP14:SAS_Fail to SP8:SAS_Start 116
6.8.3.8 SP15:SAS_PHY_Ready state 116
6.8.3.8.1 State description 116
6.8.3.8.2 Transition SP15:SAS_PHY_Ready to SP1:OOB_COMINIT 117
6.8.4 SATA host emulation states 117
6.8.4.1 SP16:SATA_COMWAKE state 119
6.8.4.1.1 State description 119
6.8.4.1.2 Transition SP16:SATA_COMWAKE to SP17:SATA_AwaitCOMWAKE 119
6.8.4.2 SP17:SATA_AwaitCOMWAKE state 119
6.8.4.2.1 State description 119
6.8.4.2.2 Transition SP17:SATA_AwaitCOMWAKE to SP18:SATA_AawitNoCOMWAKE 119
6.8.4.3 SP18:SATA_AwaitNoCOMWAKE state 119
6.8.4.3.1 State description 119
6.8.4.3.2 Transition SP18:SATA_AwaitNoCOMWAKE to SP19:SATA_AwaitALIGN 119
6.8.4.4 SP19:SATA_AwaitALIGN state 119
6.8.4.4.1 State description 119
6.8.4.4.2 Transition SP19:SATA_AwaitALIGN to SP20:SATA_AdjustSpeed 119
6.8.4.4.3 Transition SP19:SATA_AwaitALIGN to SP1:OOB_COMINIT 119
6.8.4.5 SP20:SATA_AdjustSpeed state 119
6.8.4.5.1 State description 119
6.8.4.5.2 Transition SP20:SATA_AdjustSpeed to SP21:SATA_TransmitALIGN 120
6.8.4.6 SP21:SATA_TransmitALIGN state 120
6.8.4.6.1 State description 120
6.8.4.6.2 Transition SP21:SATA_TransmitALIGN to SP22:SATA_PHY_Ready 120
6.8.4.7 SP22:SATA_PHY_Ready state 120
6.8.4.7.1 State description 120

6.8.4.7.2 Transition SP22:SATA_PHY_Ready to SP1:Reset 120
6.8.4.7.3 Transition SP22:SATA_PHY_Ready to SP24:SATA_PM_Partial 120
6.8.4.7.4 Transition SP22:SATA_PHY_Ready to SP23:SATA_PM_Slumber 120
6.8.4.8 SP23:SATA_PM_Partial state 120
6.8.4.8.1 State description 120
6.8.4.8.2 Transition SP23:SATA_PM_Partial to SP16:SATA_COMWAKE 120
6.8.4.8.3 Transition SP23:SATA_PM_Partial to SP18:SATA_AwaitNoCOMWAKE 120
6.8.4.9 SP24:SATA_PM_Slumber state 120
6.8.4.9.1 State description 120
6.8.4.9.2 Transition SP24:SATA_PM_Slumber to SP16:SATA_COMWAKE 120
6.8.4.9.3 Transition SP24:SATA_PM_Slumber to SP18:SATA_AwaitNoCOMWAKE 121
6.9 SAS phy dword synchronization (SP_DWS) state machine 121
6.9.1 Overview 121
6.9.2 SP_DWS0:AcquireSync state 123
6.9.2.1 State description 123
6.9.2.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1 123
6.9.3 SP_DWS1:Valid1 state 123
6.9.3.1 State description 123
6.9.3.2 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2 123
6.9.4 SP_DWS2:Valid2 state 123
6.9.4.1 State description 123
6.9.4.2 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired 123
6.9.4.3 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync 123
6.9.5 SP_DWS3:SyncAcquired state 123
6.9.5.1 State description 123
6.9.5.2 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1 123
6.9.6 SP_DWS4:Lost1 state 124
6.9.6.1 State description 124
6.9.6.2 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered 124
6.9.6.3 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2 124
6.9.7 SP_DWS5:Lost1Recovered state 124
6.9.7.1 State description 124
6.9.7.2 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired 124
6.9.7.3 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2 124
6.9.8 SP_DWS6:Lost2 state 124
6.9.8.1 State description 124
6.9.8.2 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered 124
6.9.8.3 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3 124
6.9.9 SP_DWS7:Lost2Recovered state 124
6.9.9.1 State description 124
6.9.9.2 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1 125
6.9.9.3 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3 125
6.9.10 SP_DWS8:Lost3 state 125
6.9.10.1 State description 125
6.9.10.2 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered 125
6.9.10.3 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync 125
6.9.11 SP_DWS9:Lost3Recovered state 125
6.9.11.1 State description 125
6.9.11.2 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2 125
6.9.11.3 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync 125
6.10 Spin-up 125

7 Link layer 127
7.1 Primitives 127
7.1.1 Primitives overview 127
7.1.2 Primitive summary 127
7.1.3 Primitive sequences 134

7.1.3.1 Primitive sequence overview 134
7.1.3.2 Single primitive sequence 135
7.1.3.3 Repeated primitive sequence 135
7.1.3.4 Triple primitive sequence 135
7.1.3.5 Redundant primitive sequence 135
7.1.4 Primitives not specific to type of connections 136
7.1.4.1 AIP (Arbitration in progress) 136
7.1.4.2 ALIGN 136
7.1.4.3 BREAK 137
7.1.4.4 BROADCAST 137
7.1.4.5 CLOSE 137
7.1.4.6 EOAF (End of address frame) 138
7.1.4.7 ERROR 138
7.1.4.8 HARD_RESET 138
7.1.4.9 NOTIFY 138
7.1.4.10 OPEN_ACCEPT 138
7.1.4.11 OPEN_REJECT 139
7.1.4.12 SOAF (Start of address frame) 141
7.1.5 Primitives used only inside SSP and SMP connections 141
7.1.5.1 ACK (Acknowledge) 141
7.1.5.2 CREDIT_BLOCKED 141
7.1.5.3 DONE 141
7.1.5.4 EOF (End of frame) 141
7.1.5.5 NAK (Negative acknowledgement) 141
7.1.5.6 RRDY (Receiver ready) 142
7.1.5.7 SOF (Start of frame) 142
7.1.6 Primitives used only inside STP connections and on SATA physical links 142
7.1.6.1 SATA_ERROR 142
7.1.6.2 SATA_PMACK, SATA_PMNAK, SATA_PMREQ_P, and SATA_PMREQ_S (Power management acknowledgements and requests) 142
7.1.6.3 SATA_HOLD and SATA_HOLDA (Hold and hold acknowledge) 142
7.1.6.4 SATA_R_RDY and SATA_X_RDY (Receiver ready and transmitter ready) 143
7.1.6.5 Other primitives used inside STP connections and on SATA physical links 143
7.2 Clock skew management 143
7.3 Idle links 144
7.4 CRC 144
7.4.1 CRC overview 144
7.4.2 CRC generation 145
7.4.3 CRC checking 146
7.5 Scrambling 147
7.6 Bit order of CRC and scrambler 148
7.7 Address frames 149
7.7.1 Address frames overview 149
7.7.2 IDENTIFY address frame 150
7.7.3 OPEN address frame 152
7.8 Identification and hard reset sequence 154
7.8.1 Overview 154
7.8.2 Initiator device specific rules 154
7.8.3 Fanout expander device specific rules 155
7.8.4 Edge expander device specific rules 155
7.8.5 Identification and hard reset (SL_IR) state machines 155
7.8.5.1 Overview 155
7.8.6 SL_IR transmitter and receiver 157
7.8.6.1 Transmit IDENTIFY or HARD_RESET (SL_IR_TIR) state machine 157
7.8.6.1.1 Overview 157
7.8.6.1.2 SL_IR_TIR1:Idle state 157
7.8.6.1.2.1 State description 157

7.8.6.1.2.2 Transition SL_IR_TIR1:Idle to SL_IR_TIR2:Transmit_Identify 157
7.8.6.1.2.3 Transition SL_IR_TIR1:Idle to SL_IR_TIR3:Transmit_Hard_Reset 157
7.8.6.1.3 SL_IR_TIR2:Transmit_Identify state 157
7.8.6.1.3.1 State description 157
7.8.6.1.3.2 Transition SL_IR_TIR2:Transmit_Identify to SL_IR_TIR4:Completed 158
7.8.6.1.4 SL_IR_TIR3:Transmit_Hard_Reset state 158
7.8.6.1.4.1 State description 158
7.8.6.1.4.2 Transition SL_IR_TIR3:Transmit_Hard_Reset to SL_IR_TIR3:Completed 158
7.8.6.1.5 SL_IR_TIR4:Completed state 158
7.8.6.2 Receive IDENTIFY Address Frame (SL_IR_RIF) state machine 158
7.8.6.2.1 Overview 158
7.8.6.2.2 SL_IR_RIF1:Idle state 158
7.8.6.2.2.1 State description 158
7.8.6.2.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame 158
7.8.6.2.3 SL_IR_RIF2:Receive_Identify_Frame state 158
7.8.6.2.3.1 State description 158
7.8.6.2.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed . 158
7.8.6.2.4 SL_IR_RIF3:Completed state 159
7.8.6.3 Identification and hard reset control (SL_IR_IRC) state machine 159
7.8.6.3.1 Overview 159
7.8.6.3.2 SL_IR_IRC1:Idle state 159
7.8.6.3.2.1 State description 159
7.8.6.3.2.2 Transition SL_IR_IRC1:Idle to SL_IR_IRC2:Wait 159
7.8.6.3.3 SL_IR_IRC2:Wait state 159
7.8.6.3.3.1 State description 159
7.8.6.3.3.2 Transition SL_IR_IRC2:Wait to SL_IR_IRC3:Completed 160
7.8.6.3.4 SL_IR_IRC3:Completed state 160
7.9 Power management 160
7.10 Near-end analog loopback test 160
7.11 Domain changes 161
7.12 Connections 162
7.12.1 Connection overview 162
7.12.2 Opening a connection 162
7.12.2.1 Connection request 162
7.12.2.2 Connection request responses 163
7.12.3 Arbitration fairness 163
7.12.3.1 Arbitration and resource management in an expander device 164
7.12.3.1.1 Arbitration overview 164
7.12.3.1.2 Arbitration status 165
7.12.3.1.3 Partial Pathway Timer 165
7.12.3.1.4 Pathway Recovery 165
7.12.4 Expander devices and connection requests 166
7.12.4.1 All expander devices 166
7.12.4.2 Edge expander devices 166
7.12.4.3 Fanout expander devices 167
7.12.5 Abandoning a connection request 167
7.12.6 Breaking a connection 169
7.12.7 Closing a connection 170
7.13 SAS link layer state machine for initiator phys and target phys (SL) 170
7.13.1 Overview 170
7.13.2 SL transmitter and receiver 173
7.13.3 SL0:Idle state 174
7.13.3.1 State description 174
7.13.3.2 Transition SL0:Idle to SL1:ArbSel 174
7.13.3.3 Transition SL0:Idle to SL2:Selected 174
7.13.4 SL1:ArbSel state 175

7.13.4.1 State description 175
7.13.4.2 Transition SL1:ArbSel to SL0:Idle 175
7.13.4.3 Transition SL1:ArbSel to SL2:Selected 176
7.13.4.4 Transition SL1:ArbSel to SL3:Connected 176
7.13.4.5 Transition SL1:ArbSel to SL5:BreakWait 176
7.13.4.6 Transition SL1:ArbSel to SL6:Break 176
7.13.5 SL2:Selected state 176
7.13.5.1 State description 176
7.13.5.2 Transition SL2:Selected to SL0:Idle 177
7.13.5.3 Transition SL2:Selected to SL3:Connected 177
7.13.5.4 Transition SL2:Selected to SL6:Break 177
7.13.6 SL3:Connected state 178
7.13.6.1 State description 178
7.13.6.2 Transition SL3:Connected to SL4:DisconnectWait 178
7.13.6.3 Transition SL3:Connected to SL5:BreakWait 178
7.13.6.4 Transition SL3:Connected to SL6:Break 178
7.13.7 SL4:DisconnectWait state 178
7.13.7.1 State description 178
7.13.7.2 Transition SL4:DisconnectWait to SL0:Idle 178
7.13.7.3 Transition SL4:DisconnectWait to SL5:BreakWait 178
7.13.7.4 Transition SL4:DisconnectWait to SL6:Break 179
7.13.8 SL5:BreakWait state 179
7.13.8.1 State description 179
7.13.8.2 Transition SL5:BreakWait to SL0:Idle 179
7.13.9 SL6:Break state 179
7.13.9.1 State description 179
7.13.9.2 Transition SL6:Break to SL0:Idle 179
7.14 SAS link layer state machine for expander phys (XL) 179
7.14.1 Overview 179
7.14.2 XL0:Idle state 184
7.14.2.1 State description 184
7.14.2.2 Transition XL0:Idle to XL1:Request_Path 184
7.14.2.3 Transition XL0:Idle to XL5:Forward_Open 184
7.14.2.4 Transition XL0:Idle to XL9:Break 184
7.14.2.5 Transition XL0:Idle to XL10:Break_Wait 184
7.14.3 XL1:Request_Path state 184
7.14.3.1 State description 184
7.14.3.2 Transition XL1:Request_Path to XL2:Request_Open 185
7.14.3.3 Transition XL1:Request_Path to XL4:Open_Reject 185
7.14.3.4 Transition XL1:Request_Path to XL0:Idle 185
7.14.3.5 Transition XL1:Request_Path to XL9:Break 185
7.14.4 XL2:Request_Open state 185
7.14.4.1 State description 185
7.14.4.2 Transition XL2:Request_Open to XL3:Open_Confirm_Wait 185
7.14.5 XL3:Open_Confirm_Wait state 186
7.14.5.1 State description 186
7.14.5.2 Transition XL3:Open_Confirm_Wait to XL0:Idle 186
7.14.5.3 Transition XL3:Open_Confirm_Wait to XL7:Connected 186
7.14.5.4 Transition XL3:Open_Confirm_Wait to XL9:Break 186
7.14.5.5 Transition XL3:Open_Confirm_Wait to XL10:Break_Wait 186
7.14.6 XL4:Open_Reject state 186
7.14.6.1 State description 186
7.14.6.2 Transition XL4:Open_Reject to XL0:Idle 187
7.14.7 XL5:Forward_Open state 187
7.14.7.1 State description 187
7.14.7.2 Transition XL5:Forward_Open to XL6:Open_Response_Wait 187
7.14.8 XL6:Open_Response_Wait state 187

7.14.8.1 State description 187
7.14.8.2 Transition XL6:Open_Response_Wait to XL0:Idle 188
7.14.8.3 Transition XL6:Open_Response_Wait to XL2:Request_Open 188
7.14.8.4 Transition XL6:Open_Response_Wait to XL7:Connected 188
7.14.8.5 Transition XL6:Open_Response_Wait to XL9:Break 188
7.14.8.6 Transition XL6:Open_Response_Wait to XL10:Break_Wait 188
7.14.9 XL7:Connected state 188
7.14.9.1 State description 188
7.14.9.2 Transition XL7:Connected to XL8:Close_Wait 188
7.14.9.3 Transition XL7:Connected to XL9:Break 189
7.14.9.4 Transition XL7:Connected to XL10:Break_Wait 189
7.14.10 XL8:Close_Wait state 189
7.14.10.1 State description 189
7.14.10.2 Transition XL8:Close_Wait to XL0:Idle 189
7.14.10.3 Transition XL8:Close_Wait to XL9:Break 189
7.14.10.4 Transition XL8:Close_Wait to XL10:Break_Wait 189
7.14.11 XL9:Break state 189
7.14.11.1 State description 189
7.14.11.2 Transition XL9:Break to XL0:Idle 189
7.14.12 XL10:Break_Wait state 189
7.14.12.1 State description 189
7.14.12.2 Transition XL10:Break_Wait to XL0:Idle 190
7.15 Rate matching 190
7.16 SSP link layer 190
7.16.1 Opening an SSP connection 190
7.16.2 Full duplex 190
7.16.3 SSP frame transmission 191
7.16.4 SSP flow control 191
7.16.5 Interlocked frames 191
7.16.6 Preparing to close an SSP connection 193
7.16.7 SSP link layer (SSP) state machines 193
7.16.7.1 Overview 193
7.16.7.2 SSP_TIM1:Tx_Interlock_Monitor state 197
7.16.7.3 SSP_TCM1:Tx_credit_monitor state 198
7.16.7.4 SSP_D1:DONE_Wait state 198
7.16.7.4.1 State description 198
7.16.7.5 SSP_TF1:Connected_idle state 199
7.16.7.5.1 State description 199
7.16.7.5.2 Transition SSP_TF1:Connected_Idle to SSP_TF2:Tx_Wait 199
7.16.7.5.3 Transition SSP_TF1:Connected_Idle to SSP_TF4:Indicate_Done_Tx 199
7.16.7.6 SSP_TF2:Tx_Wait state 199
7.16.7.6.1 State description 199
7.16.7.6.2 Transition SSP_TF2:Tx_Wait to SSP_TF3:Indicate_Frame_Tx 199
7.16.7.6.3 Transition SSP_TF2:Tx_Wait to SSP_TF4:Indicate_Done_Tx 200
7.16.7.7 SSP_TF3:Indicate_Frame_Tx state 200
7.16.7.7.1 State description 200
7.16.7.7.2 Transition SSP_TF3:Indicate_Frame_Tx to SSP_TF1:Connected_idle 200
7.16.7.8 SSP_TF4:Indicate_Done_Tx state 200
7.16.7.9 SSP_RF1:Rcv_Frame state 201
7.16.7.10 SSP_RCM1:Rcv_Credit_Monitor state 201
7.16.7.11 SSP_RIM1:Rcv_Interlock_Monitor state 202
7.16.7.12 SSP_TC1:Idle state 202
7.16.7.12.1 State description 202
7.16.7.12.2 Transition SSP_TC1:Idle to SSP_TC2:Indicate_Credit_Tx 202
7.16.7.13 SSP_TC2:Indicate_Credit_Tx state 202
7.16.7.13.1 State description 202
7.16.7.13.2 Transition SSP_TC2:Indicate_Credit_Tx to SSP_TC1:Idle 203

7.16.7.14 SSP_TAN1:Idle state 203
7.16.7.14.1 State description 203
7.16.7.14.2 Transition SSP_TAN1:Idle to SSP_TAN2:Indicate_ACK/NAK_Tx 203
7.16.7.15 SSP_TAN2:Indicate_ACK/NAK_Tx state 203
7.16.7.15.1 State description 203
7.16.7.15.2 Transition SSP_TAN2:Indicate_ACK/NAK_tx to SSP_TAN1:Idle 203
7.17 STP link layer 203
7.17.1 STP frame transmission 203
7.17.2 STP flow control 205
7.17.3 Preparing to close an STP connection 207
7.17.4 STP link layer (STP) state machines 207
7.18 SMP link layer 207
7.18.1 SMP frame transmission 207
7.18.2 SMP flow control 207
7.18.3 Preparing to close an SMP connection 207
7.18.4 SMP link layer (SMP) state machines 208
7.18.4.1 Overview 208
7.18.4.2 SMP Initiator Link state machine 210
7.18.4.2.1 SMP_IL1:Command_idle state 210
7.18.4.2.1.1 State description 210
7.18.4.2.1.2 Transition SMP_IL1:Command_idle to SMP_IL2:Indicate_frame_tx 210
7.18.4.2.2 SMP_IL2:Indicate_frame_tx state 211
7.18.4.2.2.1 State description 211
7.18.4.2.2.2 Transition SMP_IL2:Indicate_frame_tx to SMP_IL3:Rcv_response_frame ... 211
7.18.4.2.3 SMP_IL3:Rcv_response_frame state 211
7.18.4.2.3.1 State description 211
7.18.4.3 SMP Target Link state machine 211
7.18.4.3.1 SMP_TL1:Wait_originate_frame state 211
7.18.4.3.1.1 State description 211
7.18.4.3.1.2 Transition SMP_TL1:Wait_originate_frame to SMP_TL2:Wait_transmit_frame 211
7.18.4.3.2 SMP_TL2:Wait_transmit_frame state 212
7.18.4.3.2.1 State description 212

8 Port layer 213
8.1 Overview 213
8.2 Port layer timers and counters 214
8.2.1 Timers and counters overview 214
8.2.2 Bus inactivity time limit timer 214
8.2.3 Maximum connect time limit timer 214
8.2.4 I_T nexus loss timer 214
8.2.5 Arbitration wait time (AWT) timer 215
8.2.6 Pathway blocked count (PBC) counter 215
8.3 Port layer overall control (PL_OC) state machine 215
8.3.1 Overview 215
8.3.2 8.3.2 PL_OC1:Idle state 216
8.3.2.1 8.3.2.1 State description 216
8.3.2.2 8.3.2.2 Transition PL_OC1:Idle to PC_OC2:Overall_Control 217
8.3.3 PL_OC2:Overall_Control state 217
8.3.3.1 State description 217
8.3.3.1.1 State description overview 217
8.3.3.1.2 Keep track of connections/frame requests 217
8.3.3.1.3 Select a request to process and the phy on which to process it 218
8.3.3.1.4 SSP wide port rules 218
8.3.3.1.5 Filling in the Tx Frame arguments 219
8.3.3.1.6 Confirmations 220

8.3.3.1.7 Handling Cancel requests 221
8.3.3.1.8 Handling other requests 221
8.3.3.2 Transition PL_OC2:Overall_Control to PL_OC1:Idle 221
8.4 Port layer phy manager (PL_PM) state machine 221
8.4.1 Overview 221
8.4.2 PL_PM1:Idle state 223
8.4.2.1 State description 223
8.4.2.2 Transition PL_PM1:Idle to PL_PM2:ReqWait 224
8.4.2.3 Transition PL_PM1:Idle to PL_PM3:Connected 224
8.4.3 PL_PM2:ReqWait state 224
8.4.3.1 State description 224
8.4.3.1.1 State description overview 224
8.4.3.1.2 PL_PM I_T nexus loss timer 224
8.4.3.1.3 Connection Opened handling 224
8.4.3.1.4 Open Failed handling 225
8.4.3.2 Transition PL_PM2:ReqWait to PL_PM1:Idle 225
8.4.3.3 Transition PL_PM2:ReqWait to PL_PM3:Connected 225
8.4.3.4 Transition PL_PM2:ReqWait to PL_PM4:Wait_For_Close 225
8.4.4 PL_PM3:Connected state 225
8.4.4.1 State description 225
8.4.4.2 Transition PL_PM3:Connected to PL_PM1:Idle 226
8.4.5 PL_PM4:Wait_For_Close state 226
8.4.5.1 State description 226
8.4.5.2 Transition PL_PM4:Wait_For_Close to PL_PM1:Idle 226

9 Transport layer 227
9.1 Transport layer overview 227
9.2 SSP transport layer 228
9.2.1 SSP frame format 228
9.2.2 Information units 230
9.2.2.1 COMMAND information unit 230
9.2.2.2 TASK information unit 231
9.2.2.3 XFER_RDY information unit 233
9.2.2.4 DATA information unit 233
9.2.2.5 RESPONSE information unit 235
9.2.2.5.1 RESPONSE information unit overview 235
9.2.2.5.2 RESPONSE information unit NO_DATA format 236
9.2.2.5.3 RESPONSE information unit RESPONSE_DATA format 236
9.2.2.5.4 RESPONSE information unit SENSE_DATA format 237
9.2.3 Frame sequences 237
9.2.4 SSP transport layer handling of link layer errors 239
9.2.4.1 COMMAND frame 239
9.2.4.2 TASK frame 239
9.2.4.3 XFER_RDY frame 239
9.2.4.4 DATA frame 240
9.2.4.5 RESPONSE frame 240
9.2.5 SSP transport layer error handling 240
9.2.5.1 Target port error handling 240
9.2.5.2 Initiator port error handling 241
9.2.6 SSP transport layer state machines 242
9.2.6.1 Overview 242
9.2.6.2 Initiator device state machines 242
9.2.6.2.1 Overview 242
9.2.6.2.2 ST_ISF1:Send_Frame state 245
9.2.6.2.2.1 State description 245
9.2.6.2.2.2 Transition ST_ISF1:Send_Frame to ST_ISF2:Prepare_Command_Request 246

9.2.6.2.2.3 Transition ST_ISF1:Send_Frame to ST_ISF3:Prepare_Send_Data_Out . 247
9.2.6.2.3 ST_ISF2:Prepare_Command_Request state 247
9.2.6.2.3.1 State description 247
9.2.6.2.3.2 Transition ST_ISF2:Prepare_Command_Request to ST_ISF1:Send_Frame . 247
9.2.6.2.4 ST_ISF3:Prepare_Send_Data_Out state 247
9.2.6.2.4.1 State description 247
9.2.6.2.4.2 Transition ST_ISF3:Prepare_Send_Data_Out to ST_ISF1:Send_Frame . 248
9.2.6.2.5 ST_IRD1:Receive_Data_In state 248
9.2.6.2.5.1 State description 248
9.2.6.2.5.2 Transition ST_IRD1:Receive_Data_In to ST_IRD2:Process_Received_Data_In 248
9.2.6.2.6 ST_IRD2:Process_Received_Data_In state 248
9.2.6.2.7 ST_IPR1:Process_Received_Response state 248
9.2.6.2.8 ST_IFR1:Initiator_Frame_Router state 249
9.2.6.3 Target device state machines 250
9.2.6.3.1 Overview 250
9.2.6.3.2 ST_TFR1:Target_Frame_Router state 252
9.2.6.3.3 ST_TTS1:Target_Request_Response_Router state 253
9.2.6.3.3.1 State description 253
9.2.6.3.3.2 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS2:Send_Frame 253
9.2.6.3.3.3 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS4:Receive_Data_Out 253
9.2.6.3.3.4 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS7:Prepare_Response 253
9.2.6.3.4 ST_TTS2:Send_Frame state 253
9.2.6.3.4.1 State description 253
9.2.6.3.4.2 Transition ST_TTS2:Send_Frame to ST_TTS3:Prepare_Send_Data_In .. 255
9.2.6.3.4.3 Transition ST_TTS2:Send_Frame to ST_TTS4:Receive_Data_Out 255
9.2.6.3.4.4 Transition ST_TTS2:Send_Frame to ST_TTS7:Prepare_Response 255
9.2.6.3.5 ST_TTS3:Prepare_Send_Data_In state 255
9.2.6.3.5.1 State description 255
9.2.6.3.5.2 Transition ST_TTS3:Prepare_Send_Data_In to ST_TTS2:Send_Frame .. 255
9.2.6.3.6 ST_TTS4:Receive_Data_Out state 256
9.2.6.3.6.1 State description 256
9.2.6.3.6.2 Transition ST_TTS4:Receive_Data_Out to ST_TTS2:Send_Frame 257
9.2.6.3.6.3 Transition ST_TTS4:Receive_Data_Out to ST_TTS5:Prepare_XFER_RDY .. 257
9.2.6.3.6.4 Transition ST_TTS4:Receive_Data_Out to ST_TTS6:Process_Received_Data_Out 257
9.2.6.3.7 ST_TTS5:Prepare_XFER_RDY state 257
9.2.6.3.7.1 State description 257
9.2.6.3.7.2 Transition ST_TTS5:Prepare_XFER_RDY to ST_TTS4:Receive_Data_Out .. 257
9.2.6.3.8 ST_TTS6:Process_Received_Data_Out state 257
9.2.6.3.8.1 State description 257
9.2.6.3.8.2 Transition ST_TTS6:Process_Received_Data_Out to ST_TTS4:Receive_Data_Out 257
9.2.6.3.9 ST_TTS7:Prepare_Response state 257
9.2.6.3.9.1 State description 257
9.2.6.3.9.2 Transition ST_TTS7:Prepare_Response to ST_TTS2:Send_Frame 259
9.3 STP transport layer 259
9.3.1 Initial FIS 259
9.3.2 SATA tunneling for multiple STP initiator ports 259
9.3.3 BIST Activate FIS 259
9.3.4 STP transport layer (TT) state machines 259

9.4 SMP transport layer 259
9.4.1 SMP overview 259
9.4.2 SMP_REQUEST frame 260
9.4.3 SMP_RESPONSE frame 261
9.4.4 SMP transport layer state machines 262
9.4.4.1 Overview 262
9.4.4.2 Initiator device state machine 262
9.4.4.2.1 Overview 262
9.4.4.2.2 MT_ID1:Idle state 263
9.4.4.2.2.1 State description 263
9.4.4.2.2.2 Transition MT_ID1:Idle to MT_ID2:Send 263
9.4.4.2.3 MT_ID2:Send state 264
9.4.4.2.3.1 State description 264
9.4.4.2.3.2 Transition MT_ID2:Send to MT_ID1:Idle 264
9.4.4.2.3.3 Transition MT_ID2:Send to MT_ID3:Receive 264
9.4.4.2.4 MT_ID3:Receive state 264
9.4.4.2.4.1 State description 264
9.4.4.2.4.2 Transition MT_ID3:Receive to MT_ID1:Idle 264
9.4.4.3 Expander device and target device state machine 264
9.4.4.3.1 Overview 264
9.4.4.3.2 MT_TD1:Idle state 265
9.4.4.3.2.1 State description 265
9.4.4.3.2.2 Transition MT_TD1:Idle to MT_TD2:Respond 265
9.4.4.3.3 MT_TD2:Respond state 265
9.4.4.3.3.1 State description 265
9.4.4.3.3.2 Transition MT_TD2:Respond to MT_TD1:Idle 265

10 Application layer 266
10.1 SCSI application layer 266
10.1.1 SCSI transport protocol services 266
10.1.1.1 Transport protocol services overview 266
10.1.1.2 Send SCSI Command transport protocol service 267
10.1.1.3 SCSI Command Received transport protocol service 268
10.1.1.4 Send Command Complete transport protocol service 269
10.1.1.5 Command Complete Received transport protocol service 269
10.1.1.6 Send Data-In transport protocol service 270
10.1.1.7 Data-In Delivered transport protocol service 271
10.1.1.8 Receive Data-Out transport protocol service 271
10.1.1.9 Data-Out Received transport protocol service 272
10.1.1.10 Send Task Management Request transport protocol service 272
10.1.1.11 Task Management Request Received transport protocol service 273
10.1.1.12 Task Management Function Executed transport protocol service 273
10.1.1.13 Received Task Management Function-Executed transport protocol service 274
10.1.2 Device server error handling 275
10.1.3 Application client error handling 276
10.1.4 SCSI transport protocol events 276
10.1.5 SCSI commands 277
10.1.5.1 INQUIRY command 277
10.1.5.2 LOG SELECT and LOG SENSE commands 277
10.1.5.3 MODE SELECT and MODE SENSE commands 277
10.1.5.4 START STOP UNIT command 277
10.1.6 SCSI mode parameters 277
10.1.6.1 Disconnect-Reconnect mode page 277
10.1.6.1.1 Disconnect-Reconnect mode page overview 277
10.1.6.1.2 BUS INACTIVITY TIME LIMIT field 278
10.1.6.1.3 MAXIMUM CONNECT TIME LIMIT field 279
10.1.6.1.4 MAXIMUM BURST SIZE field 279

10.1.6.1.5 FIRST BURST SIZE field 279
10.1.6.2 Protocol-Specific Port mode page 279
10.1.6.2.1 Overview 279
10.1.6.2.2 Protocol-Specific Port mode page - short format 280
10.1.6.2.3 Protocol-Specific Port mode page - Phy Control And Discover subpage 281
10.1.6.3 Protocol-Specific Logical Unit mode page 283
10.1.7 SCSI log parameters 283
10.1.7.1 Protocol-Specific log page for SAS 283
10.1.8 SCSI power condition states 287
10.1.8.1 SA_PC_0:Powered_On state 288
10.1.8.1.1 State description 288
10.1.8.1.2 Transition SA_PC_0:Powered_On to SA_PC_4:Stopped 288
10.1.8.1.3 Transition SA_PC_0:Powered_On to SA_PC_5:Active_Wait 288
10.1.8.2 SA_PC_1:Active state 289
10.1.8.2.1 State description 289
10.1.8.2.2 Transition SA_PC_1:Active to SA_PC_2:Idle 289
10.1.8.2.3 Transition SA_PC_1:Active to SA_PC_3:Standby 289
10.1.8.2.4 Transition SA_PC_1:Active to SA_PC_4:Stopped 289
10.1.8.3 SA_PC_2:Idle state 289
10.1.8.3.1 State description 289
10.1.8.3.2 Transition SA_PC_2:Idle to SA_PC_1:Active 289
10.1.8.3.3 Transition SA_PC_2:Idle to SA_PC_3:Standby 289
10.1.8.3.4 Transition SA_PC_2:Idle to SA_PC_4:Stopped 290
10.1.8.4 SA_PC_3:Standby state 290
10.1.8.4.1 State description 290
10.1.8.4.2 Transition SA_PC_3:Standby to SA_PC_4:Stopped 290
10.1.8.4.3 Transition SA_PC_3:Standby to SA_PC_5:Active_Wait 290
10.1.8.4.4 Transition SA_PC_3:Standby to SA_PC_6:Idle_Wait 290
10.1.8.5 SA_PC_4:Stopped state 290
10.1.8.5.1 State description 290
10.1.8.5.2 Transition SA_PC_4:Stopped to SA_PC_3:Standby 290
10.1.8.5.3 Transition SA_PC_4:Stopped to SA_PC_5:Active_Wait 290
10.1.8.5.4 Transition SA_PC_4:Stopped to SA_PC_6:Idle_Wait 291
10.1.8.6 SA_PC_5:Active_Wait state 291
10.1.8.6.1 State description 291
10.1.8.6.2 Transition SA_PC_5:Active_Wait to SA_PC_1:Active 291
10.1.8.6.3 Transition SA_PC_5:Active_Wait to SA_PC_3:Standby 291
10.1.8.6.4 Transition SA_PC_5:Active_Wait to SA_PC_4:Stopped 291
10.1.8.6.5 Transition SA_PC_5:Active_Wait to SA_PC_6:Idle_Wait 291
10.1.8.7 SA_PC_6:Idle_Wait state 291
10.1.8.7.1 State description 291
10.1.8.7.2 Transition SA_PC_6:Idle_Wait to SA_PC_2:Idle 291
10.1.8.7.3 Transition SA_PC_6:Idle_Wait to SA_PC_3:Standby 291
10.1.8.7.4 Transition SA_PC_6:Idle_Wait to SA_PC_4:Stopped 292
10.1.8.7.5 Transition SA_PC_6:Idle_Wait to SA_PC_5:Active_Wait 292
10.1.9 SCSI vital product data 292
10.2 ATA application layer 293
10.3 Management application layer 294
10.3.1 SMP functions 294
10.3.1.1 Function overview 294
10.3.1.2 REPORT GENERAL function 295
10.3.1.3 REPORT MANUFACTURER INFORMATION function 297
10.3.1.4 DISCOVER function 299
10.3.1.5 REPORT PHY ERROR LOG function 303
10.3.1.6 REPORT PHY SATA function 305
10.3.1.7 REPORT ROUTE INFORMATION function 307
10.3.1.8 CONFIGURE ROUTE INFORMATION function 310

10.3.1.9 PHY CONTROL function 312

Annex A Compliant jitter test pattern (CJTPAT) 317
A.1 Compliant jitter test pattern (CJTPAT) 317

Annex B SAS phy reset sequence examples 324
B.1 SAS phy reset sequence examples 324

Annex C CRC 326
C.1 CRC generator and checker implementation examples 326
C.2 CRC implementation in C 326
C.3 CRC implementation with XORs 327
C.4 CRC examples 329

Annex D SAS address hashing 330
D.1 Hashing overview 330
D.2 Hash collision probability 330
D.3 Hash generation 330
D.4 Hash implementation in C 331
D.5 Hash implementation with XORs 332
D.6 Hash examples 333

Annex E Scrambling 336
E.1 Scrambler implementation in C 336
E.2 Scrambler implementation with XORs 337
E.3 Scrambler examples 338

Annex F ATA architectural notes 339
F.1 STP differences from SATA 339
F.2 STP differences from Serial ATA II 339
F.3 Byte and bit ordering 339

Annex G Expander handling of connections 343
G.1 Overview 343
G.2 Connection request - Open accept 345
G.3 Connection request - Open reject by end device 346
G.4 Connection request - Open reject by expander device 347
G.5 Connection request - Arbitration lost 348
G.6 Connection request - Backoff and retry 349
G.7 Connection request - Backoff and reverse path 350
G.8 Connection close - single step 351
G.9 Connection close - simultaneous 352
G.10 Break handling during path arbitration 353
G.11 Break handling during connection 354
G.12 STP connection - originated by STP initiator port 355
G.13 STP connection - originated by expander device 356
G.14 STP connection close - originated by STP initiator port 357
G.15 STP connection close - originated by expander device 358
G.16 Pathway blocked and recover example 359

Annex H Primitive encoding 360
H.1 Overview 360

Annex I Discover process example implementation 363
I.1 Overview 363
I.2 Header file 363
I.3 Source file 371

Annex J SAS logo 382

Tables

Page
1 Standards bodies 3
2 ISO and American numbering conventions 15
3 Data dword containing a value 17
4 Data dword containing four one-byte fields 18
5 Names and identifiers 33
6 SAM-3 object mapping 33
7 SAS address format 33
8 Hashed SAS address code parameter 34
9 Requests from management application layer to phy layer 39
10 Confirmations from phy layer to link layer or expander function 40
11 Requests between link layer, port layer, and transport layer for SSP 40
12 Confirmations between SL link layer, port layer, and SSP transport layer 41
13 Confirmations between SSP link layer, port layer, and SSP transport layer 42
14 Requests between SL/SMP link layer, port layer, and SMP transport layer 42
15 Confirmations between link layer, port layer, and SMP transport layer 43
16 Confirmations between link layer and port layer 44
17 Requests from management application layer to link layer 44
18 Confirmations between link layer and port layer, expander function or application layer 45
19 Requests from SCSI application layer to SSP transport layer 45
20 Confirmations from SSP transport layer to SCSI application layer 46
21 Requests from management application layer to SMP transport layer 46
22 Confirmations from SMP transport layer to management application layer 46
23 Expander connection manager interface 54
24 Expander connection router interface 55
25 Broadcast primitive processor interface 56
26 Expander route table levels 61
27 Expander route table entries for edge expander E0 phy 0 63
28 Expander route table entries for fanout expander device F phy 0 63
29 Connectors 67
30 SAS internal connector pins 68
31 Physical link usage in SAS external connector 69
32 Output characteristics of the READY LED signal 71
33 Compliance points 72
34 General interface characteristics 74
35 Transmitted signal characteristics at Tx compliance points 77
36 Delivered signal characteristic at Rx compliance points 77
37 Jitter compliance points 79
38 Jitter tolerance compliance points 79
39 Impedance requirements 81
40 Special character usage 86
41 Bit designations 87
42 Conversion example 87
43 Valid data characters 89
44 Valid special characters 91
45 Delayed code violation example 92
46 OOB signal timing specifications 95
47 OOB signal transmitter requirements 95
48 OOB signal receiver requirements 97
49 SAS speed negotiation sequence timing specifications 104
50 Primitive format 127
51 Primitives not specific to type of connection 127
52 Primitives used only inside SSP and SMP connections 129
53 Primitives used only inside STP connections and on SATA physical links 130
54 Primitive encoding for primitives not specific to type of connection 131
55 Primitive encoding for primitives used only inside SSP and SMP connections 133

56 Primitive encoding for primitives used only inside STP connections and on SATA physical links 134
57 Primitive sequences 134
58 AIP primitives 136
59 BROADCAST primitives 137
60 CLOSE primitives 137
61 OPEN_REJECT abandon primitives 139
62 OPEN_REJECT retry primitives 140
63 DONE primitives 141
64 NAK primitives 142
65 RRDY primitives 142
66 ALIGN insertion requirements 144
67 CRC polynomials 145
68 Scrambling types 147
69 Scrambling endianness 147
70 Address frame format 149
71 Address frame types 150
72 IDENTIFY address frame format 150
73 Device types 151
74 OPEN address frame format 152
75 Protocol 152
76 Connection rate 153
77 Arbitration wait time 154
78 Connection request responses 163
79 Pathway Recovery Priority 165
80 Edge expander device routing table 166
81 Abandon connection request responses 167
82 Break connection responses 169
83 Frame interlock requirements 191
84 SATA target port transmitting a frame 204
85 STP initiator port transmitting a frame 204
86 Confirmations to transport layer from the PL_OC state machine 220
87 Retry Frame conditions 225
88 SSP frame format 228
89 FRAME TYPE field 229
90 COMMAND information unit 230
91 TASK ATTRIBUTE field 231
92 TASK information unit 231
93 Task management functions 232
94 XFER_RDY information unit 233
95 DATA information unit 233
96 RESPONSE information unit 235
97 DATAPRES field 235
98 RESPONSE DATA field 236
99 RESPONSE CODE field 237
100 Delivery Failure to Command Complete Received mapping 249
101 SMP frame types 260
102 SMP_REQUEST frame format 260
103 SMP_RESPONSE frame format 261
104 Function results 261
105 SCSI architecture mapping 267
106 Send SCSI Command transport protocol service arguments 268
107 SCSI Command Received transport protocol service arguments 268
108 Send Command Complete transport protocol service arguments 269
109 Command Complete Received transport protocol service arguments 270
110 Send Data-In transport protocol service arguments 271
111 Data-In Delivered transport protocol service arguments 271
112 Receive Data-Out transport protocol service arguments 272

113 Data-Out Received transport protocol service arguments 272
114 Send Task Management Request transport protocol service arguments 273
115 Task Management Request Received transport protocol service arguments 273
116 Task Management Function Executed transport protocol service arguments 274
117 Received Task Management Function-Executed transport protocol service arguments 275
118 SCSI transport protocol events 276
119 Disconnect-Reconnect mode page for SSP 278
120 Protocol-Specific Port Control mode page subpages 280
121 Protocol-Specific Port Control mode page for SAS SSP - short format 280
122 Protocol-Specific Port Control mode page for SAS SSP - Phy Control And Discover subpage 281
123 SAS phy mode descriptor 282
124 Protocol-Specific log page for SAS 283
125 Protocol-Specific log parameter format for SAS 284
126 Parameter control bits for SAS log parameters 285
127 SAS phy log descriptor 286
128 Device Identification VPD page required identification descriptors 292
129 Management functions 294
130 REPORT GENERAL request 295
131 REPORT GENERAL response 296
132 REPORT MANUFACTURER INFORMATION request 297
133 REPORT MANUFACTURER INFORMATION response 298
134 DISCOVER request 299
135 DISCOVER response 300
136 Function results for DISCOVER 301
137 Routing attributes 301
138 Attached device types 301
139 Negotiated physical link rate 302
140 Hardware and programmed physical link rates 303
141 REPORT PHY ERROR LOG request 303
142 REPORT PHY ERROR LOG response 304
143 Function results for REPORT PHY ERROR LOG 305
144 REPORT PHY SATA request 305
145 REPORT PHY SATA response 306
146 Function results for REPORT PHY SATA 307
147 REPORT ROUTE INFORMATION request 308
148 REPORT ROUTE INFORMATION response 309
149 Function results for REPORT ROUTE INFORMATION 310
150 CONFIGURE ROUTE INFORMATION request 311
151 CONFIGURE ROUTE INFORMATION response 312
152 Function results for CONFIGURE ROUTE INFORMATION 312
153 PHY CONTROL request 313
154 Phy operation 314
155 Programmed physical link rate 315
156 PHY CONTROL response 315
157 Function results for PHY CONTROL 316
A.1 CJTPAT for RD+ 317
A.2 CJTPAT for RD- 318
A.3 CJTPAT for RD+ and RD- 319
A.4 CJTPAT scrambled in an SSP DATA frame 320
C.1 CRC examples 329
D.1 Monte-Carlo simulation results 330
D.2 Hash results for realistic SAS addresses 333
D.3 Hash results for a walking ones pattern 334
D.4 Hash results for a walking zeros pattern 335
E.1 Scrambler examples 338
G.1 Column descriptions for connection examples 343
H.1 Primitives with Hamming distance of 8 360

I.1 C program files 363

Figures

Page
1 SCSI document relationships 1
2 ATA document relationships 1
3 State machine conventions 16
4 SAS object model 20
5 Physical links and phys 21
6 Ports (narrow ports and wide ports) 22
7 SAS devices 23
8 Initiator device 24
9 Target device 24
10 Expander device 25
11 Edge expander device set 26
12 Domains and connections 27
13 Devices spanning domains 27
14 Maximum expander device topology 28
15 Fanout expander device topology 29
16 Edge expander device set topology 29
17 Multiple connections on wide ports 31
18 Pathways 32
19 State machines 35
20 Transmit data path and state machines 36
21 SSP link, SSP transport, and SCSI application layer state machines 37
22 SMP link, SMP transport, and management application layer state machines 38
23 STP link, STP transport, and ATA application layer state machines 39
24 Reset terminology 48
25 Expander device model 50
26 Expander device interfaces 52
27 Expander device interface detail 53
28 Expander route table example 58
29 Expander route index levels 60
30 Expander route index order 62
31 SATA cables and connectors (informative) 65
32 SAS cables and connectors - external environment 66
33 SAS cables and connectors - internal environment 66
34 SAS itnernal cable assembly and destination pin assignments 70
35 Transmitter transient test circuit 73
36 Receiver transient test circuit 73
37 Eye mask at IR, CR, and XR 75
38 Deriving a tolerance mask at IR, CR, or XR 75
39 Sinusoidal jitter mask 76
40 Compliance interconnect test load 83
41 Zero-length test load 83
42 ISI loss example at 3,0 Gbps 84
43 ISI loss example at 1,5 Gbps 84
44 SAS bit transmission logic 93
45 SAS bit reception logic 94
46 OOB signal transmission 96
47 OOB signal detection 98
48 SATA OOB sequence (informative) 99
49 SATA speed negotiation sequence (informative) 99
50 SAS to SATA OOB sequence 100
51 SAS to SAS OOB sequence 102
52 SAS speed negotiation window 103
53 SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G2 only) 105
54 SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G1, G2) with failure 106
55 Hot-plug and the phy reset sequence 107

56 SAS phy (SP) state machine - OOB sequence states 109
57 SAS phy (SP) state machine - SAS speed negotiation states 113
58 SAS phy (SP) state machine - SATA host emulation states 118
59 SAS phy dword synchronization (SP_DWS) state machine 122
60 Repeated primitive sequence 135
61 Triple primitive sequence 135
62 Redundant primitive sequence 136
63 Elasticity buffers 143
64 CRC generator bit order 146
65 Transmit path bit ordering 148
66 Receive path bit ordering 149
67 SAS link layer identification and hard reset (SL_IR) state machines 156
68 Test modes 161
69 BREAK usage 168
70 Connection request timeout example 169
71 Closing an SSP connection example 170
72 SAS link layer (SL) state machine (part 1) 172
73 SAS link layer (SL) state machine (part 2) 173
74 Expander link layer (XL) state machine (part 1) 181
75 Expander link layer (XL) state machine (part 2) 182
76 Expander link layer (XL) state machine (part 3) 183
77 Rate matching example 190
78 SSP frame transmission 191
79 Interlocked frames 192
80 Non-interlocked frames with the same tag 192
81 Non-interlocked frames with different tags 193
82 SSP link layer (SSP) state machines (part 1 - frame transmission) 195
83 SSP link layer (SSP) state machines (part 2 - frame reception) 196
84 SSP link layer (SSP) state machines (part 3 - primitive transmission) 197
85 STP frame transmission 203
86 STP expander device buffering and HOLD/HOLDA handshake 206
87 SMP frame transmission 207
88 SMP link layer (SMP) state machines – initiator device 209
89 SMP link layer (SMP) state machines – target device 210
90 Port layer state machines 213
91 Port layer overall control (PL_OC) state machine 216
92 Port layer phy manager (PL_PM) state machine (part 1) 222
93 Port layer phy manager (PL_PM) state machine (part 2) 223
94 Task management function sequence 237
95 Write command sequence 238
96 Read command sequence 238
97 Bidirectional command sequence 239
98 SSP transport layer (ST) state machines - initiator device 244
99 SSP transport layer (ST) state machines - target device 251
100 SMP request/response sequence 260
101 SMP transport layer state machine - initiator device (MT_ID) 263
102 SMP transport layer (MT) state machines - target device 265
103 SCSI application layer power condition (SA_PC) state machine for SAS 288
B.1SAS speed negotiation sequence (phy A: G1 only, phy B: G1 only) 324
B.2SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G1, G2) 325
C.1CRC generator example 326
C.2CRC checker example 326
D.1BCH(69, 39, 9) code generator 331
E.1Scrambler 336
F.1STP CRC bit ordering 340
F.2STP transmit path bit ordering 341
F.3STP receive path bit ordering 342

G.1Example topology 343
G.2Open accept 345
G.3Open reject by end device 346
G.4Open reject by expander device 347
G.5Arbitration lost 348
G.6Backoff and retry 349
G.7Backoff and reverse path 350
G.8Connection close - single step 351
G.9Connection close - simultaneous 352
G.10Break handling during path arbitration 353
G.11Break handling during connection 354
G.12STP connection - originated by STP initiator port 355
G.13STP connection - originated by expander device 356
G.14STP connection close - originated by STP initiator port 357
G.15STP connection close - originated by expander device 358
G.16Partial pathway recovery 359
J.1SAS logo 382

Foreword (This foreword is not part of this standard)

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 of it approved this standard, INCITS had the following members:

<<Insert INCITS member list after T10 letter ballot>>

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

<<Insert T10 member list after T10 letter ballot>>

Introduction

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

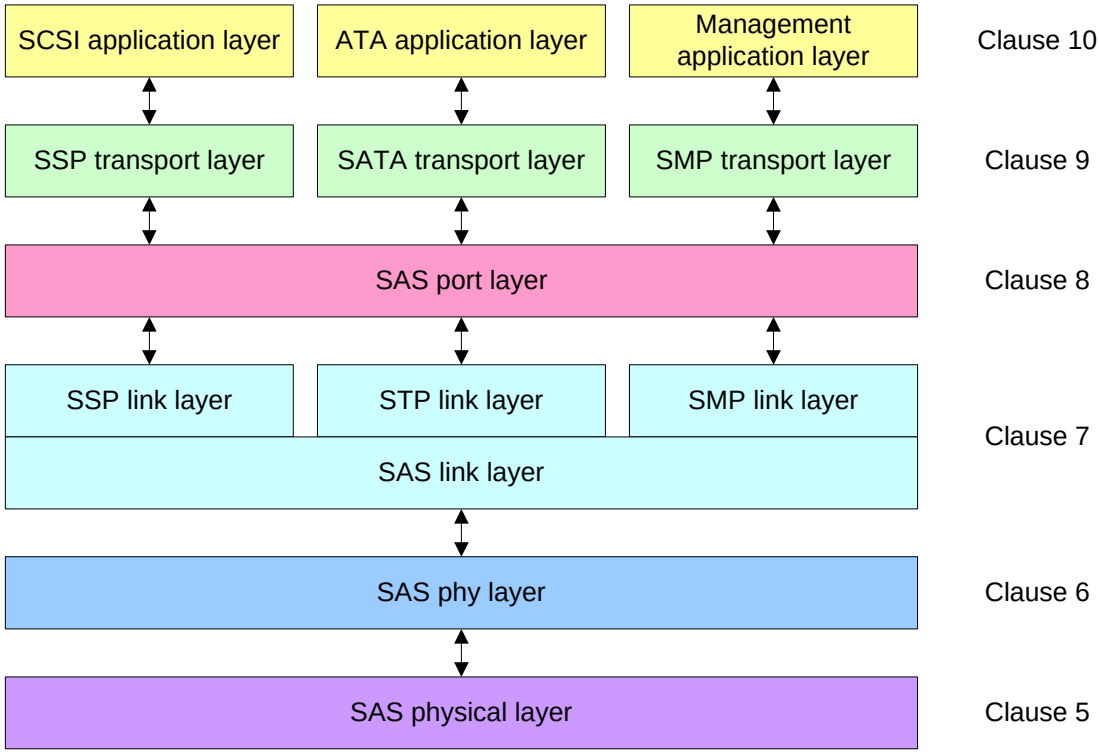
a) Serial SCSI Protocol (SSP): a mapping of SCSI supporting multiple initiators and targets;
b) Serial ATA Tunneled Protocol (STP): a mapping of Serial ATA expanded to support multiple initiators and targets; and
c) Serial Management Protocol (SMP): a management protocol.

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 interconnect layer. This includes connectors, cables, backplanes, and the driver and receiver electrical characteristics.
Clause 6 (Phy layer) describes the phy layer. It describes encoding, bit order, out of band (OOB) signals, phy reset sequences, SAS phy layer state machines, and spin-up.
Clause 7 (Link layer) describes the link layer. It describes primitives, clock skew management, idle links, CRC, scrambling, address frames, identification sequence, power management, tests, domain changes, connections, rate matching, SSP, STP, and SMP link layer specifics, 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 machine descriptions.
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. SCSI, ATA, and management specific features are described (e.g., SCSI mode pages and log pages).

Normative Annex A (Compliant jitter test pattern (CJTPAT)) describes the jitter test patterns.
Informative Annex B (SAS phy reset sequence examples) provides phy reset sequence examples.
Informative Annex C (CRC) provides information on the CRC algorithm.
Informative Annex D (SAS address hashing) provides information on the hashing algorithm.
Informative Annex E (Scrambling) provides information on the scrambling algorithm.
Informative Annex F (ATA architectural notes) describes ATA architectural issues.
Informative Annex G (Expander handling of connections) describes expander behavior in a variety of connection examples.
Informative Annex H (Primitive encoding) lists the primitive encodings available for future versions of this standard.
Informative Annex I (Discover process example implementation) provides an example implementation of the discover process.
Informative Annex J (SAS logo) defines the SAS logo.

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 (SAS)

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.

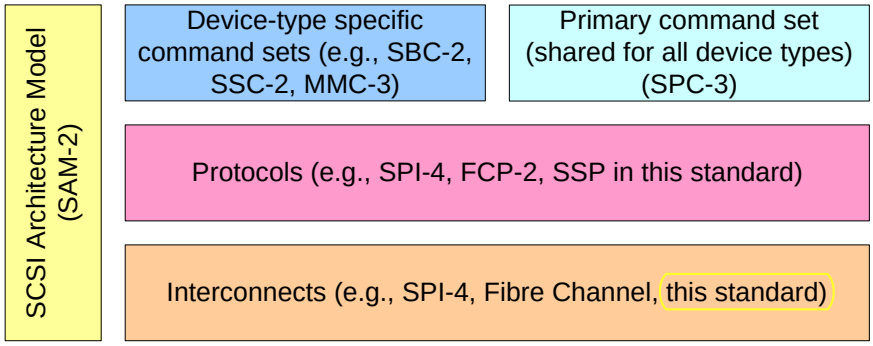


Figure 1 — SCSI document relationships

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

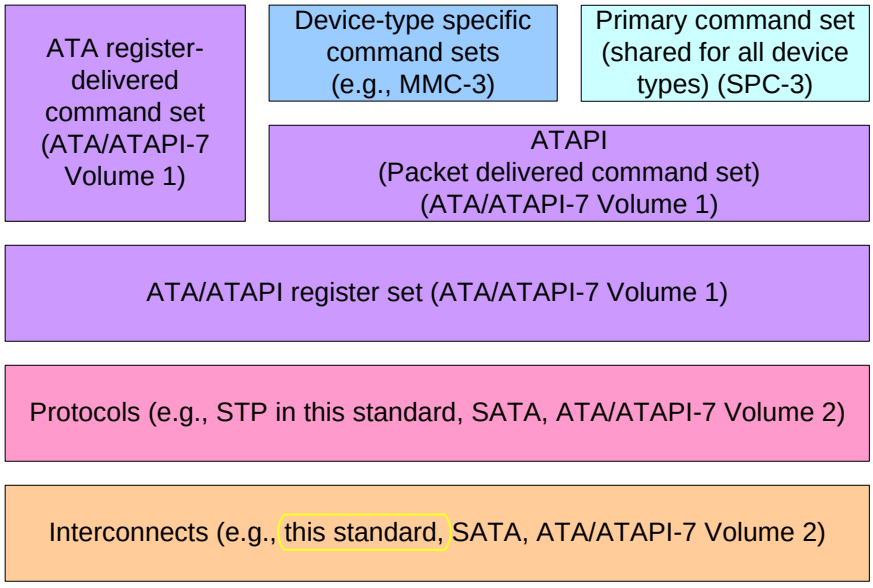


Figure 2 — ATA document relationships

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

The following standards 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 (ISO, IEC, CEN/CENELEC, ITUT); and
c) approved and draft foreign standards (including BSI, 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
BSI	British Standards Institution	http://www.bsi-global.com
CEN	European Committee for Standardization	http://www.cenorm.be
CENELEC	European Committee for Electrotechnical Standardization	http://www.cenelec.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
ISO	International Standards Organization	http://www.iso.ch
ITI	Information Technology Industry Council	http://www.itic.org
ITUT	International Telecommunications Union Telecommunications Standardization Sector	http://www.itu.int
JIS	Japanese Industrial Standards Committee	http://www.jisc.org
INCITS	International Committee for Information Technology Standards	http://www.incits.org

2.2 Approved references

At the time of publication, there were no approved references.

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.

ANSI INCITS.xxx, *AT Attachment with Packet Interface-7 Volume 1 (ATA/ATAPI-7 V1) standard* (T13/1532-D)

ANSI INCITS.xxx, *AT Attachment with Packet Interface-7 Volume 2 (ATA/ATAPI-7 V2) standard* (T13/1532-D)

ISO/IEC 14776-xxx, *SCSI Architecture Model-3 standard* (T10/1561-D)

ISO/IEC 14776-xxx, *SCSI Primary Commands-3 standard* (T10/1416-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 this document, 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: High Speed Serialized AT Attachment (SATA). Revision 1.0, 29 August 2001

Serial ATA 1.0 Errata. 5 April 2002

Serial ATA 1.0 Design Guides. 5 April 2002

Serial ATA II: Extensions to Serial ATA 1.0. Revision 1.0. 16 October 2002

NOTE 2 For information on the current status of the Serial ATA documents, contact the Serial ATA Working Group at info@serialata.org. To obtain copies of these documents, see http://www.serialata.org.

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-8460, *HSS Backplane Design Guidelines*

SFF-8470, *Multi Lane Copper Connector*

SFF-8482, *SAS Plug Connector*

NOTE 3 For more information on the current status of the SFF documents, 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.

3 Definitions, symbols, abbreviations, keywords, and conventions

3.1 Definitions

3.1.1 application client: An object that is the source of SCSI commands (see SAM-3), ATA commands, or management function requests.

3.1.2 AT Attachment (ATA): A standard for the internal attachment of storage devices to host systems (see ATA/ATAPI-7 V1).

3.1.3 ATA device: A device that contains one or more ATA ports that are connected to a service delivery subsystem and supports an ATA application protocol (see 4.1.1).

NOTE 4 ATA uses the term device where this standard uses the term target device (see ATA/ATAPI V1).

3.1.4 ATA domain: An I/O system consisting of a set of ATA devices that communicate with one another by means of a service delivery subsystem (see 4.1.1).

3.1.5 ATA initiator device: An ATA device containing application clients and ATA initiator ports that originate device service and task management requests to be processed by an ATA target device. Equivalent to a host system in ATA (see ATA/ATAPI V1).

3.1.6 ATA initiator port: An ATA initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests and responses are routed. Equivalent to a host adapter in ATA (see ATA/ATAPI V1).

3.1.7 ATA port: An ATA initiator port or an ATA target port (see 4.1.1).

3.1.8 ATA target device: An ATA device containing logical units and ATA target ports that receives device service and task management requests for processing. Equivalent to a device in ATA (see ATA/ATAPI V1).

3.1.9 ATA target port: An ATA target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which requests and responses are routed. Equivalent to a device in ATA (see ATA/ATAPI V1).

3.1.10 ATA target/initiator device: A device that has all the characteristics of an ATA target device and an ATA initiator device.

3.1.11 ATA target/initiator port: An ATA target/initiator device object that has all the characteristics of an ATA target port and an ATA initiator port.

3.1.12 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).

3.1.13 broadcast primitive processor (BPP): The portion of an expander function that manages broadcast primitives (see 4.6.5).

3.1.14 byte: A sequence of 8 contiguous bits considered as a unit.

3.1.15 character: A sequence of 10 contiguous bits considered as a unit. A byte is encoded as a character using 8b10b coding (see 6.2).

3.1.16 configurable expander device: An expander device that contains an expander route table that is configured with expander route entries (see 4.1.8.3).

3.1.17 confirmation: A parameter passed from a lower layer state machine to a higher layer state machine.

3.1.18 connection: A temporary association between an initiator port and a target port (see 7.12). During a connection all transmitted dwords are associated with the I_T nexus formed by that initiator port and target port.

3.1.19 connection rate: The effective rate of dwords through the pathway between an initiator phy and a target phy, established through the connection request.

3.1.20 control character (Kxx.y): A character that does not represent a byte of data (see 6.2).

3.1.21 cyclic redundancy check (CRC): An error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum (see 7.4).

3.1.22 data character (Dxx.y): A character representing a byte of data (see 6.2).

3.1.23 data dword: A dword that starts with a Dxx.y (data character).

3.1.24 deterministic jitter: All jitter from sources that do not have tails on their probability distribution functions (i.e., values outside the bounds have probability of zero). Includes duty cycle distortion, data dependent, sinusoidal, and uncorrelated (to the data) bounded.

~~**3.1.25 device:** A physical entity.~~

3.1.26 device server: An object within a target device that processes SCSI tasks (see SAM-3), ATA commands, or management functions.

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

3.1.28 direct routing attribute: The attribute of an expander phy that allows it to be used by the expander connection manager to route a connection request to an end device (see 4.6.11.1).

3.1.29 direct routing method: The method the expander connection manager uses to establish a connection with an end device (see 4.6.11.1).

3.1.30 discover process: The management application client that configures the domain (see 4.6.11.5).

~~**3.1.31 domain:** Synonymous with SAS domain.~~

~~**3.1.32 downstream phy:**~~ The next phy along the pathway of a connection between two SAS phys in the same direction as the direction frame transmission. The term is relevant only in contexts where the ~~primary~~ direction of frame transmission is clear.

3.1.33 dword: A sequence of four contiguous bytes or four contiguous characters considered as a unit.

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

3.1.35 edge expander device: An expander device containing phys with subtractive routing attributes (see 4.1.8.1).

3.1.36 edge expander device set: A group of one or more edge expander devices (see 4.1.8.2).

3.1.37 end device: An initiator device or target device that is not contained within an expander device.

3.1.38 expander connection manager (ECM): The portion of an expander function that manages routing (see 4.6.3).

3.1.39 expander connection router (ER): The portion of an expander function that routes signals between expander phys (see 4.6.4).

3.1.40 expander device: A device that contains expander ports. An expander device receives primitives, device service requests, and task management requests on one expander port and routes those requests to another expander port. An expander device supports SMP and may also contain a SCSI device and/or an ATA device.

3.1.41 expander function: The portion of an expander device that contains an expander connection manager, expander connection router, and broadcast primitive processor (see 4.6.1).

3.1.42 expander phy: A phy in an expander device.

3.1.43 expander port: A SAS expander device object that routes SSP, SMP, and STP frames to and from physical links or to internal initiator ports and/or target ports. Contains one or more phys.

3.1.44 expander route entry: A single destination SAS address and a method to enable or disable the route entry within an expander route table.

3.1.45 expander route index: A value used in combination with the phy identifier to select a expander route entry within an expander route table.

3.1.46 expander route table: A table of expander route entries within an expander device. The table is used by the expander function to resolve connection requests (see 4.6.11.3).

3.1.47 fanout expander device: An expander device containing no phys with subtractive routing attributes (see 4.1.8.1).

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

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

3.1.50 frame information structure (FIS): The SATA frame format (see SATA).

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

3.1.52 hard reset sequence: A sequence ~~following the SAS speed negotiation sequence~~ that causes a hard reset (see 4.4).

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

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

3.1.55 hash function: A mathematical function that maps values from a larger domain into a smaller domain, and that reduces a long value into a hashed value.

3.1.56 I_T nexus: A nexus that exists between an initiator port and a target port.

3.1.57 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.

3.1.58 I_T_L_Q nexus: A nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a queue tag following the successful receipt of a queue tag. This relationship extends the prior I_T nexus or I_T_L nexus.

3.1.59 identification sequence: A sequence ~~following the SAS speed negotiation sequence~~ where SAS devices exchange IDENTIFY address frames (see 4.4).

3.1.60 idle dword: A vendor-specific data dword that is scrambled and is transmitted outside a frame (see 7.5).

3.1.61 idle time: The part of an OOB signal where ALIGNs are not being transmitted (see 5.7.4).

3.1.62 indication: A parameter passed from a lower layer state machine to a higher layer state machine.

3.1.63 information unit (IU): Portion of an SSP frame that carries command, task management function, data, response, or transfer ready information.

3.1.64 initiator device: Synonymous with SAS initiator device (see 3.1.102).

3.1.65 initiator phy: A phy in an initiator device.

3.1.66 initiator port: Synonymous with SAS initiator port (see 3.1.103).

3.1.67 invalid dword: A dword with an illegal character, with a control character in other than the first character position, with a control character other than K28.5 or K28.3 in the first character position, or with a running disparity error.

3.1.68 jitter: Abrupt and unwanted variations in the interval between successive pulses.

3.1.69 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).

3.1.70 link: A physical link.

3.1.71 link reset: Performing the link reset sequence (see 4.4).

3.1.72 link reset sequence: For SATA, a phy reset sequence. For SAS, a phy reset sequence followed by ~~an identification sequence, or a phy reset sequence followed by a hard reset sequence, another phy reset sequence, and an identification sequence~~ (see 4.4).

3.1.73 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).

3.1.74 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).

3.1.75 narrow link: A physical link that attaches a narrow port to another narrow port (see 4.1.3).

3.1.76 narrow port: A port that contains exactly one phy (see 4.1.3).

3.1.77 negotiated physical link rate: The current operational physical link rate established after speed negotiation between two phys.

3.1.78 nexus: A relationship between a SCSI initiator port and a SCSI target port that may extend to a logical unit and a queue tag (see SAM-3).

3.1.79 object: An architectural abstraction or container that encapsulates data types, services, or other objects that are related in some way.

3.1.80 OOB sequence: A sequence where two phys exchange OOB signals. Part of the phy reset sequence (see 4.4).

3.1.81 out-of-band (OOB) signal: Pattern of ALIGNs and idle time used during the link reset sequence (see 6.5).

3.1.82 partial pathway: The set of ~~physical~~ links participating in a connection request which has not reached a SAS endpoint (i.e., the connection request has been transmitted by the source device and confirmed as received by at least one expander device with AIP)(see 4.1.12).

3.1.83 pathway: A set of ~~physical~~ links between a SAS initiator port and a SAS target port (see 4.1.12).

3.1.84 phy: A SAS device object that interfaces to a service delivery subsystem (see 4.1.2).

3.1.85 phy reset sequence: An OOB sequence followed by a speed negotiation sequence (see 4.4).

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

3.1.87 port: A SAS port or an expander port. Each port contains one or more phys (see 4.1.3).

3.1.88 power on: Power being applied.

3.1.89 primitive: A dword starting with K28.5 or K28.3 followed by three data characters (see 7.1).

3.1.90 primitive sequence: A set of primitives treated as a single entity (see 7.1.3).

3.1.91 programmed maximum physical link rate: The maximum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function or the Phy Control and Discover subpage); ~~defaults to the hardware maximum physical link rate.~~

3.1.92 programmed minimum physical link rate: The minimum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function or the Phy Control and Discover subpage); ~~defaults to the hardware maximum physical link rate.~~

3.1.93 random jitter: Jitter that is assumed to have a Gaussian distribution.

3.1.94 rate: Data transfer rate of a physical link (e.g., 1,5 Gbps or 3,0 Gbps).

3.1.95 reflection coefficient (Γ): The reflection coefficient of the transmission media (i.e., the ratio of the reflected voltage divided by the voltage applied to the transmission media).

3.1.96 request: A parameter passed from a higher layer state machine to a lower layer state machine.

3.1.97 reset event: An event that triggers a hard reset from a SAS device (see 4.4.2).

3.1.98 response: A parameter passed from a higher layer state machine to a lower layer state machine.

3.1.99 SAS address: A worldwide unique name assigned to an initiator port, target port, expander device, initiator device, or target device (see 4.2.2).

3.1.100 SAS device: A SCSI device, an ATA device, or an expander device in a SAS domain (see 4.1.4).

3.1.101 SAS domain: The I/O system defined by this standard that may serve as an ATA domain and/or a SCSI domain (see 4.1.9).

3.1.102 SAS initiator device: A SCSI initiator device or an ATA initiator device in a SAS domain (see 4.1.5).

3.1.103 SAS initiator port: A SCSI initiator port or an ATA initiator port in a SAS domain.

3.1.104 SAS port: A SAS initiator port or a SAS target port.

3.1.105 SAS primitive: A primitive using a K28.5 for the control character.

3.1.106 SAS target device: A SCSI target device or an ATA target device in a SAS domain (see 4.1.6).

3.1.107 SAS target port: A SCSI target port or an ATA target port in a SAS domain.

3.1.108 SAS target/initiator device: A SCSI target/initiator device or an ATA target/initiator device in a SAS domain (see 4.1.7).

3.1.109 SAS target/initiator port: A SCSI target/initiator port or an ATA target/initiator port in a SAS domain.

3.1.110 SATA domain: The I/O system defined by SATA that serves as an ATA domain.

3.1.111 SATA target port: An ATA target port containing one phy in a SAS domain or a SATA domain.

3.1.112 scrambling: Modifying data by XORing each bit with a pattern generated by a linear feedback shift register to minimize repetitive character patterns (see 7.5).

3.1.113 SCSI device: A device that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol (see SAM-3).

3.1.114 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 SAM-3).

3.1.115 SCSI initiator device: A SCSI device containing application clients and SCSI initiator ports that originate 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-3).

3.1.116 SCSI initiator port: A SCSI initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests and responses are routed (see SAM-3).

3.1.117 SCSI port: A SCSI initiator port or a SCSI target port (see SAM-3).

3.1.118 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-3).

3.1.119 SCSI target port: A SCSI target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which requests and responses are routed (see SAM-3).

3.1.120 SCSI target/initiator device: A device that has all the characteristics of a SCSI target device and a SCSI initiator device (see SAM-3).

3.1.121 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-3).

3.1.122 Serial ATA (SATA): The protocol defined by SATA.

3.1.123 Serial ATA Tunneled Protocol (STP): The protocol defined in this standard used by ATA initiator ports to communicate with SATA target ports in a SAS domain.

3.1.124 Serial Attached SCSI (SAS): The protocol defined by this standard.

3.1.125 Serial Management Protocol (SMP): The protocol defined in this standard used by SAS devices to communicate management information with other SAS devices in a SAS domain.

3.1.126 Serial SCSI Protocol (SSP): The protocol defined in this standard used by SCSI initiator ports to communicate with SCSI target ports in a SAS domain.

3.1.127 service delivery subsystem: That part of a SCSI or ATA I/O system that transmits service requests to a logical unit or target port and returns logical unit or target port responses to an initiator port.

3.1.128 speed negotiation sequence: A sequence where two phys determine the highest common supported physical link rate. ~~Part of the phy reset sequence~~ (see 4.4).

~~**3.1.129 spread spectrum clocking (SSC):** A signaling technique where the frequency of a clock is varied over time to reduce the peaks but increase the frequency spectrum of electromagnetic emissions.~~

3.1.130 SSP initiator port: A SCSI initiator port containing one or more phys in a SAS domain.

3.1.131 SSP target port: A SCSI target port containing one or more phys in a SAS domain.

3.1.132 STP initiator port: An ATA initiator port containing one or more phys in a SAS domain.

3.1.133 STP target port: An ATA target port containing one or more phys in a SAS domain.

3.1.134 subtractive routing attribute: The attribute of an edge expander phy that allows it to be used by the expander connection manager to route connection requests not resolved using the direct routing method or table routing method (see 4.6.11.1).

3.1.135 subtractive routing method: The method the expander connection manager uses to route connection requests not resolved using the direct routing method or table routing method (see 4.6.11.1).

3.1.136 table routing attribute: The attribute of an expander phy that allows it to be used by the expander connection manager to route connection requests using an expander route table (see 4.6.11.1).

3.1.137 table routing method: The method the expander connection manager uses to route connection requests using an expander route table (see 4.6.11.1).

3.1.138 target device: Synonymous with SAS target device (see 3.1.106).

3.1.139 target phy: A phy in a target device.

3.1.140 target port: Synonymous with SAS target port (see 3.1.107).

3.1.141 task: An object within the logical unit representing the work associated with a command or group of linked commands.

3.1.142 task management function: A task manager service capable of being requested by an application client to affect the processing of one or more tasks (see SAM-3).

3.1.143 task manager: An agent within the device server that processes task management functions (see SAM-3).

3.1.144 total jitter: Measured jitter including deterministic jitter and random jitter.

3.1.145 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 5.7.11).

3.1.146 transport protocol service confirmation: A parameter passed from the transport layer to the application layer that notifies the application layer that a SCSI transport protocol service has completed.

3.1.147 transport protocol service indication: A parameter passed from the transport layer to the application layer notifying the application layer to begin a SCSI transport protocol service.

3.1.148 transport protocol service request: A parameter passed from the application layer to the transport layer to begin a SCSI transport protocol service.

3.1.149 transport protocol service response: A parameter passed from the application layer to the transport layer that completes the SCSI transport protocol service.

3.1.150 unit interval: The time required to transmit one bit on a physical link (e.g., 666,667 ps at 1,5 Gbps and 333,333 ps at 3,0 Gbps).

~~**3.1.151 upstream phy:**~~ The next phy along the pathway of a connection between two SAS phys in the direction opposite the direction of frame transmission. The term is relevant only in contexts where the ~~primary~~ direction of frame transmission is clear.

3.1.152 valid dword: A dword that is not an invalid dword.

3.1.153 wide link: A group of physical links that attaches a wide port to another wide port (see 4.1.3).

3.1.154 wide port: A port that contains more than one phy (see 4.1.3).

3.2 Symbols and abbreviations

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

Abbreviation	Meaning
A.C.	alternating current
ACK	acknowledge
AIP	arbitration in progress primitive
ATA	AT attachment
ATAPI	AT attachment packet interface
AWG	American wire gauge
AWT	arbitration wait time
BER	bit error rate
BPP	broadcast primitive processor
CDB	command descriptor block
CRC	cyclic redundancy check
dB	decibel
D.C.	direct current
Dxx.y	data character
ECM	expander connection manager
ECR	expander connection router
~~EOAF~~	~~end of address frame primitive~~
~~EOF~~	~~end of frame primitive~~
FIS	frame information structure
G1	generation 1 physical link rate (1,5 Gbps)
G2	generation 2 physical link rate (3,0 Gbps)
G3	generation 3 physical link rate (defined in a future version of this standard)
Gbps	gigabits per second (10^9 bits per second)
IU	information unit

Abbreviation	Meaning
kHz	kilohertz (10^3 bits per second)
Kxx.y	control character
LED	light-emitting diode
LSB	least significant bit
LUN	logical unit number
µA	microampere (10^{-6} amperes)
µs	microsecond (10^{-6} seconds)
Mbaud	megabaud (10^6 transitions per second)
MBps	megabytes per second (10^6 bytes per second)
MHz	megahertz (10^6 bits per second)
MSB	most significant bit
ms	millisecond (10^{-6} seconds)
mV	millivolt (10^{-6} volts)
N/A	not applicable
NAA	name address authority
~~NAK~~	~~negative acknowledge~~ primitive
nF	nanofarad (10^{-6} Farads)
ns	nanosecond (10^{-9} seconds)
OOB	out-of-band
PLL	phase lock loop
P-P	peak-to-peak
PPT	partial pathway timer
ppm	parts per million (10^{-6})
ps	picosecond (10^{-12} seconds)
Γ	reflection coefficient (rho)
RCD	rate change delay
RRDY	receiver ready
SAM-3	SCSI Architecture Model - 3 standard
SAS	Serial Attached SCSI
SATA	Serial ATA
SCSI	Small Computer System Interface ~~-3~~ family of standards
SMP	Serial Management Protocol
SNLT	speed negotiation lock time
SNTT	speed negotiation transmit time
~~SOAF~~	~~start of address frame~~
~~SOF~~	~~start of frame~~
SPC-3	SCSI Primary Commands - 3 standard
SSC	spread spectrum clocking
SSP	Serial SCSI Protocol
STP	Serial ATA Tunneled Protocol
s	second (unit of time)
TCTF	transmitter compliance transfer function
UI	unit interval
V	volt
XOR	exclusive logical OR
^	exclusive logical OR
x	multiplication
/	division

3.3 Keywords

3.3.1 ignored: A keyword used to describe an unused bit, byte, word, field or code value. The contents or value of an ignored bit, byte, word, field or code value shall not be examined by the receiving SCSI device and may be set to any value by the transmitting SCSI device.

3.3.2 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.

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

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

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

3.3.6 need not:: Keywords indicating a feature that is not required to be implemented (equivalent to “is not required to”).

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

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

3.3.9 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 error.

3.3.10 restricted: A keyword referring to bits, bytes, words, and fields that are set aside for use in other SCSI standards. 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.

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

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

3.3.13 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, commands, SCSI mode pages, SCSI log pages, SCSI statuses, SCSI sense keys, and SCSI additional sense codes are in all uppercase (e.g., REQUEST SENSE).

Names of fields are in small uppercase (e.g., STATE OF SPARE). 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.

The ISO convention of numbering is used (i.e., the thousands and higher multiples are separated by a space and a comma is used as the decimal point). Table 1 shows a comparison of 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

~~Fields containing only one bit are usually referred to as the name bit instead of the name field.~~

Numbers that are not immediately followed by lower-case b or h are decimal values (e.g., 25).

Numbers immediately followed by lower-case b (e.g., 0101b) are binary values. Underscores may be included in binary values to increase readability or delineate field boundaries (e.g., 0101_1010b).

Numbers immediately followed by lower-case h (e.g., 15h) are hexadecimal values. A sequence of numbers and/or upper case letters 'A' through 'F' immediately followed by lower-case h (e.g., FA23h) are hexadecimal values. Underscores may be included in hexadecimal values to increase readability or delineate field boundaries (e.g., FD8C_FA23h).

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

3.5.1 State machine conventions overview

Figure 3 shows how state machines are described. See 4.3 for a summary of the state machines in this standard.

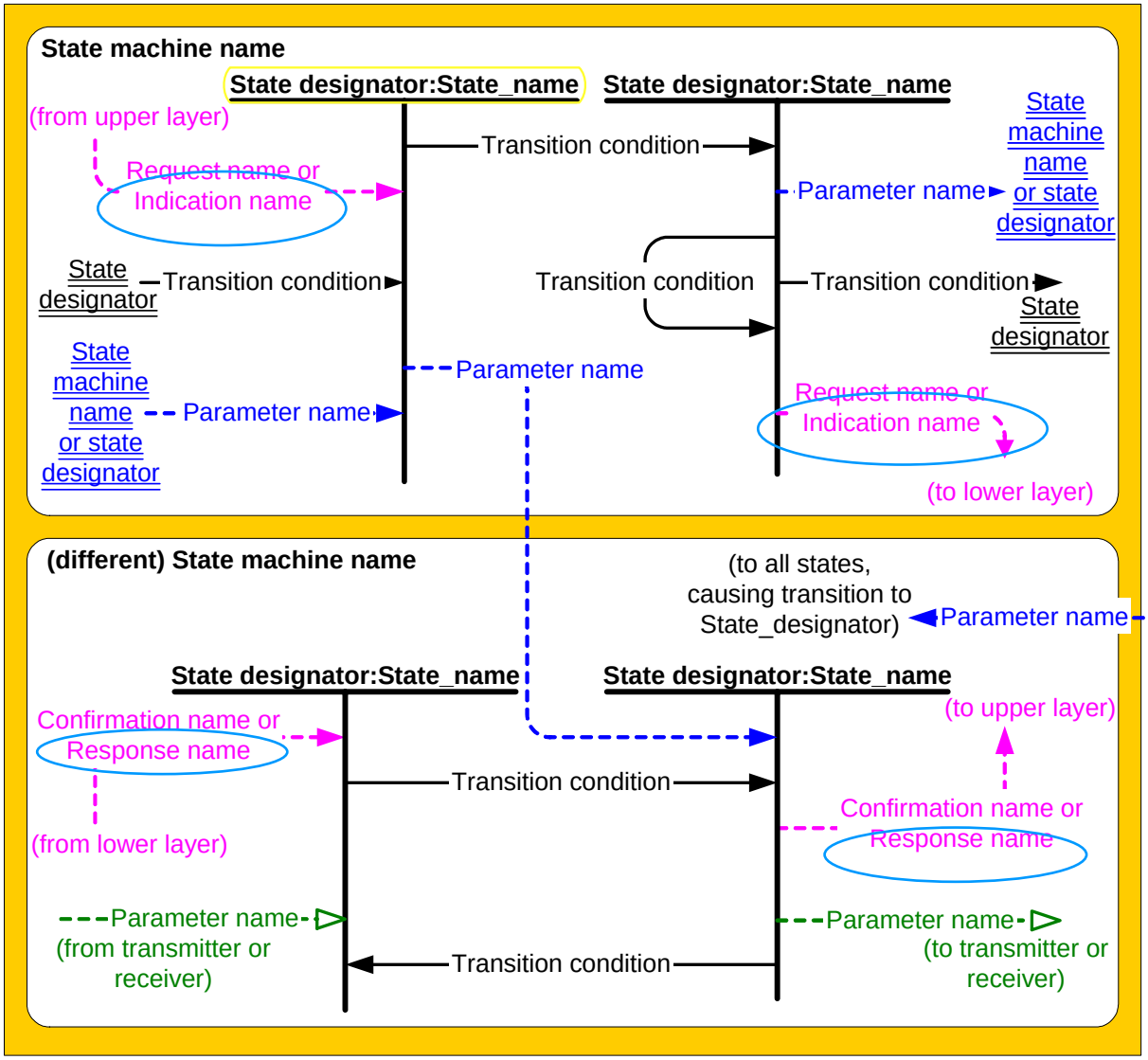


Figure 3 — State machine conventions

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

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.

3.5.2 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 to 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 in entry into the state. Transitions between states are instantaneous.

3.5.3 Parameters, requests, indications, confirmations, and responses

Parameters passed between state machines are shown with dashed lines labeled with a parameter name. When parameters 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 parameter is described in the state description text.

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

Parameters with unfilled arrowheads are passed to or from the transmitter or receiver, not shown in the state machine figures, and are directly related to data being sent onto or received from the physical link.

3.6 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 5 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.

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

A data dword consists of 32 bits. Table 3 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 3 — 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 4 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 4 — 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

3.7 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 enclosing optional or conditional parameters and arguments.

This notation allows data objects to be specified as inputs and outputs. The following is an example of a procedure specification:

Found = Search (IN (Pattern, Item List), OUT ([Item Found]))

Where:

Found = Flag
If set, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of Pattern object */
Object containing the search pattern.
Item List = Item<NN> /* Definition of Item List as an array of NN Item objects*/
Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure */
This object is only returned if the search succeeds.

4 General

4.1 Architecture

4.1.1 Architecture overview

This standard defines a SAS domain (see 4.1.9), which is an I/O system that may serve as an ATA domain and/or a SCSI domain:

a) An ATA domain is an I/O system consisting of a set of ATA devices that communicate with one another by means of a service delivery subsystem (see ATA/ATAPI-7 V1); and
b) A SCSI domain is an I/O system consisting of a set of SCSI devices that communicate with one another by means of a service delivery subsystem (see SAM-3).

A SAS device (see 4.1.4) is an ATA device or SCSI device with ports in a SAS domain:

a) ATA devices are either ATA initiator devices or ATA target devices (see ATA/ATAPI-7 V1); and
b) SCSI devices are either SCSI initiator devices, SCSI target devices, or SCSI target/initiator devices (see SAM-3).

SAS devices contain SAS ports (see 4.1.3) which interface to the service delivery subsystem through phys (see 4.1.2).

The service delivery subsystem in a SAS domain may include expander devices (see 4.1.8). Expander devices contain expander ports (see 4.1.3) which interface to the service delivery subsystem through phys (see 4.1.2).

Figure 4 shows the SAS object model.

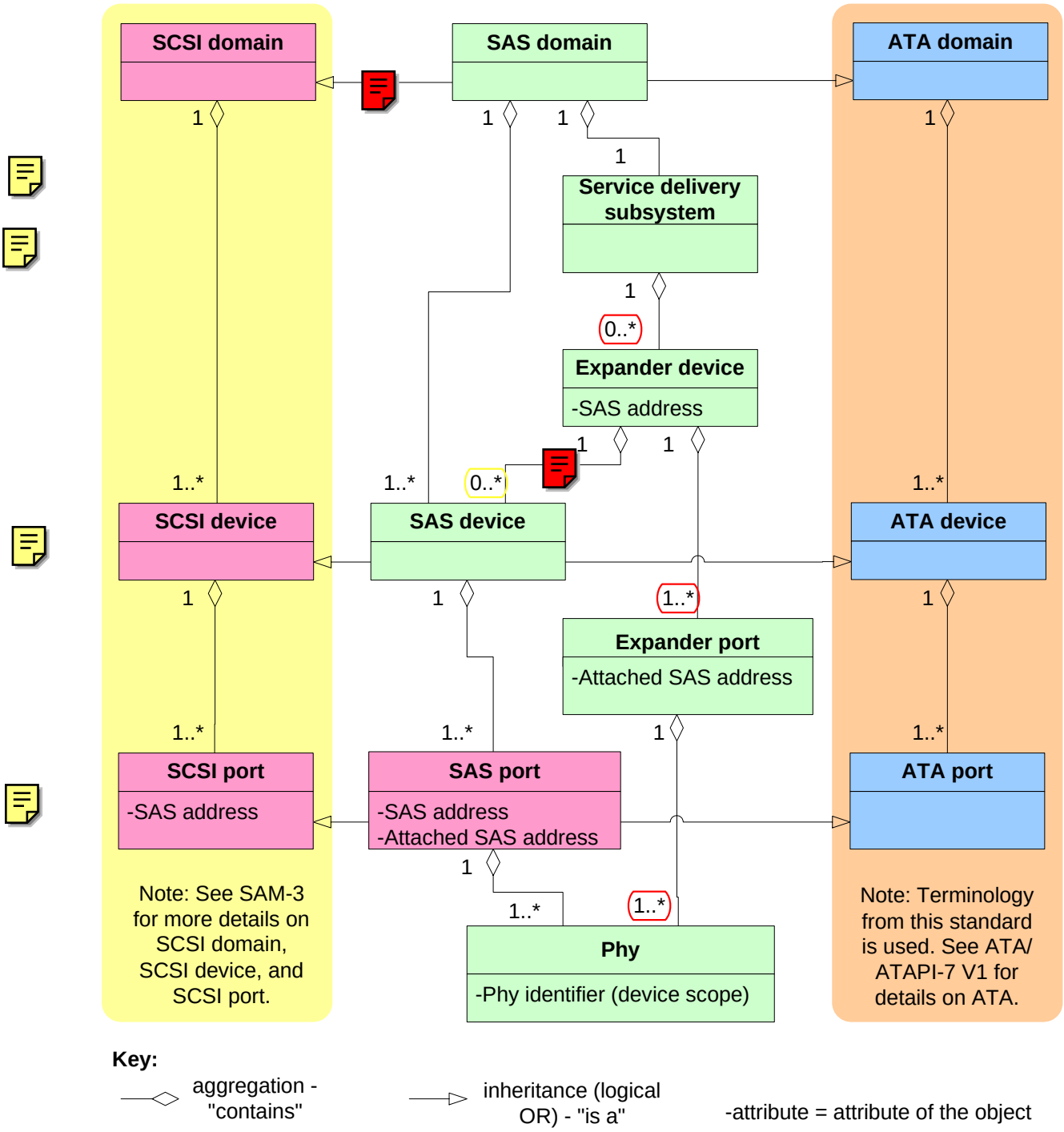


Figure 4 — SAS object model

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 phy is a transceiver; it is the object in a device that electrically interfaces to a physical link. Phys are contained in ports (see 4.1.3).

Figure 5 shows two phys attached with a physical link.

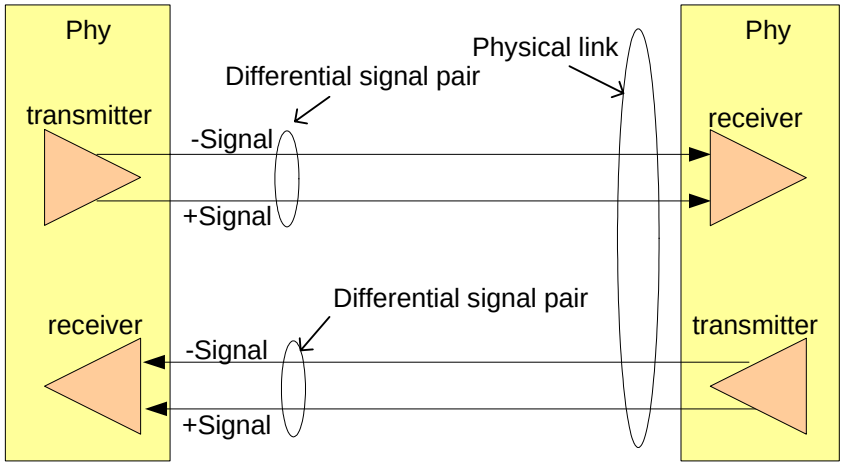


Figure 5 — Physical links and phys

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

A device may contain one or more phys. Each phy has a unique phy identifier (see 4.2.6) within the device.

~~Phys transmit and receive bits at physical link rates of 1,5 Gbps or 3,0 Gbps (see 5.7). The bits are part of dwords (see 6.1) which have been 8b10b coded into 10-bit characters (see 6.2).~~

4.1.3 Ports (narrow ports and wide ports)

A port may contain one or more phys. Ports in a device are associated with phys based on physical link initialization.

A port is created if:

a) one or more phys contained within a device share the same SAS address; and
b) the corresponding attached phy or phys share a SAS address.

A wide port is created if there is more than one phy in the port. A narrow port is a port with one phy.

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.

NOTE 6 Phys within a wide port may be attached to other phys in the same wide port, primarily for test purposes. This creates a wide link if more than one pair are so attached or a narrow link if only one pair are so attached.

Figure 6 shows examples of narrow ports and wide ports.

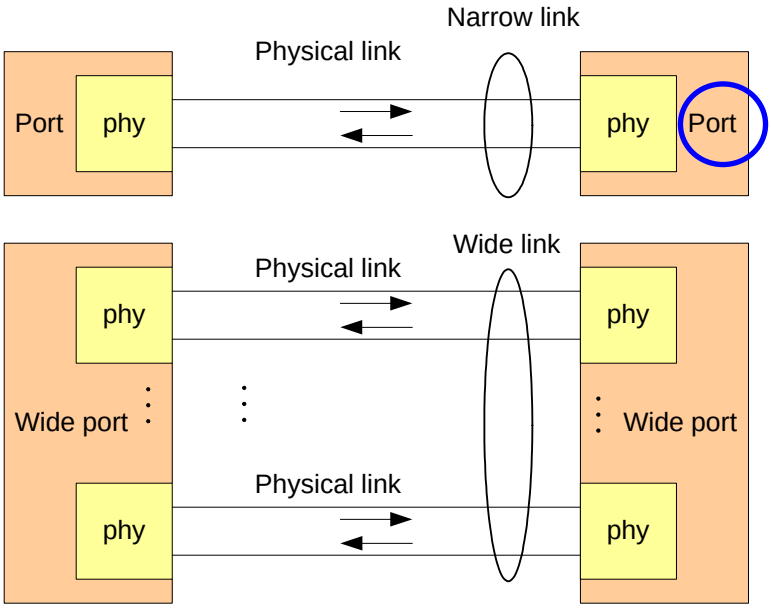


Figure 6 — Ports (narrow ports and wide ports)

4.1.4 SAS devices

A SAS device contains one or more ports, each containing one or more phys.

Figure 7 shows examples of devices with different port and phy configurations.

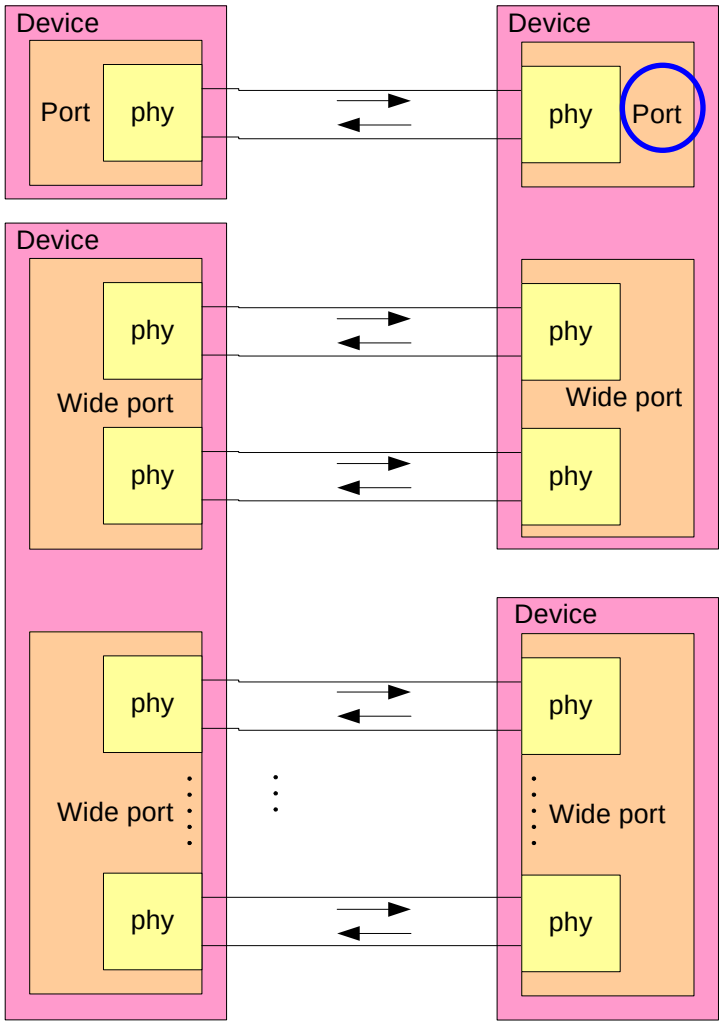


Figure 7 — SAS devices

In figures that show ports but no phys, the ports still contain phys and may or may not be wide ports.

4.1.5 Initiator devices

Initiator devices may support SCSI and/or ATA. Initiator devices include one or more initiator ports.

Initiator ports may support SSP and/or STP ~~and/or SATA.~~ Initiator ports shall support SMP in SAS domains. ~~Initiator ports supporting SATA are outside the scope of this standard.~~ SATA-only initiator ports are not supported in SAS domains.

Figure 8 shows an initiator device and the type of ports it may support.

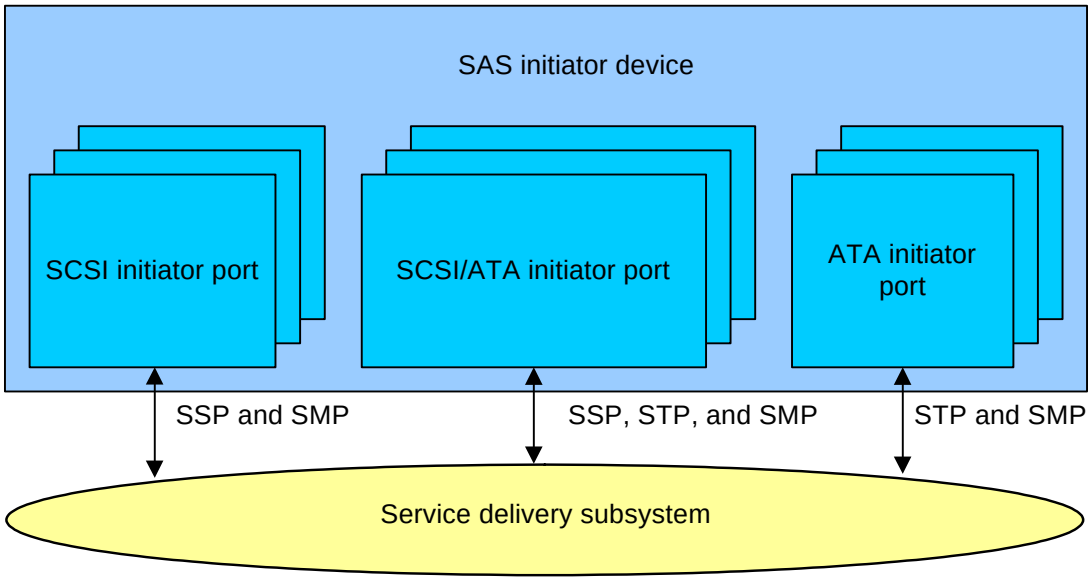


Figure 8 — Initiator device

4.1.6 Target devices

Target devices may support SCSI and/or ATA. Target devices include one or more target ports.

Target ports may support SSP and/or STP and/or SATA. SSP target ports and STP target ports may support SMP in SAS domains. SATA-only target ports may be included in SAS domains if the expander device to which they are attached supports STP.

Figure 9 shows a target device and the type of ports it may support.

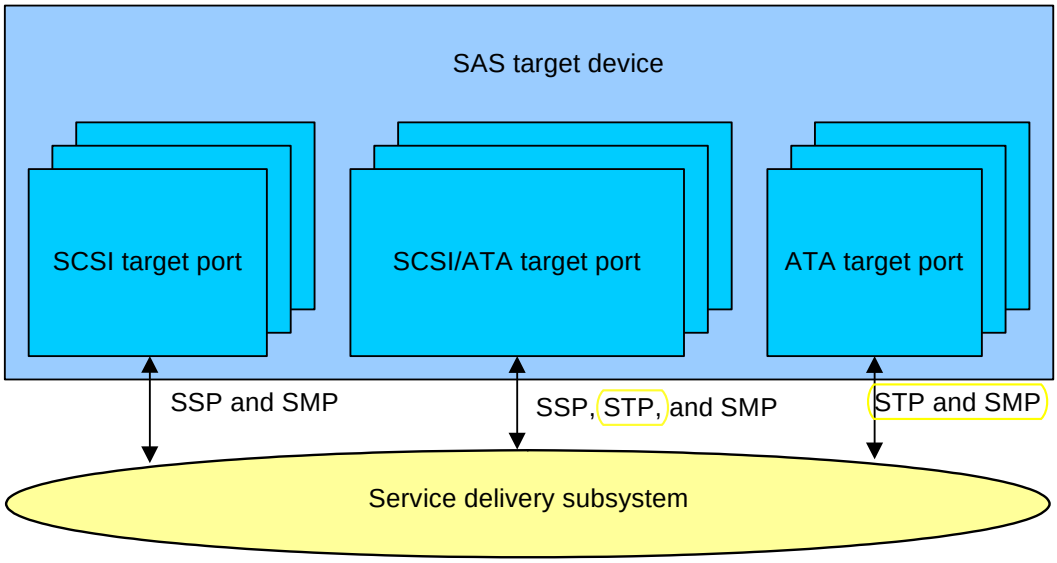


Figure 9 — Target device

4.1.7 Target/initiator devices

Target/initiator devices are devices that support both initiator device and target device roles. Target/initiator devices may have ports that support both initiator port and target port roles. Those ports are called target/initiator ports.

4.1.8 Expander devices

4.1.8.1 Expander device overview

Expander devices are part of the service delivery subsystem. Expander devices contain two or more external expander ports. Expander devices may also include initiator ports and target ports (e.g., an expander device may include an embedded SCSI enclosure services target port) attached to internal expander ports. Expander ports may support being attached to SAS initiator ports, SAS and/or SATA target ports, and to other expander ports.

Figure 10 shows an expander device and the type of ports it may support.

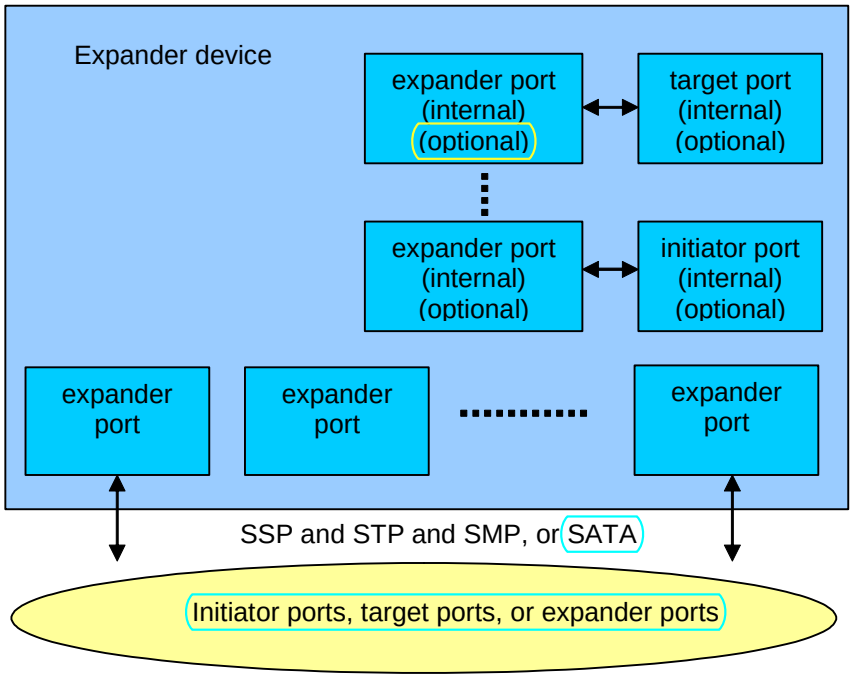


Figure 10 — Expander device

There are two types of expander devices differentiated by the routing attributes of their phys, edge expander devices and fanout expander devices. Edge expander devices may contain phys with the subtractive routing attribute. A fanout expander device shall not contain phys with the subtractive routing attribute. Edge expander devices and the fanout expander device may also contain phys with the direct routing and table routing attributes.

4.1.8.2 Edge expander device set

One or more edge expander devices may be grouped into edge expander device sets. The edge expander device sets are bounded by the direct routing phys that are either attached to end devices or not attached to any device. Edge expander device sets are further bounded by the phys supporting subtractive routing that are:

a) attached to the phys of a fanout device;
b) attached to the phys supporting subtractive routing on another edge expander device set;
c) attached to an end device; or
d) unattached.

The phys that support table routing within an edge expander device provide attachments to other edge expander devices in the edge expander device set.

The number of devices attached to an edge expander device set shall not exceed 64.

Figure 11 shows an edge expander device set.

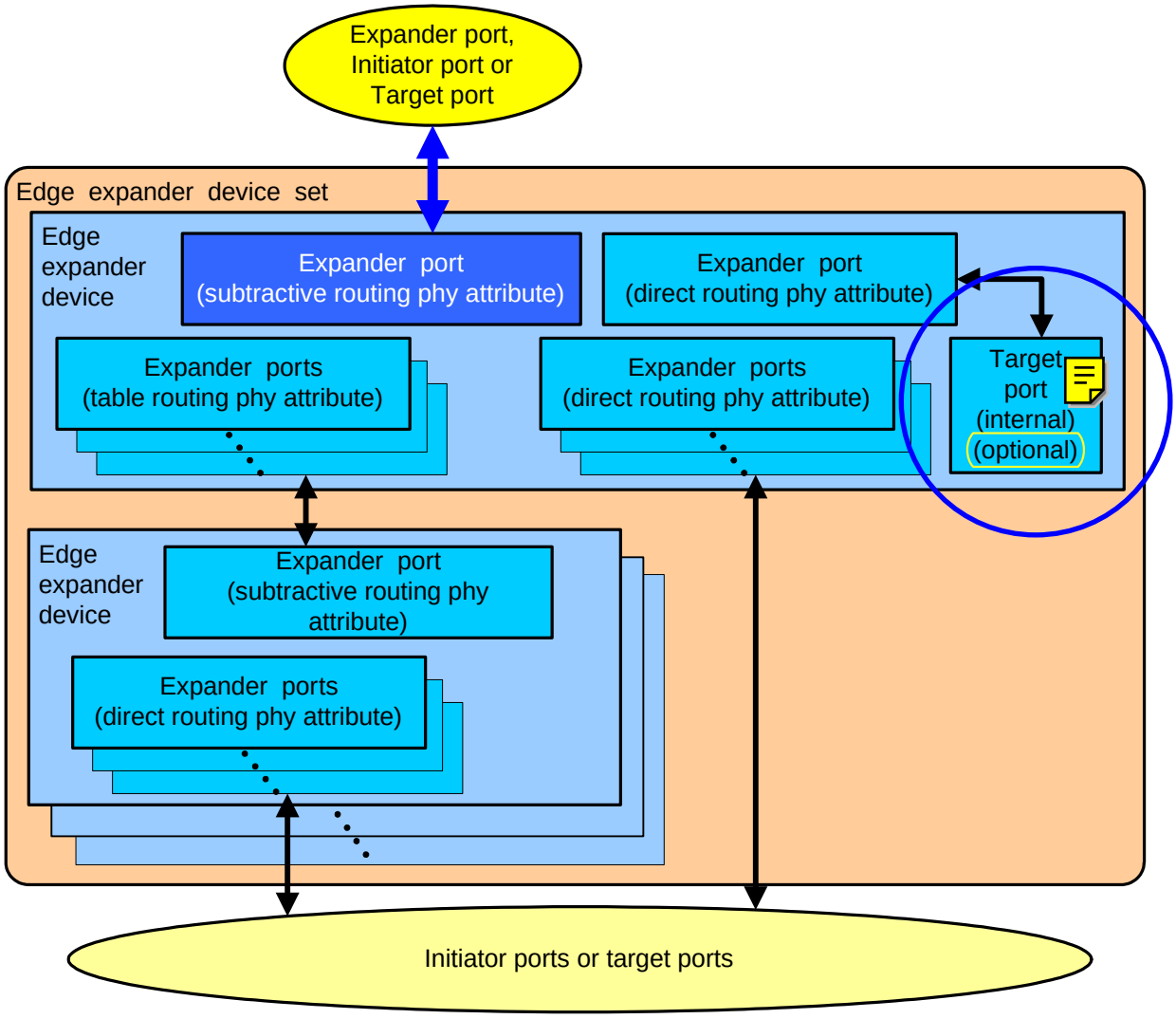


Figure 11 — Edge expander device set

4.1.8.3 Configurable expander device

Expander devices may have a configurable route table. Expander devices with a configurable route table depend on the application client within one or more initiator devices to use the discover process (see 4.6.11.5) to configure the expander route table. The expander route table is used by the expander connection manager to route connection requests to expander phys with the table routing attribute.

4.1.9 Domains

This standard defines SAS domains containing SCSI initiator ports, ATA initiator ports, SCSI target ports, ATA target ports, and expander ports.

SATA defines the SATA domain, which supports a single SATA initiator port and a single SATA target port (see SATA).

Figure 12 shows the possible domains.

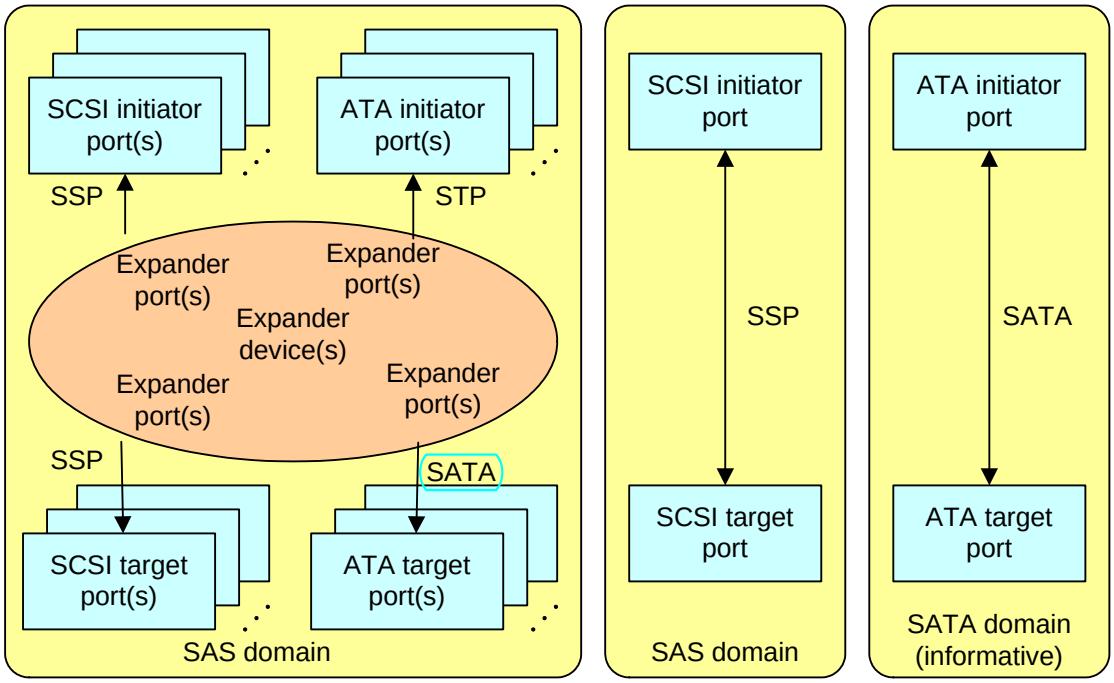


Figure 12 — Domains and connections

The expander port attached to a SATA target port translates STP to SATA; other expander ports pass through STP traffic. All expander ports pass through SSP traffic. Expander ports are not required to translate SSP to SATA.

Figure 13 shows initiator devices and target devices with ports in the same SAS domains and in different SAS domains.

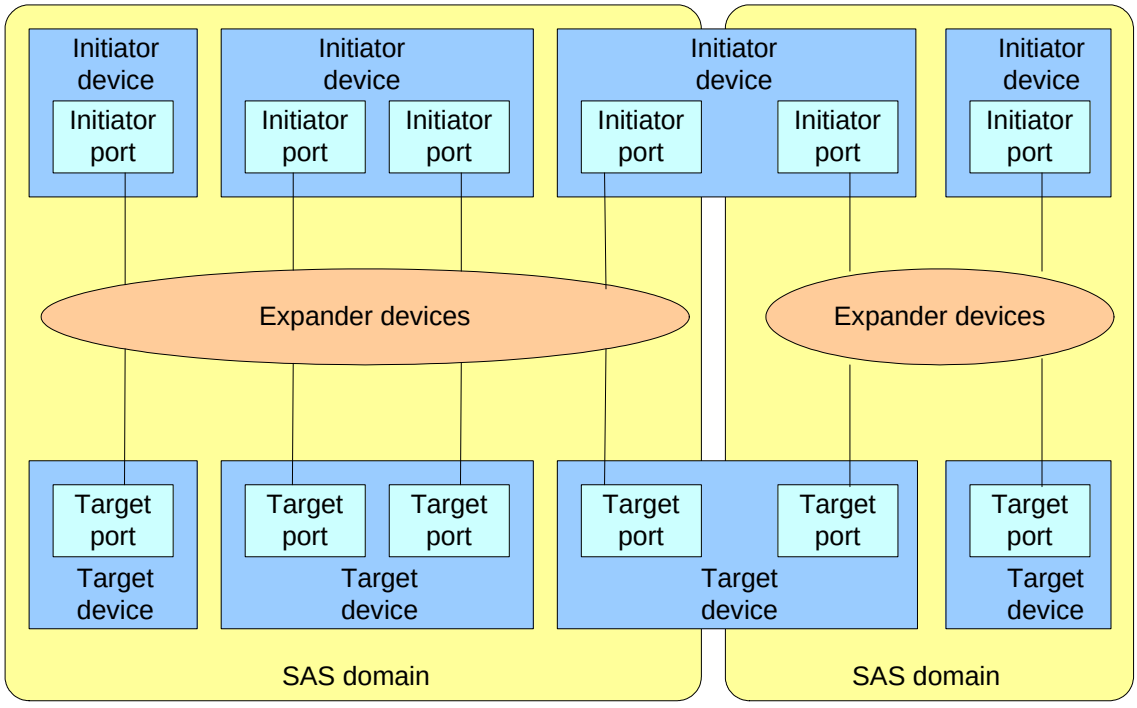


Figure 13 — Devices spanning domains

4.1.10 Expander device topologies

~~The domain consists of initiator devices, target devices, and expander devices.~~

No more than one fanout expander device shall be included in a SAS domain. The fanout expander device may be attached to up to 64 edge expander device sets, initiator ports, or target ports.

Each edge expander device set shall contain no more than 64 physical links to edge expander devices, initiator devices, or target devices. Each edge expander device set shall not be attached to more than one fanout expander device. An edge expander device set may be attached to one other edge expander device set if that is the only other edge expander device set in the domain and there are no fanout expander devices in the domain.

The number of edge expander devices and the phy route attributes of edge expander devices within an edge expander device set shall be established when the edge expander device set is configured.

Figure 14 shows a SAS domain with the maximum number of expander devices.

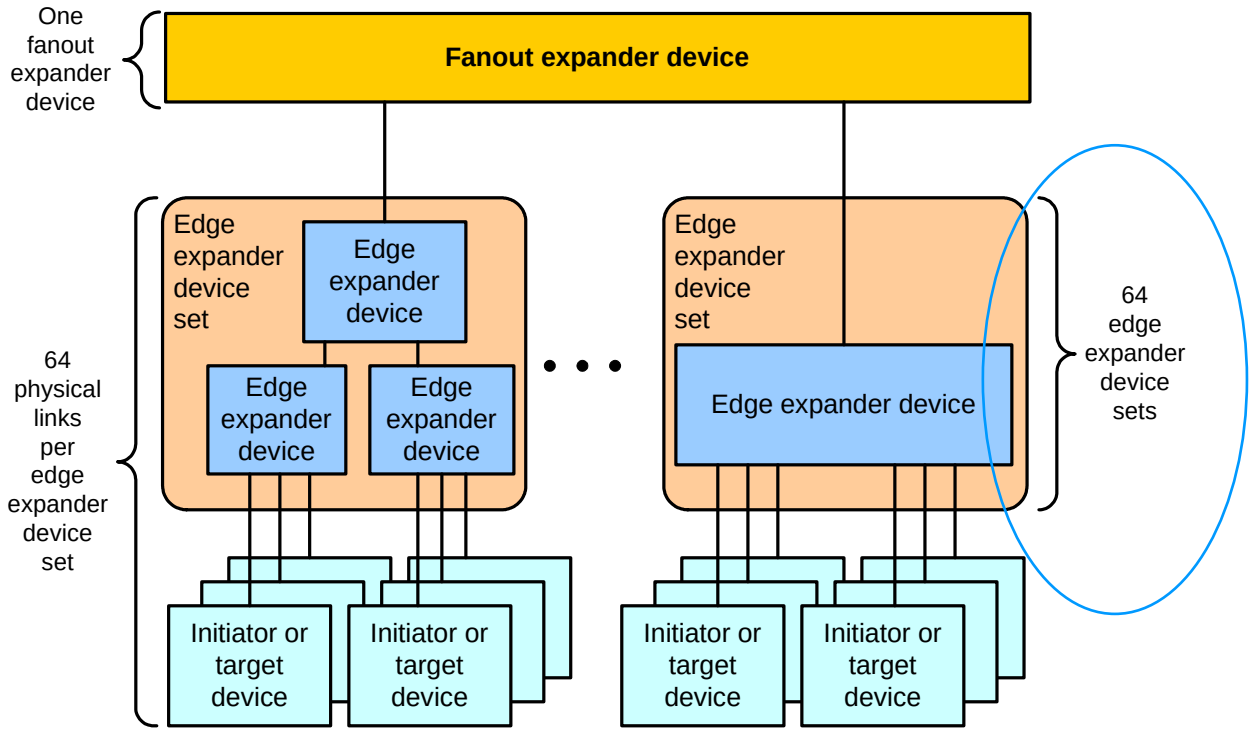


Figure 14 — Maximum expander device topology

Initiator devices and target devices may be attached directly to the fanout expander device, as shown in figure 15.

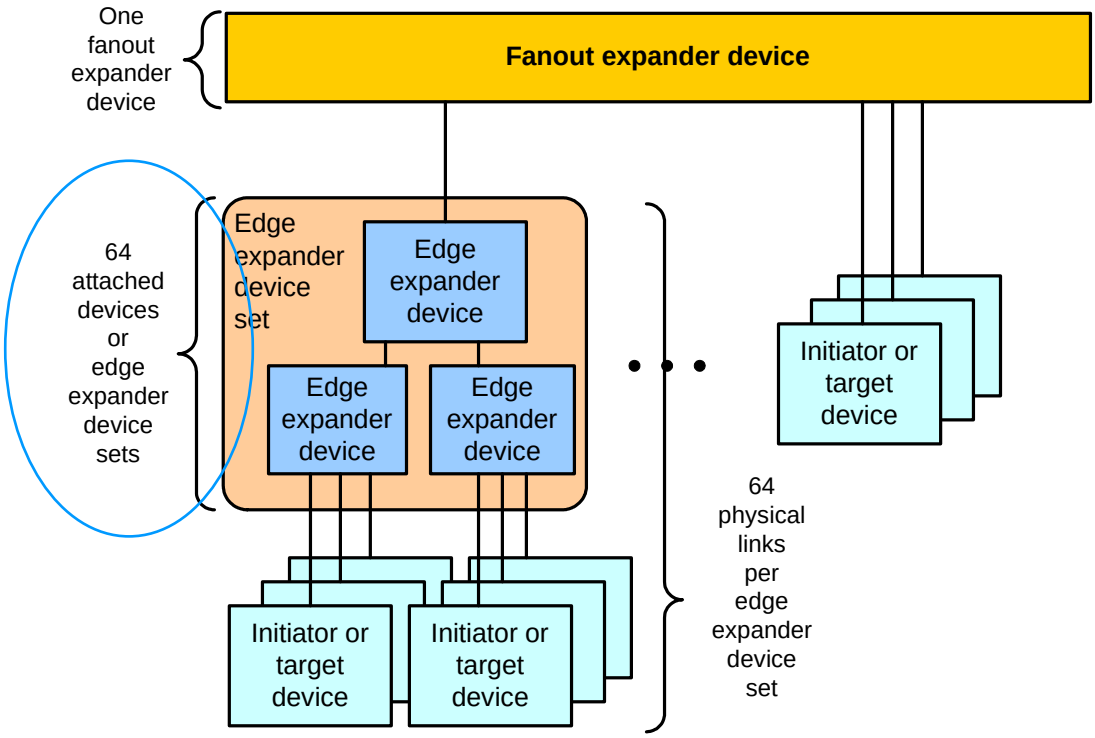


Figure 15 — Fanout expander device topology

Initiator devices and target devices may be attached to any of the edge expander devices within an edge expander device set and at most, two edge expander device sets may be attached together without a fanout expander, as shown in figure 16.

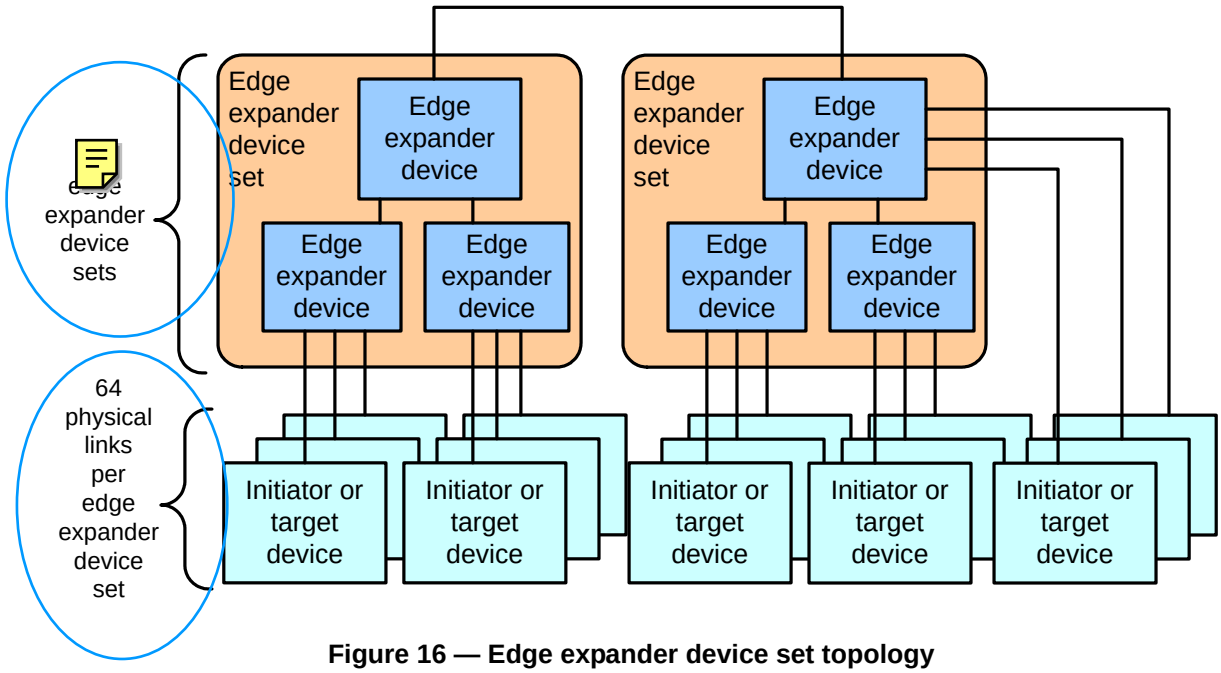


Figure 16 — Edge expander device set topology

4.1.11 Connections

A connection is an association between an initiator port and a target port.

SAS domains support the following connections:

a) SCSI initiator port to SCSI target port using SSP;
b) SCSI initiator port(s) to expander port(s) to SCSI target port(s); and
c) ATA initiator port(s) using STP to expander port(s) to ATA target port(s) using SATA.

One connection may be active on a physical 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, SATA_SYNCs, SATA_CONTs, or vendor-specific scrambled data dwords (after a SATA_CONT) are transmitted. If there is no connection on a physical link then idle dwords are transmitted.

Ports may establish multiple connections on a wide port. However, the number of connections shall not exceed the number of phys within the wide port (i.e., only one connection per phy is allowed) and there shall be a separate connection on each physical link.

Multiple connections may be established by a port between the following if multiple pathways exist between the initiator port(s) and the target port(s):

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

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

Figure 17 shows examples of connections between wide and narrow ports. All the connections shown may occur simultaneously ditionally:

a) the connections labeled A and B are an example of one initiator port with connections to multiple target ports;
b) the connections labeled A and C are an example of one target port with connections to multiple initiator ports;
c) the connections labeled E and F are an example of multiple connections between one initiator port and one target port; and

d) the connections labeled C, D, E, and F are an example of one initiator port with connections to multiple target ports with one of those target ports having multiple connections with that initiator port.

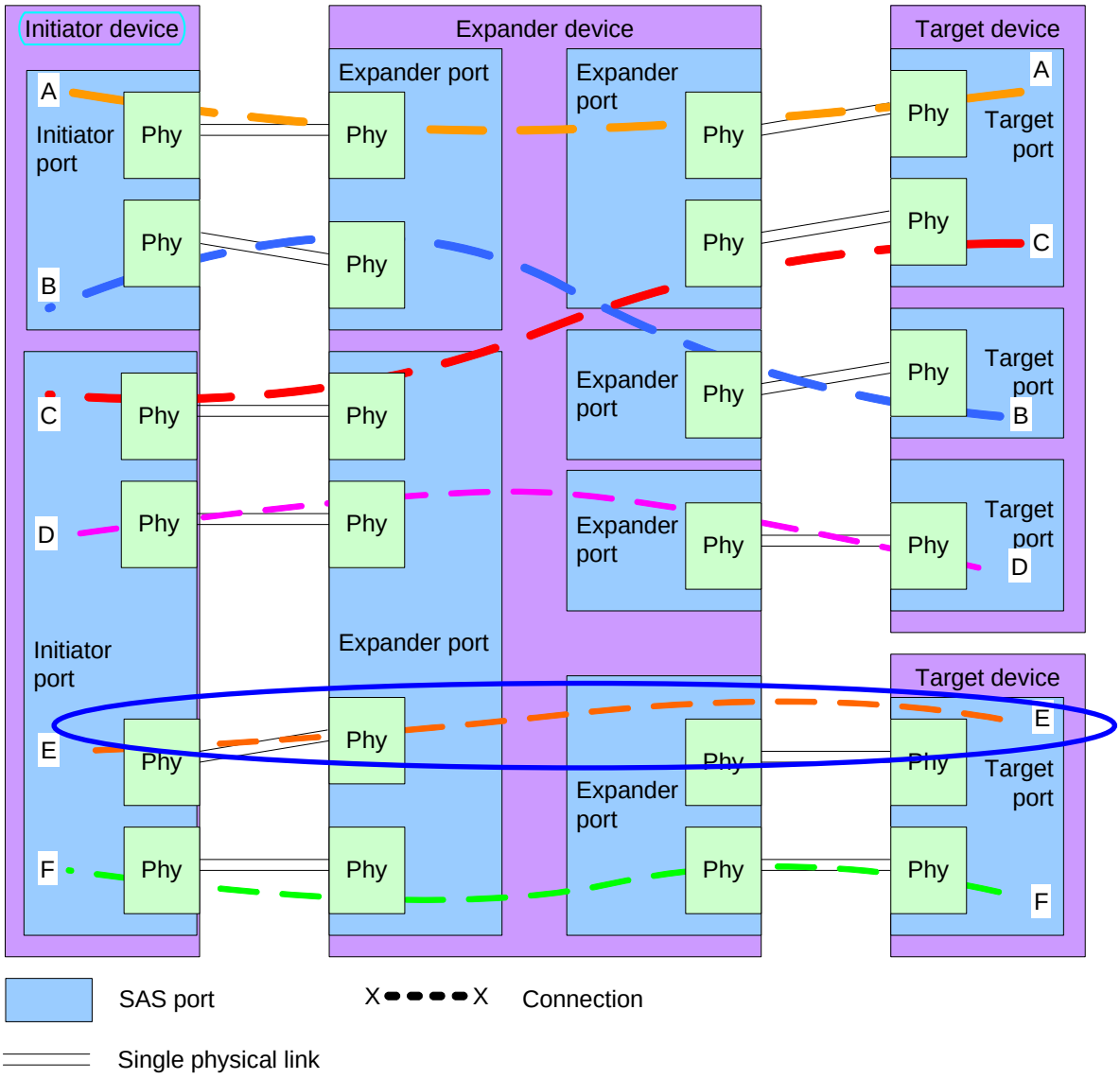


Figure 17 — Multiple connections on wide ports

4.1.12 Pathways

A pathway is the physical route of a connection. In the case where an initiator phy is directly attached to a target phy the pathway and the physical link are identical. In the case where there are expander devices between an initiator phy and a target phy, the pathway consists of all the physical links required to route dwords between the initiator phy and the target phy. The physical links may or may not be using the same physical link rate.

Figure 18 shows examples of pathways.

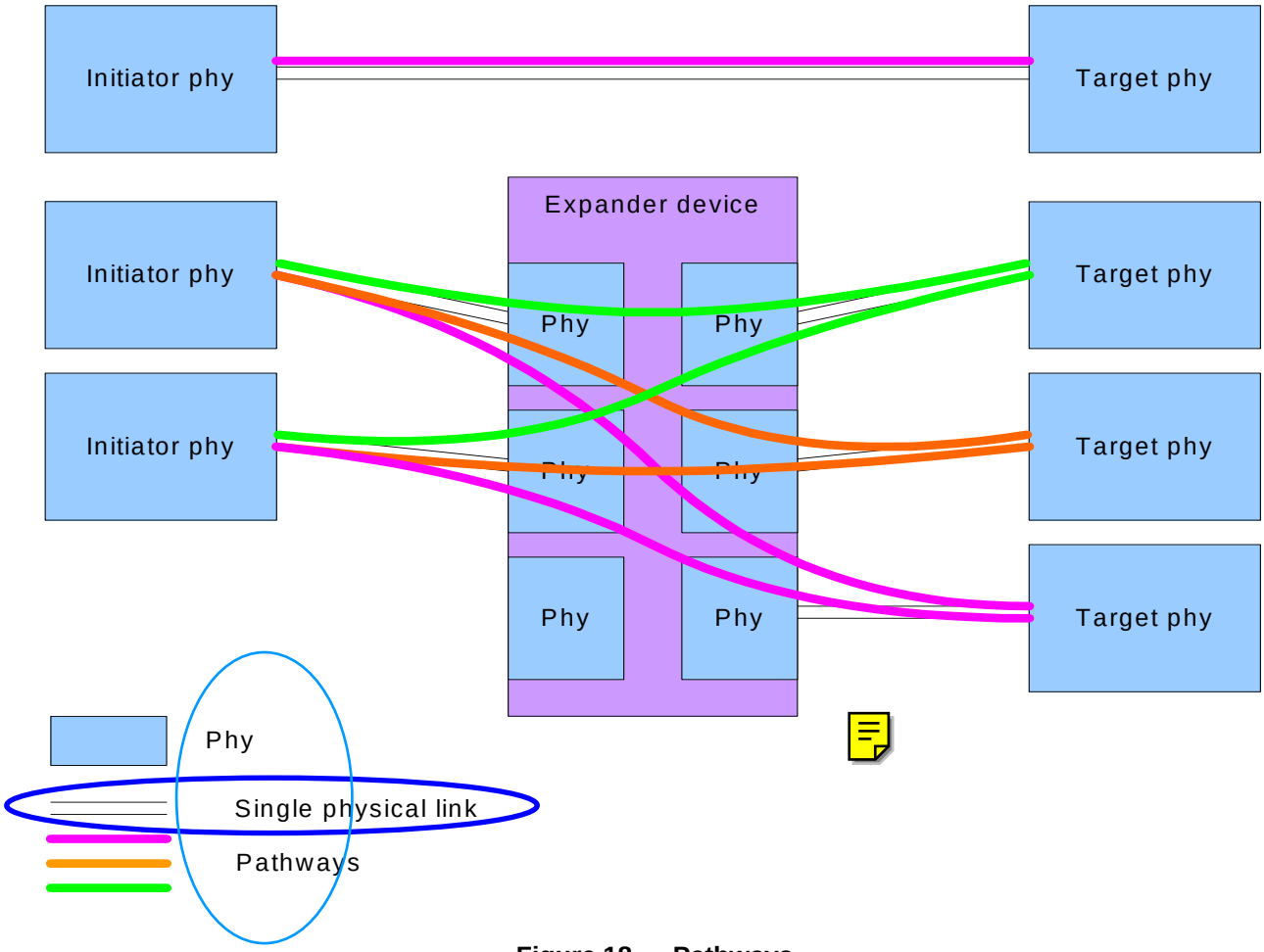


Figure 18 — Pathways

A partial pathway is the set of ~~physical~~ links participating in a connection request that has not reached the destination (e.g., the connection request (OPEN address frame) has been transmitted by the connection originator and the originator has received at least one AIP).

A partial pathway is blocked when path resources it requires are held by either another connection or another partial pathway.

A pending connection exists when an OPEN address frame has been delivered along a completed pathway to the destination but the destination has not yet responded to the connection request.

4.2 Names and identifiers

4.2.1 Names and identifiers overview

Device names are worldwide unique names for devices within a transport protocol (see SAM-3). 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, and are used as SAS addresses. Phy identifiers are unique within a device.

Table 5 shows the definition of names and identifiers for SAS.

Table 5 — Names and identifiers

Object	SAS implementation
Port identifier	SAS address
Port name	Not defined
Device name	SAS address
Phy identifier	Phy identifier

Table 6 describes how various SAM-3 objects are implemented in SSP.

Table 6 — SAM-3 object mapping

SAM-3 object	SSP implementation
Initiator port identifier	Initiator port SAS address
Initiator port name	Not defined
Target port identifier	Target port SAS address
Target port name	Not defined
Device name	SAS address

4.2.2 SAS addresses

Each initiator port, target port, target/initiator port, expander device, initiator device, and target device shall include a SAS address.

NOTE 7 There is no way to retrieve initiator device names in this standard.

Table 7 defines the SAS address format. SAS addresses shall be compatible with the NAA (Name Address Authority) IEEE Registered format identification descriptor defined in SPC-3.

Table 7 — SAS address format

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

The NAA field contains 5h.

The IEEE COMPANY ID field contains a 24-bit canonical form company identifier assigned by the IEEE. Information about IEEE company identifiers may be obtained from the http://standards.ieee.org/regauth/oui web site.

The VENDOR-SPECIFIC IDENTIFIER contains a 36-bit numeric value that is uniquely assigned by the organization associated with the IEEE COMPANY IDENTIFIER.

The SAS address shall be worldwide unique. A SAS address of 00000000_00000000h indicates an invalid SAS address.

4.2.3 Hashed SAS address

SSP frames include a hashed version of the SAS address for optional 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 8 lists the parameters for the code.

Table 8 — 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:

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

4.2.4 Port names

Port names are not defined in SAS.

NOTE 8 Port names are not defined in SAS, because the SAS address used by ports in different SAS domains may be the same. Also, there is no login process in SSP to exchange port names.

4.2.5 Port identifiers

The SAS address serves as the port identifier for each SAS initiator port, target port, and target/initiator port.

4.2.6 Phy identifier

Each phy in a device shall be assigned a unique 8-bit identifier within the device. The phy identifier is used for management functions.

Phy identifiers shall be greater than or equal to 00h and less than 40h.

4.3 State machines

4.3.1 State machine overview

Figure 19 shows the state machines for initiator devices and target devices and their relationships to each other and to the SAS device, SAS port, and phy objects.

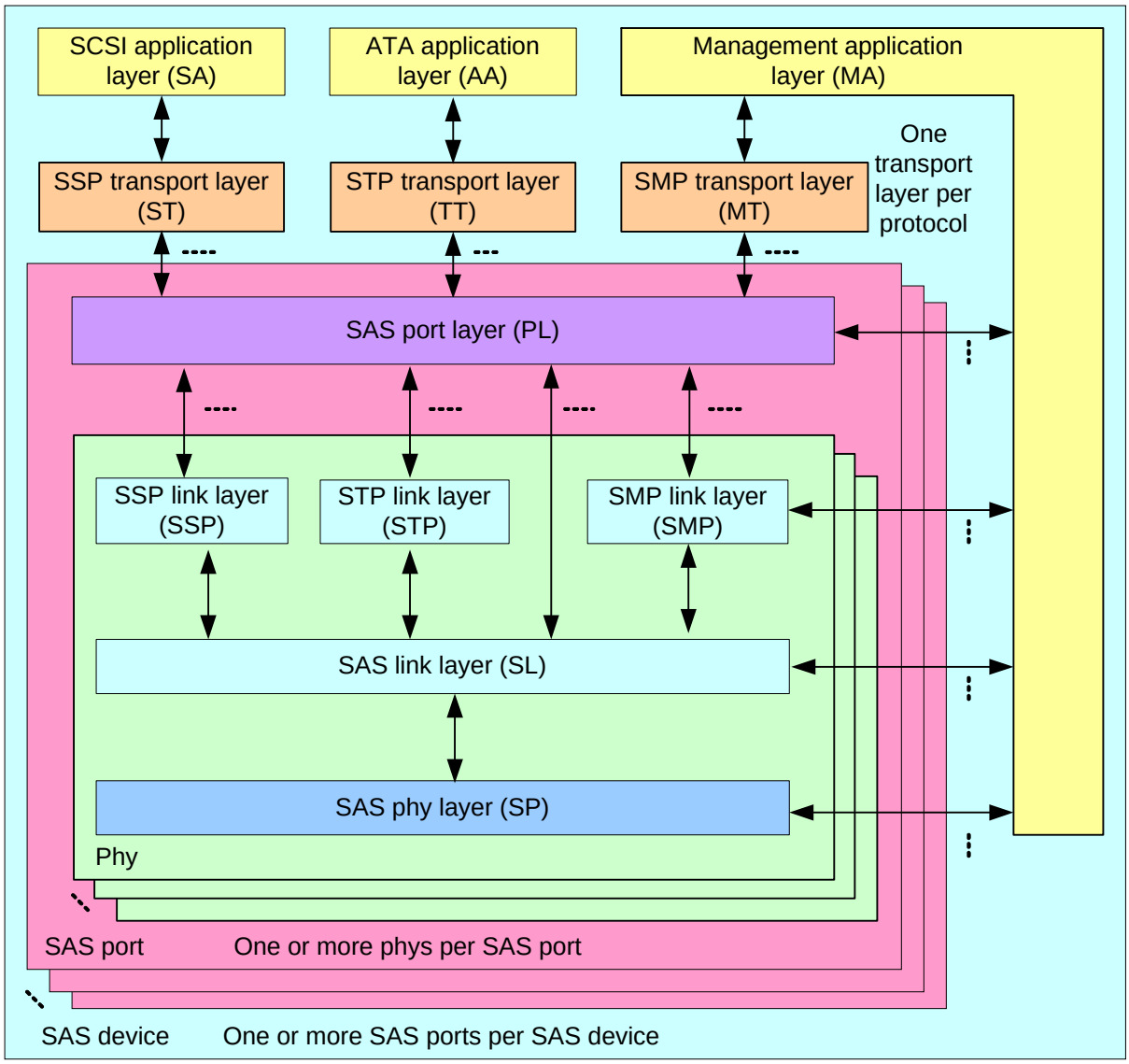


Figure 19 — State machines

4.3.2 Transmit data path

Figure 20 shows the transmit data path inside an initiator port or target port, indicating where each link layer and phy layer state machine fits.

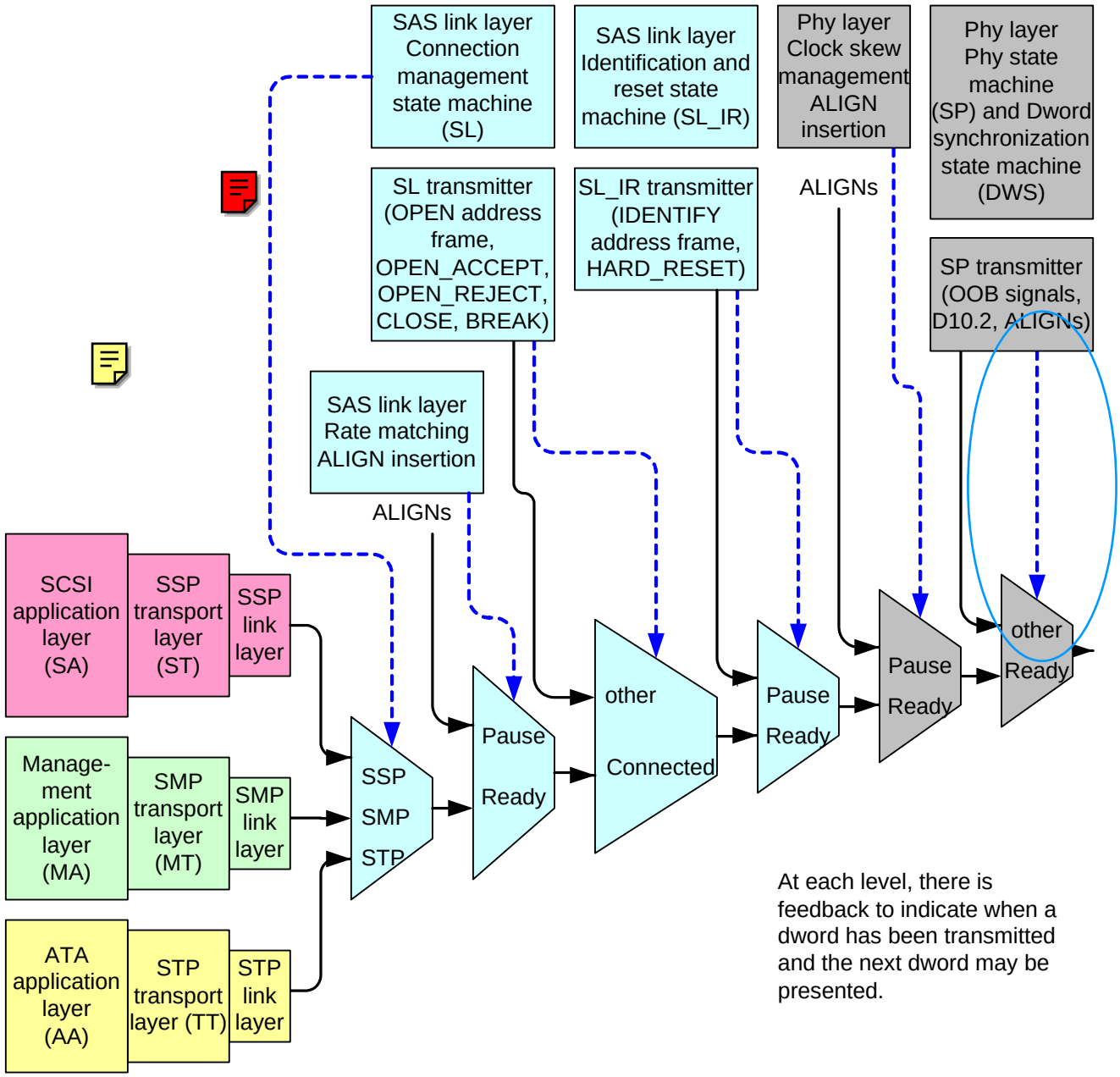


Figure 20 — Transmit data path and state machines

Figure 21 shows the transmit data path for the SSP link, SSP transport, and SCSI application layers. Only the SSP link layer directly transmits dwords.

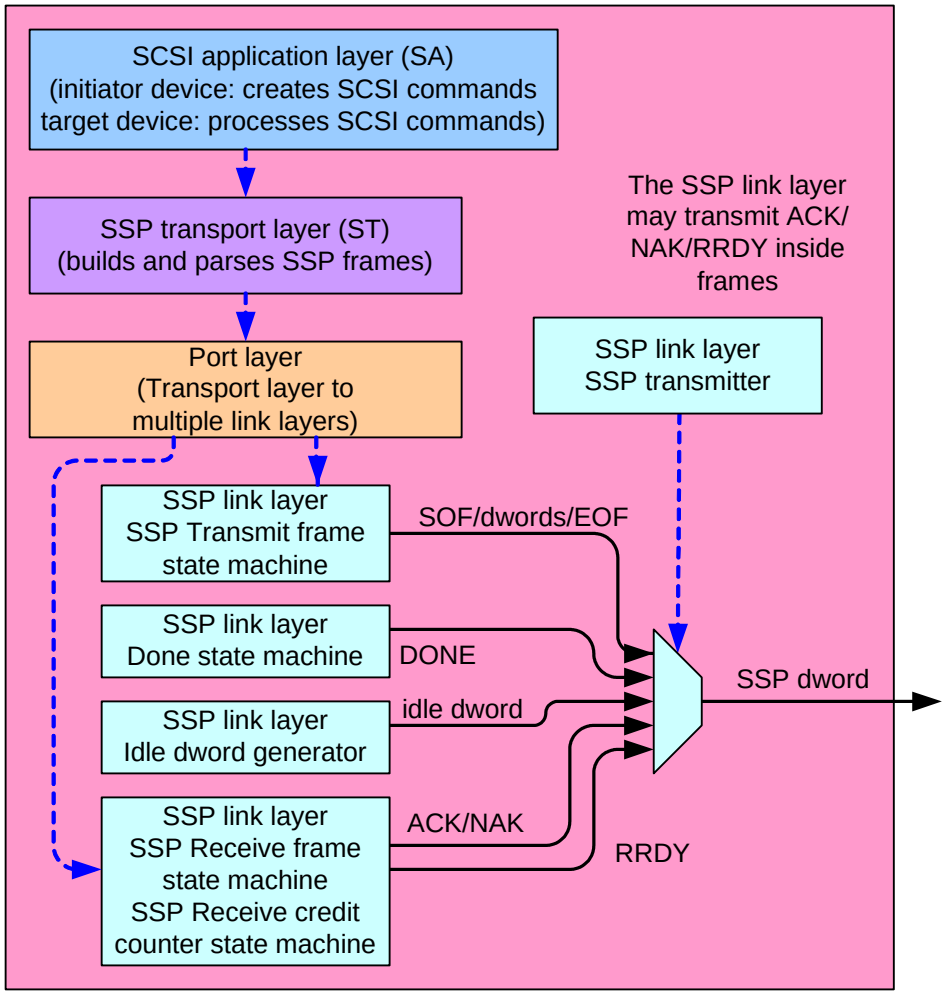


Figure 21 — SSP link, SSP transport, and SCSI application layer state machines

Figure 22 shows the transmit data path for the SMP link, SMP transport, and Management application layers. Only the SMP link layer transmits dwords.

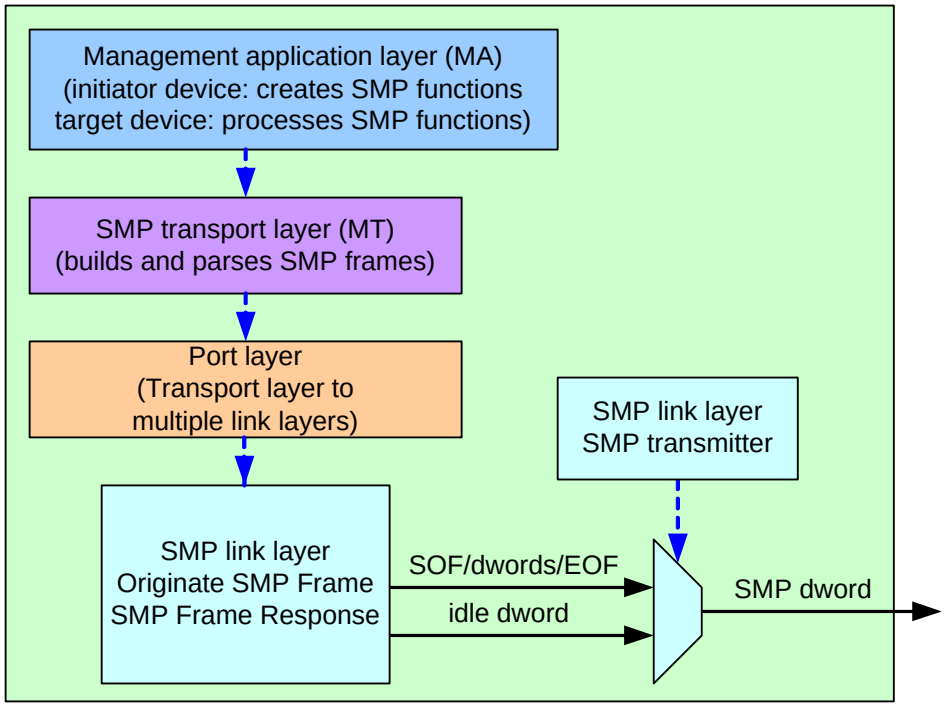


Figure 22 — SMP link, SMP transport, and management application layer state machines

Figure 23 shows the transmit data path for the STP link, STP transport, and ATA application layers. Only the STP link layer transmits dwords.

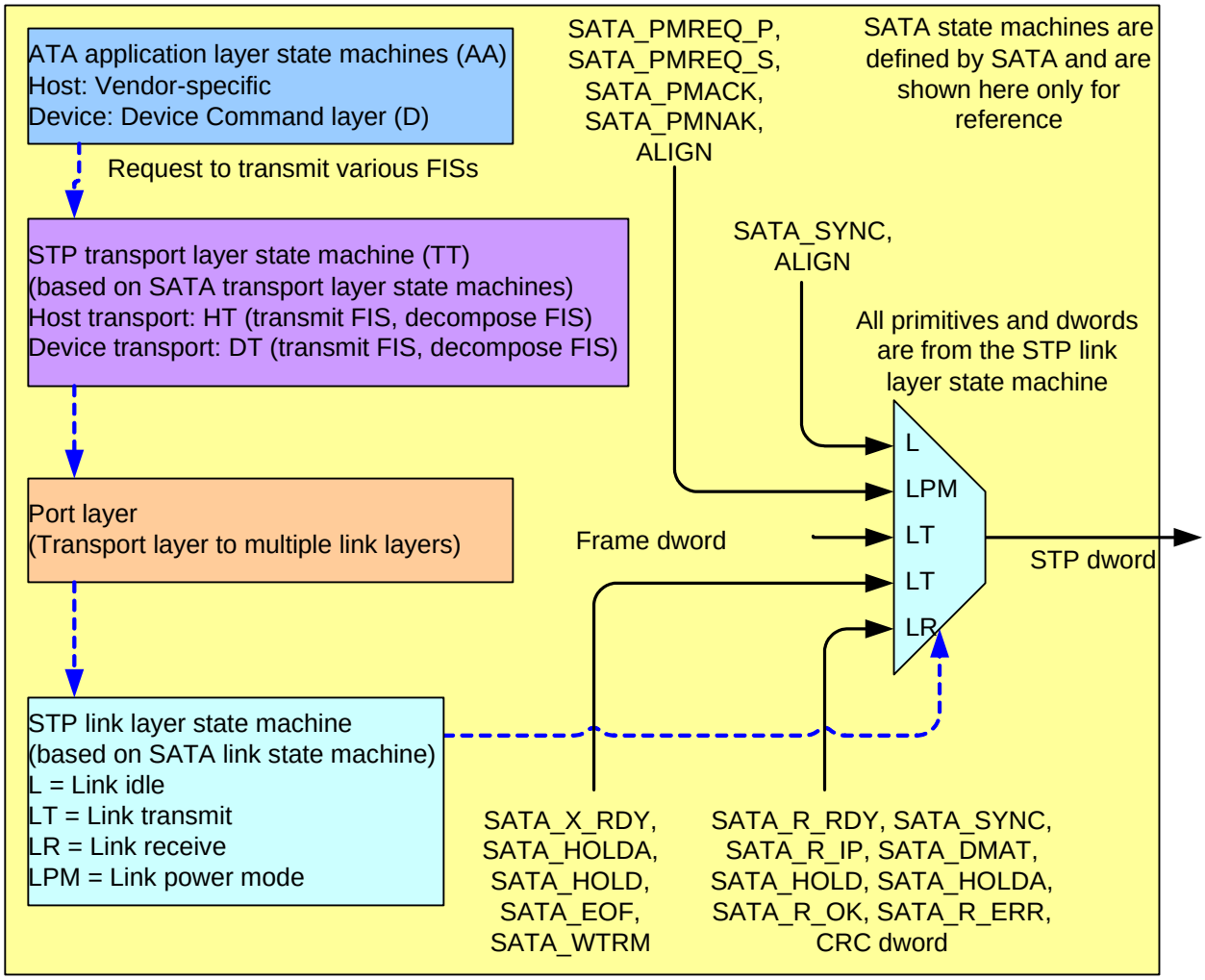


Figure 23 — STP link, STP transport, and ATA application layer state machines

4.3.3 Signals between state machines

4.3.3.1 Signals between phy layer and other layers

Table 9 shows the requests from the management application layer to the phy layer.

Table 9 — Requests from management application layer to phy layer

Phy layer	<- Request --	Application layer
SP	Enter Partial	MA
	Enter Slumber	
	Exit Partial	
	Exit Slumber	
	Power on, hard reset, or Management Reset	

Table 10 shows the confirmations from the phy layer to the link layer or the expander function.

Table 10 — Confirmations from phy layer to link layer or expander function

Phy layer	-- Confirmation ->	Link layer or expander function
SP	Broadcast Event Notify (Phy Not Ready)	Expander function (in expander device only)
	Broadcast Event Notify (SATA Spinup Hold)	
SP_DWS	Enable Disable Link Layer (Disable)	SL_IR
	Enable Disable Link Layer (SAS Enable)	
	Enable Disable Link Layer (SATA Enable)	

4.3.3.2 Signals between link layer, port layer, and transport layer for SSP

Table 11 shows the requests between the link layer, port layer, and SSP transport layer.

Table 11 — Requests between link layer, port layer, and transport layer for SSP

Link layer	<- Request --	Port layer	<- Request --	Transport layer
SL	Open Connection	PL		ST
	Stop Arb		Cancel	
	Accept_Reject Opens (Accept SSP)		Accept_Reject Opens (Accept SSP)	
	Accept_Reject Opens (Reject SSP)		Accept_Reject Opens (Reject SSP)	
SSP			Transmit Frame (Interlocked)	ST
			Transmit Frame (Non-Interlocked)	
	Tx Frame (Balance Required)			
	Tx Frame (Balance Not Requred)			
	Close Connection			

Table 12 shows the confirmations between the SL link layer, port layer, and SSP transport layer.

Table 12 — Confirmations between SL link layer, port layer, and SSP transport layer

Link layer	-- Confirmation ->	Port layer	-- Confirmation ->	Transport layer
SL		PL	Cancel Acknowledge	ST
	Open Failed (Wrong Destination		Transmission Status (Open Failed - Wrong Destination	
	Open Failed (Connection Rate Not Supported)		Transmission Status (Open Failed - Connection Rate Not Supported)	
	Open Failed (Protocol Not Supported)		Transmission Status (Open Failed - Protocol Not Supported)	
	Open Failed (No Destination)		Transmission Status (Open Failed - No Destination)	
	Open Failed (Bad Destination)		Transmission Status (Open Failed - Bad Destination)	
	Open Failed (STP Resources Busy)		Transmission Status (Open Failed - STP Resources Busy)	
	Open Failed (Open Timeout Occurred)		Transmission Status (Open Failed - Open Timeout Occurred)	
	Open Failed (Break Received)		Transmission Status (Open Failed - Break Received)	
	Open Failed (Pathway Blocked)		Transmission Status (Open Failed - Pathway Blocked)	

Table 13 shows the confirmations between the SSP link layer, port layer, and SSP transport layer.

Table 13 — Confirmations between SSP link layer, port layer, and SSP transport layer

Link layer	-- Confirmation ->	Port layer	-- Confirmation ->	Transport layer
SSP	Frame Transmitted	PL	Transmission Status (Frame Transmitted)	ST
	ACK/NAK Timeout		Connection Failed (ACK/NAK Timeout)	
	Credit Timeout		Transmission Status (Credit Timeout)	
	DONE Transmitted			
	DONE Timeout			
	ACK Received		ACK Received	
	NAK Received		NAK Received	
	DONE Received		DONE Received	
			Connection Failed (Connection Lost Without ACK/NAK)	
	Frame Received (ACK/NAK Balanced)		Frame Received (ACK/NAK Balanced)	
	Frame Received (ACK/NAK Not Balanced)		Frame Received (ACK/NAK Not Balanced)	
			Transmission Status (Connection Lost)	
			Transmission Status (No Phys In Port)	

4.3.3.3 Signals between link layer, port layer, and transport layer for SMP

Table 14 shows the requests between the link layer, port layer, and SMP transport layer.

Table 14 — Requests between SL/SMP link layer, port layer, and SMP transport layer

Link layer	<- Request --	Port layer	<- Request --	Transport layer
SL	Open Connection	PL		MT
	Stop Arb			
	Accept_Reject Opens (Accept SMP)		Accept_Reject Opens (Accept SMP)	
	Accept_Reject Opens (Reject SMP)		Accept_Reject Opens (Reject SMP)	
SMP	Tx Frame (SMP)	PL	Transmit Frame (SMP)	MT
	SMP Transmit Break		SMP Transmit Break	

Table 15 shows the confirmations between the link layer, port layer, and SMP transport layer.

Table 15 — Confirmations between link layer, port layer, and SMP transport layer

Link layer	-- Confirmation ->	Port layer	-- Confirmation ->	Transport layer
SMP	Frame Transmitted	PL	Transmission Status (Frame Transmitted)	MT
	Frame Received (SMP)		Frame Received (SMP)	
	Frame Received (SMP Failure)		Frame Received (SMP Failure)	
SL	~~Connection Opened (SMP, Source Opened)~~	PL		MT
	~~Connection Opened (SMP, Destination Opened)~~			
	~~Connection Closed (Close~~ Timeout)		Connection Closed	
	~~Connection Closed (Break Received)~~			
	~~Connection Closed (Link Broken)~~			
	~~Connection Closed (Normal)~~			
	Open Failed (Wrong Destination		Transmission Status (Open Failed - Wrong Destination	
	Open Failed (Connection Rate Not Supported)		Transmission Status (Open Failed - Connection Rate Not Supported)	
	Open Failed (Protocol Not Supported)		Transmission Status (Open Failed - Protocol Not Supported)	
	Open Failed (No Destination)		Transmission Status (Open Failed - No Destination)	
	Open Failed (Bad Destination)		Transmission Status (Open Failed - Bad Destination)	
	Open Failed (STP Resources Busy)		Transmission Status (Open Failed - STP Resources Busy)	
	Open Failed (Open Timeout Occurred)		Transmission Status (Open Failed - Open Timeout Occurred)	
	Open Failed (Break Received)		Transmission Status (Open Failed - Break Received)	
	~~Open Failed (Retry)~~			
	Open Failed (Pathway Blocked)		Transmission Status (Open Failed - Pathway Blocked)	
	~~Open Failed (Port Layer Request)~~			

4.3.3.4 Signals between link layer, port layer, and management application layer for all protocols

Table 16 shows the confirmations between the link layer and the port layer for all protocols.

Table 16 — Confirmations between link layer and port layer

Link layer	-- Confirmation ->	Port layer
SL	Open Failed (Retry)	PL
	Open Failed (Port Layer Request)	
	Connection Opened (SSP, Source Opened)	
	Connection Opened (STP, Source Opened)	
	Connection Opened (SMP, Source Opened)	
	Connection Opened (SSP, Destination Opened)	
	Connection Opened (STP, Destination Opened)	
	Connection Opened (SMP, Destination Opened)	
	Connection Closed (Break Received)	
	Connection Closed (Normal)	
	Connection Closed (Close Timeout)	
	Connection Closed (Link Broken)	

Table 17 shows the requests from the management application layer to the link layer for all protocols.

Table 17 — Requests from management application layer to link layer

Link layer	<- Request --	Application layer
SL_IR	Tx HARD_RESET	MA
	Tx IDENTIFY Address Frame	

Table 18 shows the confirmations between the link layer and the expander function or management application layer for all protocols.

Table 18 — Confirmations between link layer and port layer, expander function or application layer

Link layer	-- Confirmation ->	Port layer, expander function or application layer
SL	Change Received	MA
SL_IR	Phy Enabled	MA
	Phy Disabled	
	Address Frame Failed	
	HARD_RESET Transmitted	
	HARD_RESET Received	
	Identify Timeout	
	Identify Sequence Complete	
SL_IR	Broadcast Event Notify (Identification Sequence Complete)	Expander function
SL_IR	Phy Enabled	PL
	Phy Disabled	

4.3.3.5 Transport layer to application layer for SSP

Table 19 shows the requests from the SCSI application layer to the SSP transport layer.

Table 19 — Requests from SCSI application layer to SSP transport layer

Transport layer	<- Request --	Application layer
ST (initiator device)	Send SCSI Command	SA (initiator device)
	Send Task Management Request	
	Accept_Reject OPENs (Accept SSP)	
	Accept_Reject OPENs (Reject SSP)	
ST (target device)	Send SCSI Command Complete	SA (target device)
	Task Management Function Executed	
	Send Data-In	
	Receive Data-Out	
	Accept_Reject OPENs (Accept SSP)	
	Accept_Reject OPENs (Reject SSP)	

Table 20 shows the confirmations from the SSP transport layer to the the SCSI application layer.

Table 20 — Confirmations from SSP transport layer to SCSI application layer

Transport layer	-- Confirmation ->	Application layer
ST (initiator device)	Command Complete Received	SA (initiator device)
	Received Task Management Function - Executed	
	DONE Received	
	Nexus Lost	
ST (target device)	SCSI Command Received	SA (target device)
	Task Management Request Received	
	Data-In Delivered	
	Data-Out Received	
	Nexus Lost	

4.3.3.6 Transport layer to application layer for SMP

Table 21 shows the requests from the management application layer to the SMP transport layer.

Table 21 — Requests from management application layer to SMP transport layer

Transport layer	<- Request --	Application layer
MT (initiator device)	Send/Receive Frame Pair	MA (initiator device)
MT (target device)	Tx SMP Frame	MA (target device)
	Accept_Reject OPENs (Accept SMP)	
	Accept_Reject OPENs (Reject SMP)	

Table 22 shows the confirmations from the SMP transport layer to the the management application layer.

Table 22 — Confirmations from SMP transport layer to management application layer

Transport layer	-- Confirmation ->	Application layer
MT (initiator device)	Open Failed	MA (initiator device)
	SMP Frame Receive Timeout	
	SMP Frame Tx/Rcv Failure	
	SMP Frame Pair Sent/Received	
MT (target device)	SMP Frame Received	MA (target device)
	SMP Connection Closed	

4.4 Resets

4.4.1 Reset overview

Figure 24 describes the reset terminology used in this standard:

a) link reset sequence;
b) phy reset sequence;
c) SATA OOB sequence (see 6.6.2.1);
d) SATA speed negotiation sequence (see 6.6.2.2);
e) SAS OOB sequence (see 6.6.4.1);
f) SAS speed negotiation sequence (see 6.6.4.2);
g) hard reset sequence (see 7.8); and
h) identification sequence (see 7.8).

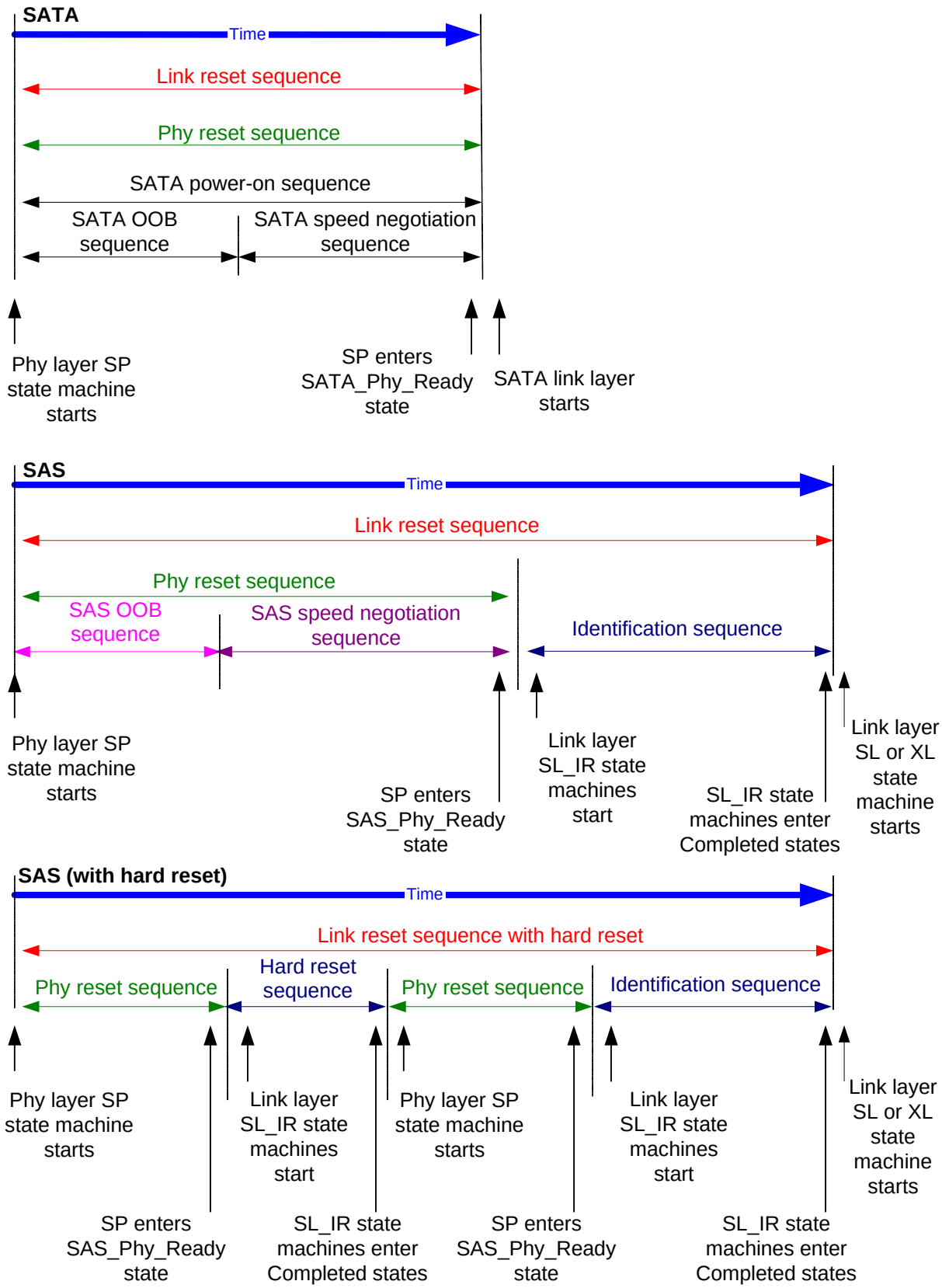


Figure 24 — Reset terminology

The phy reset sequences, including the OOB sequence and speed negotiation sequences, are implemented by the SP state machine and are described in 6.6 and 6.8. The hard reset sequence and identification sequence are implemented by the SL_IR state machine and are described in 7.8.

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

4.4.2 Hard reset

Between the link reset sequence and the IDENTIFY address frame, if a SAS phy receives a HARD_RESET, it shall be considered a reset event and initiate a hard reset of the port containing that phy.

When a port detects a hard reset, it shall stop transmitting on each of the phys contained in that port until a phy reset sequence occurs on that phy. The hard reset shall not affect any other ports in the device. Expander devices shall not forward HARD_RESETs to other phys.

If the port is part of a SCSI device, this causes a Transport Reset event notification to the SCSI application layer (see 10.1.4); the SCSI device shall perform the actions defined for hard reset in SAM-3.

If the port is part of an ATA device, the ATA device shall perform the actions defined for power-on or hardware reset in ATA.

If the port is part of an expander device, the expander function and other expander ports in the expander device shall not be affected by hard reset. If the expander device is also a SCSI device and/or an ATA device (i.e., it contains internal SCSI or ATA initiator ports or target ports), the SCSI and/or ATA device roles are reset.

After processing a hard reset, a phy shall originate a link reset sequence. After the link reset sequence completes, each logical unit to which the SSP target port has access shall create a unit attention condition for all SSP initiator ports. The sense key shall be set to UNIT ATTENTION with the additional sense code set to HARD RESET OCCURRED.

4.5 I_T nexus loss

When a port receives OPEN_REJECT (NO DESTINATION), OPEN_REJECT (CONNECTION RATE NOT SUPPORTED), or an open connection timeout in response to a connection request, it shall retry the connection request until:

a) the connection is established;
b) for SCSI target ports, the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port Control mode page (see 10.1.6.2) expires;
c) for SCSI initiator ports, the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port Control mode page (see 10.1.6.2) for that target port expires; or
d) for STP or SMP connection requests, a vendor-specific time expires.

If the timer expires, then the port shall send a Nexus Lost event notification to the SCSI application layer (see 10.1.4); the SCSI device shall perform the actions defined for I_T nexus loss in SAM-3. I_T nexus loss is handled by the port layer state machine (see 8).

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);
   B) expander connection router (ECR); and
   C) SL_IR primitive processor (BPP);
b) two or more external expander phys. ~~For the maximum number of phys, see 4.1.8;~~
c) an expander port available per phy; and
d) an internal SMP target port.

An expander device may contain the following:

a) additional internal SMP target ports;
b) internal SMP initiator port(s);
c) internal SSP target port(s);
d) internal SSP initiator port(s);
e) internal STP target port(s); and
f) internal STP initiator port(s).

Figure 25 shows the model of an expander device.

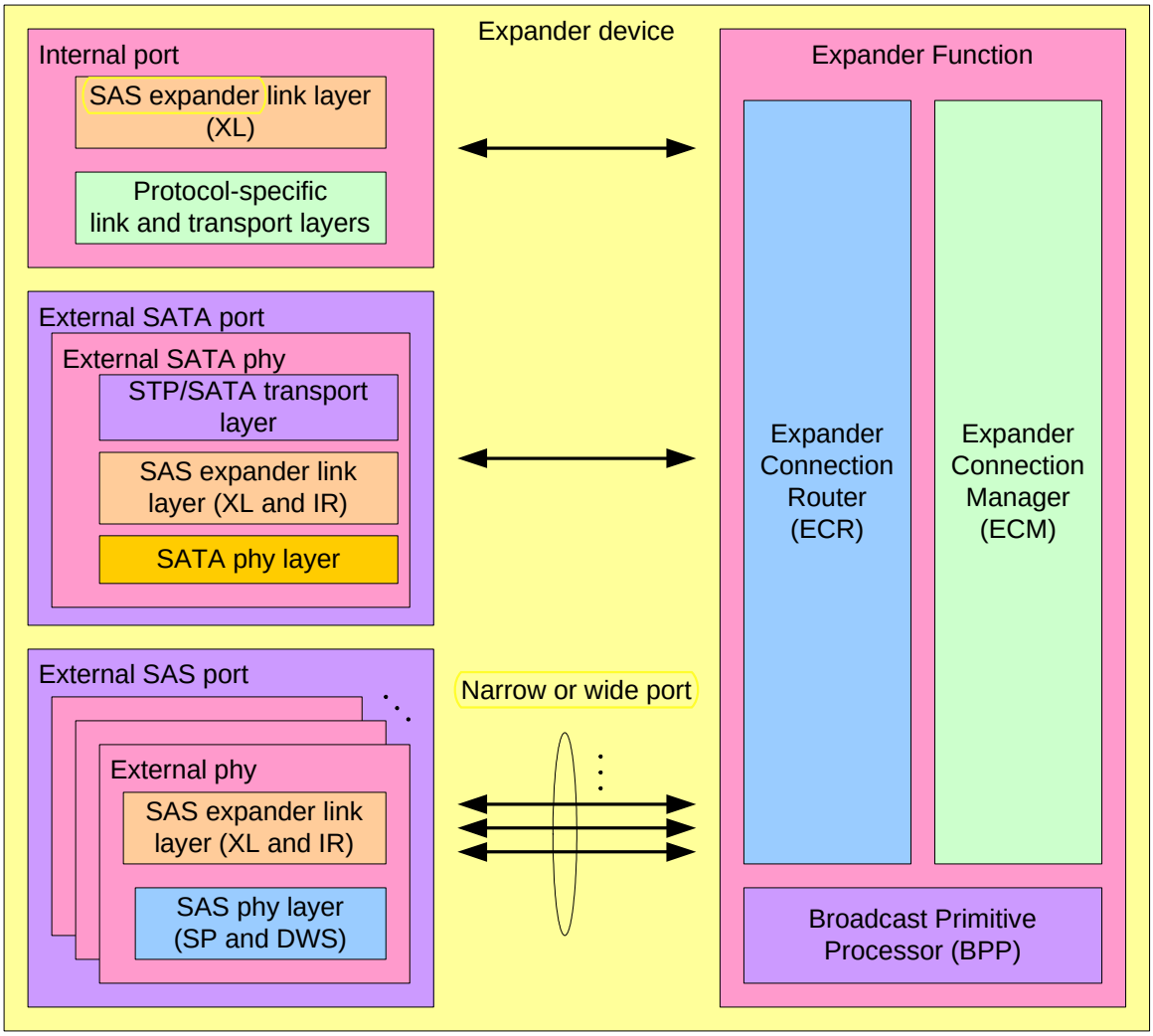


Figure 25 — Expander device model

4.6.2 Expander ports

An external expander port contains one or more phys (see 4.1.2). Each expander phy contains an expander link layer with an XL state machine and an SL_IR state machine. The expander link layers within an expander port request and respond to connection requests independently.

An internal expander port contains an expander link layer and a protocol-specific transport layer (e.g., to provide access to a SCSI enclosure services device as an SSP target).

One internal SMP port, one internal SSP port, and one internal STP port may each share the expander device's SAS address. If there is more than one internal SMP port, one internal SSP port, and one internal STP port, the additional ports shall include SAS addresses different from that of the expander device.

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 routed addressing 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 signals between pairs of expander phys as configured by the ECM. Enough routing resources shall be provided to support at least one active connection.

4.6.5 Broadcast primitive processor (BPP)

The BPP receives SL_IR primitive requests from each expander phy and requests transmission of those requests on all expander ports except the expander port from which the SL_IR primitive request was received.

4.6.6 Expander device interface

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

Figure 26 describes the interfaces present within an expander device.

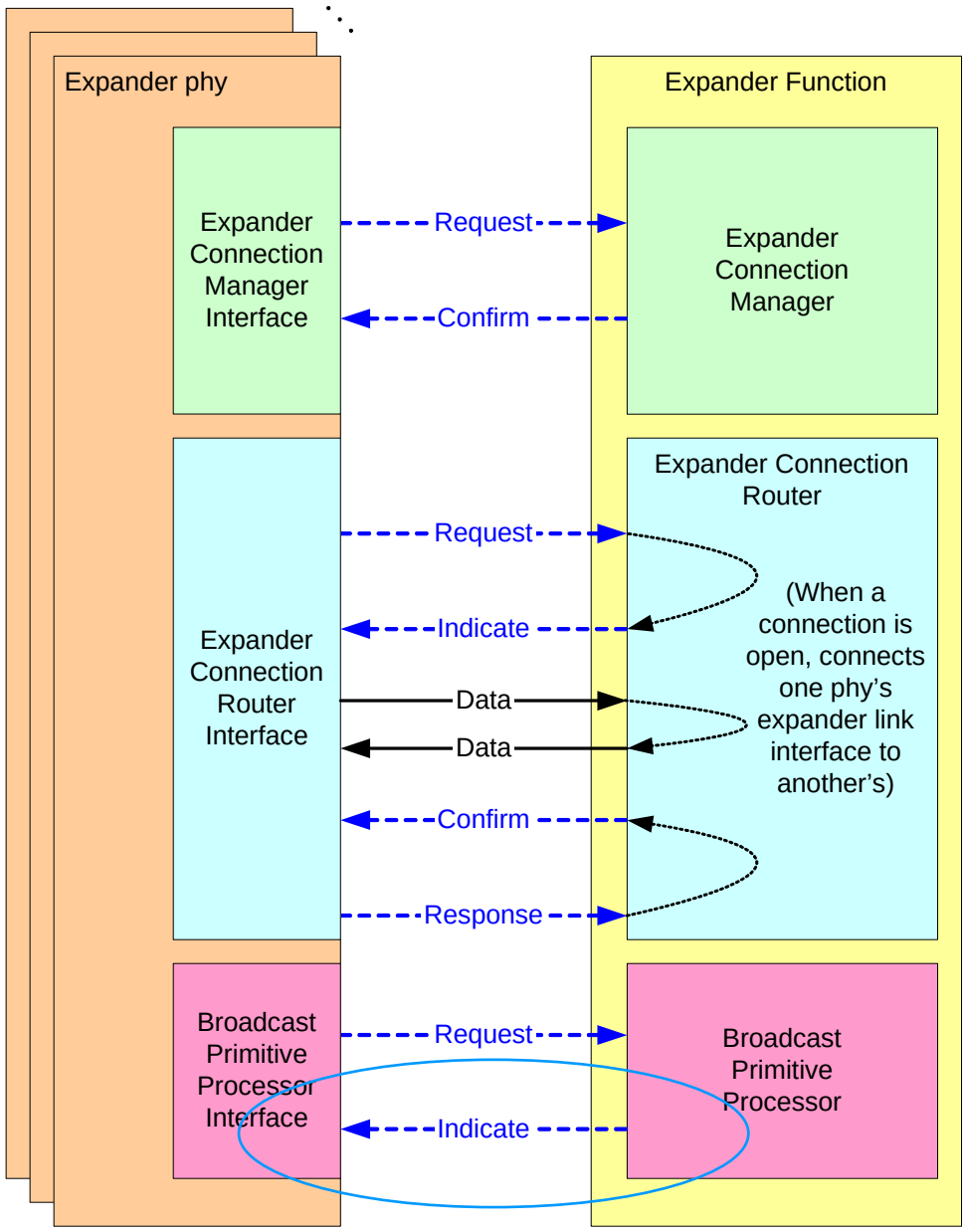


Figure 26 — Expander device interfaces

4.6.7 Expander device interface detail

Figure 27 shows the interface requests, confirmations, indications, and responses used by an expander device to manage connections.

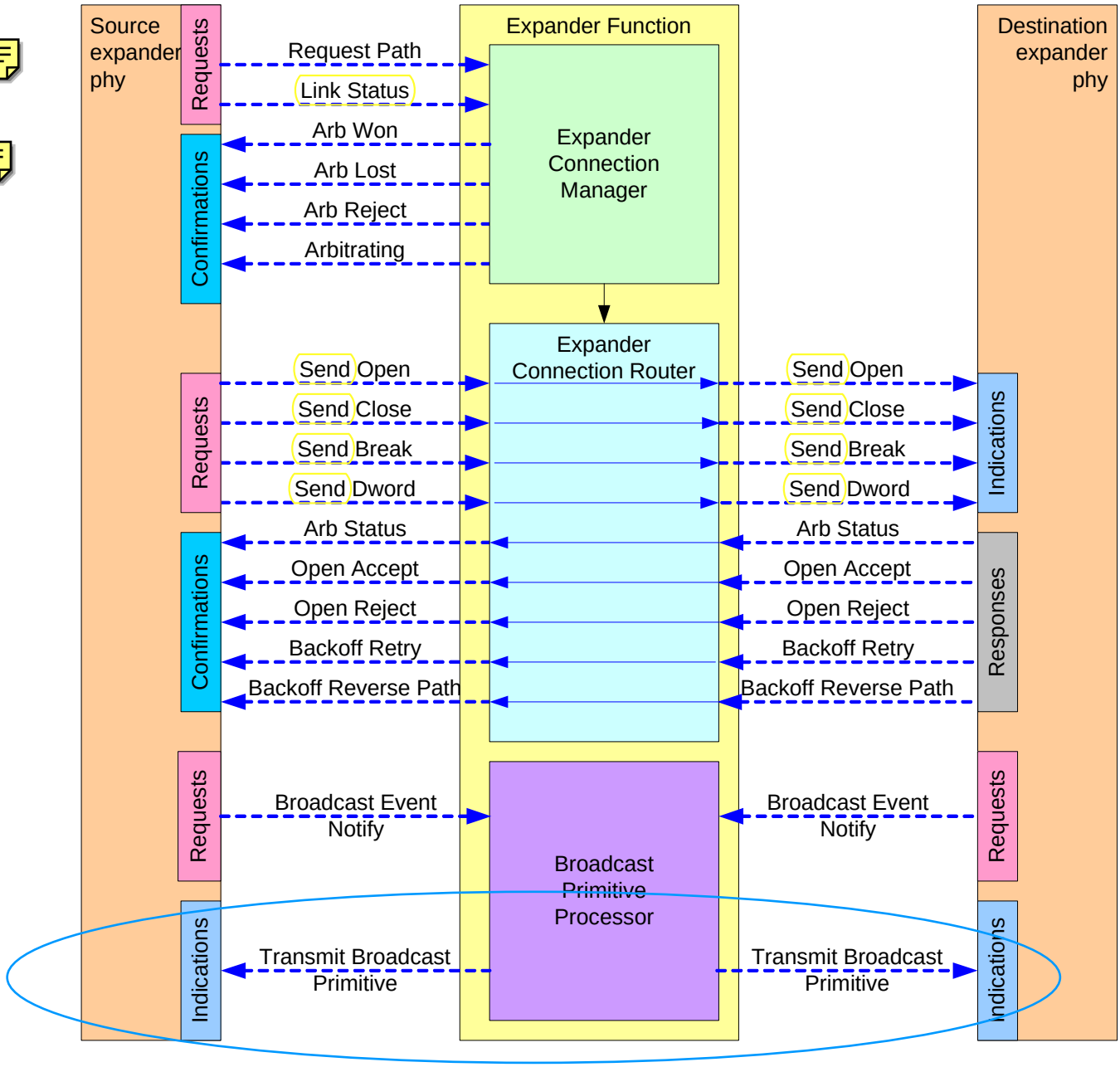


Figure 27 — Expander device interface detail

4.6.8 Expander connection manager interface

Table 23 describes the interface between an expander phy and the expander connection manager (ECM).

Table 23 — Expander connection manager interface

Signal	Source	Destination	Description
Request Path (arguments)	Expander phy	ECM	Request for a connection.
Phy Status (Partial Pathway)	Expander phy	ECM	Request specifying that an expander phy contains a partial pathway.
Phy Status (Blocked Partial Pathway)	Expander phy	ECM	Request specifying that an expander phy contains a partial pathway and the most recent AIP received or indicated is AIP (WAITING ON PARTIAL).
Phy Status (Connected)	Expander phy	ECM	Request specifying that an expander phy contains an active connection.
Arb Won	ECM	Expander phy	Confirmation that an expander phy has won path arbitration.
Arb Lost	ECM	Expander phy	Confirmation that an expander phy has lost path arbitration.
Arb Reject (No Destination)	ECM	Expander phy	Confirmation that the expander connection manager did not find an operational phy configured to match the requested destination SAS address.
Arb Reject (Bad Destination)	ECM	Expander phy	Confirmation that the expander connection manager has determined that the requested destination SAS address maps back to the requesting port.
Arb Reject (Bad Connection Rate)	ECM	Expander phy	Confirmation that the expander connection manager has determined that at least one destination phy matches the requested destination SAS address but no phys within the destination port are configured to support the requested connection rate.
Arb Reject (Pathway Blocked)	ECM	Expander phy	Confirmation that the expander connection manager has determined that the requesting expander phy shall back off according to SAS pathway recovery rules.
Arbitrating (Waiting On Partial)	ECM	Expander phy	Confirmation that the expander connection manager has determined that: a) at least one destination phy matches the requested destination SAS address; and b) all of the phys within the destination port contain a Phy Status of Partial Pathway.
Arbitrating (Blocked On Partial)	ECM	Expander phy	Confirmation that the expander connection manager has determined that: a) at least one destination phy matches the requested destination SAS address; and b) all of the phys within the destination port contain a Phy Status of Blocked Partial Pathway.
Arbitrating (Waiting On Connection)	ECM	Expander phy	Confirmation that the expander connection manager has determined that: a) at least one destination phy matches the requested destination SAS address; b) no phys within the destination port are available; and c) at least one of the phys within the destination port contain a Phy Status of Connected.

4.6.9 Expander connection router interface

Table 24 describes the interface between an expander phy and the expander connection router. All signals are routed to another expander phy.

Table 24 — Expander connection router interface

Signal	Description
Transmit Open (arguments)	Request/indication to transmit an OPEN address frame.
Transmit Close	Request/indication to transmit an CLOSE.
Transmit Break	Request/indication to transmit a BREAK.
Transmit Dword	Request/indication to transmit a dword.
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 AIP (WAITING ON DEVICE) has been received.
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.

4.6.10 Broadcast primitive processor interface

Table 25 describes the interface between expander phys and the broadcast primitive processor (BPP).

Table 25 — Broadcast primitive processor interface

Signal	Source	Destination	Description
Broadcast Event Notify (Phy Not Ready)	Expander phy	BPP	Request/indication to transmit a BROADCAST (CHANGE) on all other ports because a phy lost dword synchronization (see 6.8).
Broadcast Event Notify (SATA Spinup Hold)	Expander phy	BPP	Request/indication to transmit a BROADCAST (CHANGE) on all other ports because the SATA spinup hold state has been reached (see 6.10).
Broadcast Event Notify (Identification sequence complete)	Expander phy	BPP	Request/indication to transmit a BROADCAST (CHANGE) on all other ports because a phy has completed the identification sequence (see 7.8).
Broadcast Event Notify (CHANGE Received)	Expander phy	BPP	Request/indication to transmit a BROADCAST (CHANGE) on all other ports because a BROADCAST (CHANGE) was received.
Broadcast Event Notify (RESERVED CHANGE Received)	Expander phy	BPP	Request/indication to transmit a BROADCAST (RESERVED CHANGE) on all other ports because a BROADCAST (RESERVED CHANGE) was received.
Broadcast Event Notify (RESERVED 0 Received)	Expander phy	BPP	Request/indication to transmit a BROADCAST (RESERVED 0) on all other ports because a BROADCAST (RESERVED 0) was received.
Broadcast Event Notify (RESERVED 1 Received)	Expander phy	BPP	Request/indication to transmit a BROADCAST (RESERVED 1) on all other ports because a BROADCASE (RESERVED 1) was received.
Transmit Broadcast Primitive (CHANGE)	BPP	Expander phy	Request/indication to transmit a BROADCAST (CHANGE).
Transmit Broadcast Primitive (RESERVED CHANGE)	BPP	Expander phy	Request/indication to transmit a BROADCAST (RESERVED CHANGE).
Transmit Broadcast Primitive (RESERVED 0)	BPP	Expander phy	Request/indication to transmit a BROADCAST (RESERVED 0).
Transmit Broadcast Primitive (RESERVED 1)	BPP	Expander phy	Request/indication to transmit a BROADCAST (RESERVED 1).

4.6.11 Expander device routing

4.6.11.1 Routing attributes and 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 expander connection manager to determine which routing method to use when routing connection requests to 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.

A phy that has the direct routing attribute allows the expander connection manager to use the direct routing method.

A phy that has the table routing attribute allows the expander connection manager to use one of the following methods to route connection requests:

a) the table routing method if attached to an expander device; or
b) the direct routing method if attached to an end device.

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

A phy that has the subtractive routing attribute allows the expander connection manager to use one of the following methods to route connection requests:

a) the subtractive routing method if attached to an expander device; or
b) the direct routing method if attached to an end device.

An edge expander device shall have at most one defined port with phys that have the subtractive routing attribute. Phys in a fanout expander device shall not have the subtractive routing attribute.

An edge expander device may only use phys with the table routing attribute to attach to phys with the subtractive routing attribute in other edge expander devices within an edge expander device set.

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 processing the discover process (see 4.6.11.5) sh eport an error in a vendor-specific manner.

4.6.11.2 Expander device connection request routing

The expander connection manager of an expander device containing phys that have different route attributes shall determine how to route a connection request from a source phy using the following precedence:

1) route to a phy with a direct routing attribute when the destination SAS address in the connection request matches the attached SAS address;
2) route to a phy with a table routing attribute when the destination SAS address in the connection request matches the ATTACHED SAS ADDRESS in the expander route entry and the DISABLE ROUTE ENTRY bit is set to zero in the expander route entry;
3) route to a phy with a subtractive routing attribute; or
4) return an Arb Reject (No Destination) confirmation (see 4.6.8) to the source phy.

If the destination SAS address of a connection request matches the attached SAS address of an expander route entry and the DISABLE ROUTE ENTRY bit is set to one in the expander route entry, then the expander connection manager shall ignore the expander route entry.

The discover process shall allow the following attachments between expander phys:

a) edge expander phy with subtractive routing attribute attached to an edge expander phy with subtractive routing attribute;
b) edge expander phy with subtractive routing attribute attached to an edge expander phy with table routing attribute; or
c) edge expander phy with subtractive routing attribute attached to a fanout expander phy with table routing attribute.

During the discover process, if an application client in an initiator device detects an illegal expander phy attachment, it shall report an error in a vendor-specific manner.

4.6.11.3 Expander route table

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 and expander phy identifiers. Each association represents an expander route entry.

An expander device reports the size of its expander route table and indicates if the expander route table is configurable in the SMP REPORT GENERAL function (see 10.3.1.2).

An application client may reference a specific expander route entry within an expander route table with the SMP REPORT ROUTE INFORMATION function (see 10.3.1.7) and the SMP CONFIGURE ROUTE INFORMATION function (see 10.3.1.8).

Figure 28 shows a representation of an expander route table.

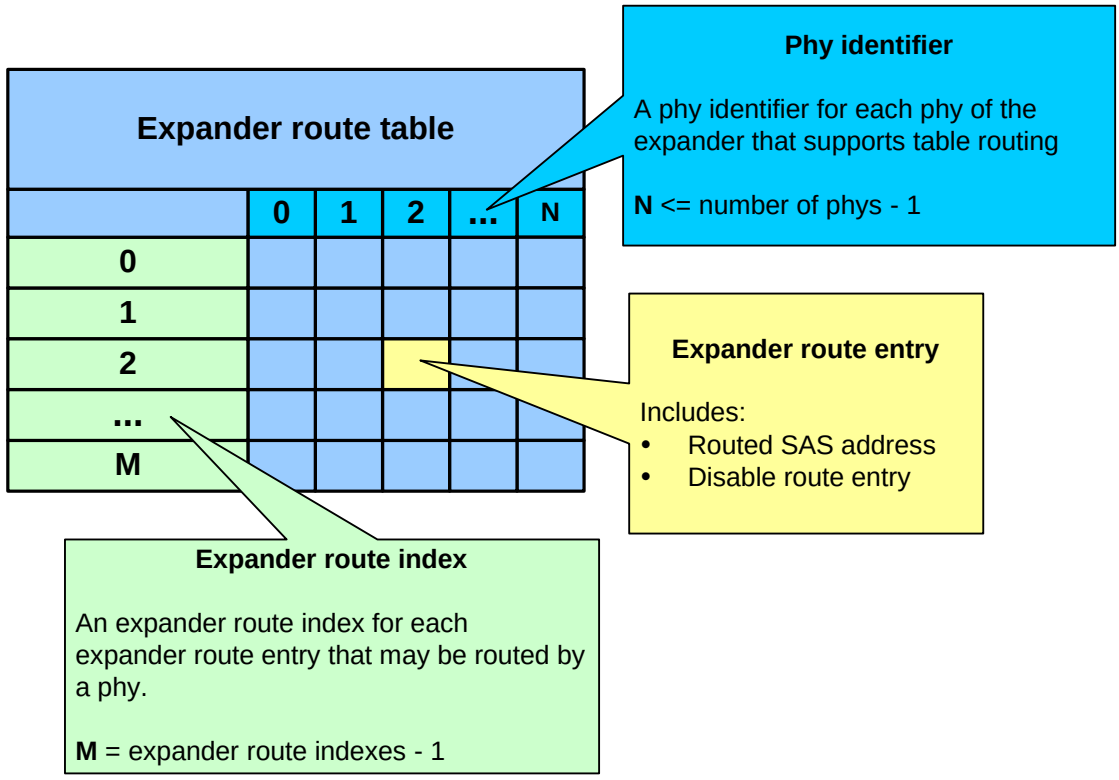


Figure 28 — Expander route table example

The number of end devices that may be attached to an edge expander device set is dependent on the number of expander route entries in the expander route table of the edge expander devices.

~~To avoid an overflow of an edge expander route index during the discover process (see 4.6.11.5) the number of edge expander route indexes per phy identifier shall be greater than or equal to the sum of all cascaded edge expander phys addressable through the edge expander phy.~~

~~An expander device is considered cascaded to another expander device when the expander device table routing phy is attached to the subtractive routing phy of another edge expander device.~~

Fanout expander devices shall not be cascaded.

During the discover process, if an application client in an initiator device detects an overflow of the edge expander route index, it shall report an error in a vendor-specific manner.

If the discover process detects an expander route table entry that references the SAS address of the expander itself (i.e., self-reference), it shall set the DISABLE ROUTE ENTRY bit to one in the expander route table entry.

If the discover process detects an expander route table entry that references the SAS address of an attached edge expander device, it shall set the DISABLE ROUTE ENTRY bit to one in the expander route table entry.

The discover process shall set the DISABLE ROUTE ENTRY bit to one in each table route entry for each phy that has its ATTACHED DEVICE TYPE set to zero.

4.6.11.4 Expander route index order

The expander route table shall be configured for each phy that:

a) has a table routing attribute; and
b) is attached to another edge expander device.

For purposes of configuring the expander route table, the edge expander devices attached to the phy are assigned levels:

1) the attached edge expander device is considered level 1;
2) devices attached to it are considered level 2;
3) devices attached to level 2 edge expander devices are considered level 3; and
4) etc.

The expander route table shall be configured starting from expander route index 0 by level (e.g., all level 1 entries first, then all level 2 entries, then all level 3 ~~entries, etc.)~~ up to the EXPANDER ROUTE INDEXES reported by the SMP REPORT GENERAL function (see 10.3.1.2).

Assuming the attached edge expander devices has N phys, the first N entries shall be the SAS addresses of the devices attached to the attached edge expander device, ordered from phy 0 through phy N.

For each of the level 2 devices that:

a) is an edge expander device with M phys; and
b) is attached to a phy in the level 1 edge expander device with the table routing attribute,

the next M entries shall be the SAS addresses of the devices (level 3) attached to that level 2 edge expander device.

This process shall repeat for all levels of edge expander devices in the edge expander device set.

Figure 29 shows a portion of an expander device set with levels labeled.

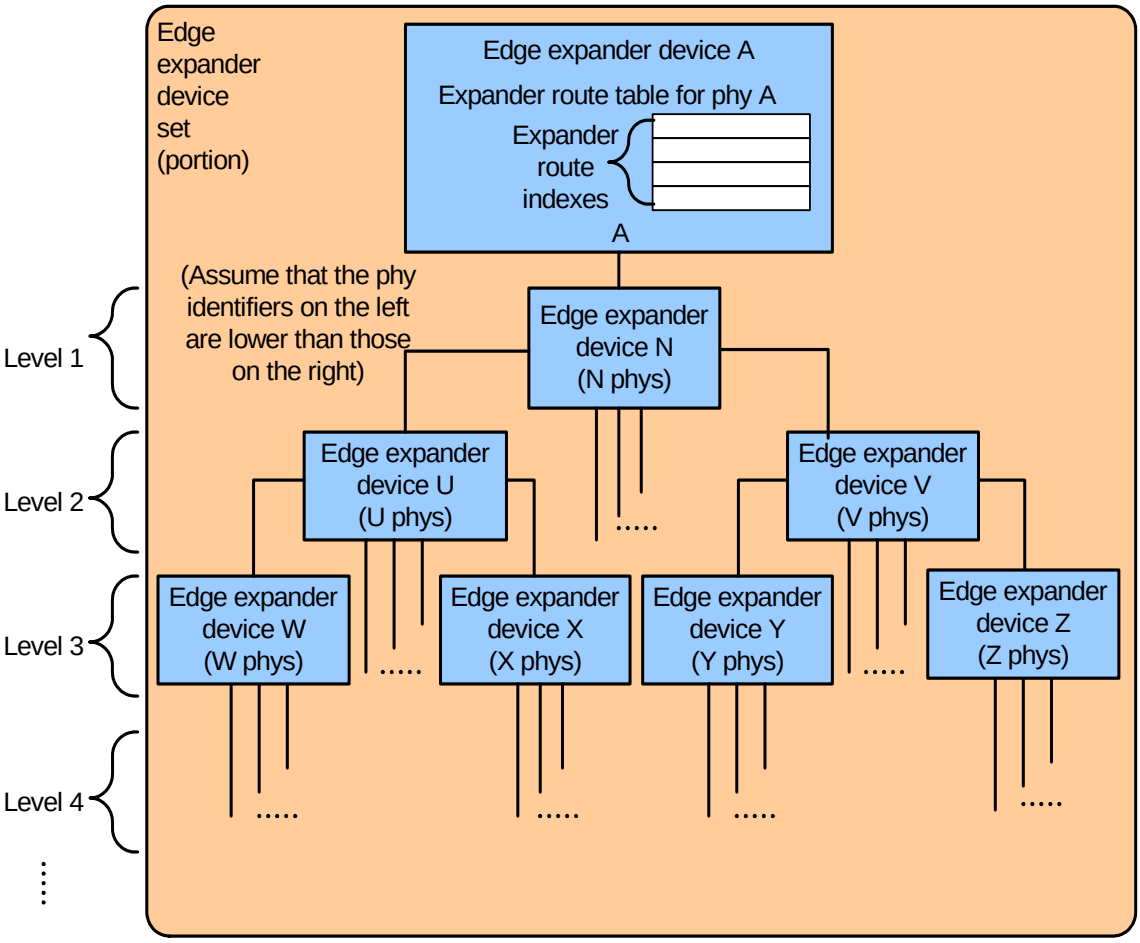


Figure 29 — Expander route index levels

Table 26 shows how the expander route table is configured for phy A in figure 29.

Table 26 — Expander route table levels

Expander route index	Routed SAS address description
Level 1 entries	
0	SAS address of the device attached to edge expander device N phy 0
1	SAS address of the device attached to edge expander device N phy 1
...	...
N	SAS address of the device attached to edge expander device N phy N
Level 2 entries	
N + 1	SAS address of the device attached to edge expander device U phy 0
	...
N + 1 + U	SAS address of the device attached to edge expander device U phy U
	...
	SAS address of the device attached to edge expander device V phy 0
	...
	SAS address of the device attached to edge expander device V phy U
Level 3 entries	
	SAS address of the device attached to edge expander device W phy 0
	...
	SAS address of the device attached to edge expander device W phy W
	...
	SAS address of the device attached to edge expander device Z phy 0
	...
	SAS address of the device attached to edge expander device Z phy Z
Entries for additional levels	
	...

Figure 30 is an example topology.

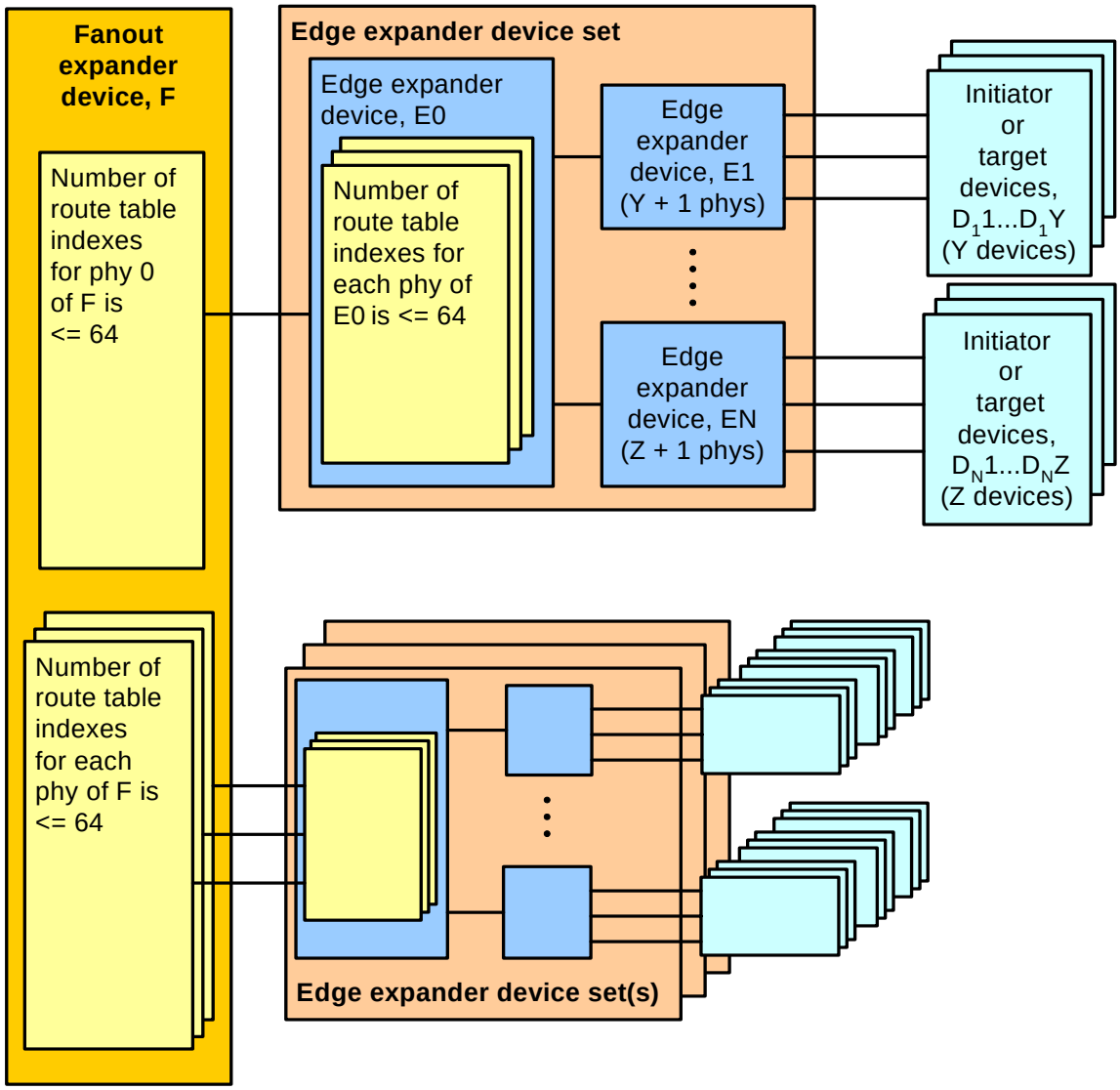


Figure 30 — Expander route index order

Table 27 shows the expander route index order for the edge expander E0 in figure 30.

Table 27 — Expander route table entries for edge expander E0 phy 0

Expander route index	Routed SAS address description
Level 1 entries	
0	SAS address (e.g., E0) of the device attached to phy 0 of edge expander device E1. The DISABLE ROUTE ENTRY bit is set to one in this entry because E0 is the expander device being configured.
1	SAS address (e.g., $D_1$1) of the device attached to phy 1 of edge expander device E1
2	SAS address (e.g., $D_1$2) of the device attached to phy 2 of edge expander device E1
...	...
Y	SAS address (e.g., D_1Y) of the device attached to phy N of the attached edge expander device E1
Level 2: no entries since all devices attached to E1 through EN, except for E0, are end devices	

Table 28 shows the expander route index order for the fanout expander F phy 0 in figure 30.

Table 28 — Expander route table entries for fanout expander device F phy 0

Expander route index	Routed SAS address description
Level 1 entries	
0	SAS address (e.g., F) of the device attached to phy 0 of edge expander device E0. The DISABLE ROUTE ENTRY bit is set to one in this entry because F is the expander device being configured.
1	SAS address (e.g., E1) of the device attached to phy 1 of edge expander device E0
2	SAS address (e.g., E2) of the device attached to phy 2 of edge expander device E0
...	...
N	SAS address (e.g., EN) of the device attached to phy N of the attached edge expander device E0
Level 2 entries	
N + 1	SAS address (e.g., E0) of the device attached to phy 0 of the edge expander device E1. The DISABLE ROUTE ENTRY bit is set to one in this entry because E0 is attached to the expander device being configured.
...	...
N + 1 + Y	SAS address (e.g., D_1Y) of the device attached to phy Y of the edge expander device E1
...	...
	SAS address (e.g., E0) of the device attached to phy 0 of the edge expander device EN. The DISABLE ROUTE ENTRY bit is set to one in this entry because E0 is attached to the expander device being configured.
...	...
	SAS address (e.g., D_NZ) of the device attached to phy Z of the edge expander device EN
Level 3 entries: none since all devices attached to E1 through EN, except for E0, are end devices	

4.6.11.5 Discover process

The discover process shall perform a traversal of the SAS domain to identify the initiator devices, target devices and expander devices. The order of traversal shall be to discover:

1) the expander device to which the initiator port is attached;
2) every device attached to that expander device; and
3) as each expander device is found, every device attached to that expander device.

The discover process is repeated until all expander devices have been traversed.

The discover process begins with the application client of an initiator device determining that an expander device is attached.

If either an edge device or fanout device is attached, then the discover process determines if the expander device requires configuration. If the expander device requires configuration, then expander route entries are configured with the SAS addresses of the devices routed to the phys with the table routing attribute. The order of the expander route entries is described in 4.6.11.4.

The result of the discover process is that the application client has the necessary information to communicate with each device in the SAS domain and each configurable expander device is configured with the expander route entries to allow routing of connection requests through the SAS domain.

Annex I contains an example algorithm used to perform the discover process.

5 Physical layer

5.1 SATA cables and connectors (informative)

Figure 31 shows the cables and connectors defined by SATA (for reference). A SATA host is equivalent to a SAS initiator device; a SATA device is equivalent to a SAS target device.

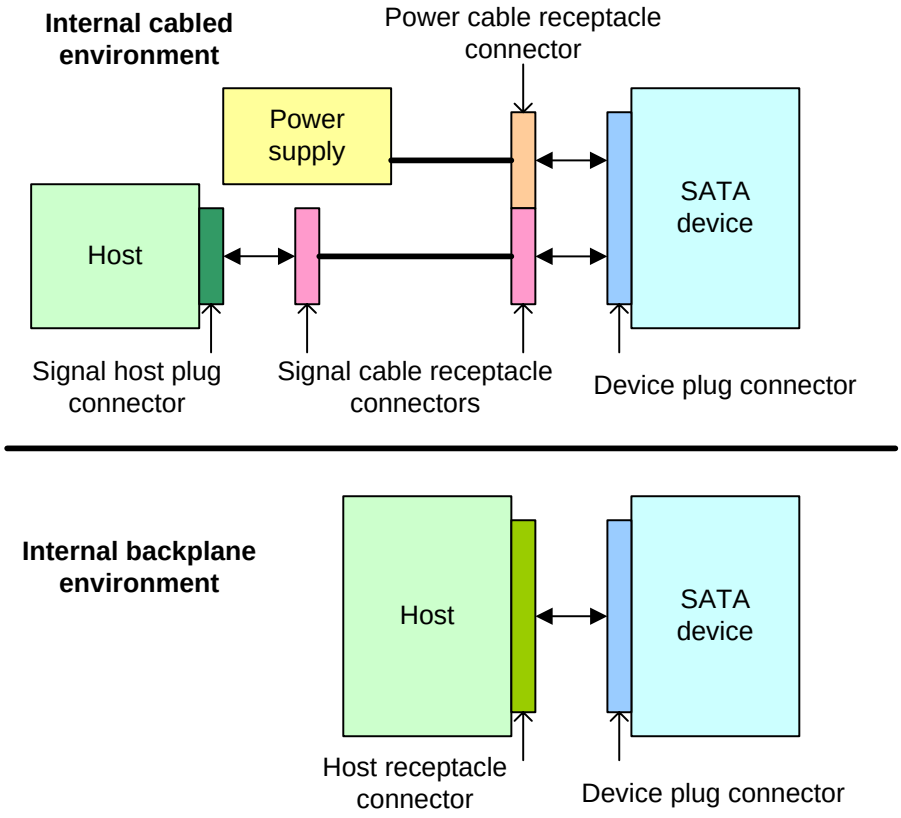


Figure 31 — SATA cables and connectors (informative)

5.2 SAS cables and connectors

This standard supports external cable, internal cable, and internal backplane environments.

Figure 32 shows the cables and connectors defined in this standard to support an external environment.

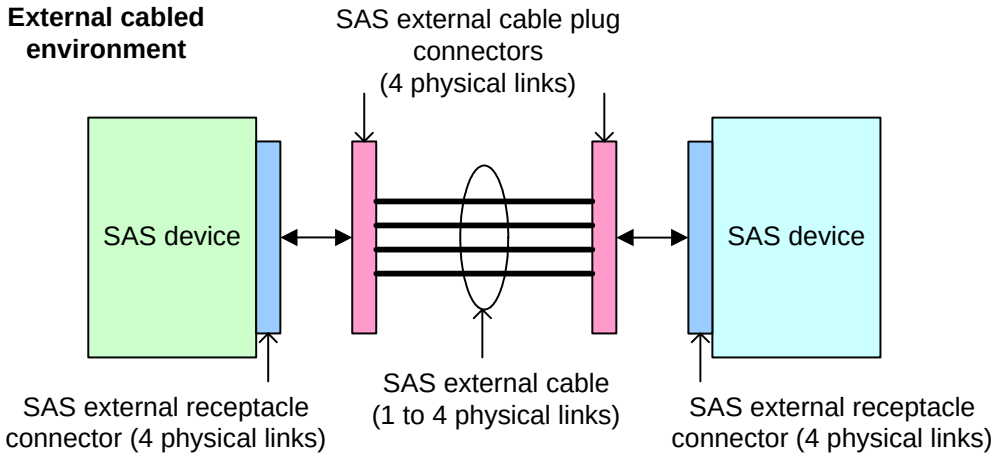


Figure 32 — SAS cables and connectors - external environment

Figure 33 shows the connectors defined in this standard for internal environments.

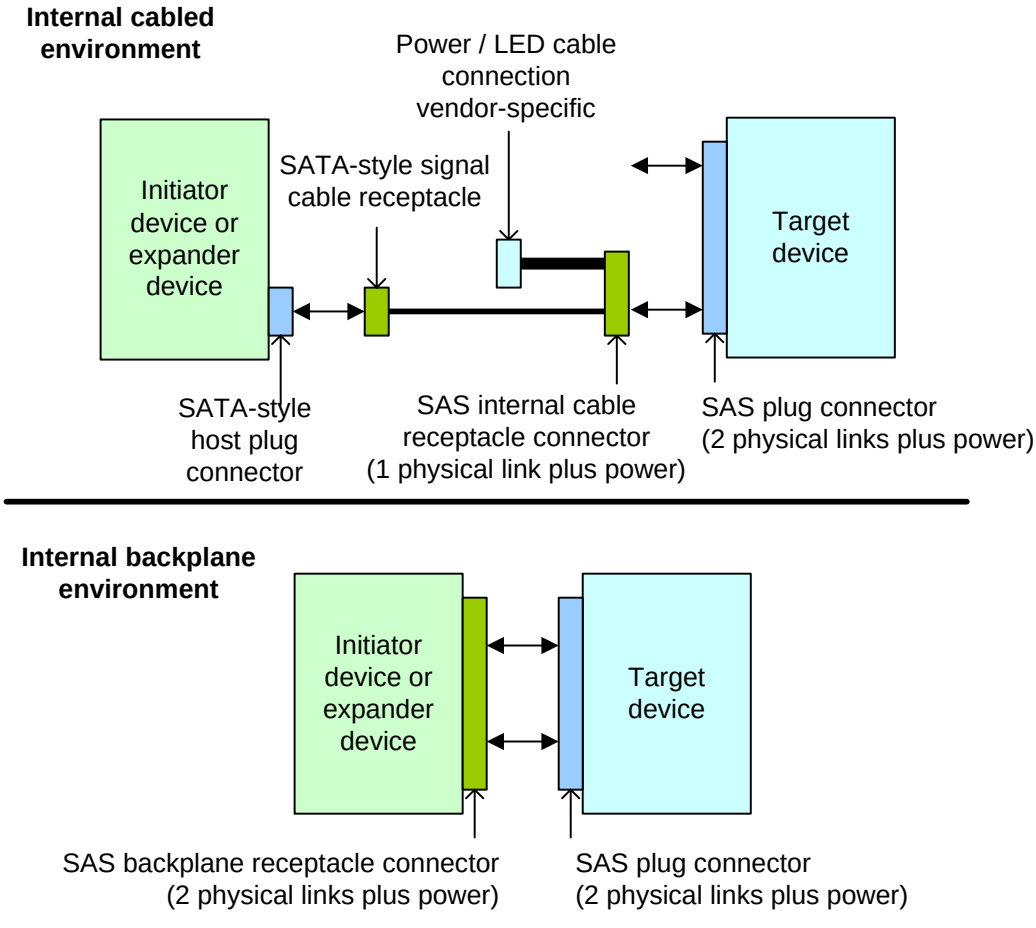


Figure 33 — SAS cables and connectors - internal environment

Table 29 summarizes the connectors defined in this standard.

Table 29 — Connectors

Type of connector	Reference	Attaches to	Reference
SAS plug	5.3.2	SAS internal cable receptacle	5.3.3
		SAS backplane receptacle	5.3.4
SAS internal cable receptacle	5.3.3	SAS plug	5.3.2
		SATA device plug	SATA
SAS backplane receptacle	5.3.4	SAS plug	5.3.2
		SATA device plug (single-port)	SATA
SAS external cable plug	5.3.6	SAS external receptacle	5.3.7
SAS external receptacle	5.3.7	SAS external cable plug	5.3.6

The SATA device plug connector (e.g., used by a disk drive) may be attached to a SAS backplane receptacle connector or a SAS internal cable receptacle connector, connecting the primary signal pairs and leaving the second signal pairs unconnected.

See SFF-8223, SFF-8323, and SFF-8523 for the connector locations on common ~~drive~~ form factors.

5.3 Connectors

5.3.1 Connectors overview

~~SAS connectors should be marked with the SAS logo (see Annex J).~~

5.3.2 SAS plug connector

5.3.2.1 SAS plug connector overview

SAS devices with internal ports shall use the SAS plug connector. The SAS plug connector is defined in SFF-8482. It attaches to SAS internal cable receptacle connectors (for SAS cables) and SAS backplane receptacle connectors (for SAS backplanes).

Table 30 defines the pin assignments.

5.3.3 SAS internal cable receptacle connector

SAS internal cables shall use the SAS internal cable receptacle connector on the target device end. The SAS internal cable receptacle connector is defined in SFF-8482. It attaches to either:

a) a SAS plug connector, providing contact for the power pins and the primary physical link; or
b) a SATA device plug connector, providing contact for ~~only~~ the power pins and the primary physical link.

Table 30 defines the pin assignments. The secondary physical link, pins S8 through S14, is not supported by the internal cable receptacle.

5.3.4 SAS backplane receptacle connector

SAS backplanes shall use the SAS backplane receptacle connector. The SAS backplane receptacle connector (see SFF-8482) attaches to either:

a) a SAS plug connector, providing contact for the power pins and both primary and secondary physical links; or
b) a SATA device plug connector, providing contact for ~~only~~ the power pins and the primary physical link.

Table 30 defines the pin assignments.

5.3.5 SAS internal connector pin assignments

Table 30 shows the pins in the SAS internal connectors.

Table 30 — SAS internal connector pins

Segment	Pin	Name
Primary Signal	S1	GROUND
Primary Signal	S2	AR+
Primary Signal	S3	AR-
Primary Signal	S4	GROUND
Primary Signal	S5	AT-
Primary Signal	S6	AT+
Primary Signal	S7	GROUND
Secondary Signal [b]	S8	GROUND
Secondary Signal [b]	S9	BR+
Secondary Signal [b]	S10	BR-
Secondary Signal [b]	S11	GROUND
Secondary Signal [b]	S12	BT-
Secondary Signal [b]	S13	BT+
Secondary Signal [b]	S14	GROUND
Power [a]	P1	V_{33}
Power [a]	P2	V_{33}
Power [a]	P3	V_{33}, precharge
Power [a]	P4	GROUND
Power [a]	P5	GROUND
Power [a]	P6	GROUND
Power [a]	P7	V_5, precharge
Power [a]	P8	V_5
Power [a]	P9	V_5
Power [a]	P10	GROUND
Power [a]	P11	READY LED
Power [a]	P12	GROUND
Power [a]	P13	V_{12}, precharge
Power [a]	P14	V_{12}
Power [a]	P15	V_{12}

[a] Precharge and voltage pins shall be connected together.
[b] S8 through S14 are no-connects on single port implementations.

SAS target device signal assignments, except for the addition of the secondary physical link when present, are in the same locations as they are in a SATA target device. On cable assemblies, backplanes, or any other connection media, the Tx signal from one internal connector pair shall be connected to the corresponding Rx signal of the other internal connector pair (i.e., AT+ of connector 1 shall connect to AR+ of connector 2) if there is an internal connector at both ends of the transmission media.

The AT+, AT-, AR+, and AR- signals are used by the primary physical link. The BT+, BT-, BR+, and BR- signals are used by the secondary physical link.

5.3.6 SAS external cable plug connector

SAS external cables shall use the SAS external cable plug connector. The SAS external cable plug connector is defined in SFF-8470 as the 4x configuration with thumbscrews. No special SAS keying is provided. It attaches to a SAS external receptacle connector, providing contact for up to four physical links.

Table 31 defines the pin assignments.

5.3.7 SAS external receptacle connector

SAS devices with external ports shall use the SAS external receptacle connector. The SAS external receptacle connector is defined in SFF-8470 as the 4x configuration with thumbscrews. No special SAS keying is provided. It attaches to a SAS external cable plug connector, providing contact for up to four physical links.

Table 31 defines the pin assignments.

5.3.8 SAS external connector pin assignments

Table 31 shows how the connector signal pairs are used in external connectors for applications using one, two, three, or four of the physical links.

Table 31 — Physical link usage in SAS external connector

Signal	Signal pin to use based on number of physical links supported by the cable			
	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			
Key: N/C = not connected				

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the cable connector.

5.4 Cables

5.4.1 SAS internal cables

SAS internal cables shall use SAS internal cable receptacle connector on the target device end and a SATA-style cable receptacle on the initiator device or expander device end. The power and READY LED signal connection is vendor-specific.

A SAS initiator device shall use a SATA-style host plug connector for connection to the SAS internal cable. The SATA host plug connector is defined in SATA. The signal assignment for the SAS initiator device or expander device with this connector shall be the same as defined for a SATA host in SATA.

Figure 34 shows destination signal assignments and a connection diagram for the SAS internal cable.

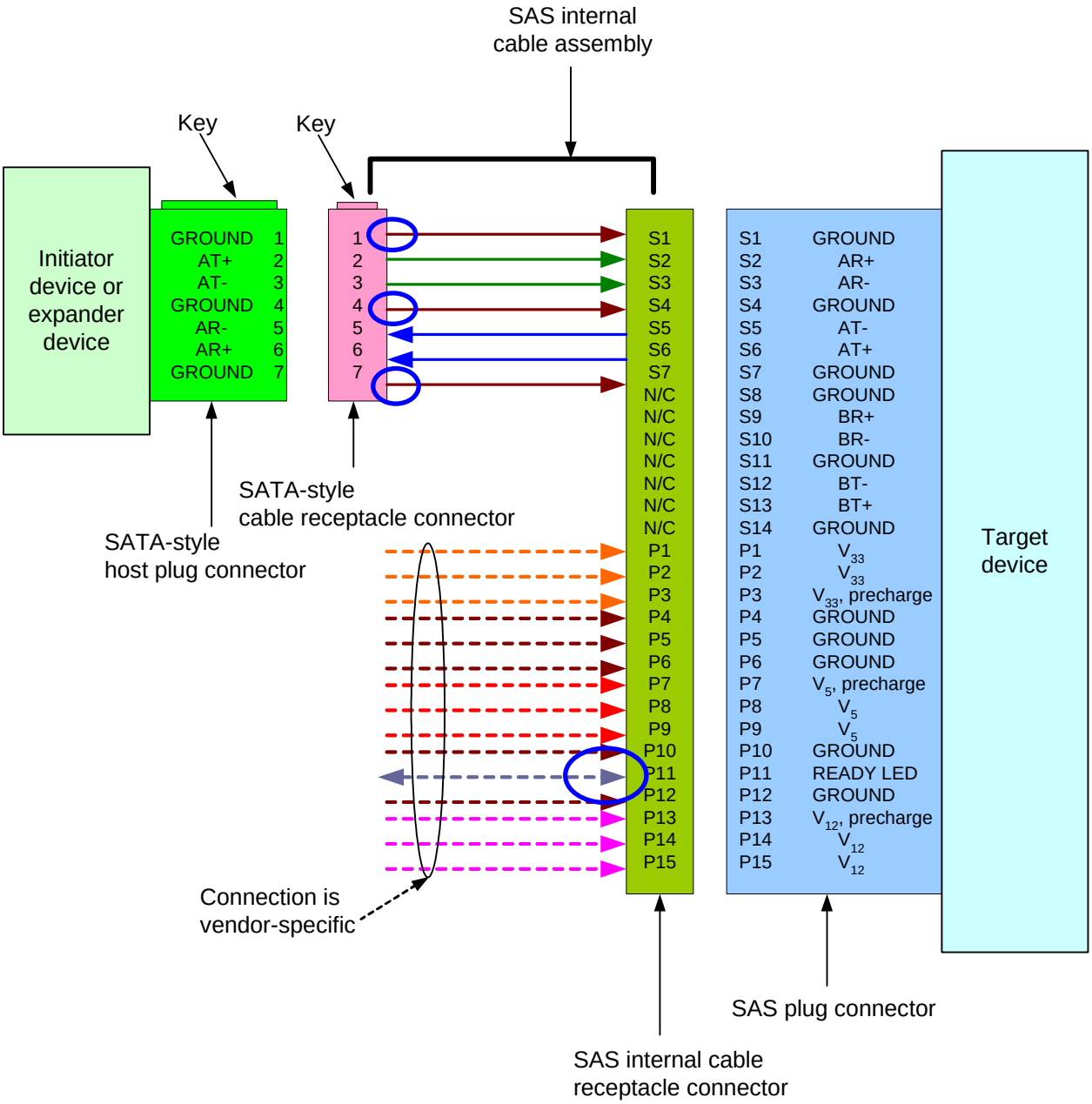


Figure 34 — SAS itnernal cable assembly and destination pin assignments

5.4.2 SAS external cables

The SAS external cable connectors are defined in SFF-8470 as the 4x configuration with thumbscrews. The external cable does not carry power signals or the READY LED signal.

Although the connector always supports four physical links, the cable may support one, two, three, or four physical links.

On external cable assemblies, the Tx signal from one connector shall be connected to the corresponding Rx signal of the other connector (e.g., Tx 0+ (S16) of connector shall connect to Rx 0+ (S1) of the other connector) (see 5.3.6).

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the cable.

5.5 Backplanes

Backplane designs should follow the recommendations in SFF-8460.

5.6 READY LED pin

The READY LED signal may be driven by a target device to turn on an externally visible LED that indicates the state of readiness and activity of the target device.

All target devices shall support the READY LED signal. The system is not required to generate any visual output when the READY LED signal is raised, but if such a visual output is provided, it shall be white or green to indicate that normal activity is being performed. Additional vendor-specific flashing patterns may be used to signal vendor-specific conditions.

The READY LED signal is designed to pull down the cathode of an LED using an open collector or open drain driver circuit. The target device circuitry shall be GROUND-tolerant, ~~since this pin may be connected by a system directly to power supply GROUND.~~ The LED anode shall be attached to an appropriate supply through a current limiting resistor. The LED and the current limiting resistor may be external to the target device.

Table 32 describes the output characteristics of the READY LED signal.

Table 32 — Output characteristics of the READY LED signal

State	Output current	Output voltage
Drive LED off	-100 μA < I_{OH} < 100 μA	0 ≤ V_{OH} ≤ 3,6 V
Drive LED on	I_{OL} ≥ 15 mA	0 ≤ V_{OL} ≤ 0,225 V at I_{OL} = 15 mA

The READY LED signal may optionally be driven by a backplane to turn on the external LED. This usage is vendor specific.

The READY LED signal shall be driven by a target device using the following patterns:

a) If the target device is in the standby or stopped power condition state, it shall assert READY LED while processing a command (i.e., the LED is usually off, but flashes on when commands are processed). The target device may be removed with no danger of mechanical damage in this state;
b) If the target device has rotating media and is in the process of performing a spin-up or spin-down, it shall assert and negate READY LED with a 1 s ± 0,1 s cycle using a 50 % ± 10 % duty cycle (i.e., LED is on for 0,5 s and off for 0,5 s);
c) If the target device is in the active or idle power condition state, it shall assert READY LED continuously except when the target device is processing a command. When processing a command, the target device shall negate READY LED for a period long enough to be detected by an observer (i.e., LED is usually on, but flashes off when commands are processed); or
d) If the target device is formatting the media, it shall toggle READY LED between asserted and negated at significant intervals during the format operation (e.g., with each cylinder change on a disk drive). The interval is vendor-specific.

Target devices with rotating media transition from pattern c) to pattern b) during the spin-down process. When the target device has reached a state stable enough for it to be removed without mechanical damage, it shall change from pattern b) to pattern a).

5.7 Driver and receiver electrical characteristics

5.7.1 Compliance points

Signal behavior at separable connectors and integrated circuit package connections that satisfy the physical definition for a compliance point require compliance with transmitter and receiver characteristic tables only if the connectors or integrated circuit package connections are identified as compliance points by the supplier of the parts that contain or comprise the candidate compliance point. Table 33 lists the compliance points.

Table 33 — Compliance points

Compliance point	Type	Description
IT	intra-enclosure	Internal connector; transmit serial port
IR	intra-enclosure	Internal connector; receive serial port
CT	inter-enclosure	External connector; transmit serial port
CR	inter-enclosure	External connector; receive serial port
XT	intra-enclosure	Expander or initiator phy; transmit serial port
XR	intra-enclosure	Expander or initiator phy; receive serial port

5.7.2 General interface specification

A TxRx connection is the complete simplex signal path between the output reference point of one SAS phy or retimer to the input reference point of a second SAS phy or retimer, ~~over which a BER of < 10^{-12} is achieved.~~

A TxRx connection segment is that portion of a TxRx connection delimited by separable connectors or changes in media.

This subclause defines the interfaces of the serial electrical signal at the interoperability points IT, IR, CT, and CR in a TxRx connection. The IT, IR, CT, and CR points are located at the connectors of a TxRx Connection.

Each conforming SAS phy shall be compatible with this serial electrical interface to allow interoperability within a SAS environment. All TxRx connections described in this subclause shall operate within the BER objective of 10^{-12}. The parameters specified in this section support meeting that requirement under all conditions including the minimum input and output amplitude levels.

TxRx connections operating at the maximum specified distances may require some form of equalization (e.g., transmitter pre-emphasis, receiver adaptive equalization, or passive cable equalization) to enable the signal requirements to be met.

Figure 35 shows the transmitter transient test circuit.

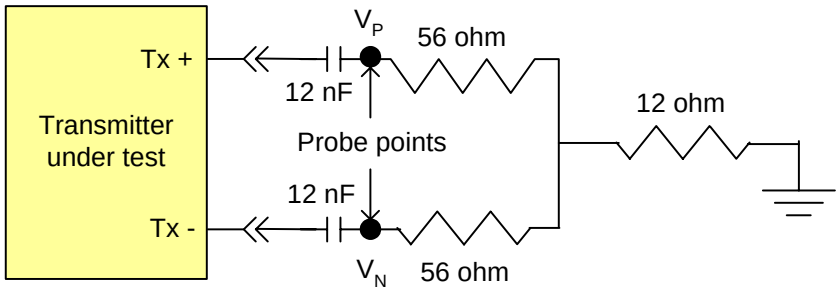


Figure 35 — Transmitter transient test circuit

Figure 36 shows the receiver transient test circuit.

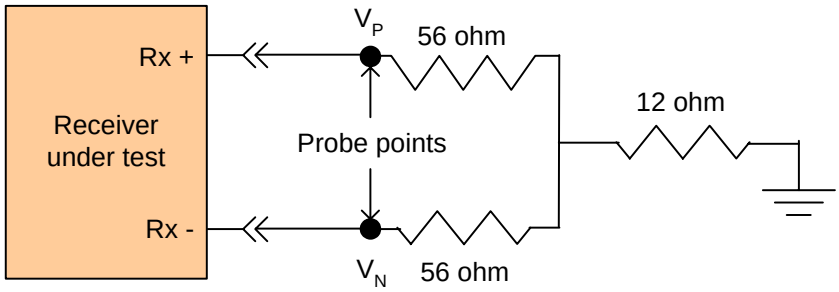


Figure 36 — Receiver transient test circuit

Table 34 defines the general interface characteristics.

Table 34 — General interface characteristics

Characteristic	Units	1,5 Gbps	3,0 Gbps
Physical link rate	MBps	150	300
Bit rate (nominal)	Mbaud	1 500	3 000
Unit interval (UI)	ps	666,667	333,333
Physical link rate tolerance at XR [b]	ppm	+350 / -5 350	+350 / -5 350
Physical link rate tolerance at IR and CR	ppm	± 100	± 100
Physical link rate tolerance at IT, CT, and XT	ppm	± 100	± 100
Media Impedance [a]	ohm	100	100
A.C. coupling capacitor, maximum [c]	nF	12	12
Transmitter transients, maximum [d]	V	± 1,2	± 1,2
Receiver transients, maximum [d]	V	± 1,2	± 1,2
Receiver A.C. common mode voltage tolerance V_{CM}, minimum [e]	mV(P-P)	150	150
Receiver A.C. common mode frequency tolerance range F_{CM} [e]	MHz	2 to 200	2 to 200

[a] The media impedances are the differential, ~~or odd mode,~~ impedances.
[b] Allows support for SATA devices with spread spectrum clocking.
[c] The coupling capacitor value for A.C. coupled transmit and receive pairs.
[d] The maximum transmitter and receiver transients are measured at nodes V_P and V_N on the test loads shown in figure 35 (for the transmitter) and figure 36 (for the receiver) during all power state and mode transitions. Test conditions shall include the system power supply ramping at the fastest possible rate (both up and down).
[e] Receivers shall tolerate sinusoidal common mode noise components within the peak-to-peak amplitude (V_{CM}) and the frequency range (F_{CM}).

5.7.3 Eye masks

5.7.3.1 Eye masks overview

The eye masks shown in this subclause shall be interpreted as graphical representations of the voltage and time limits. The time values between X1 and (1 - X1) cover all but 10^{-12} of the jitter population. The random content of the total jitter population has a range of ± 7 sigma.

5.7.3.2 Delivered (receive) eye mask at IR, CR, and XR

Figure 37 describes the delivered (receive) eye mask. 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 the (bit rate) / 1 667.

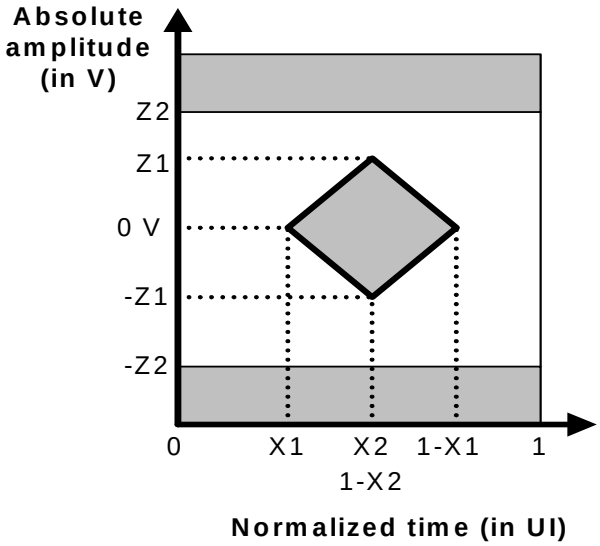


Figure 37 — Eye mask at IR, CR, and XR

Verifying compliance with the limits represented by the received eye mask should be done with reverse channel traffic present in order that the effects of crosstalk are taken into account.

5.7.3.3 Jitter tolerance masks

Figure 38 describes the receiver tolerance eye masks at IR, CR, and XR and shall be constructed using the X2 and Z2 values given in table 36. $X1_{OP}$ shall be half the value for total jitter in table 37 and $X1_{TOL}$ shall be half the value for total jitter in table 38, for jitter frequencies above (bit rate) / 1 667.

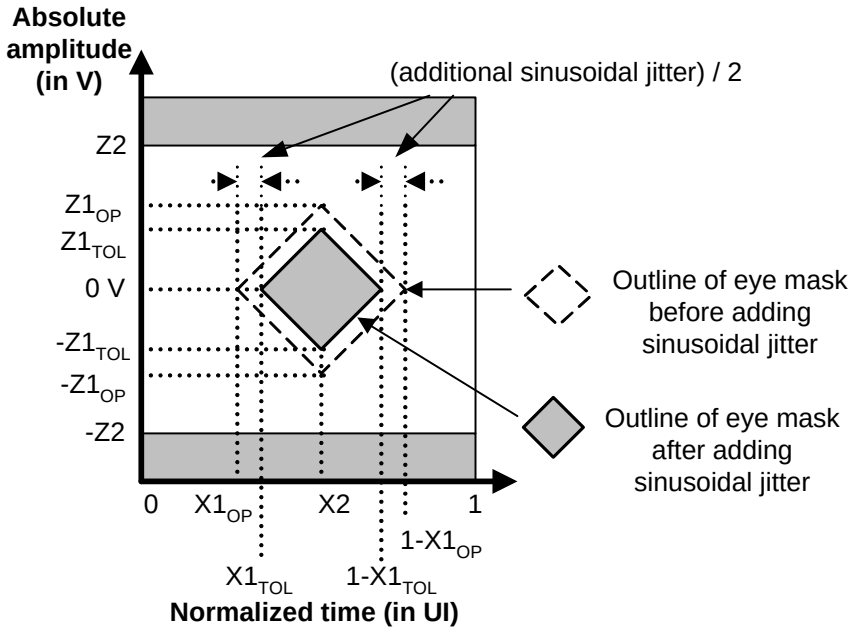


Figure 38 — Deriving a tolerance mask at IR, CR, or XR

However, the leading and trailing edge slopes of figure 37 shall be preserved. As a result the amplitude value of Z1 is less than that given in table 36 and shall be calculated from those slopes as follows:

$$Z1_{TOL} = Z1_{OP} \times (X2_{OP} - (0,5 \times (\text{additional sinusoidal jitter})) - X1_{OP}) / (X2_{OP} - X1_{OP})$$

Where:

a) $Z1_{TOL}$ is the value for Z1 to be used for the tolerance masks; and
b) $Z1_{OP}$, $X1_{OP}$, and $X2_{OP}$ are the values in table 36 for Z1, X1, and X2.

The X1 points in the receive tolerance masks are greater than the X1 points in the receive masks, due to the addition of sinusoidal jitter.

Figure 39 defines the sinusoidal jitter mask.

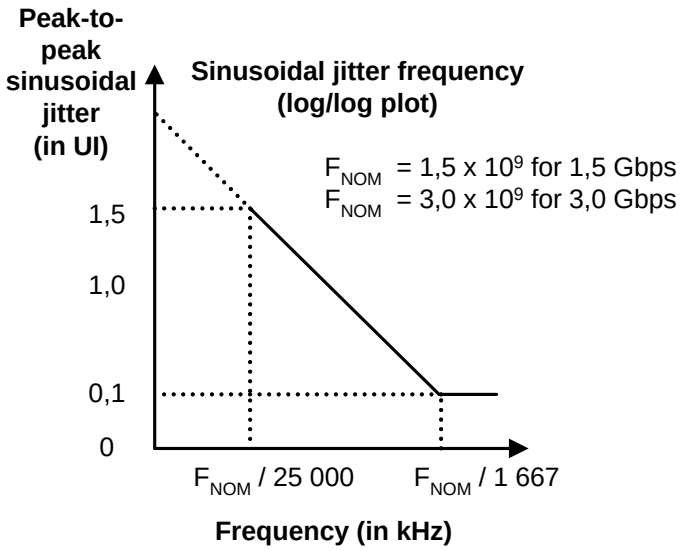


Figure 39 — Sinusoidal jitter mask

5.7.4 Transmitted signal characteristics

This subclause defines the inter-operability requirements of the transmitted signal at the driver end of a TxRx connection as measured into the zero-length test load specified in figure 41. All specifications are based on differential measurements.

The OOB sequence shall be performed at signal voltage levels corresponding to the lowest supported transfer rate. Expander phys supporting attachment to SATA target devices shall use SATA 1.0 signal levels during the first OOB sequence after a power on or hard reset if the 1,5 Gbps transfer rate is supported. As soon as COMSAS has been exchanged, the expander phy shall increase its transmit levels to the SAS voltage levels specified in table 36. If a COMINIT is not received within a hot plug timeout at SATA 1.0 signal levels, the expander phy shall increase its transmit levels to the to SAS voltage levels and perform the OOB sequence again. If no COMINIT is received within a hot plug timeout of the second OOB sequence the expander phy shall initiate another OOB sequence using SATA 1.0 signal levels. The expander phy shall continue alternating between sending COMINIT at SATA 1.0 signal levels and SAS signal levels until a COMINIT is received.

If the OOB sequence is completed at the SAS voltage level and the target device is determined to be a SATA device, the initiator phy or expander phy shall switch to SATA 1.0 voltage levels and repeat the OOB sequence.

NOTE 9 SAS initiator phys supporting attachment to SATA target devices may use the same algorithm as SAS expander phys.

SAS initiator phys and SAS target phys shall transmit OOB signals at the lowest supported transfer rate using SAS signal levels.

~~During the idle time of an OOB signal the transmitter output shall meet the following requirements:~~

~~a) The differential voltage D.C. (offset) component shall be equal to the average differential voltage during the burst time of an OOB signal ± 25 mV;~~
~~b) The differential voltage A.C. component shall be less than 50 mV(P-P); and~~
~~c) The common mode voltage shall be equal to the average of common mode during the burst time of an OOB signal, ± 50 mV.~~

Table 35 specifies the transmitted signal characteristics at Tx compliance points.

Table 35 — Transmitted signal characteristics at Tx compliance points

Compliance point	Signal characteristic [a]	Units	1,5 Gbps	3,0 Gbps
IT, CT, XT	Skew [b]	ps	20	15
	Tx Off Voltage [c]	mV(P-P)	< 50	< 50
	Maximum rise/fall time [d]	ps	273	137
	Minimum rise/fall time [d]	ps	133	67
	Maximum transmitter output imbalance [e]	%	10	10

[a] All tests in this table shall be performed with zero-length test load shown in figure 41.
[b] The skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Tx+ and Tx- signals. 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 off voltage is the maximum A.C. voltage measured at compliance points IT, CT, and XT when the transmitter is logically turned off or unpowered.
[d] Rise/fall times are measured from 20 % to 80 % of the transition with a repeating 0101b pattern on the physical link.
[e] The maximum difference between the V+ and V- A.C. RMS transmitter amplitudes measured on a CJTPAT test pattern into the test load shown in figure 41, as a percentage of the average of the V+ and V- A.C. RMS amplitudes.

5.7.5 Received signal characteristics

Table 36 defines the compliance point requirements of the delivered signal at the receiver end of a TxRx connection as measured into the test loads specified in figure 40 and figure 41.

Table 36 — Delivered signal characteristic at Rx compliance points (part 1 of 2)

Compliance point	Signal characteristic	Units	SATA	1,5 Gbps	3,0 Gbps
IR [e]	Jitter (see figure 37) [b]	N/A	N/A	See table 37	See table 37
	2 x Z2	mV(P-P)	N/A	1 200	1 600
	2 x Z1	mV(P-P)	N/A	325	275
	X1 [a]	UI	N/A	0,275	0,275
	X2	UI	N/A	0,50	0,50
	Skew [d]	ps	N/A	80	75
	Max voltage (non-op)	mV(P-P)	N/A	2 000	2 000
	OOB detect guaranteed on (eye opening) [c]	mV(P-P)	N/A	240	240
	OOB detect guaranteed off signal level [c]	mV(P-P)	N/A	120	120
	Max near-end crosstalk	mV(P-P)	N/A	100	100

Table 36 — Delivered signal characteristic at Rx compliance points (part 2 of 2)

Compliance point	Signal characteristic	Units	SATA	1,5 Gbps	3,0 Gbps
CR	Jitter (see figure 37) [b]		N/A	See table 37	See table 37
	2 x Z2	mV(P-P)	N/A	1 600	1 600
	2 x Z1	mV(P-P)	N/A	275	275
	X1 [a]	UI	N/A	0,275	0,275
	X2	UI	N/A	0,50	0,50
	Skew [d]	ps	N/A	80	75
	Max voltage (non-op)	mV(P-P)	N/A	2 000	2 000
	OOB detect guaranteed on (eye opening) [c]	mV(P-P)	N/A	240	240
	OOB detect guaranteed off signal level [c]	mV(P-P)	N/A	120	120
	Max near-end crosstalk	mV(P-P)	N/A	100	100
XR	Jitter (see figure 37) [b]		See table 37	See table 37	See table 37
	2 x Z2	mV(P-P)	600	1 200	1 600
	2 x Z1	mV(P-P)	225	325	275
	X1 [a]	UI	0,275	0,275	0,275
	X2	UI	0,50	0,50	0,50
	Skew [d]	ps	50	80	75
	Max voltage (non-op)	mV(P-P)	2 000	2 000	2 000
	OOB detect guaranteed on (eye opening) [c]	mV(P-P)	240	240	240
	OOB detect guaranteed off signal level [c]	mV(P-P)	120	120	120
	Max near-end crosstalk	mV(P-P)	< 50	100	100

[a] The value for X1 shall be half the value given for total jitter in table 37. 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.

[b] 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 output requirements.

[c] Worst case signals are trapezoidal. The guaranteed on level is based on a 2x ratio to the guaranteed off level.

[d] The skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Rx+ and Rx- signals. Skew is defined as the time difference between the means of the midpoint crossing times of the Rx+ signal and the Rx- signal.

[e] If SATA device attachment is supported at the IR location, requirements of SATA shall be met at IR.

5.7.6 Jitter

Table 37 defines the allowable jitter at the compliance points.

Table 37 — Jitter compliance points

Compliance point	1,5 Gbps		3,0 Gbps	
	Deterministic jitter	Total jitter	Deterministic jitter	Total jitter
IR	0,35	0,55	0,35	0,55
CR	0,35	0,55	0,35	0,55
XR	0,35	0,55	0,35	0,55

[a] Units are in UI.
[b] The values for jitter in this section are measured at the average amplitude point.
[c] Total jitter is the sum of deterministic jitter and random jitter. If the actual deterministic jitter is less than the maximum specified, then the random jitter may increase as long as the total jitter does not exceed the specified maximum total jitter.
[d] Total jitter is specified at a probability of 10^{-12}.
[e] The deterministic and total values in this table apply to jitter after 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.
[f] If total jitter delivered at any point is less than the maximum allowed, then the jitter distribution of the signals is allowed to be asymmetric. The total jitter plus the magnitude of the asymmetry shall not exceed the allowed maximum total jitter. 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 total jitter - |Asymmetry|).

5.7.7 Jitter tolerance

Table 38 defines the minimum allowable jitter at the compliance points.

Table 38 — Jitter tolerance compliance points

Compliance point	1,5 Gbps			3,0 Gbps		
	Sinusoidal jitter [b]	Deterministic jitter [c, f]	Total jitter [f]	Sinusoidal jitter [d]	Deterministic jitter [e, f]	Total jitter [f]
CR	0,10	0,35	0,65	0,10	0,35	0,65
IR	0,10	0,35	0,65	0,10	0,35	0,65
XR	0,10	0,35	0,65	0,10	0,35	0,65

[a] Units are in UI.
[b] Sinusoidal swept frequency: 900 kHz to > 5 MHz. The jitter values given are normative for a combination of deterministic jitter, random jitter, and sinusoidal jitter that receivers shall be able to tolerate without exceeding a BER of 10^{-12}.
[c] Deterministic jitter: 900 kHz to 750 MHz. No value is given for random jitter. For compliance with this standard, the actual random jitter amplitude shall be the value that brings total jitter to the stated value at a probability of 10^{-12}.
[d] Sinusoidal swept frequency: 1 800 kHz to > 5 MHz. Receivers shall tolerate sinusoidal jitter of progressively greater amplitude at lower frequencies, according to the mask in figure 39 with the same deterministic jitter and random jitter levels as were used in the high frequency sweep.
[e] Deterministic jitter: 1 800 kHz to 1 500 MHz. The additional 0,1 UI of sinusoidal jitter is added to ensure the receiver has sufficient operating margin in the presence of external interference.
[f] The deterministic and total values in this table apply to jitter after 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.

5.7.8 Jitter compliance test pattern (CJTPAT)

The jitter test pattern within a compliant protocol frame (CJTPAT) shall be used for all jitter testing unless otherwise specified. Annex A defines the required pattern on the physical link and information regarding special considerations for scrambling and running disparity.

5.7.9 Impedance specifications

Table 39 defines impedance requirements.

Table 39 — Impedance requirements

Requirement	Units	1,5 Gbps	3,0 Gbps
Time domain reflectometer rise time 20 % to 80 % [a, b]	ps	100	50
Media (PCB or cable)			
Differential impedance [b, c, d]	ohm	100 ± 10	100 ± 10
Single-ended impedance match [b, c, d]	ohm	± 5	± 5
Common mode impedance [b, c, d]	ohm	32,5 ± 7,5	32,5 ± 7,5
Mated connectors			
Differential impedance [b, c, d]	ohm	100 ± 15	100 ± 15
Single-ended impedance match [b, c, d]	ohm	± 5	± 5
Common mode impedance [b, c, d]	ohm	32,5 ± 7,5	32,5 ± 7,5
Receiver termination			
Differential impedance [b, e, f]	ohm	100 ± 15	100 ± 15
Single-ended impedance match [b, e, f]	ohm	± 5	± 5
Receiver termination time constant [b, e, f]	ps	150 max	100 max
Common mode impedance [b, e]	ohm	20 min/40 max	20 min/40 max
Transmitter source termination			
Differential impedance [b]	ohm	60 min/115 max	60 min/115 max
Single-ended impedance match [b]	ohm	± 5	± 5
Common mode impedance [b]	ohm	15 min/40 max	15 min/40 max

[a] All times indicated for time domain reflectometer measurements are recorded times. Recorded times are twice the transit time of the time domain reflectometer signal.

[b] All measurements are made through mated connector pairs.

[c] The media impedance measurement identifies the impedance mismatches present in the media when terminated in its characteristic impedance. This measurement includes mated connectors at both ends of the media, when present, and any intermediate connectors or splices.

[d] Where the media 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 receiver termination impedance specification applies to all receivers in a TxRx connection and covers all time points between the connector nearest the receiver, the receiver, and the transmission line terminator. This measurement shall be made from that connector.

[f] At the time point corresponding to the connection of the receiver to the transmission line the input capacitance of the receiver and its connection to the transmission line may cause the measured impedance to fall below the minimum impedances specified in this table. The area of the impedance dip (amplitude in units of Γ, the reflection coefficient, and duration in time) caused by this capacitance is the receiver termination time constant. The receiver 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 Γ) and its width (in ps) measured at the half amplitude point. The amplitude is defined as being 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 excess input capacitance is given by C = (receiver termination time constant) / (R0 || RR), where (R0 || RR) is the parallel combination of the transmission line characteristic impedance and termination resistance at the receiver

5.7.10 Electrical TxRx connections

TxRx connections may be divided into TxRx connection segments. In a single TxRx connection individual TxRx connection segments may be formed from differing media and materials, including traces on printed wiring boards and optical fibers. This subclause applies only to TxRx connection segments that are formed from an electrically conductive media.

Each electrical TxRx connection segment shall comply with the impedance requirements of table 39 for the media from which they are formed. An equalizer network, if present, shall be part of the TxRx connection.

TxRx connections that are composed entirely of electrically conducting media 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).

5.7.11 Transmitter characteristics

For all inter-enclosure TxRx connections, the transmitter shall be A. C. coupled to the interconnect through a transmission network.

For intra-enclosure TxRx connections the expander transmitter shall be A. C. coupled to the interconnect. Other transmitters may be A. C. or D. C. coupled.

A combination of a zero-length test load and the transmitter compliance transfer function (TCTF) test load methodology is used for the specification of the external, initiator, expander or target device transmitter characteristics. This methodology specifies the transmitter signal at the test points on the required test loads. The transmitter shall use the same settings (e.g., pre-emphasis, voltage swing, etc.) with both the zero-length test load and the TCTF test load. The received signal specifications shall be met under each of these loading conditions.

The TCTF is the mathematical statement of the transfer function through which the transmitter shall be capable of producing acceptable signals as defined by a receive mask. The transmission magnitude response, $[S_{21}]$, of the TCTF in dB satisfies the following equation:

$$| S_{21} | \leq -20 \log (e) \{[6,5 \times 10^{-6} (f^{0,5})] + [2,0 \times 10^{-10} (f)] + [3,3 \times 10^{-20} (f^{2})]\} \text{ dB}$$

The TCTF is used to specify the requirements on transmitters that may or may not incorporate pre-emphasis or other forms of compensation. A compliance interconnect is any physical interconnect with equal or greater loss at all frequencies than that required by the TCTF.

Compliance with the TCTF test load requirement shall be determined either:

a) by measuring the signal produced by the transmitter through a physical compliance interconnect attached to the transmitter; or
b) by mathematically processing through the TCTF the signal captured using a zero-length test load.

Compliance with the zero-length test load requirement shall be determined by measurement made across a load equivalent to the zero-length load shown in figure 41.

For both test load cases, the transmitter shall deliver the output voltages and timing listed in table 36 at the designated compliance points. The default mask shall be CR for inter-cabinet TxRx connections and IR for intra-cabinet TxRx connections. The eye masks are shown in 5.7.3.

Figure 40 shows the compliance interconnect test load.

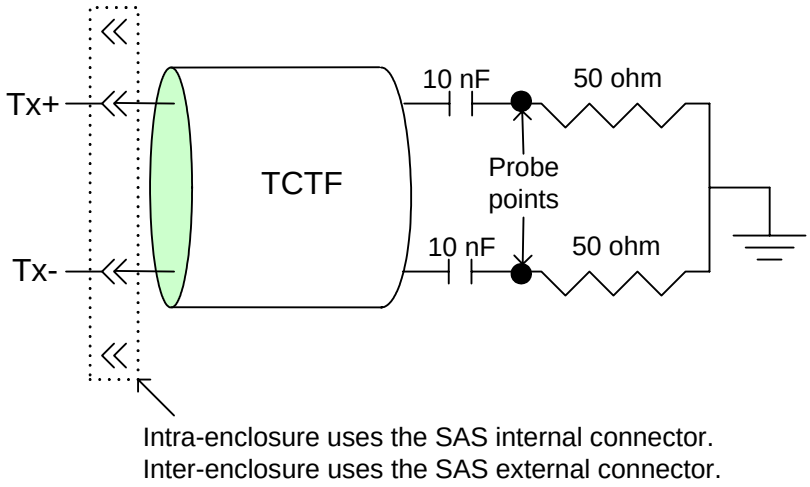


Figure 40 — Compliance interconnect test load

Figure 41 shows the zero-length test load.

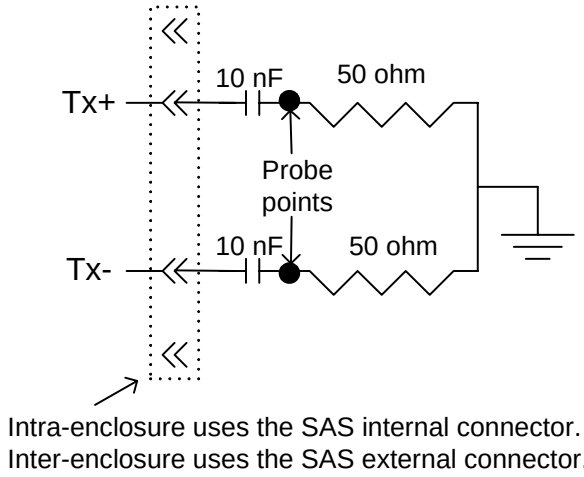


Figure 41 — Zero-length test load

Figure 42 shows an ISI loss example at 3,0 Gbps.

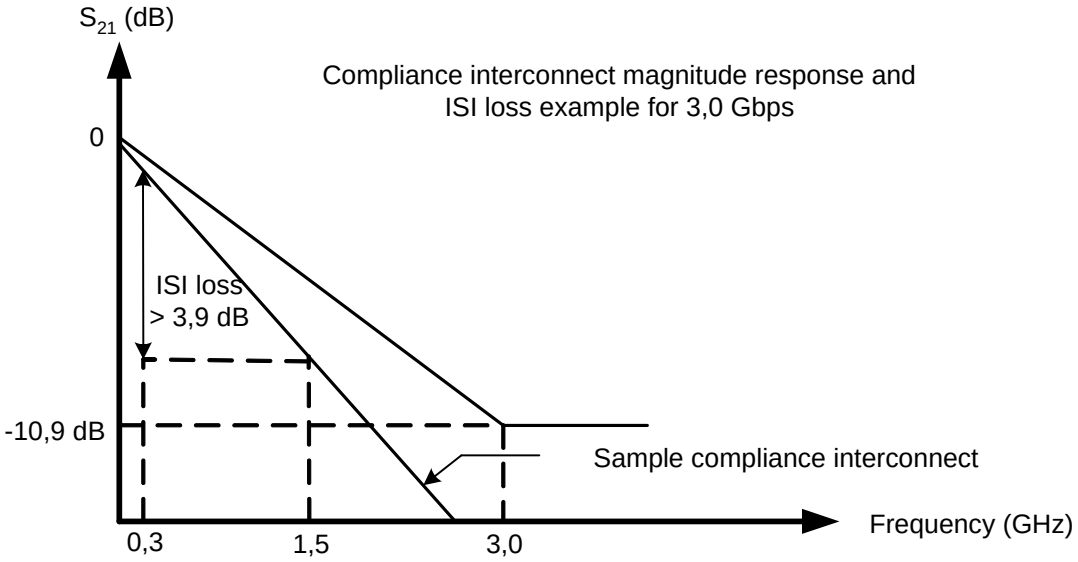


Figure 42 — ISI loss example at 3,0 Gbps

Figure 43 shows an ISI loss example at 1,5 Gbps.

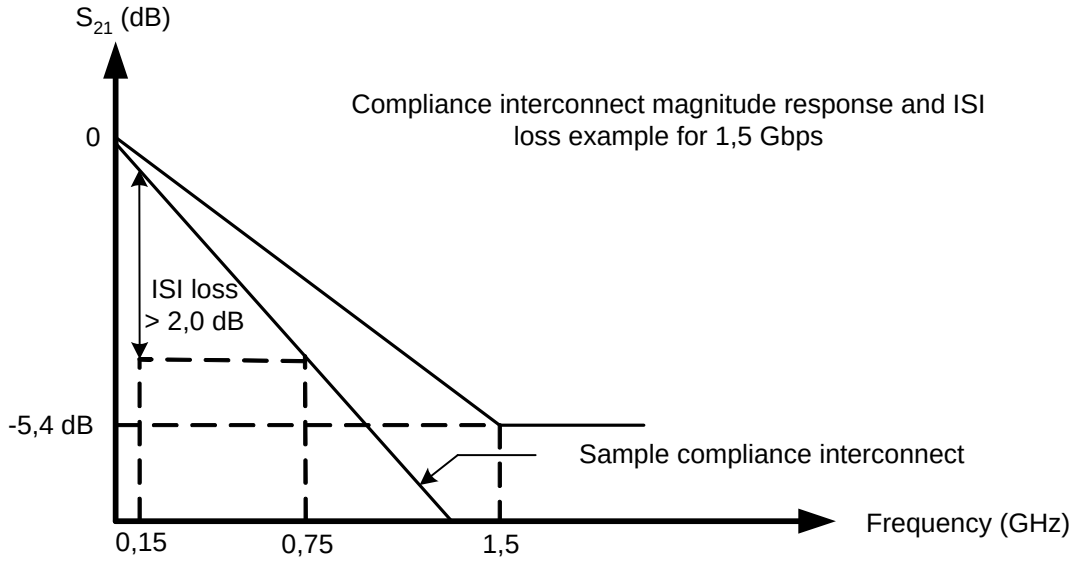


Figure 43 — ISI loss example at 1,5 Gbps

5.7.12 Receiver characteristics

The receiver 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 39.

The receiver shall operate within a BER of 10^{-12} when a SAS signal with valid voltage and timing characteristics is delivered to the compliance point from a 100 ohm source. The delivered SAS signal shall be considered valid if it meets the voltage and timing limits specified in table 36.

Additionally the receiver shall also operate within the BER objective when the signal at a receiving phy has the additional sinusoidal jitter present that is specified in table 38 and the common mode signal V_{CM} over frequency range F_{CM} as specified in table 34. The jitter tolerance figure is given in figure 38 for all Rx compliance points in a TxRx connection. The figure given 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, 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 receiving phy. The additional jitter reduces the eye opening in both voltage and time.

5.7.13 Spread spectrum clocking

Phys shall not transmit with spread spectrum clocking. Expander phys that support being attached to SATA target phys shall support receiving with spread spectrum clocking (see SATA). The expander device shall retime data from a SATA target phy with an internal clock before forwarding to the rest of the domain.

5.8 Non-tracking clock architecture

Phys shall be designed with a non-tracking clock architecture; the receive clock derived from the received bit stream shall not be used as the transmit clock. Expander phys that support being attached to SATA target phys shall tolerate clock tracking by the SATA target phy.

6 Phy layer

6.1 Phy layer overview

The phy layer interfaces to the physical layer, performs 8b10b coding, and manages the phy reset sequence.

6.2 Encoding (8b10b)

6.2.1 Encoding overview

All data bytes transferred in SAS are encoded into 10 bit data characters using 8b10b coding. Additional characters not related to data bytes are called control characters.

All characters transferred in SAS are grouped into four byte sequences called dwords. A primitive is a dword whose first byte is a control character and remaining three bytes are data characters.

Primitives are defined with both negative and positive starting disparity. SAS defines numerous primitives starting with the K28.5 control character. Table 40 shows special character usage in SAS and SATA.

Table 40 — Special character usage

First character	Usage in SATA (informative)	Usage in SAS
K28.3	All primitives except ALIGN	Primitives used only inside STP connections
K28.5	ALIGN	ALIGN and all primitives defined in this standard
Dxx.y	Data	Data

Primitives are defined in 7.1.

A data dword is a dword starting with a data character.

Disparity shall be maintained separately on each physical link. 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. Physical links 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 an 10-bit transmission character and then transmitted serially bit-by-bit across the link. Information received over the link shall be collected ten bits at a time, and those transmission characters that are used for data, called data characters, shall be decoded into the correct eight-bit codes. The 10-bit transmission code supports all 256 eight-bit combinations. Some of the remaining transmission characters, referred to as special characters, are used for functions that are to be distinguishable from the contents of a frame.

The encodings defined by the transmission code ensure 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 special characters of the transmission code contain a distinct and ~~easily~~ recognizable bit pattern (a comma) which assists a receiver in achieving word alignment on the incoming bit stream.

6.2.3 8b10b encoding 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 transmission 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, i, f, g, h, j of a 10-bit transmission 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 valid data byte. When the control variable associated with an unencoded information byte contains the value K, that byte is referred to as a special code.

The information bit labeled A corresponds to bit 0 in the numbering scheme of this specification, B corresponds to bit 1, and so on, as shown in table 41. The control variable is typically not specified. When the control variable is not specified, this standard assumes its value to be D (data).

Table 41 — 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 transmission character has been given a name using the following convention:

Zxx.y

Where:

a) Z is the control variable of the unencoded information byte. The value of Z is used to indicate whether the transmission character is a data character (Z = D) or a special character (Z = K);
b) 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; and
c) 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 42 shows the conversion from byte notation to the transmission character naming convention described above.

Table 42 — Conversion 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
Transmission character name	K	28	.	5					

Most Kxx.y combinations do not result in valid transmission characters within the 8b10b transmission code. Only those combinations that result in special characters as specified by table 44 are considered valid.

6.3 Character encoding and decoding

6.3.1 Introduction

This subclause describes how to select valid transmission characters (encoding) and check the validity of received transmission characters (decoding). It also specifies the ordering rules to be followed when transmitting the bits within a character and the characters within the higher-level constructs specified by the document (i.e., primitives and frames).

6.3.2 Transmission order

Within the definition of the 8b10b transmission code, the bit positions of the transmission 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. Bit i shall be 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 special character used to distinguish the primitive (e.g., K28.5 or K28.3) 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 (SOF delimiter) 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 (EOF delimiter) is transmitted.

6.3.3 Valid and invalid transmission characters

6.3.3.1 Definitions

Table 43 and table 44 define the valid data characters (Dxx.y characters) and valid special characters (Kxx.y characters), respectively, and shall be used for both generating valid transmission characters (encoding) and checking the validity of received transmission characters (decoding). Each Valid-Data-Byte or special code entry has two columns that represent two (not necessarily different) transmission characters, corresponding to the current value of the running disparity (Current RD - or Current RD +). Running disparity is a binary parameter with either the value negative (-) or the value positive (+). The running disparity at the beginning of an primitive is the beginning running disparity (beginning RD).

After powering on, the transmitter shall initialize the Current RD to negative. Upon transmission of any transmission character, the transmitter shall calculate a new value for its running disparity based on the contents of the transmitted character.

After powering on or exiting diagnostic mode (the definition of diagnostic mode is beyond the scope of this standard), the receiver should assume either the positive or negative value for its initial running disparity. Upon reception of any transmission character, the receiver shall determine whether the transmission character is valid or invalid according to the following rules and shall calculate a new value for its running disparity based on the contents of the received character.

The following rules for running disparity shall be used to calculate the new running disparity value for transmission characters that have been transmitted (i.e. transmitter's running disparity) and that have been received (i.e. receiver's running disparity).

Running disparity for a transmission character shall be calculated on the basis of sub-blocks, where the first six bits ('abcdei' b) form one sub-block (six-bit sub-block) and the second four bits ('fghj' b) form the other sub-block (four-bit sub-block). Running disparity at the beginning of the six-bit sub-block is the running disparity at the end of the last transmission character. Running disparity at the beginning of the four-bit sub-block is the running disparity at the end of the six-bit sub-block. Running disparity at the end of the transmission character is the running disparity at the end of the four-bit sub-block.

Running disparity for the sub-blocks shall be calculated as follows:

a) Running disparity at the end of any sub-block is positive if the sub-block contains more ones than zeros. It is also positive at the end of the six-bit sub-block if the six-bit sub-block is 000111b, and it is positive at the end of the four-bit sub-block if the four-bit sub-block is 0011b.
b) Running disparity at the end of any sub-block is negative if the sub-block contains more zeros than ones. It is also negative at the end of the six-bit sub-block if the six-bit sub-block is 111000b, and it is negative at the end of the four-bit sub-block if the four-bit sub-block is 1100b.
c) Otherwise, running disparity 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 are disparity neutral. In order to limit the run length of zeros or ones between sub-blocks, the 8b10b transmission code rules specify that sub-blocks encoded as 000111b or 0011b are generated only when the running disparity at the beginning of the sub-block is positive;

thus, running disparity at the end of these sub-blocks shall also be positive. Likewise, sub-blocks containing 111000b or 1100b are generated only when the running disparity at the beginning of the sub-block is negative; thus, running disparity at the end of these sub-blocks shall also be negative.

Table 43 defines the valid data characters (Dxx.y characters).

Table 43 — Valid data characters (part 1 of 3)

Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)	Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D00.0	000 00000	100111 0100	011000 1011	D00.1	001 00000	100111 1001	011000 1001
D01.0	000 00001	011101 0100	100010 1011	D01.1	001 00001	011101 1001	100010 1001
D02.0	000 00010	101101 0100	010010 1011	D02.1	001 00010	101101 1001	010010 1001
D03.0	000 00011	110001 1011	110001 0100	D03.1	001 00011	110001 1001	110001 1001
D04.0	000 00100	110101 0100	001010 1011	D04.1	001 00100	110101 1001	001010 1001
D05.0	000 00101	101001 1011	101001 0100	D05.1	001 00101	101001 1001	101001 1001
D06.0	000 00110	011001 1011	011001 0100	D06.1	001 00110	011001 1001	011001 1001
D07.0	000 00111	111000 1011	000111 0100	D07.1	001 00111	111000 1001	000111 1001
D08.0	000 01000	111001 0100	000110 1011	D08.1	001 01000	111001 1001	000110 1001
D09.0	000 01001	100101 1011	100101 0100	D09.1	001 01001	100101 1001	100101 1001
D10.0	000 01010	010101 1011	010101 0100	D10.1	001 01010	010101 1001	010101 1001
D11.0	000 01011	110100 1011	110100 0100	D11.1	001 01011	110100 1001	110100 1001
D12.0	000 01100	001101 1011	001101 0100	D12.1	001 01100	001101 1001	001101 1001
D13.0	000 01101	101100 1011	101100 0100	D13.1	001 01101	101100 1001	101100 1001
D14.0	000 01110	011100 1011	011100 0100	D14.1	001 01110	011100 1001	011100 1001
D15.0	000 01111	010111 0100	101000 1011	D15.1	001 01111	010111 1001	101000 1001
D16.0	000 10000	011011 0100	100100 1011	D16.1	001 10000	011011 1001	100100 1001
D17.0	000 10001	100011 1011	100011 0100	D17.1	001 10001	100011 1001	100011 1001
D18.0	000 10010	010011 1011	010011 0100	D18.1	001 10010	010011 1001	010011 1001
D19.0	000 10011	110010 1011	110010 0100	D19.1	001 10011	110010 1001	110010 1001
D20.0	000 10100	001011 1011	001011 0100	D20.1	001 10100	001011 1001	001011 1001
D21.0	000 10101	101010 1011	101010 0100	D21.1	001 10101	101010 1001	101010 1001
D22.0	000 10110	011010 1011	011010 0100	D22.1	001 10110	011010 1001	011010 1001
D23.0	000 10111	111010 0100	000101 1011	D23.1	001 10111	111010 1001	000101 1001
D24.0	000 11000	110011 0100	001100 1011	D24.1	001 11000	110011 1001	001100 1001
D25.0	000 11001	100110 1011	100110 0100	D25.1	001 11001	100110 1001	100110 1001
D26.0	000 11010	010110 1011	010110 0100	D26.1	001 11010	010110 1001	010110 1001
D27.0	000 11011	110110 0100	001001 1011	D27.1	001 11011	110110 1001	001001 1001
D28.0	000 11100	001110 1011	001110 0100	D28.1	001 11100	001110 1001	001110 1001
D29.0	000 11101	101110 0100	010001 1011	D29.1	001 11101	101110 1001	010001 1001
D30.0	000 11110	011110 0100	100001 1011	D30.1	001 11110	011110 1001	100001 1001
D31.0	000 11111	101011 0100	010100 1011	D31.1	001 11111	101011 1001	010100 1001
D00.2	010 00000	100111 0101	011000 0101	D00.3	011 00000	100111 0011	011000 1100
D01.2	010 00001	011101 0101	100010 0101	D01.3	011 00001	011101 0011	100010 1100
D02.2	010 00010	101101 0101	010010 0101	D02.3	011 00010	101101 0011	010010 1100
D03.2	010 00011	110001 0101	110001 0101	D03.3	011 00011	110001 1100	110001 0011
D04.2	010 00100	110101 0101	001010 0101	D04.3	011 00100	110101 0011	001010 1100
D05.2	010 00101	101001 0101	101001 0101	D05.3	011 00101	101001 1100	101001 0011
D06.2	010 00110	011001 0101	011001 0101	D06.3	011 00110	011001 1100	011001 0011
D07.2	010 00111	111000 0101	000111 0101	D07.3	011 00111	111000 1100	000111 0011
D08.2	010 01000	111001 0101	000110 0101	D08.3	011 01000	111001 0011	000110 1100
D09.2	010 01001	100101 0101	100101 0101	D09.3	011 01001	100101 1100	100101 0011
D10.2	010 01010	010101 0101	010101 0101	D10.3	011 01010	010101 1100	010101 0011
D11.2	010 01011	110100 0101	110100 0101	D11.3	011 01011	110100 1100	110100 0011
D12.2	010 01100	001101 0101	001101 0101	D12.3	011 01100	001101 1100	001101 0011
D13.2	010 01101	101100 0101	101100 0101	D13.3	011 01101	101100 1100	101100 0011
D14.2	010 01110	011100 0101	011100 0101	D14.3	011 01110	011100 1100	011100 0011
D15.2	010 01111	010111 0101	101000 0101	D15.3	011 01111	010111 0011	101000 1100
D16.2	010 10000	011011 0101	100100 0101	D16.3	011 10000	011011 0011	100100 1100
D17.2	010 10001	100011 0101	100011 0101	D17.3	011 10001	100011 1100	100011 0011
D18.2	010 10010	010011 0101	010011 0101	D18.3	011 10010	010011 1100	010011 0011

Table 43 — Valid data characters (part 2 of 3)

Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)	Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D19.2	010 10011	110010 0101	110010 0101	D19.3	011 10011	110010 1100	110010 0011
D20.2	010 10100	001011 0101	001011 0101	D20.3	011 10100	001011 1100	001011 0011
D21.2	010 10101	101010 0101	101010 0101	D21.3	011 10101	101010 1100	101010 0011
D22.2	010 10110	011010 0101	011010 0101	D22.3	011 10110	011010 1100	011010 0011
D23.2	010 10111	111010 0101	000101 0101	D23.3	011 10111	111010 0011	000101 1100
D24.2	010 11000	110011 0101	001100 0101	D24.3	011 11000	110011 0011	001100 1100
D25.2	010 11001	100110 0101	100110 0101	D25.3	011 11001	100110 1100	100110 0011
D26.2	010 11010	010110 0101	010110 0101	D26.3	011 11010	010110 1100	010110 0011
D27.2	010 11011	110110 0101	001001 0101	D27.3	011 11011	110110 0011	001001 1100
D28.2	010 11100	001110 0101	001110 0101	D28.3	011 11100	001110 1100	001110 0011
D29.2	010 11101	101110 0101	010001 0101	D29.3	011 11101	101110 0011	010001 1100
D30.2	010 11110	011110 0101	100001 0101	D30.3	011 11110	011110 0011	100001 1100
D31.2	010 11111	101011 0101	010100 0101	D31.3	011 11111	101011 0011	010100 1100
D00.4	100 00000	100111 0010	011000 1101	D00.5	101 00000	100111 1010	011000 1010
D01.4	100 00001	011101 0010	100010 1101	D01.5	101 00001	011101 1010	100010 1010
D02.4	100 00010	101101 0010	010010 1101	D02.5	101 00010	101101 1010	010010 1010
D03.4	100 00011	110001 1101	110001 0010	D03.5	101 00011	110001 1010	110001 1010
D04.4	100 00100	110101 0010	001010 1101	D04.5	101 00100	110101 1010	001010 1010
D05.4	100 00101	101001 1101	101001 0010	D05.5	101 00101	101001 1010	101001 1010
D06.4	100 00110	011001 1101	011001 0010	D06.5	101 00110	011001 1010	011001 1010
D07.4	100 00111	111000 1101	000111 0010	D07.5	101 00111	111000 1010	000111 1010
D08.4	100 01000	111001 0010	000110 1101	D08.5	101 01000	111001 1010	000110 1010
D09.4	100 01001	100101 1101	100101 0010	D09.5	101 01001	100101 1010	100101 1010
D10.4	100 01010	010101 1101	010101 0010	D10.5	101 01010	010101 1010	010101 1010
D11.4	100 01011	110100 1101	110100 0010	D11.5	101 01011	110100 1010	110100 1010
D12.4	100 01100	001101 1101	001101 0010	D12.5	101 01100	001101 1010	001101 1010
D13.4	100 01101	101100 1101	101100 0010	D13.5	101 01101	101100 1010	101100 1010
D14.4	100 01110	011100 1101	011100 0010	D14.5	101 01110	011100 1010	011100 1010
D15.4	100 01111	010111 0010	101000 1101	D15.5	101 01111	010111 1010	101000 1010
D16.4	100 10000	011011 0010	100100 1101	D16.5	101 10000	011011 1010	100100 1010
D17.4	100 10001	100011 1101	100011 0010	D17.5	101 10001	100011 1010	100011 1010
D18.4	100 10010	010011 1101	010011 0010	D18.5	101 10010	010011 1010	010011 1010
D19.4	100 10011	110010 1101	110010 0010	D19.5	101 10011	110010 1010	110010 1010
D20.4	100 10100	001011 1101	001011 0010	D20.5	101 10100	001011 1010	001011 1010
D21.4	100 10101	101010 1101	101010 0010	D21.5	101 10101	101010 1010	101010 1010
D22.4	100 10110	011010 1101	011010 0010	D22.5	101 10110	011010 1010	011010 1010
D23.4	100 10111	111010 0010	000101 1101	D23.5	101 10111	111010 1010	000101 1010
D24.4	100 11000	110011 0010	001100 1101	D24.5	101 11000	110011 1010	001100 1010
D25.4	100 11001	100110 1101	100110 0010	D25.5	101 11001	100110 1010	100110 1010
D26.4	100 11010	010110 1101	010110 0010	D26.5	101 11010	010110 1010	010110 1010
D27.4	100 11011	110110 0010	001001 1101	D27.5	101 11011	110110 1010	001001 1010
D28.4	100 11100	001110 1101	001110 0010	D28.5	101 11100	001110 1010	001110 1010
D29.4	100 11101	101110 0010	010001 1101	D29.5	101 11101	101110 1010	010001 1010
D30.4	100 11110	011110 0010	100001 1101	D30.5	101 11110	011110 1010	100001 1010
D31.4	100 11111	101011 0010	010100 1101	D31.5	101 11111	101011 1010	010100 1010
D00.6	110 00000	100111 0110	011000 0110	D00.7	111 00000	100111 0001	011000 1110
D01.6	110 00001	011101 0110	100010 0110	D01.7	111 00001	011101 0001	100010 1110
D02.6	110 00010	101101 0110	010010 0110	D02.7	111 00010	101101 0001	010010 1110
D03.6	110 00011	110001 0110	110001 0110	D03.7	111 00011	110001 1110	110001 0001
D04.6	110 00100	110101 0110	001010 0110	D04.7	111 00100	110101 0001	001010 1110
D05.6	110 00101	101001 0110	101001 0110	D05.7	111 00101	101001 1110	101001 0001
D06.6	110 00110	011001 0110	011001 0110	D06.7	111 00110	011001 1110	011001 0001
D07.6	110 00111	111000 0110	000111 0110	D07.7	111 00111	111000 1110	000111 0001
D08.6	110 01000	111001 0110	000110 0110	D08.7	111 01000	111001 0001	000110 1110
D09.6	110 01001	100101 0110	100101 0110	D09.7	111 01001	100101 1110	100101 0001
D10.6	110 01010	010101 0110	010101 0110	D10.7	111 01010	010101 1110	010101 0001
D11.6	110 01011	110100 0110	110100 0110	D11.7	111 01011	110100 1110	110100 1000

Table 43 — Valid data characters (part 3 of 3)

Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)	Data byte name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj (binary)	Current RD + abcdei fghj (binary)
D12.6	110 01100	001101 0110	001101 0110	D12.7	111 01100	001101 1110	001101 0001
D13.6	110 01101	101100 0110	101100 0110	D13.7	111 01101	101100 1110	101100 1000
D14.6	110 01110	011100 0110	011100 0110	D14.7	111 01110	011100 1110	011100 1000
D15.6	110 01111	010111 0110	101000 0110	D15.7	111 01111	010111 0001	101000 1110
D16.6	110 10000	011011 0110	100100 0110	D16.7	111 10000	011011 0001	100100 1110
D17.6	110 10001	100011 0110	100011 0110	D17.7	111 10001	100011 0111	100011 0001
D18.6	110 10010	010011 0110	010011 0110	D18.7	111 10010	010011 0111	010011 0001
D19.6	110 10011	110010 0110	110010 0110	D19.7	111 10011	110010 1110	110010 0001
D20.6	110 10100	001011 0110	001011 0110	D20.7	111 10100	001011 0111	001011 0001
D21.6	110 10101	101010 0110	101010 0110	D21.7	111 10101	101010 1110	101010 0001
D22.6	110 10110	011010 0110	011010 0110	D22.7	111 10110	011010 1110	011010 0001
D23.6	110 10111	111010 0110	000101 0110	D23.7	111 10111	111010 0001	000101 1110
D24.6	110 11000	110011 0110	001100 0110	D24.7	111 11000	110011 0001	001100 1110
D25.6	110 11001	100110 0110	100110 0110	D25.7	111 11001	100110 1110	100110 0001
D26.6	110 11010	010110 0110	010110 0110	D26.7	111 11010	010110 1110	010110 0001
D27.6	110 11011	110110 0110	001001 0110	D27.7	111 11011	110110 0001	001001 1110
D28.6	110 11100	001110 0110	001110 0110	D28.7	111 11100	001110 1110	001110 0001
D29.6	110 11101	101110 0110	010001 0110	D29.7	111 11101	101110 0001	010001 1110
D30.6	110 11110	011110 0110	100001 0110	D30.7	111 11110	011110 0001	100001 1110
D31.6	110 11111	101011 0110	010100 0110	D31.7	111 11111	101011 0001	010100 1110

Table 44 defines the valid special characters (Kxx.y characters). Comma patterns, five bits of the same polarity, are underlined.

Table 44 — Valid special characters

Special code name	Bits HGF EDCBA (binary)	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	001111 0100	110000 1011
K28.1	001 11100	001111 1001	110000 0110
K28.2	010 11100	001111 0101	110000 1010
K28.3	011 11100	001111 0011	110000 1100
K28.4	100 11100	001111 0010	110000 1101
K28.5	101 11100	001111 1010	110000 0101
K28.6	110 11100	001111 0110	110000 1001
K28.7	111 11100	001111 1000	110000 0111
K23.7	111 10111	111010 1000	000101 0111
K27.7	111 11011	110110 1000	001001 0111
K29.7	111 11101	101110 1000	010001 0111
K30.7	111 11110	011110 1000	100001 0111

Only K28.5 and K28.3 are used in this standard (see 7.1).

NOTE 10 K28.1, K28.5, and K28.7 are the only valid characters which contain comma patterns. The K28.7 special character is not used because it introduces a false comma pattern when followed by any of the following special or data characters: K28.x, D3.x, D11.x, D12.x, D19.x, D20.x, or D28.x, where x is a value in the range 0 to 7, inclusive.

6.3.3.2 Generating transmission characters

The appropriate entry in the table shall be found for the valid data byte or special code for which a transmission character is to be generated (encoded). The current value of the transmitter's running disparity shall be used to select the transmission character from its corresponding column. For each transmission character transmitted, a new value of the running disparity shall be calculated. This new value shall be used as the transmitter's current running disparity for the next valid data byte or special code to be encoded and transmitted.

6.3.3.3 Validity of received transmission characters

The columns in table 43 and table 44 corresponding to the current value of the receiver's running disparity shall be searched for each received transmission character. If the received transmission character is found in the proper column, then the transmission character shall be considered valid and the associated data byte or special code determined (decoded). If the received transmission character is not found in the proper column, then the transmission character shall be considered invalid and the dword containing the character shall be considered an invalid dword. Independent of the transmission character's validity, the received transmission character shall be used to calculate a new value of running disparity.

This new value shall be used as the receiver's current running disparity for the next received transmission character.

Detection of a code violation does not necessarily indicate that the transmission character in which the code violation was detected is in error. Code violations may result from a prior error that altered the running disparity of the bit stream but did not result in a detectable error at the transmission character in which the error occurred. The example shown in table 45 exhibits this behavior.

Table 45 — Delayed code violation example

	RD	Character	RD	Character	RD	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 1011	+	010101 0101	+	111010 1010	+
Decoded character stream	-	D21.0	+	D10.2	+	Code violation	+

6.4 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 44.

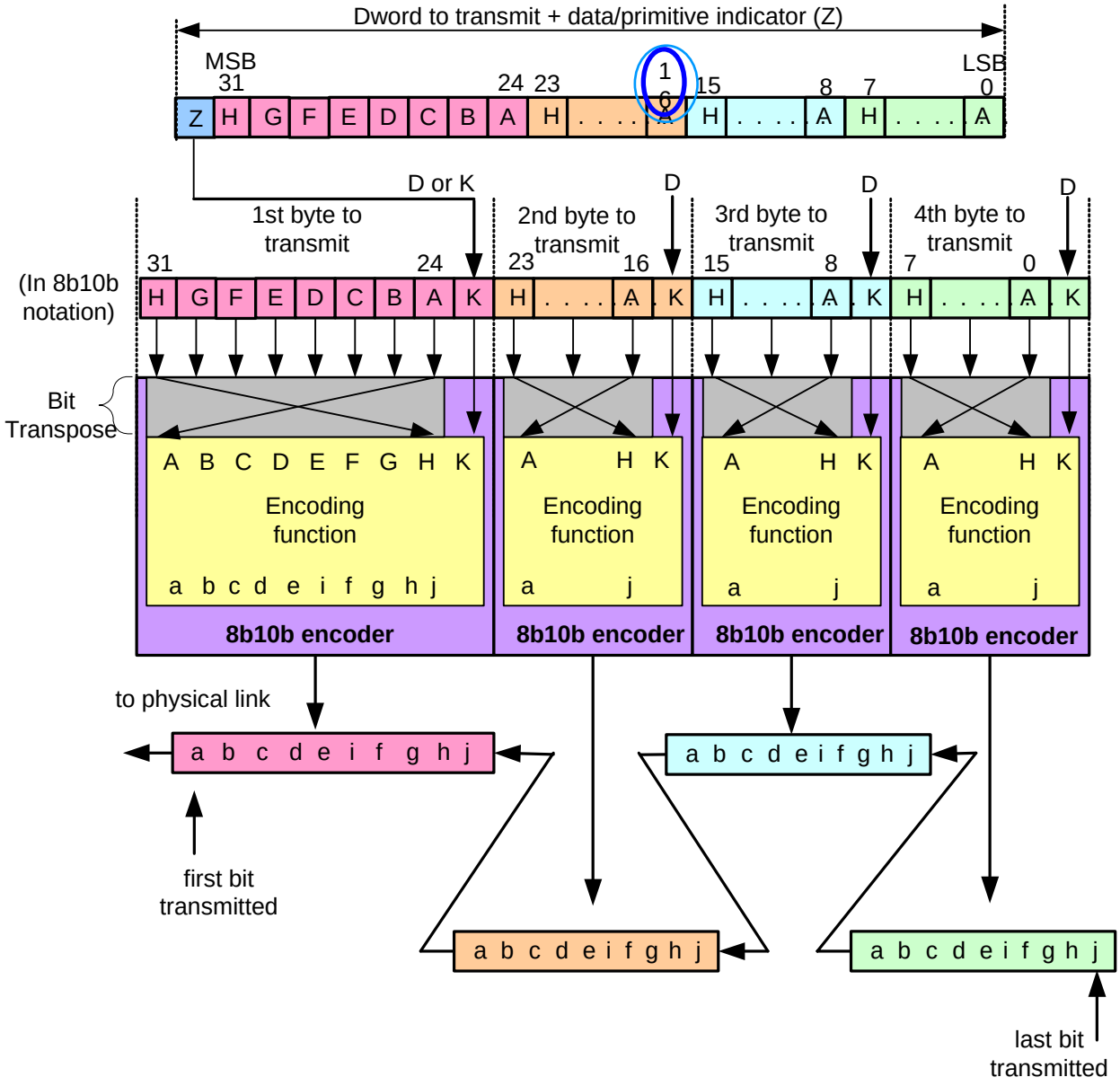


Figure 44 — SAS bit transmission logic

Figure 45 shows the SAS bit reception order.

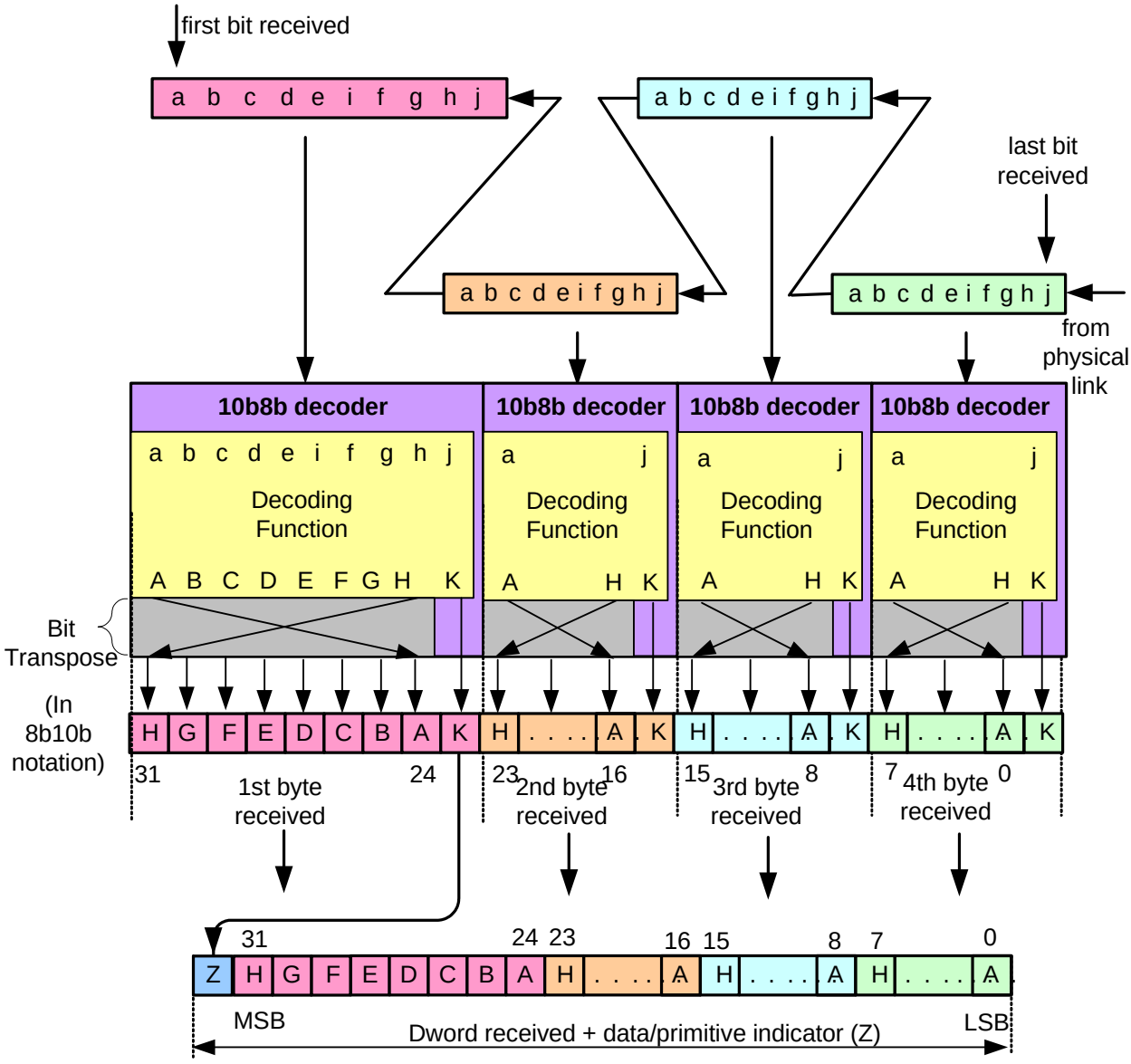


Figure 45 — SAS bit reception logic

6.5 Out of band (OOB) signals

Out of band (OOB) signals are low-speed signal patterns detected by the phy that do not appear in normal data streams. They consist of defined amounts of idle time followed by defined amounts of burst time. During the burst time, ALIGN (0) primitives are transmitted repeatedly. During the idle time, the transmitter output shall meet the requirements described in 5.7.4. 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. SAS devices identify themselves with an additional SAS-specific OOB signal called COMSAS.

Table 46 defines the timing specifications for OOB signals.

Table 46 — OOB signal timing specifications

Parameter	Minimum	Nominal	Maximum	Comments
Unit interval during OOB (UI(OOB)) [a]	666,600 ps	666,667 ps	666,734 ps	The time basis for burst times and idle times used to create OOB signals. Based on 1,5 Gbps clock tolerance.
COMSAS detect timeout	13,65 µs			The minimum time a receiver shall allow to detect COMSAS after transmitting COMSAS. Derived from: UI(OOB) x 512 x 40

[a] ~~UI(OOB) is different than that defined in SATA; SAS has tighter clock tolerance.~~ This is a fixed value equal to the UI for G1, regardless of the actual transfer rate being used to create the burst time.

Table 47 describes the OOB signal transmitter requirements for the burst time, idle time, and negation times that comprise each OOB signal.

Table 47 — OOB signal transmitter requirements

Signal	Burst time	Idle time	Negation time
COMWAKE	160 UI(OOB)	160 UI(OOB)	280 UI(OOB)
COMINIT/RESET	160 UI(OOB)	480 UI(OOB)	800 UI(OOB)
COMSAS	160 UI(OOB)	1 440 UI(OOB)	2 400 UI(OOB)

To transmit an OOB signal, a transmitter shall repeat these steps six times:

1) transmit idle for an idle time; and
2) transmit an ALIGN burst for a burst time.

It shall then transmit idle for an OOB signal negation time.

The ALIGNs used in OOB signals are not required to be at generation 1 (G1) physical link rates (i.e., 1,5 Gbps), as this rate may not be supported in future generations of SAS devices. The ALIGNs are only required to generate an envelope for the detection circuitry, as required for any signaling that may be A.C. coupled. If G2 ALIGNs are used, the number of ALIGNs doubles compared with G1 ALIGNs.

A SAS transmitter should transmit ALIGNs at the G1 physical link rate to create the burst portion of the OOB signal, but may transmit ALIGNs at its slowest supported physical link rate if it does not support the G1 physical link rate and shall not transmit them at a physical link rate faster than its slowest supported physical link rate.

Figure 46 describes OOB signal transmission by the SP transmitter. The COMWAKE Transmitted, COMINIT Transmitted, and COMSAS Transmitted parameters are sent to the SP state machine.

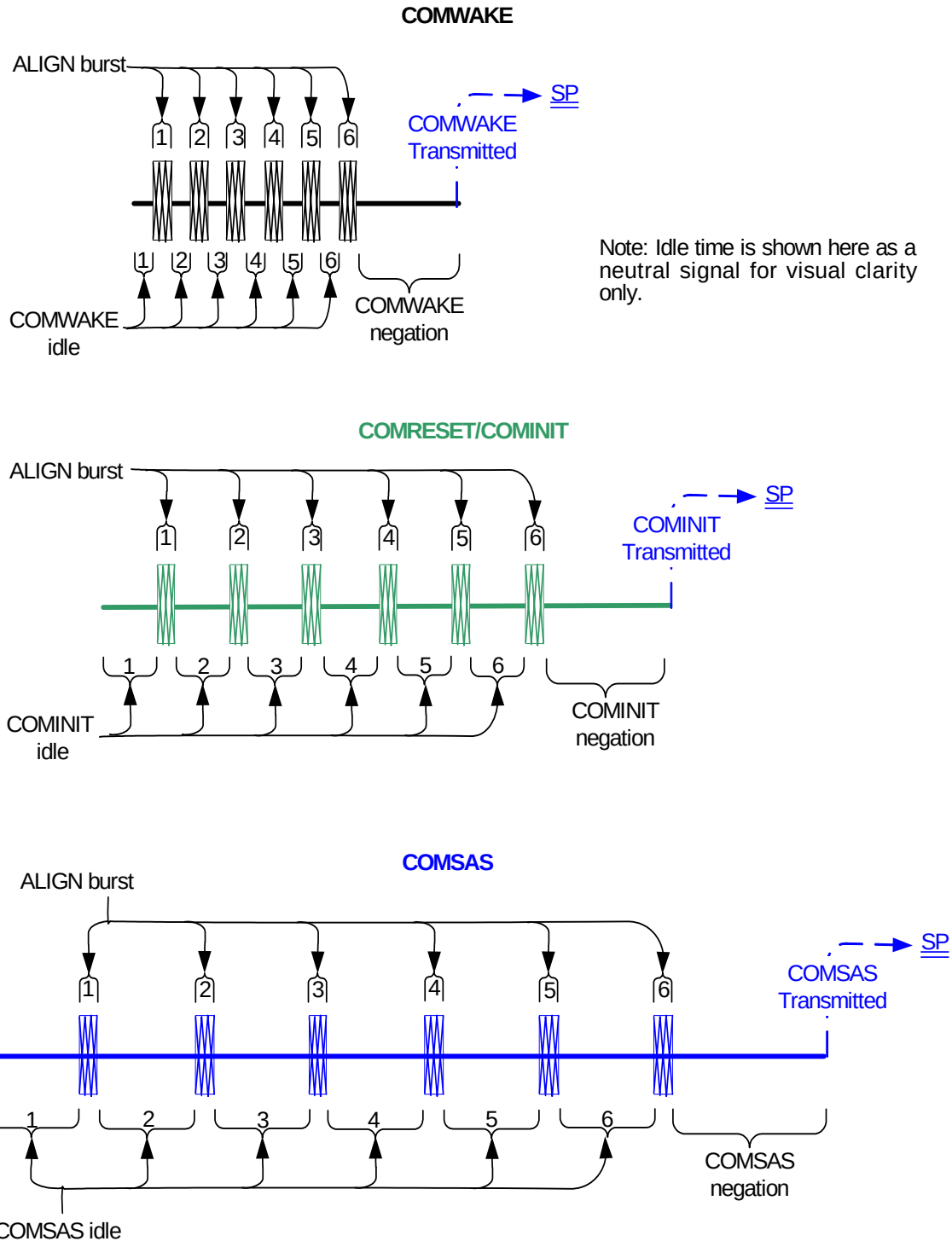


Figure 46 — OOB signal transmission

Table 48 describes the OOB signal receiver requirements for the idle times and negation times. The burst time is not used to distinguish between signals.

Table 48 — OOB signal receiver requirements

Signal	Idle time			Negation time
	may detect	shall detect	shall not detect	shall detect
COMWAKE	55 ns to 175 ns	101,3 ns to 112 ns	< 55 ns or > 175 ns	175 ns
COMINIT/ COMRESET	175 ns to 525 ns	304 ns to 336 ns	< 175 ns or > 525 ns	525 ns
COMSAS	525 ns to 1 575 ns	911,7 ns to 1008 ns	< 525 ns or > 1 575 ns	1 575 ns

A receiver shall detect an OOB signal after receiving four consecutive idle time/burst time pairs (see figure 47). It is not an error to receive more than six idle time/burst time pairs. A receiver shall not detect the same OOB signal again until it has detected a different OOB signal (e.g., if the idle time changes) or has detected lack of transitions for a time greater then the proceeding idle time (i.e., a COMINIT negation time for a COMINIT idle time or a COMSAS negation time for a COMSAS idle time).

A SAS receiver shall detect OOB signals comprised of ALIGNs transmitted at any rate up to its highest supported physical link rate. This includes physical link rates below its lowest supported physical link rate (e.g., a SAS receiver supporting only 3,0 Gbps needs to detect 1,5 Gbps based ALIGNs to interoperate with a SAS transmitter supporting both 1,5 Gbps and 3,0 Gbps).

Figure 47 describes SAS OOB signal detection by the SP receiver. The COMWAKE Detected, COMWAKE Completed, COMINIT Detected, COMINIT Completed, COMSAS Detected, and COMSAS Completed

parameters are sent to the SAS phy (SP) state machine to indicate that an OOB signal has been partially or fully detected.

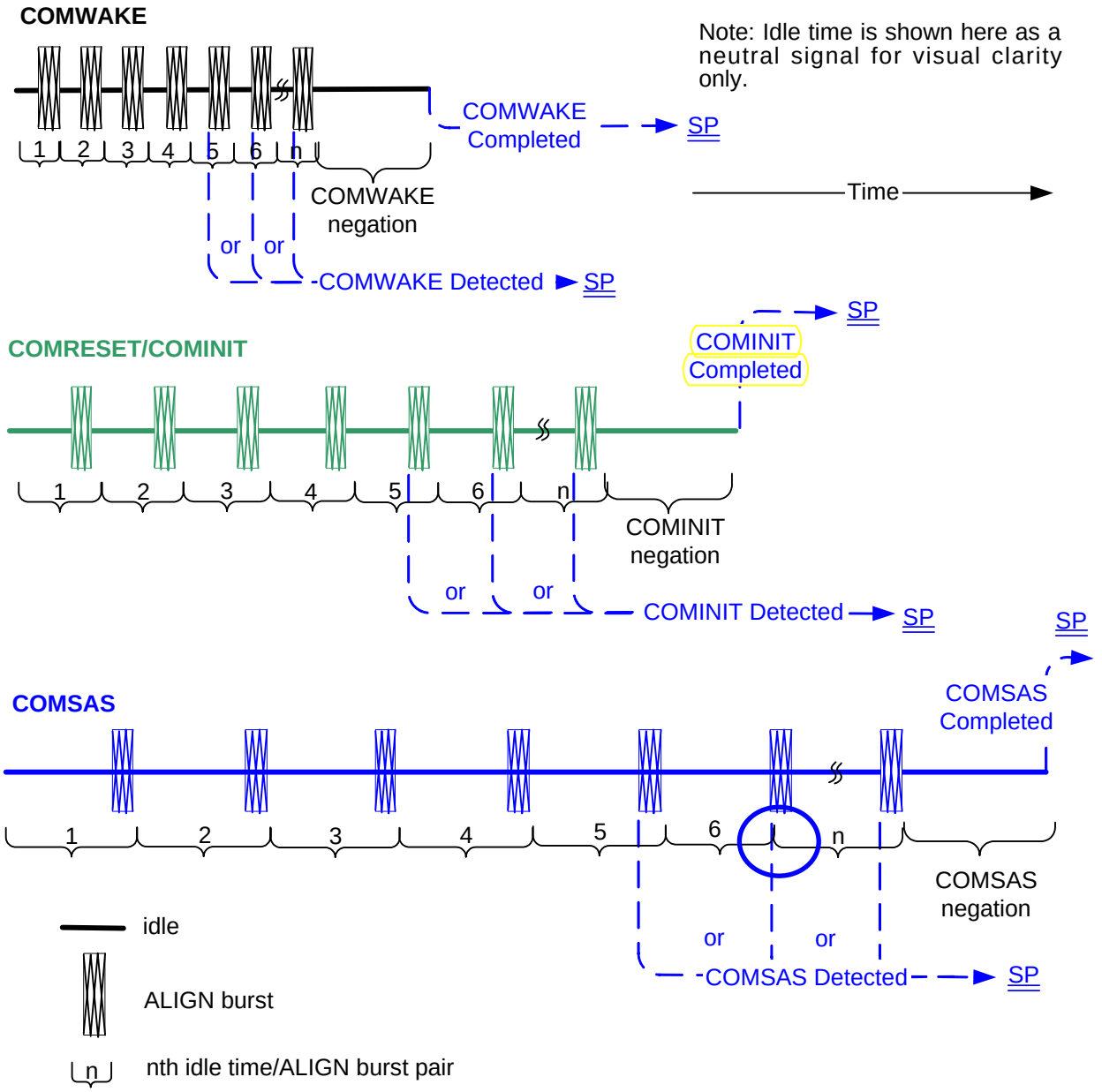


Figure 47 — OOB signal detection

Expander devices shall not forward OOB signals. An expander device shall run the link reset sequence independently on each physical link ~~(i.e., from initiator phy to expander phy, expander phy to expander phy, or expander phy to target phy).~~

6.6 Phy reset sequences

6.6.1 Overview

The phy reset sequence consists of an OOB sequence and a speed negotiation sequence.

The SAS 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.

After a HARD_RESET, a device should start the phy reset sequence within 250 ms.

6.6.2 SATA phy reset sequence (informative)

6.6.2.1 SATA OOB sequence (informative)

Figure 48 shows the SATA OOB sequence between a SATA initiator device (i.e., SATA host) and target device (i.e, SATA device). The SATA OOB sequence is defined by SATA; see SATA for detailed requirements.

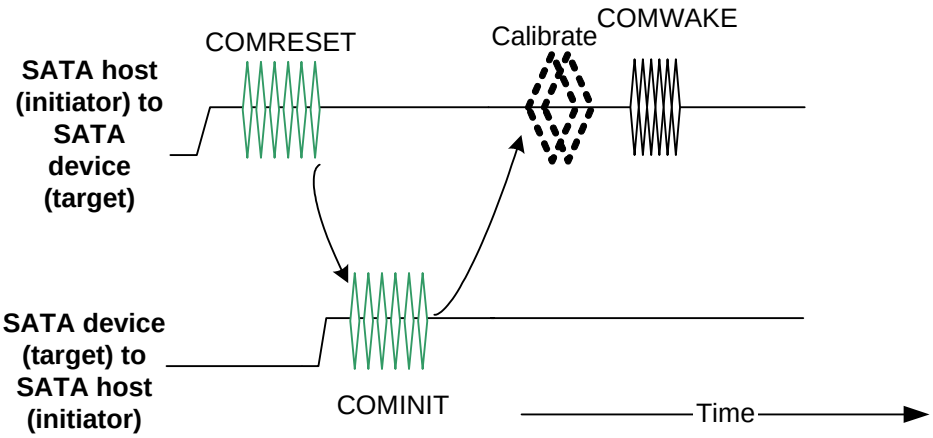


Figure 48 — SATA OOB sequence (informative)

6.6.2.2 SATA speed negotiation sequence (informative)

Figure 49 shows the speed negotiation sequence between a SATA initiator device and target device. The SATA speed negotiation sequence is defined by SATA; see SATA for detailed requirements.

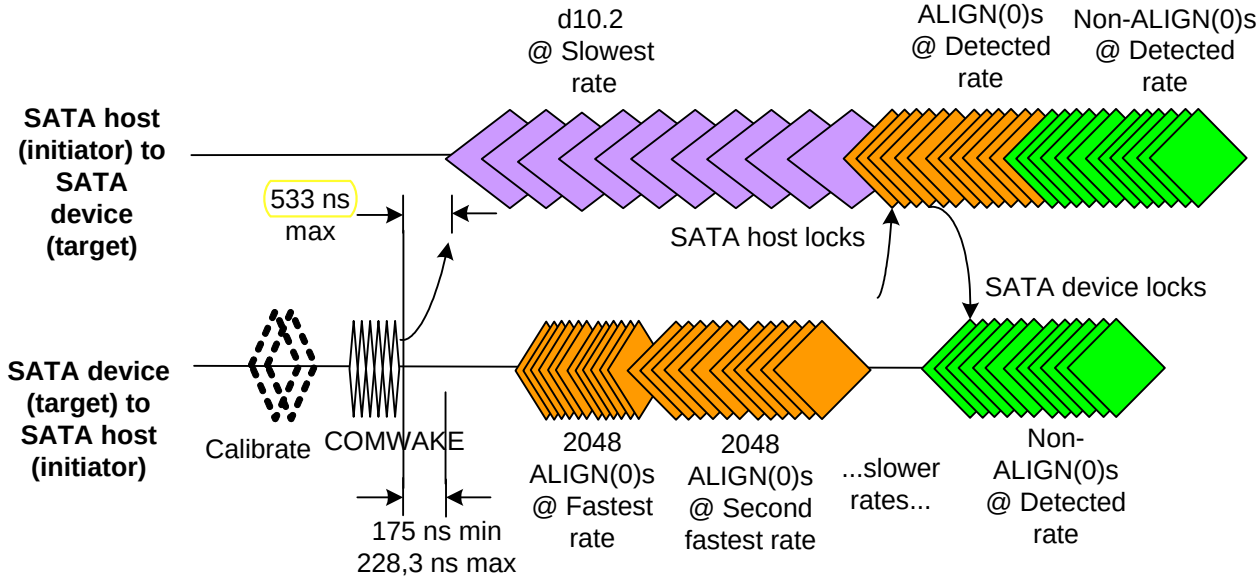


Figure 49 — SATA speed negotiation sequence (informative)

6.6.3 SAS to SATA phy reset sequence

SAS initiator devices and expander devices may be attached to SATA target devices.

To initiate a phy reset sequence a SAS phy shall:

1) transmit a COMINIT; and
2) in response to a COMINT, transmit a COMSAS.

The COMSAS identifies the phy as a SAS phy instead of a SATA phy for SAS to SAS attachments.

If a SATA phy is attached to the link it either:

a) misinterprets the COMSAS to be a COMRESET and responds with a COMINIT; or
b) ignores the COMSAS and provides no response within a COMSAS detect timeout.

Either response indicates to the SAS phy that a SATA phy is attached. As a result the SAS phy shall initiate a SATA reset sequence (i.e., transmit a COMRESET).

Figure 50 shows a reset sequence between a SAS phy and a SATA phy (i.e., between an expander device and a SATA target device, or between a SAS initiator device and a SATA target device). 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 a ~~legacy~~ SATA phy misinterprets the COMSAS to be a COMRESET and responds with a COMINIT. The SAS phy state machine treats these two cases the same, and determines that a SATA phy is attached after a COMSAS detect timeout. The normal SATA reset sequence shall be entered starting with COMWAKE.

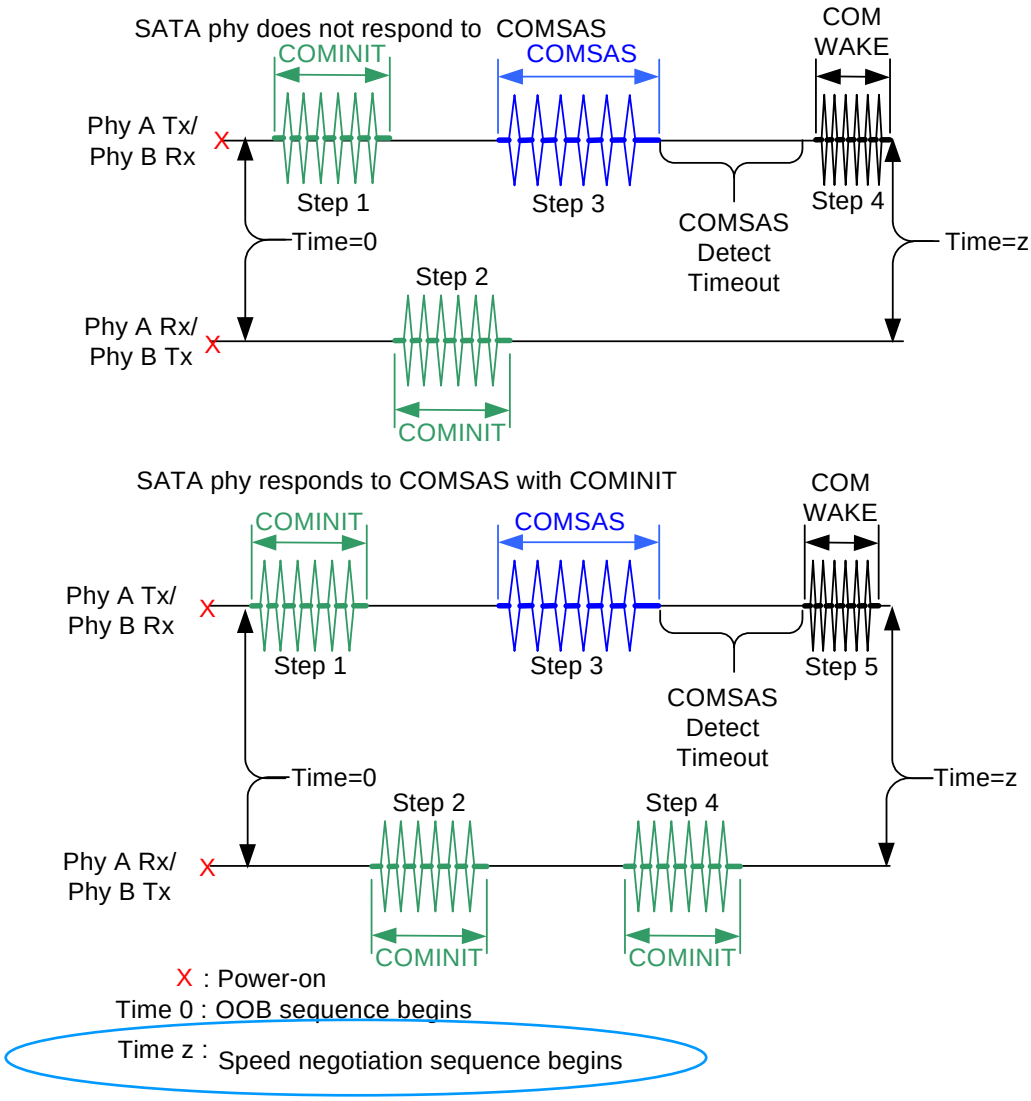


Figure 50 — SAS to SATA OOB sequence

6.6.4 SAS to SAS phy reset sequence

6.6.4.1 SAS OOB sequence

To initiate a SAS OOB sequence a SAS phy shall transmit a COMINIT. If there is no COMINIT or a COMSAS received within a hot-plug timeout then the SAS phy shall transmit another COMINIT.

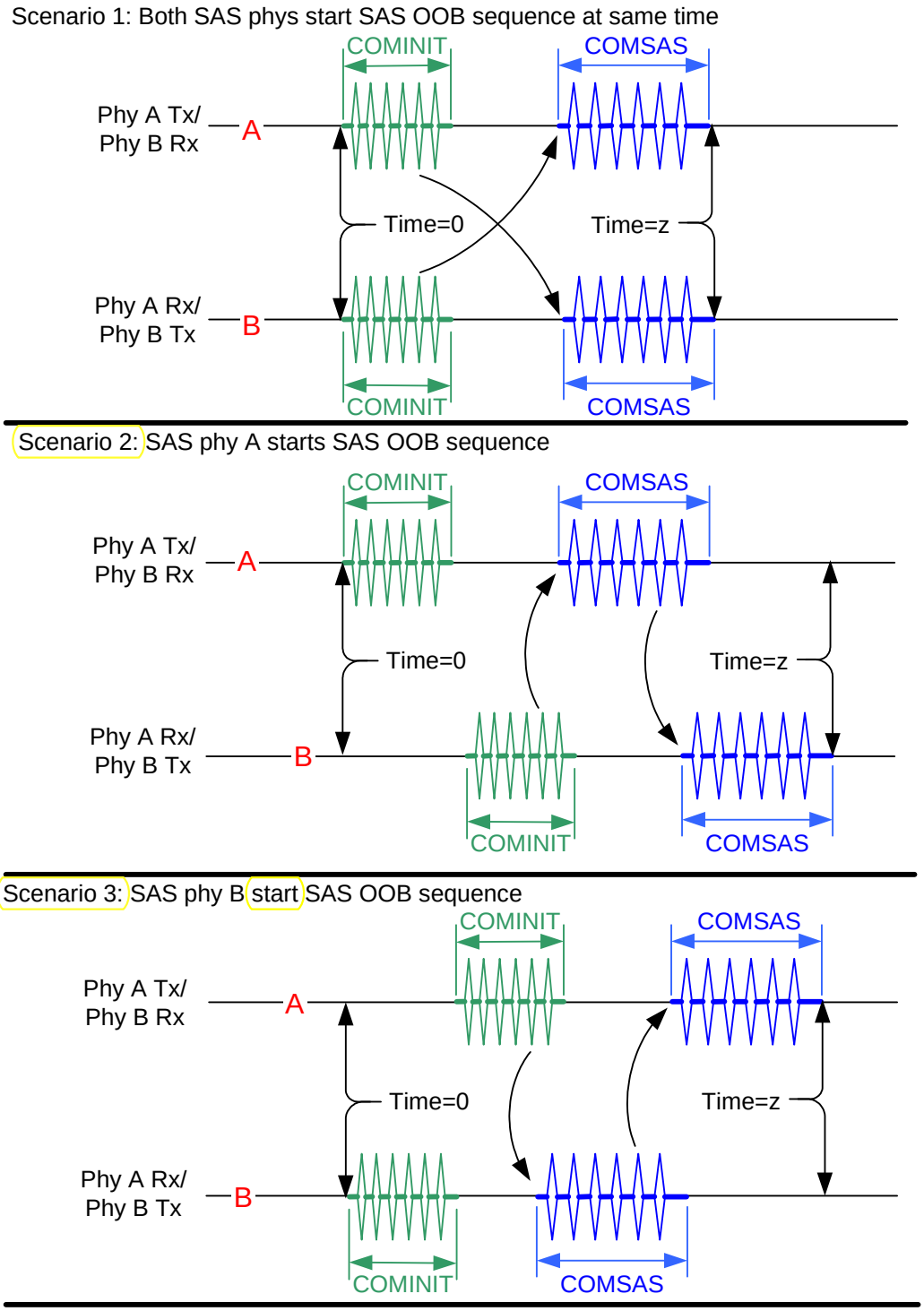
On receipt of a COMINIT a SAS phy shall either:

a) transmit a COMINIT, if the receiving SAS phys has not yet transmitted a COMSAS, followed by a COMSAS; or
b) transmit a COMSAS, if the receiving SAS phys has transmitted a COMINIT.

After completing the transmission of a COMSAS and the successful receipt a COMSAS the SAS OOB sequence is complete and the SAS speed negotiation sequence begins.

A SAS device shall distinguish between COMINIT and COMSAS and continue with a SAS speed negotiation sequence after completing the SAS OOB sequence.

Figure 51 shows several different SAS OOB sequences between a SAS phy A and SAS phy B, with SAS phy A starting the SAS OOB sequence before, after, or at the same time as SAS phy.



A : SAS phy A power on

B : SAS phy B power on

Time 0: SAS phy reset sequence begins

Time z: SAS speed negotiation sequence begins

Figure 51 — SAS to SAS OOB sequence

6.6.4.2 SAS speed negotiation sequence

The SAS speed negotiation sequence is a peer-to-peer negotiation technique that does not assume initiator device and target device roles ~~like the SATA speed negotiation sequence.~~ The sequence consists of a set of speed negotiation windows for each physical link rate, starting with 1,5 Gbps, then 3,0 Gbps, then the next rate. The length of the speed negotiation sequence is determined by the number of physical link rates supported by the phys.

Figure 52 defines the speed negotiation window, including:

a) speed negotiation window time;
b) rate change delay (RCD);
c) speed negotiation transmit time (SNTT); and
d) speed negotiation lock time (SNLT), ~~a subset of the SNTT used by the receiver.~~

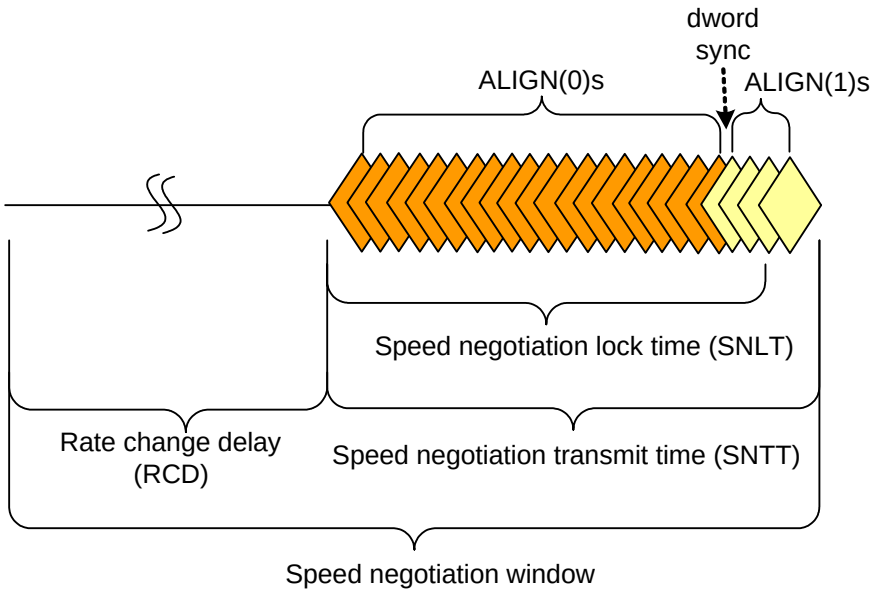


Figure 52 — SAS speed negotiation window

Table 46 defines the timing specifications for the SAS speed negotiation sequence.

Table 49 — SAS speed negotiation sequence timing specifications

Parameter	Minimum	Nominal	Maximum	Comments
Hot-plug timeout	10 ms	100 ms	500 ms	How often a device should retransmit COMINIT to detect if a device has been attached.
Rate change delay (RCD)		750 000 UI(OOB)		The time the transmitter shall transmit idle between rates during speed negotiation. Used by the transmitter and receiver to calculate the speed negotiation window time.
Speed negotiation transmit time (SNTT) for transmitter		163 840 UI(OOB)		The time during which ALIGN (0) or ALIGN (1) is transmitted at each physical link rate during the speed negotiation sequence. Derived from: UI(OOB) x 4 096 x 40.
Speed negotiation transmit time (SNTT) for receiver	109,22 µs	109,23 µs	109,24 µs	The time during which ALIGN (0) or ALIGN (1) is received at each physical link rate during negotiation. Derived from: UI(OOB) x 4 096 x 40.
Speed negotiation lock time (SNLT) for transmitter		153 600 UI(OOB)		The maximum time during the speed negotiation window for a transmitter to reply with ALIGN (1). Derived from: UI(OOB) x 3 840 x 40
Speed negotiation lock time (SNLT) for receiver	102,39 µs	102,40 µs	102,41 µs	The maximum time during the speed negotiation window for a receiver to detect ALIGN (0) or ALIGN (1) and reply with ALIGN (1). Derived from: UI(OOB) x 3 840 x 40.
Speed negotiation window time	609,166 µs	609,227µs	609,288 µs	The duration of a speed negotiation window. Derived from: RCD + SNTT.
ALIGN detect timeout			880 µs	The maximum time during SATA speed negotiation that an initiator phy receiver shall allow for an ALIGN to be received after detecting the COMWAKE negation time.

[a] ~~This is not the same as UI(OOB) defined in SATA; SAS has tighter clock tolerance.~~

The speed negotiation window shall consist of the following transmission sequence for each speed negotiation window:

1) a transmission of idle for an RCD; and
2) if the phy supports the physical link rate, a transmission of ALIGNs at that physical link rate for the remainder of the entire speed negotiation window. If the phy does not supports the physical link rate, transmission of idle for the remainder of the entire speed negotiation window.

If the receiving phy supports the physical link rate, it shall attempt to synchronize on an incoming series of dwords at that rate for the SNLT. 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 speed negotiation window). If the device does not achieve dword synchronization within the SNLT, it shall continue transmitting ALIGN(0)s for the remainder of the SNTT (i.e., the remainder of the speed negotiation window).

At the end of the SNTT, if a phy is both transmitting and receiving ALIGN (1) primitives, it shall consider that physical link rate valid. The phy shall then proceed to the next speed negotiation window. A phy shall participate in all speed negotiation windows:

a) up to its highest supported physical link rate plus one; or
b) until it runs a speed negotiation window that does not detect a valid physical link rate after having detected a valid physical link rate in a previous speed negotiation window.

If the phy has detected a valid physical link rate in the previous speed negotiation window, it shall enter the final speed negotiation window using the highest previously successful link rate.

Figure 53 shows speed negotiation between a SAS phy A that supports G1 through G3 link rates and a SAS phy B that only supports the G2 link rate. Both phys run the G1 speed negotiation window (supported by phy A but not by phy B, so invalid), the G2 speed negotiation window (valid), and the G3 speed negotiation window (supported by phy A but not by phy B, so invalid), that phy then selects G2 for the final speed negotiation window to establish the negotiated physical link rate.

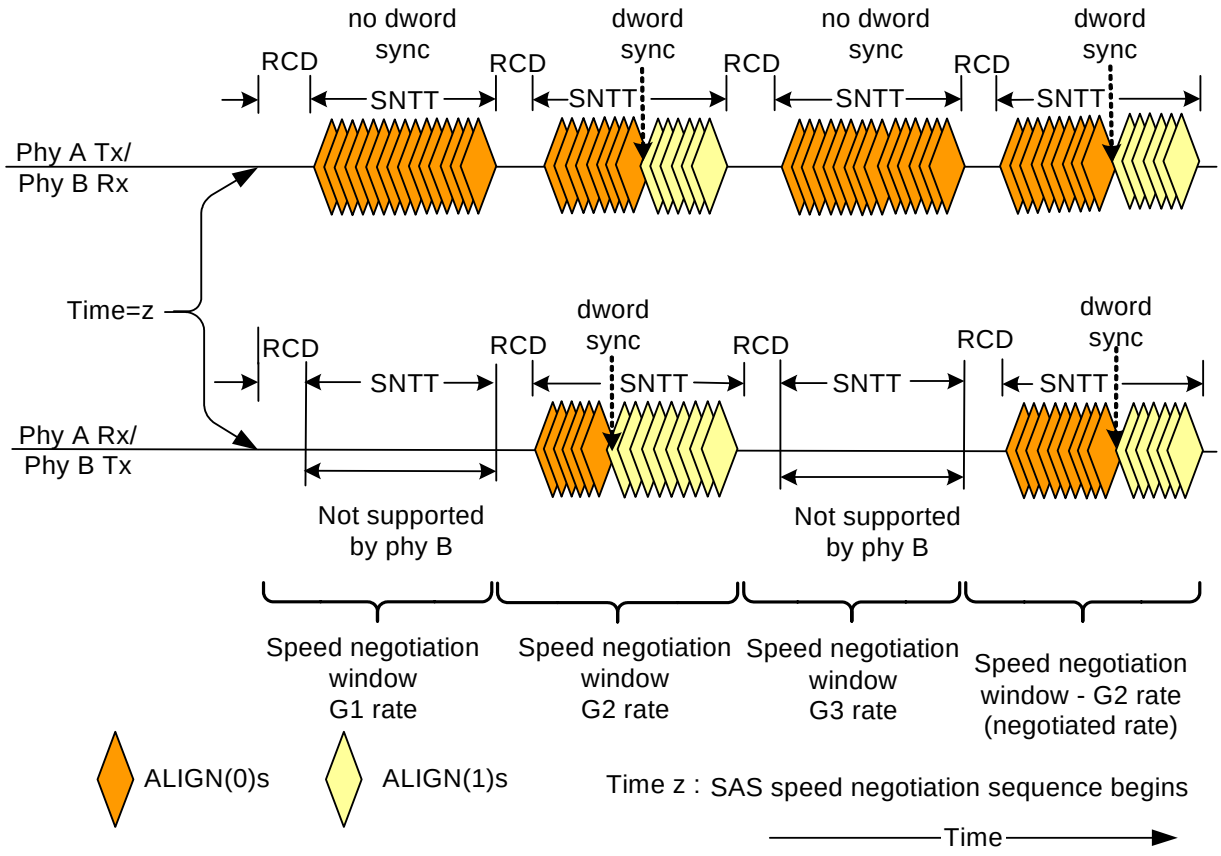


Figure 53 — SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G2 only)

If the SAS phy's PLLs does not obtain dword synchronization during the final speed negotiation window the SAS phy speed negotiation fails and the phy reset sequence shall be retried. This may be counted and reported in the PHY RESET PROBLEM field in the SMP REPORT PHY ERROR LOG page (see 10.3.1.5) and the REPORT PHY ERROR LOG log page (see 10.1.7.1).

Any time a SAS phy fails speed negotiation, it shall wait the hot-plug timeout before attempting a retry.

Figure 54 shows the same speed negotiation sequence as in figure 53 when SAS phy B does not obtain dword synchronization during the final speed negotiation window. If this occurs, the handshake is not complete and the OOB sequence shall be retried starting with COMINIT, forcing the phy to retry the whole reset sequence.

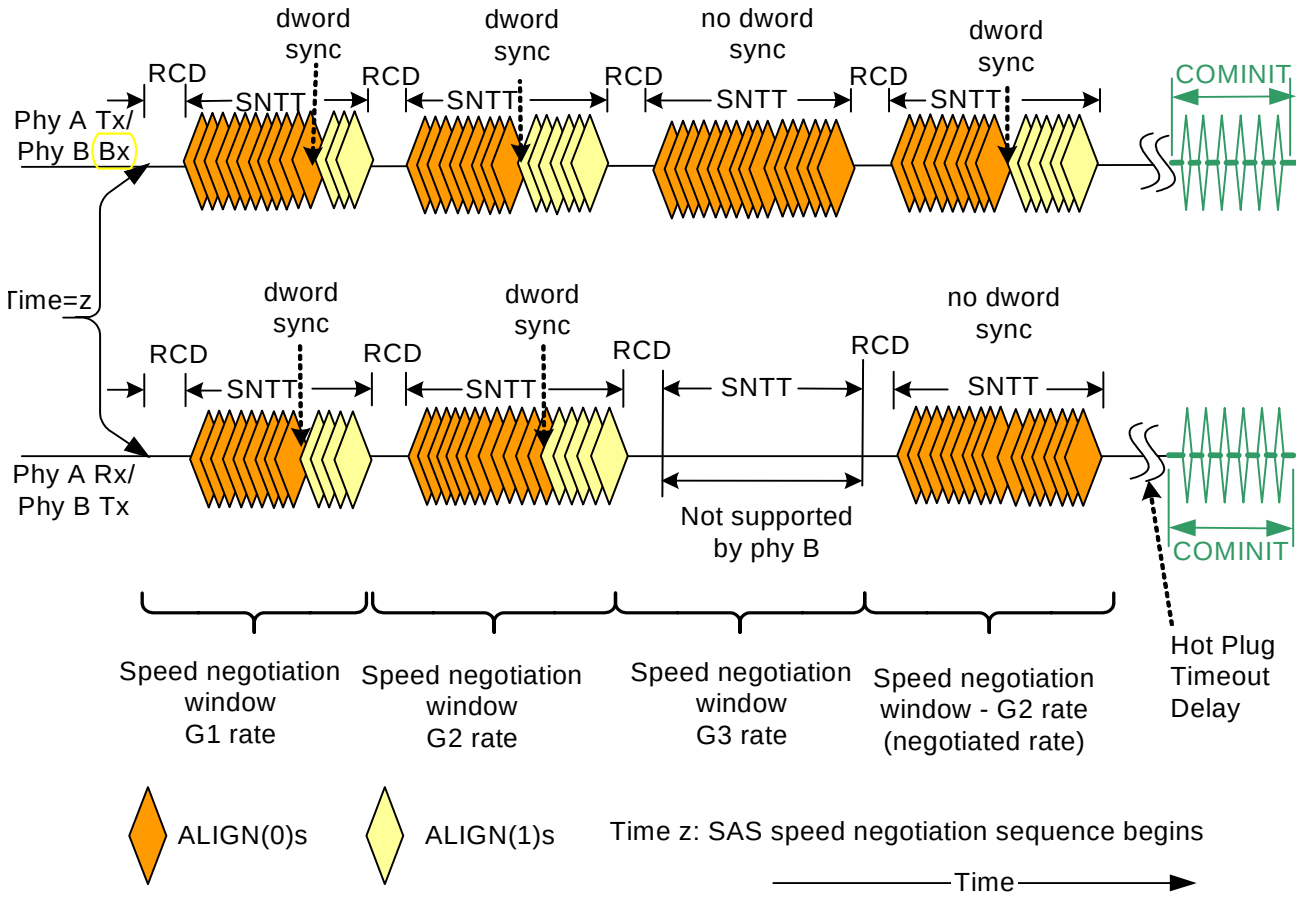


Figure 54 — SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G1, G2) with failure

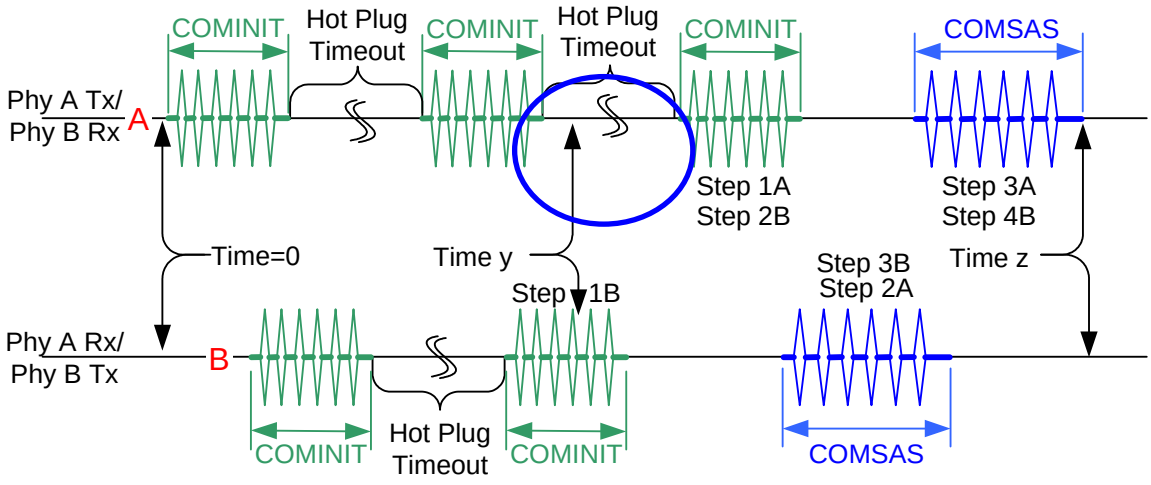
For more examples of speed negotiations between devices that support various speeds see Annex B.

6.6.5 Phy reset sequence after device is 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 into a receptacle. Non-cabled environments may similarly not have a way to detect physical insertion of a device. As a result, SAS phys should periodically transmit a COMINIT sequence if they have not detected a COMINIT sequence within a hot-plug timeout.

Figure 55 shows how two SAS phys complete the phy reset sequence if the phys are not attached at power on. In this example, SAS phy B is attached to SAS phy A some time before SAS phy B's second hot-plug timeout occurs. SAS phy B's OOB detection circuitry detects a COMINIT after the attachment, and therefore SAS phy B transmits the COMSAS sequence, since it has both transmitted and received a COMINIT

sequence. Upon receiving COMSAS, SAS phy A transmits its own COMSAS sequence, bypassing the normal requirement that COMINIT be both transmitted and received. The SAS speed negotiation sequence follows.



A : SAS A power on

Time y : SAS phy A attached to SAS phy B

Time z : SAS phy A and SAS phy B start the SAS speed negotiation sequence

B : SAS B power on

Step xA : The SAS OOB sequence at SAS phy A

Step xB : The SAS OOB sequence at SAS phy B

Figure 55 — Hot-plug and the phy reset sequence

6.7 SAS phy (SP) transmitter and receiver

The SP transmitter transmits OOB signals and dwords on the physical link based on parameters from the SP state machine.

Parameters received by the SP transmitter include:

a) Transmit COMINIT;
b) Transmit COMSAS;
c) Transmit COMWAKE;
d) Transmit D10.2;
e) Set Rate (Physical Link Rate);
f) Transmit ALIGN0; and
g) Transmit ALIGN1.

Parameters sent by the SP transmitter include:

a) COMINIT Transmitted;
b) COMSAS Transmitted; and
c) COMWAKE Transmitted.

The SP receiver receives OOB signals and dwords from the physical link and sends parameters to the SP state machine indicating what it has received.

Parameters sent by the SP receiver include:

a) COMINIT Detected;
b) COMSAS Detected;
c) COMWAKE Detected;
d) COMINIT Completed;
e) COMSAS Completed;
f) COMWAKE Completed;
g) ALIGN0 Detected; and
h) ALIGN1 Detected.

6.8 SAS phy (SP) state machine

6.8.1 Overview

The SAS phy (SP) state machine controls the phy reset sequence. The SP state machine consists of three sets of states:

a) OOB sequence (OOB) states;
b) SAS speed negotiation (SAS) states; and
c) SATA host emulation (SATA) states.

The SP states are:

a) SP1:OOB_COMINIT (see 6.8.2.1)(initial state);
b) SP2:OOB_AwaitCOMX (see 6.8.2.2);
c) SP3:OOB_AwaitCOMINIT_Sent (see 6.8.2.3);
d) SP4:OOB_COMSAS (see 6.8.2.4);
e) SP5:OOB_AwaitCOMSAS_Sent (see 6.8.2.5);
f) SP6:OOB_AwaitNoCOMSAS (see 6.8.2.6);
g) SP7:OOB_AwaitCOMSAS (see 6.8.2.7);
h) SP8:SAS_Start (see 6.8.3.1);
i) SP9:SAS_RateNotSupported (see 6.8.3.2);
j) SP10:SAS_AwaitALIGN (see 6.8.4.3);
k) SP11:SAS_AwaitALIGN1 (see 6.8.3.4);
l) SP12:SAS_AwaitSNW (see 6.8.3.5);
m) SP13:SAS_Pass (see 6.8.3.6);
n) SP14 SAS_Fail (see 6.8.3.7);
o) SP15:SAS_PHY_Ready (see 6.8.3.8);
p) SP16:SATA_COMWAKE (see 6.8.4.1);
q) SP17:SATA_AwaitCOMWAKE (see 6.8.4.2);
r) SP18:SATA_AwaitNoCOMWAKE (see 6.8.4.3);
s) SP19:SATA_AwaitALIGN (see 6.8.4.4);
t) SP20:SATA_AdjustSpeed (see 6.8.4.5);
u) SP21:SATA_Transmit_ALIGN (see 6.8.4.6);
v) SP22:SATA_PHY_Ready (see 6.8.4.7);
w) SP23:SATA_PM_Partial (see 6.8.4.8); and
x) SP24:SATA_PM_Slumber (see 6.8.4.9).

The SP state machine shall start in the SP0:SAS_PowerOn state after:

a) a power on;
b) a hard reset; or
c) receiving a Management Reset request ~~from the management layer~~ (e.g., from the SMP PHY CONTROL function in an expander device).

The SP state machine sends the following parameters to the SP_DWS state machine:

a) PhyNotReady;
b) PhyReady (SAS); and
c) PhyReady (SATA).

The SP state machine receives the following parameters from the SP_DWS state machine:

a) DWS Reset.

The SAS phy layer shall maintain the following timers:

a) Hot-plug timeout timer;
b) RCD timer;
c) SNLT timer;
d) SNTT timer; and
e) COMSAS detect timeout timer.

6.8.2 OOB sequence states

Figure 56 shows the OOB sequence states. These states are indicated by state names with a prefix of OOB.

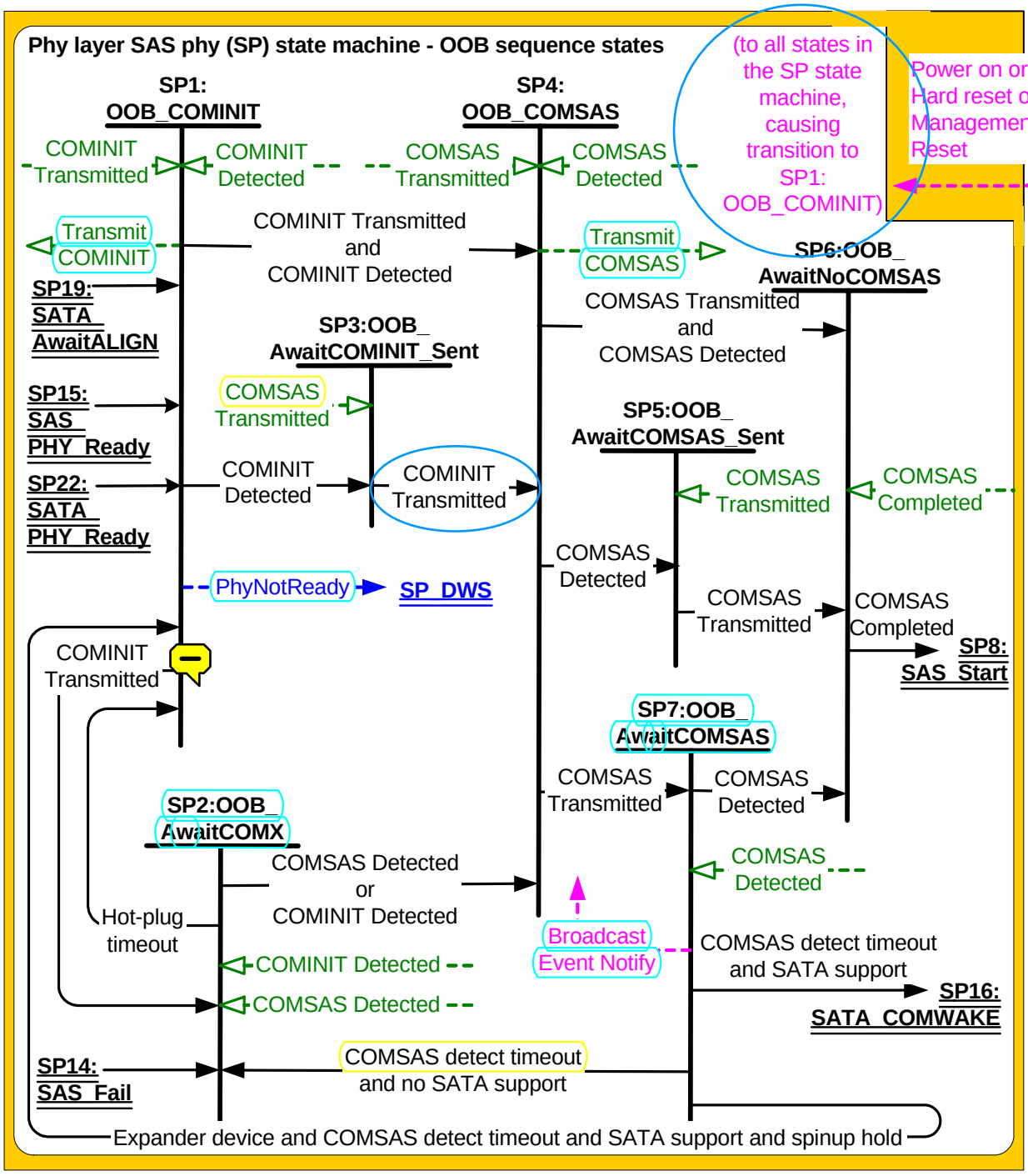


Figure 56 — SAS phy (SP) state machine - OOB sequence states

6.8.2.1 SP1:OOB_COMINIT state

6.8.2.1.1 State description

This state is the initial state for the state machine.

This state shall send a Transmit COMINIT parameter to the SP transmitter and wait for COMINIT to be transmitted and/or received.

s state shall send a PhyNotReady parameter to the SP_DWS state machine.

6.8.2.1.2 Transition SP1:OOB_COMINIT to SP2:OOB_AwaitCOMX

This transition shall occur if this state receives a COMINIT Transmitted parameter and does not receive a COMINIT Detected parameter.

6.8.2.1.3 Transition SP1:OOB_COMINIT to SP3:OOB_AwaitCOMINIT_Sent

This transition shall occur if this state receives a COMINIT Detected parameter and does not receive a COMINIT Transmitted parameter.

6.8.2.1.4 Transition SP1:OOB_COMINIT to SP4:OOB_COMSAS

This transition shall occur if this state receives both a COMINIT Transmitted parameter and a COMINIT Detected parameter.

6.8.2.2 SP2:OOB_AwaitCOMX state

6.8.2.2.1 State description

Upon entering this state, a hot-plug timeout timer shall be initialized and enabled. The state machine waits for COMINIT, COMSAS, or a hot-plug timeout.

6.8.2.2.2 Transition SP2:OOB_AwaitCOMX to SP1:OOB_COMINIT

This transition shall occur if a hot-plug timeout occurs.

6.8.2.2.3 Transition SP2:OOB_AwaitCOMX to SP4:OOB_COMSAS

This transition shall occur if this state receives either a COMINIT Detected parameter or a COMSAS Detected parameter.

6.8.2.3 SP3:OOB_AwaitCOMINIT_Sent state

6.8.2.3.1 State description

~~This state is entered when a COMINIT sequence has been detected but the COMINIT initiated in SP1:OOB_COMINIT has not been completely transmitted.~~

This state waits for COMINIT to be transmitted.

6.8.2.3.2 Transition SP3:OOB_AwaitCOMINIT_Sent to SP4:COMSAS

This transition shall occur if this state receives a COMINIT Transmitted parameter.

6.8.2.4 SP4:OOB_COMSAS state

6.8.2.4.1 State description

~~This state is reached when a COMINIT has been transmitted and detected.~~

This state shall send a Transmit COMSAS parameter to the SP transmitter and wait for COMSAS to be transmitted and/or detected.

6.8.2.4.2 Transition SP4:OOB_COMSAS to SP5:OOB_AwaitCOMSAS_Sent

This transition shall occur if this state receives a COMSAS Detected parameter and does not receive a COMSAS Transmitted parameter.

6.8.2.4.3 Transition SP4:OOB_COMSAS to SP6:OOB_AwaitNoCOMSAS

This transition shall occur if this state receives both a COMSAS Transmitted parameter and a COMSAS Detected parameter.

6.8.2.4.4 Transition SP4:OOB_COMSAS to SP7:OOB_AwaitCOMSAS

This transition shall occur if this state receives a COMSAS Transmitted parameter and does not receive a COMSAS Detected parameter.

6.8.2.5 SP5:OOB_AwaitCOMSAS_Sent state

6.8.2.5.1 State description

This state waits for COMSAS to be transmitted.

6.8.2.5.2 Transition SP5:OOB_AwaitCOMSAS_Sent to SP6:OOB_AwaitNoCOMSAS

This transition shall occur if this state receives a COMSAS Transmitted parameter.

6.8.2.6 SP6:OOB_AwaitNoCOMSAS state

6.8.2.6.1 State description

~~This state is entered when a COMSAS sequence has been both transmitted and detected.~~

The state machine waits for COMSAS to be completely received.

6.8.2.6.2 Transition SP6:OOB_AwaitNoCOMSAS to SP8:SAS_Start

This transition shall occur if this state receives a COMSAS Completed parameter. The COMSAS Completed parameter may be received before this state is entered.

6.8.2.7 SP7:OOB_AwaitCOMSAS state

6.8.2.7.1 State description

Upon entering this state the COMSAS detect timeout timer shall be initialized and enabled.

This state waits for COMSAS to be received or for a COMSAS detect timeout.

6.8.2.7.2 Transition SP7:OOB_AwaitCOMSAS to SP1:OOB_COMINIT

If all of these conditions are true:

a) this device in an expander device;
b) this device supports attachment to a SATA device on this phy;
c) the COMSAS detect timeout timer expires; and
d) this device implements SATA spinup hold and the SP1:OOB_COMINIT state was not originally entered because of an SMP Reset request

this state shall send a Broadcast Event Notify (SATA Spinup Hold) confirmation to the expander function and perform this transition.

NOTE 11 In other words, SMP PHY CONTROL-based requests to reset the phy bypass spinup hold; all other resets honor it.

6.8.2.7.3 Transition SP7:OOB_AwaitCOMSAS to SP6:OOB_AwaitNoCOMSAS

This transition shall occur if this state receives a COMSAS Detected parameter.

6.8.2.7.4 Transition SP7:OOB_AwaitCOMSAS to SP16:SATA_COMWAKE

This transition shall occur if:

a) the device supports attachment to SATA devices; and
b) the COMSAS detect timeout timer expires.

6.8.2.7.5 Transition SP7:OOB_AwaitCOMSAS to SAS_AwaitNoCOMX

This transition shall occur if the device does not support SATA and the COMSAS detect timeout timer expires.

6.8.3 SAS speed negotiation states

Figure 57 shows the SAS speed negotiation states, in which the SAS phy has detected that it is attached to a SAS phy, and performs the SAS speed negotiation sequence. These states are indicated by state names with a prefix of SAS.

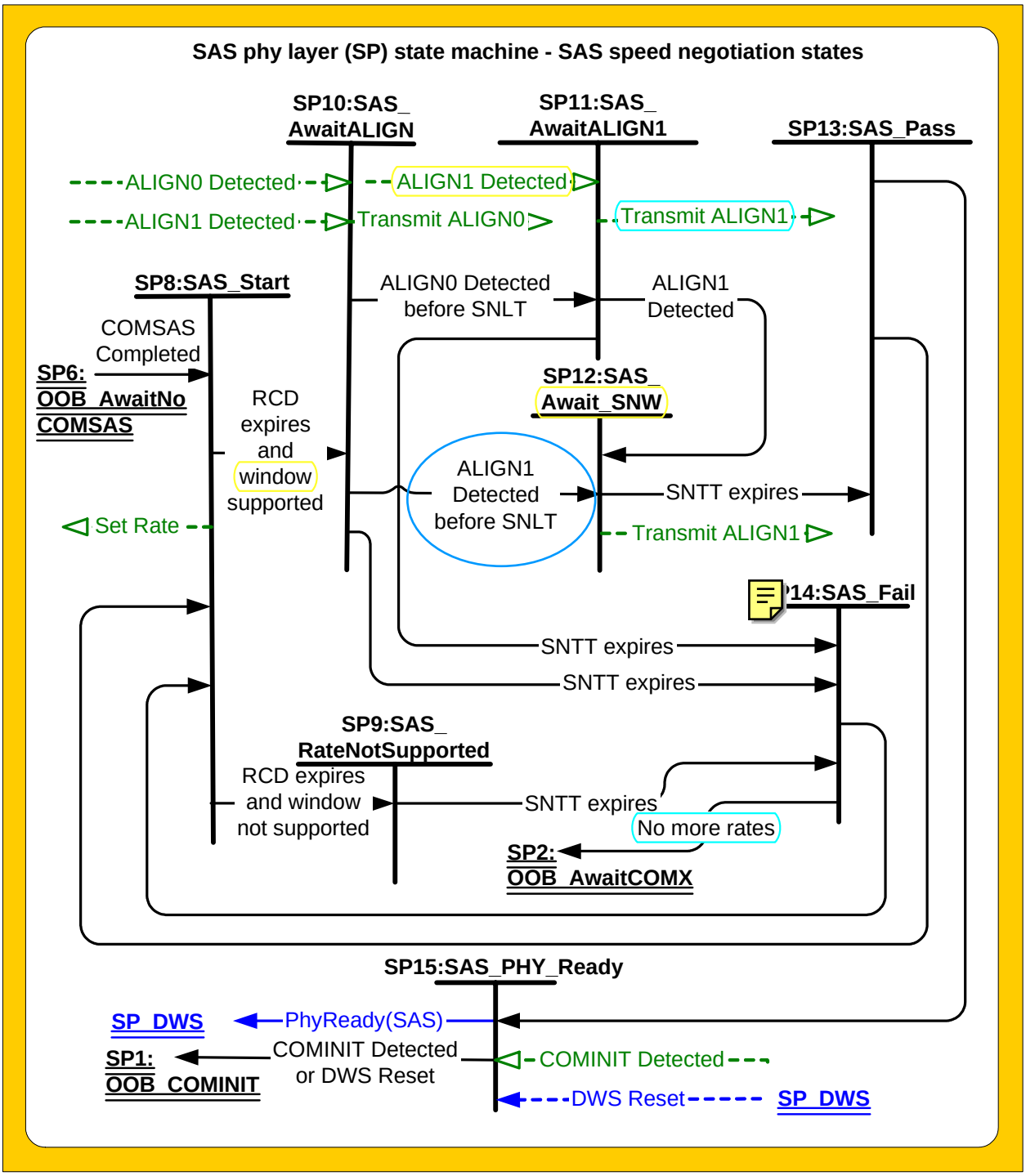


Figure 57 — SAS phy (SP) state machine - SAS speed negotiation states

6.8.3.1 SP8:SAS_Start state

6.8.3.1.1 State description

This state marks the beginning of the SAS speed negotiation process. ~~It is used to transmit idle in between SAS speed negotiation windows.~~

Upon entering this state, the RCD timer shall be initialized and started. ~~This allows time required for a transmitter to switch to either the next higher or next lower supported speed.~~ The state machine remains in this state until a RCD has elapsed.

This state shall send the Set Rate parameter to the SP transmitter selecting the next rate for attempted negotiation.

This state sets the SAS speed negotiation rate to:

a) the lowest supported speed negotiation window if the transition into this state is from the SP6:OOB_AwaitNoCOMSAS state; or
b) to the value of the speed negotiation window received as an argument.

During this state idle shall be transmitted.

6.8.3.1.2 Transition SP8:SAS_Start to SP10:SAS_AwaitALIGN

This transition shall occur if the RCD timer expires and the current speed negotiation window is supported.

6.8.3.1.3 Transition SP8:SAS_Start to SP9:SAS_RateNotSupported

This transition shall occur if the RCD timer expires and the current speed negotiation window is not supported.

6.8.3.2 SP9:SAS_RateNotSupported state

6.8.3.2.1 State description

Upon entering this state the SNTT timer shall be initialized and enabled. ~~The state machine exits from this state after the SNTT expires.~~

During this state idle shall be transmitted.

6.8.3.2.2 Transition SP9:SAS_RateNotSupported to SP14:SAS_Fail

This transition shall occur if the SNTT expires.

6.8.3.3 SP10:SAS_AwaitALIGN state

6.8.3.3.1 State description

The state machine shall start transmitting ALIGN (0) primitives at the current rate (G1, G2, G3…).

Upon entering this state, the SNTT timer and SNLT timer shall be initialized and enabled.

The state machine exits this state when SNTT expires without the pattern synchronization, or if synchronization occurs before the SNLT expires.

Editor's Note 1: To better integrate SP and SP_DWS, these changes should be made in SAS-1. SP10:SAS_AwaitALIGN and SP19:AwaitALIGN should start the DWS state machine (not the PhyReady states). When DWS reports it is dword aligned, the SP receiver can start reporting ALIGNnn Received. If DWS loses alignment in SP11, SP12, or SP15 (for SAS) or SP19, SP20, or SP21 (for SATA) the SP state machine goes back to SP1. Once SP reaches SP15 or SP21, the PhyReady confirmation goes to the upper layers. When SP goes to SP1, the PhyNotReady confirmation goes to the upper layers.

6.8.3.3.2 Transition SP10:SAS_AwaitALIGN to SP11:SAS_AwaitALIGN1

This transition shall occur if this state receives an ALIGN0 Detected parameter before the SNLT timer expires.

6.8.3.3.3 Transition SP10:SAS_AwaitALIGN to SP12:SAS_AwaitSNW

This transition shall occur if this state receives an ALIGN1 Detected parameter before the SNLT timer expires.

6.8.3.3.4 Transition SP10:SAS_AwaitALIGN to SP14:SAS_Fail

This transition shall occur if the SNTT timer expires.

6.8.3.4 SP11:SAS_AwaitALIGN1 state

6.8.3.4.1 State description

~~This state is reached after ALIGN (0) has been both transmitted and received.~~

This state shall repeatedly send a Transmit ALIGN0 parameter to the SP transmitter.

This state is exited when the SNTT expires or when ALIGN (1) primitives are received before the SNLT timer expires.

6.8.3.4.2 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 lock at the current rate.

6.8.3.4.3 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_ AwaitSNW

This transition shall occur if this state receives an ALIGN1 Detected parameter before the SNTT timer expires. This indicates that the other phy has been able to lock at the current rate.

6.8.3.5 SP12:SAS_AwaitSNW state

6.8.3.5.1 State description

~~This state is reached after ALIGN (1) has been both transmitted and received.~~

This state shall repeatedly send a Transmit ALIGN1 parameter to the SP transmitter.

This state waits for the SNTT timer to expire.

6.8.3.5.2 Transition SP12:SAS_AwaitALIGN1 to SP13:SAS_Pass

This transition shall occur if the SNTT timer expires.

6.8.3.6 SP13:SAS_Pass state

6.8.3.6.1 State description

This state determines if:

a) another SAS speed negotiation window is required; and
b) the SAS speed negotiation is complete.

6.8.3.6.2 Transition SP13:SAS_Pass to SP8:SAS_Start

This transition shall occur:

a) after setting the SAS speed negotiation window rate to one greater than the current SAS speed negotiation window rate which is sent as an argument to the SN_start state; and
b) if the state machine has not fallen back during this current SAS speed negotiation.

6.8.3.6.3 Transition SP13:SAS_Pass to SP15:SAS_PHY_Ready

This transition shall occur if speed negotiation has progressed to where it failed and then had fallen back to the last negotiated speed and then subsequently passed.

6.8.3.7 SP14:SAS_Fail state

6.8.3.7.1 State description

This state determines if the SAS speed negotiation window failure occurred because:

a) the maximum SAS speed negotiation window has been attempted and there haven't been any successful negotiated physical link rates;
b) the SAS speed negotiation failed after dropping back to the last successful SAS speed negotiation window;
c) the SAS speed negotiation has failed and there was a previous successful SAS speed negotiation; or
d) no SAS speed negotiation has previously passed and the maximum SAS speed negotiation window has not yet been attempted.

6.8.3.7.2 Transition SP14:SAS_Fail to SP2:OOB_AwaitCOMX

This transition shall occur if:

a) the maximum SAS speed negotiation window has been attempted and there haven't been any successful negotiated physical link rates; or
b) the SAS speed negotiation failed after dropping back to the last successful SAS speed negotiation window.

6.8.3.7.3 Transition SP14:SAS_Fail to SP8:SAS_Start

This transition shall occur:

a) after setting the SAS speed negotiation window to one less the current SAS speed negotiation window; and
b) if the SAS speed negotiation has failed and there was a previous successful SAS speed negotiation;

or:

a) after setting the SAS speed negotiation window to one greater than the current SAS speed negotiation window; and
b) if no SAS speed negotiation has previously passed and the maximum supported SAS speed negotiation window has not yet been attempted.

Which speed negotiation window to use is sent as an argument with this transition.

6.8.3.8 SP15:SAS_PHY_Ready state

6.8.3.8.1 State description

This state enables the SAS phy dword synchronization state machine (SP_DWS) ~~to provide rule checking for dword synchronization and determination of link failure.~~

This state machine monitors for:

a) the receipt of a COMINIT; and
b) the DWS Reset parameter.

While in this state dwords ~~from the link layer~~ are transmitted at the negotiated physical link rate.

This state shall send a PhyReady (SAS) parameter to the SP_DWS state machine to indicate that the link has been brought up successfully in SAS mode. ~~While in this state, dwords from the link layer are transmitted at the negotiated physical link rate~~

6.8.3.8.2 Transition SP15:SAS_PHY_Ready to SP1:OOB_COMINIT

This transition shall occur if:

a) a COMINIT Detected parameter is received; or
b) a DWS Reset parameter is received.

6.8.4 SATA host emulation states

Figure 58 shows the SATA host emulation states, in which the SAS device (an initiator device or expander device) has detected that it is attached to a SATA target device and behaves as if it were a SATA host, initiating the SATA speed negotiation sequence. These states are indicated by state names with a prefix of SATA.

~~During SATA host emulation, the SAS device transmits a COMWAKE sequence and then waits to receive a COMWAKE. Once the COMWAKE sequence is detected, the SAS device follows the speed negotiation sequence defined in SATA.~~

The power management states defined in this specification are for SAS initiator devices that support being attached to SATA target devices; SAS expander devices attached to SATA target devices do not support power management in this standard.

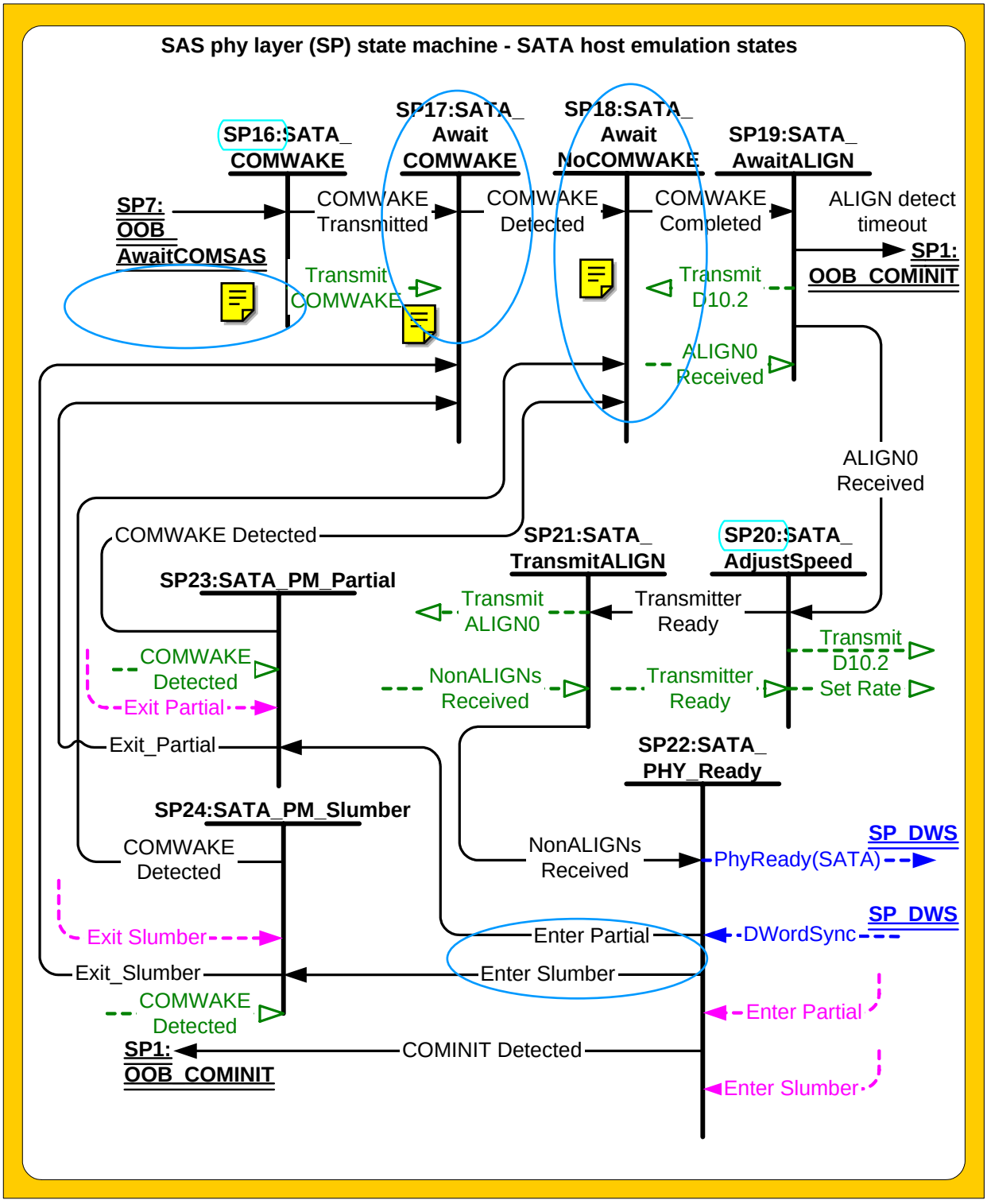


Figure 58 — SAS phy (SP) state machine - SATA host emulation states

6.8.4.1 SP16:SATA_COMWAKE state

6.8.4.1.1 State description

This state shall send a Transmit COMWAKE parameter to the SP transmitter and wait for COMWAKE to be transmitted.

6.8.4.1.2 Transition SP16:SATA_COMWAKE to SP17:SATA_AwaitCOMWAKE

This transition shall occur if this state receives a COMWAKE Transmitted parameter.

6.8.4.2 SP17:SATA_AwaitCOMWAKE state

6.8.4.2.1 State description

This state waits for COMWAKE to be received.

6.8.4.2.2 Transition SP17:SATA_AwaitCOMWAKE to SP18:SATA_AawitNoCOMWAKE

This transition shall occur if this state receives a COMWAKE Detected parameter.

6.8.4.3 SP18:SATA_AwaitNoCOMWAKE state

6.8.4.3.1 State description

This state waits for COMWAKE to be fully received.

6.8.4.3.2 Transition SP18:SATA_AwaitNoCOMWAKE to SP19:SATA_AwaitALIGN

This transition shall occur if this state receives a COMWAKE Completed parameter.

6.8.4.4 SP19:SATA_AwaitALIGN state

6.8.4.4.1 State description

This state shall:

a) repeatedly send a Transmit D10.2 parameter to the SP transmitter;
b) start the ALIGN detect timeout timer; and
c) wait for an ALIGN to be received or an ALIGN detect timeout.

~~The SAS device shall start transmitting D10.2 characters no later than 20 G1 dwords (i.e., 533 ns) after COMWAKE was deasserted.~~

6.8.4.4.2 Transition SP19:SATA_AwaitALIGN to SP20:SATA_AdjustSpeed

This transition shall occur if this state receives an ALIGN0 Received parameter ~~at any of its supported physical link rates.~~

6.8.4.4.3 Transition SP19:SATA_AwaitALIGN to SP1:OOB_COMINIT

This transition shall occur if the ALIGN detect timeout timer expires.

6.8.4.5 SP20:SATA_AdjustSpeed state

6.8.4.5.1 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 repeatedly send Transmit D10.2 parameters to the SP transmitter and shall send a Set Rate parameter to the SP transmitter.

6.8.4.5.2 Transition SP20:SATA_AdjustSpeed to SP21:SATA_TransmitALIGN

This transition shall occur when this state receives a Transmitter Ready parameter.

6.8.4.6 SP21:SATA_TransmitALIGN state

6.8.4.6.1 State description

This state shall repeatedly send the Transmit ALIGN0s parameter to the SP transmitter.

~~When the SP receiver detects three back-to-back non-ALIGNs, the state machine transitions to state SP22:SATA_PHY_Ready.~~

6.8.4.6.2 Transition SP21:SATA_TransmitALIGN to SP22:SATA_PHY_Ready

This transition shall occur when this state receives a NonALIGNS Received parameter.

6.8.4.7 SP22:SATA_PHY_Ready state

6.8.4.7.1 State description

In this state, the SP state machine hands control over dword transmission to the SP_DWS state machine. The SP receiver monitors the input dword stream looking for COMINIT.

This state shall send a PhyReady (SATA) parameter to the SP_DWS state machine.

6.8.4.7.2 Transition SP22:SATA_PHY_Ready to SP1:Reset

This transition shall occur if this state receives a COMINIT Received parameter or a DWS Reset parameter.

6.8.4.7.3 Transition SP22:SATA_PHY_Ready to SP24:SATA_PM_Partial

This transition shall occur if this state receives an Enter Partial request.

6.8.4.7.4 Transition SP22:SATA_PHY_Ready to SP23:SATA_PM_Slumber

This transition shall occur if this state receives an Enter Slumber request.

6.8.4.8 SP23:SATA_PM_Partial state

6.8.4.8.1 State description

Exit from this state is driven from receipt of COMWAKE or by request of the link layer.

6.8.4.8.2 Transition SP23:SATA_PM_Partial to SP16:SATA_COMWAKE

This transition shall occur if this state receives a Exit Partial request.

6.8.4.8.3 Transition SP23:SATA_PM_Partial to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur if this state receives a COMWAKE Detected parameter.

6.8.4.9 SP24:SATA_PM_Slumber state

6.8.4.9.1 State description

Exit from this state is driven from receipt of COMWAKE or by request of the link layer.

6.8.4.9.2 Transition SP24:SATA_PM_Slumber to SP16:SATA_COMWAKE

This transition shall occur if this state receives a Exit Slumber request.

6.8.4.9.3 Transition SP24:SATA_PM_Slumber to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur if this state receives a COMWAKE Detected parameter.

6.9 SAS phy dword synchronization (SP_DWS) state machine

6.9.1 Overview

Each SAS phy includes a SAS phy dword synchronization state machine (SP_DWS).

The SP_DWS state machine establishes the same dword boundaries at the receiver as at the attached transmitter by searching for control characters. A receiver in the SAS phy monitors and decodes the incoming data stream and forces K28.5 characters into the first byte position to effectively perform dword alignment. The receiver continues to reestablish dword alignment by forcing received K28.5 characters into the first byte position until a valid primitive is detected. The resultant primitives, dwords and valid dword indicators (e.g., encoding error indicators) are sent to the SP_DWS machine to enable the state machine to determine the dword synchronization policy.

After dword synchronization has been achieved, the SP_DWS 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.

The SP_DWS state machine contains the following states:

a) SP_DWS0:AcquireSync (see 6.9.2)(initial state);
b) SP_DWS1:Valid1 (see 6.9.3);
c) SP_DWS2:Valid2 (see 6.9.4);
d) SP_DWS3:SyncAcquired (see 6.9.5);
e) SP_DWS4:Lost1 (see 6.9.6);
f) SP_DWS5:Lost1Recovered (see 6.9.7);
g) SP_DWS6:Lost2 (see 6.9.8);
h) SP_DWS7:Lost2Recovered (see 6.9.9);
i) SP_DWS8:Lost3 (see 6.9.10); and
j) SP_DWS9:Lost3Recovered (see 6.9.11).

The state machine shall start in the SP_DWS0:AcquireSync state after:

a) power on;
b) hard reset; or
c) receiving a PhyReady (SAS) or PhyReady (SATA) parameter ~~from the SP state machine.~~

The SP_DWS state machine receives the following parameters ~~from the SP state machine:~~

a) PhyNotReady;
b) PhyReady (SAS); and
c) PhyReady (SATA).

The SP_DWS state machine sends the following parameters to the SP state machine:

a) DWS Reset.

Figure 59 shows the SP_DWS state machine.

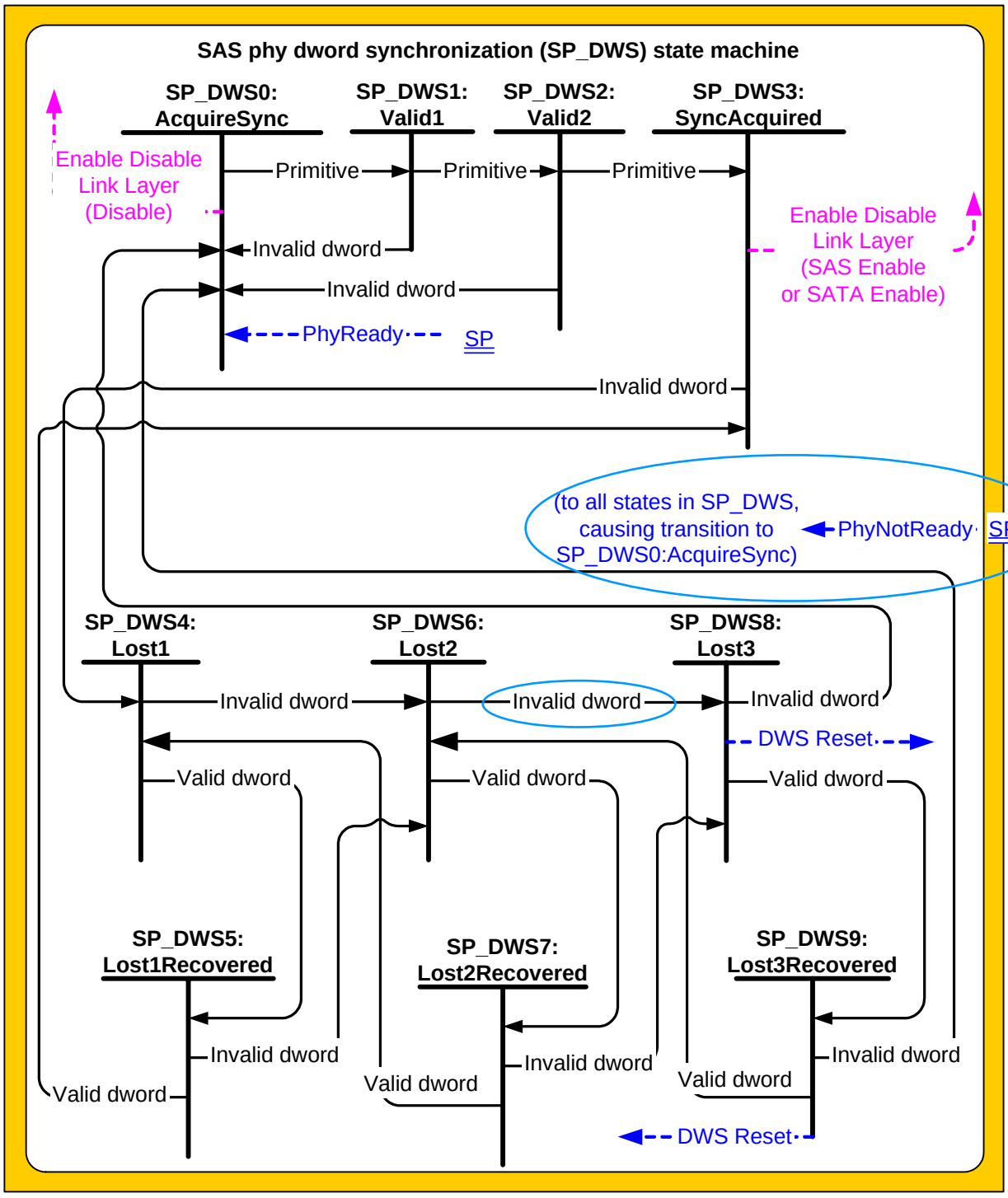


Figure 59 — SAS phy dword synchronization (SP_DWS) state machine

6.9.2 SP_DWS0:AcquireSync state

6.9.2.1 State description

~~This state is entered upon power on loss or previous dword synchronization.~~ On entry into this state, this state shall send an Enable Disable Link Layer (Disable) confirmation to the link layer.

In this state, the receiver monitors the input data stream and forces each K28.5 character it detects into the first byte position for possible dword alignment. This is the only state that shall modify dword alignment.

6.9.2.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1

This transition shall occur when a PhyReady parameter has been received and a valid primitive (e.g., a known dword starting with K28.5) is detected.

6.9.3 SP_DWS1:Valid1 state

6.9.3.1 State description

~~This state is reached after one valid primitive has been detected.~~ In this state, the receiver shall monitor the input data stream looking for a second valid primitive or an invalid dword.

6.9.3.2 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2

This transition shall occur when a valid primitive is detected.

6.9.4 SP_DWS2:Valid2 state

6.9.4.1 State description

~~This state is reached after the receiver has detected two valid primitives.~~ In this state, the receiver shall monitor the input data stream looking for a third valid primitive or an invalid dword.

6.9.4.2 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired

This transition shall occur when a valid primitive is detected.

6.9.4.3 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync

This transition shall occur when an invalid dword is detected.

6.9.5 SP_DWS3:SyncAcquired state

6.9.5.1 State description

~~This state is reached when the receiver has detected three valid primitives without adjusting the dword synchronization.~~ This state shall send one of the following confirmations to the link layer:

a) an Enable Disable Link Layer (SAS Enable) confirmation if it has received the PhyReady (SAS) parameter; or
b) an Enable Disable Link Layer (SATA Enable) confirmation if is has received the PhyReady (SATA) parameter.

The most recently received primitive shall be considered valid for processing by the link layer.

In this state, the receiver shall monitor the incoming data stream looking for an invalid dword, which indicates that dword synchronization might be lost.

6.9.5.2 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1

This transition shall occur when an invalid dword (i.e., the first invalid dword) is detected. An expander forwarding the dword to another phy shall replace the invalid dword with ERROR for a SAS physical link or SATA_ERROR for a SATA physical link.

6.9.6 SP_DWS4:Lost1 state

6.9.6.1 State description

~~This state is reached when one invalid dword has been received and not nullified.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.6.2 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered

This transition shall occur when a valid dword is detected.

6.9.6.3 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2

This transition shall occur when an invalid dword is detected. An expander forwarding the dword to another phy shall replace the invalid dword with ERROR for a SAS physical link or SATA_ERROR for a SATA physical link.

6.9.7 SP_DWS5:Lost1Recovered state

6.9.7.1 State description

~~This state is reached when a valid dword has been received, and another valid dword will nullify the previous invalid dword.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.7.2 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired

This transition shall occur when a valid dword is detected.

6.9.7.3 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2

This transition shall occur when an invalid dword is detected. An expander forwarding the dword to another phy shall replace the invalid dword with ERROR for a SAS physical link or SATA_ERROR for a SATA physical link.

6.9.8 SP_DWS6:Lost2 state

6.9.8.1 State description

~~This state is reached when two invalid dwords has been received and not nullified.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.8.2 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered

This transition shall occur when a valid dword is detected.

6.9.8.3 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3

This transition shall occur when an invalid dword is detected. An expander forwarding the dword to another phy shall replace the invalid dword with ERROR for a SAS physical link or SATA_ERROR for a SATA physical link.

6.9.9 SP_DWS7:Lost2Recovered state

6.9.9.1 State description

~~This state is reached when a valid dword has been received, and another valid dword will nullify the previous invalid dword.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.9.2 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1

This transition shall occur when a valid dword is detected.

6.9.9.3 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3

This transition shall occur when an invalid dword is detected. An expander forwarding the dword to another phy shall replace the invalid dword with ERROR for a SAS physical link or SATA_ERROR for a SATA physical link.

6.9.10 SP_DWS8:Lost3 state

6.9.10.1 State description

~~This state is reached when three invalid dwords has been received and not nullified.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.10.2 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered

This transition shall occur when a valid dword is detected.

6.9.10.3 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync

If an invalid dword (i.e., the fourth non-nullified invalid dword) is detected, this state shall send a DWS Reset parameter to the SP state machine and this transition shall occur.

6.9.11 SP_DWS9:Lost3Recovered state

6.9.11.1 State description

~~This state is reached when a valid dword has been received, and another valid dword will nullify the previous invalid dword.~~ In this state, the receiver shall monitor the incoming data stream looking for a valid or invalid dword.

6.9.11.2 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2

This transition shall occur when a valid dword is detected.

6.9.11.3 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync

If an invalid dword (i.e., the fourth non-nullified invalid dword) is detected, this state shall send a DWS Reset parameter to the SP state machine and this transition shall occur.

6.10 Spin-up

If a SAS target device receives COMSAS during the reset sequence, it shall not spin-up until allowed by the power condition state machine (see 10.1.8).

If a SAS target device supporting SATA does not receive COMSAS during the reset sequence, it shall follow SATA spin-up rules (see SATA).

~~NOTE 12 A SATA target device with rotating media spins up:~~
~~a) automatically after power on (allowed by SATA);~~
~~b) after its phy is enabled (allowed by SATA);~~
~~c) after the reset sequence has completed (recommended by SATA); or~~
~~d) after the Power Up in Standby flag is cleared by an application (if the ATA Power Up in Standby feature is implemented).~~
~~The ATA Power Up in Standby feature is not widely implemented, since it requires the target device to include a nonvolatile memory to remember the state of the Power Up in Standby flag. Desktop-class disk drives do not typically have nonvolatile memory storage.~~

Expander devices attached to SATA target devices may halt the automatic phy reset sequence to delay spin-up; this is called SATA spinup hold. This is reported in the SMP DISCOVER function (see 10.3.1.4) and is released with the SMP PHY CONTROL function (see 10.3.1.9).

NOTE 13 Enclosures supporting both SATA-only target devices and SAS target devices may need to sequence power to each target device to avoid excessive power consumption during power on, since the SATA-only target devices may spin-up automatically after power on.

7 Link layer

7.1 Primitives

7.1.1 Primitives overview

Primitives are dwords whose first byte is a K28.5 or K28.3 control character. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last bytes. Table 50 shows the primitive format.

Table 50 — Primitive format

Byte	Description
First	K28.5 control character (for primitives defined in this standard) or K28.3 control character (for SATA primitives).
Second	Constant data character.
Third	Constant data character.
Last	Constant data character.

7.1.2 Primitive summary

Table 51 defines the primitives not specific to the type of connection.

Table 51 — 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		I	E	T	Single
AIP (RESERVED 0)	NoConn				I	E	T	Single
AIP (RESERVED 1)	NoConn				I	E	T	Single
AIP (RESERVED 2)	NoConn				I	E	T	Single
AIP (RESERVED WAITING ON PARTIAL)	NoConn				I	E	T	Single
AIP (WAITING ON CONNECTION)	NoConn		E		I	E	T	Single
AIP (WAITING ON DEVICE)	NoConn		E		I	E	T	Single
AIP (WAITING ON PARTIAL)	NoConn		E		I	E	T	Single
ALIGN (0)	All	I	E	T	I	E	T	Single
ALIGN (1)	All	I	E	T	I	E	T	Single
ALIGN (2)	All	I	E	T	I	E	T	Single
ALIGN (3)	All	I	E	T	I	E	T	Single
BREAK	All	I	E	T	I	E	T	Redundant
BROADCAST (CHANGE)	NoConn	I	E		I	E	T	Redundant
BROADCAST (RESERVED 0)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED 1)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED CHANGE)	NoConn				I	E	T	Redundant
CLOSE (CLEAR AFFILIATION)	STP	I					T	Triple
CLOSE (NORMAL)	Conn	I		T	I		T	Triple
CLOSE (RESERVED 0)	Conn				I		T	Triple
CLOSE (RESERVED 1)	Conn				I		T	Triple
EOAF	NoConn	I	E	T	I	E	T	Single

Table 51 — 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	
ERROR	All		E		I	E	T	Single
HARD_RESET	NoConn	I	E		I	E	T	Redundant
NOTIFY (ENABLE_SPINUP)	All	I	E				T	Single
NOTIFY (RESERVED 0)	All				I	E	T	Single
NOTIFY (RESERVED 1)	All				I	E	T	Single
NOTIFY (RESERVED 2)	All				I	E	T	Single
OPEN_ACCEPT	NoConn	I		T	I		T	Single
OPEN_REJECT (BAD DESTINATION)	NoConn		E		I		T	Single
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	NoConn	I	E	T	I		T	Single
OPEN_REJECT (NO DESTINATION)	NoConn		E		I		T	Single
OPEN_REJECT (PATHWAY BLOCKED)	NoConn		E		I		T	Single
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	NoConn	I		T	I		T	Single
OPEN_REJECT (RESERVED ABANDON 0)	NoConn				I		T	Single
OPEN_REJECT (RESERVED ABANDON 1)	NoConn				I		T	Single
OPEN_REJECT (RESERVED ABANDON 2)	NoConn				I		T	Single
OPEN_REJECT (RESERVED ABANDON 3)	NoConn				I		T	Single
OPEN_REJECT (RESERVED CONTINUE 0)	NoConn				I		T	Single
OPEN_REJECT (RESERVED CONTINUE 1)	NoConn				I		T	Single
OPEN_REJECT (RESERVED INITIALIZE 0)	NoConn				I		T	Single
OPEN_REJECT (RESERVED INITIALIZE 1)	NoConn				I		T	Single
OPEN_REJECT (RESERVED STOP 0)	NoConn				I		T	Single
OPEN_REJECT (RESERVED STOP 1)	NoConn				I		T	Single
OPEN_REJECT (RETRY)	NoConn	I		T	I		T	Single
OPEN_REJECT (STP RESOURCES BUSY)	NoConn		E		I			Single
OPEN_REJECT (WRONG DESTINATION)	NoConn	I		T	I		T	Single
SOAF	NoConn	I	E	T	I	E	T	Single

[a] The Use column indicates when the primitive is used:
a) NoConn: SAS physical links, outside connections;
b) Conn: SAS physical links, inside connections;
c) All: SAS physical links, both outside connections or inside any type of connection; or
d) STP: SAS physical links, inside STP 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 initiator ports;
b) E for expander ports; and
c) T for 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 repeated primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.1.3).

Table 52 defines the primitives used only inside SSP and SMP connections.

Table 52 — 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)	SSP	I		T	I		T	Single
DONE (NORMAL)	SSP	I		T	I		T	Single
DONE (RESERVED 0)	SSP				I		T	Single
DONE (RESERVED 1)	SSP				I		T	Single
DONE (RESERVED TIMEOUT 0)	SSP				I		T	Single
DONE (RESERVED TIMEOUT 1)	SSP				I		T	Single
EOF	SSP, SMP	I		T	I		T	Single
NAK (CRC ERROR)	SSP	I		T	I		T	Single
NAK (RESERVED 0)	SSP				I		T	Single
NAK (RESERVED 1)	SSP				I		T	Single
NAK (RESERVED 2)	SSP				I		T	Single
RRDY (NORMAL)	SSP	I		T	I		T	Single
RRDY (RESERVED 0)	SSP				I		T	Single
RRDY (RESERVED 1)	SSP				I		T	Single
SOF	SSP, SMP	I		T	I		T	Single

[a] The Use column indicates when the primitive is used:
a) SSP: SAS physical links, inside SSP connections; or
b) SMP: SAS physical 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 initiator ports;
b) E for expander ports; and
c) T for 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 repeated primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.1.3).

Table 53 lists the primitives used only inside STP connections and on SATA physical links.

Table 53 — 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	SATA		E				T	Single
SATA_HOLD	STP, SATA	I		T	I		T	Repeated
SATA_HOLDA	STP, SATA	I		T	I		T	Repeated
SATA_PMACK	STP, SATA							Single
SATA_PMNAK	STP, SATA	I	E				T	Single
SATA_PMREQ_P	STP, SATA							Repeated
SATA_PMREQ_S	STP, SATA							Repeated
SATA_R_ERR	STP, SATA	I		T	I		T	Repeated
SATA_R_IP	STP, SATA	I		T	I		T	Repeated
SATA_R_OK	STP, SATA	I		T	I		T	Repeated
SATA_R_RDY	STP, SATA	I		T	I		T	Repeated
SATA_SOF	STP, SATA	I		T	I		T	Single
SATA_SYNC	STP, SATA	I		T	I		T	Repeated
SATA_WTRM	STP, SATA	I		T	I		T	Repeated
SATA_X_RDY	STP, SATA	I		T	I		T	Repeated

[a] The Use column indicates when the primitive is used:
a) STP: SAS physical 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 initiator ports;
b) E for expander ports; and
c) T for 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 repeated primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.1.3).

Table 54 defines the primitive encoding for primitives used outside connections.

Table 54 — Primitive encoding for primitives not specific to type of connection (part 1 of 2)

Primitive	Character			
	1st	2nd	3rd	4th (last)
AIP (NORMAL)	K28.5	D27.4	D27.4	D27.4
AIP (RESERVED 0)	K28.5	D27.4	D31.4	D16.7
AIP (RESERVED 1)	K28.5	D27.4	D16.7	D30.0
AIP (RESERVED 2)	K28.5	D27.4	D29.7	D01.4
AIP (RESERVED WAITING ON PARTIAL)	K28.5	D27.4	D01.4	D07.3
AIP (WAITING ON CONNECTION)	K28.5	D27.4	D07.3	D24.0
AIP (WAITING ON DEVICE)	K28.5	D27.4	D30.0	D29.7
AIP (WAITING ON PARTIAL)	K28.5	D27.4	D24.0	D04.7
ALIGN (0)	K28.5	D10.2	D10.2	D27.3
ALIGN (1)	K28.5	D07.0	D07.0	D07.0
ALIGN (2)	K28.5	D01.3	D01.3	D01.3
ALIGN (3)	K28.5	D27.3	D27.3	D27.3
BREAK	K28.5	D02.0	D24.0	D07.3
BROADCAST (CHANGE)	K28.5	D04.7	D02.0	D01.4
BROADCAST (RESERVED 0)	K28.5	D04.7	D07.3	D29.7
BROADCAST (RESERVED 1)	K28.5	D04.7	D01.4	D24.0
BROADCAST (RESERVED CHANGE)	K28.5	D04.7	D24.0	D31.4
CLOSE (CLEAR AFFILIATION)	K28.5	D02.0	D07.3	D04.7
CLOSE (NORMAL)	K28.5	D02.0	D30.0	D27.4
CLOSE (RESERVED 0)	K28.5	D02.0	D31.4	D30.0
CLOSE (RESERVED 1)	K28.5	D02.0	D04.7	D01.4
EOAF	K28.5	D24.0	D07.3	D31.4
ERROR	K28.5	D02.0	D01.4	D29.7
HARD_RESET	K28.5	D02.0	D02.0	D02.0
NOTIFY (ENABLE_SPINUP)	K28.5	D31.3	D31.3	D31.3
NOTIFY (RESERVED 0)	K28.5	D31.3	D07.0	D01.3
NOTIFY (RESERVED 1)	K28.5	D31.3	D01.3	D07.0
NOTIFY (RESERVED 2)	K28.5	D31.3	D27.3	D10.2
OPEN_ACCEPT	K28.5	D16.7	D16.7	D16.7
OPEN_REJECT (BAD DESTINATION)	K28.5	D31.4	D31.4	D31.4

Table 54 — Primitive encoding for primitives not specific to type of connection (part 2 of 2)

Primitive	Character			
	1st	2nd	3rd	4th (last)
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	K28.5	D31.4	D04.7	D29.7
OPEN_REJECT (NO DESTINATION)	K28.5	D29.7	D29.7	D29.7
OPEN_REJECT (PATHWAY BLOCKED)	K28.5	D29.7	D16.7	D04.7
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	K28.5	D31.4	D29.7	D07.3
OPEN_REJECT (RESERVED ABANDON 0)	K28.5	D31.4	D02.0	D27.4
OPEN_REJECT (RESERVED ABANDON 1)	K28.5	D31.4	D30.0	D16.7
OPEN_REJECT (RESERVED ABANDON 2)	K28.5	D31.4	D07.3	D02.0
OPEN_REJECT (RESERVED ABANDON 3)	K28.5	D31.4	D01.4	D30.0
OPEN_REJECT (RESERVED CONTINUE 0)	K28.5	D29.7	D02.0	D30.0
OPEN_REJECT (RESERVED CONTINUE 1)	K28.5	D29.7	D24.0	D01.4
OPEN_REJECT (RESERVED INITIALIZE 0)	K28.5	D29.7	D30.0	D31.4
OPEN_REJECT (RESERVED INITIALIZE 1)	K28.5	D29.7	D07.3	D16.7
OPEN_REJECT (RESERVED STOP 0)	K28.5	D29.7	D31.4	D07.3
OPEN_REJECT (RESERVED STOP 1)	K28.5	D29.7	D04.7	D27.4
OPEN_REJECT (RETRY)	K28.5	D29.7	D27.4	D24.0
OPEN_REJECT (STP RESOURCES BUSY)	K28.5	D31.4	D27.4	D01.4
OPEN_REJECT (WRONG DESTINATION)	K28.5	D31.4	D16.7	D24.0
SOAF	K28.5	D24.0	D30.0	D01.4

Table 55 defines the primitive encodings for primitives used inside SSP and SMP connections.

Table 55 — Primitive encoding for primitives used only inside SSP and SMP connections

Primitive	Character			
	1st	2nd	3rd	4th (last)
ACK	K28.5	D01.4	D01.4	D01.4
CREDIT_BLOCKED	K28.5	D01.4	D07.3	D30.0
DONE (ACK/NAK TIMEOUT)	K28.5	D30.0	D01.4	D04.7
DONE (CREDIT TIMEOUT)	K28.5	D30.0	D07.3	D27.4
DONE (NORMAL)	K28.5	D30.0	D30.0	D30.0
DONE (RESERVED 0)	K28.5	D30.0	D16.7	D01.4
DONE (RESERVED 1)	K28.5	D30.0	D29.7	D31.4
DONE (RESERVED TIMEOUT 0)	K28.5	D30.0	D27.4	D29.7
DONE (RESERVED TIMEOUT 1)	K28.5	D30.0	D31.4	D24.0
EOF	K28.5	D24.0	D16.7	D27.4
NAK (CRC ERROR)	K28.5	D01.4	D27.4	D04.7
NAK (RESERVED 0)	K28.5	D01.4	D31.4	D29.7
NAK (RESERVED 1)	K28.5	D01.4	D04.7	D24.0
NAK (RESERVED 2)	K28.5	D01.4	D16.7	D07.3
RRDY (NORMAL)	K28.5	D01.4	D24.0	D16.7
RRDY (RESERVED 0)	K28.5	D01.4	D02.0	D31.4
RRDY (RESERVED 1)	K28.5	D01.4	D30.0	D02.0
SOF	K28.5	D24.0	D04.7	D07.3

Table 56 lists the primitive encodings for primitives used inside STP connections and on SATA physical links.

Table 56 — Primitive encoding for primitives used only inside STP connections and on SATA physical links

Primitive	Character			
	1st	2nd	3rd	4th (last)
SATA_CONT	K28.3	D10.5	D25.4	D25.4
SATA_DMAT	K28.3	D21.5	D22.1	D22.1
SATA_EOF	K28.3	D21.5	D21.6	D21.6
SATA_ERROR [a]	K28.6	D02.0	D01.4	D29.7
SATA_HOLD	K28.3	D10.5	D21.6	D21.6
SATA_HOLDA	K28.3	D10.5	D21.4	D21.4
SATA_PMACK	K28.3	D21.4	D21.4	D21.4
SATA_PMNAK	K28.3	D21.4	D21.7	D21.7
SATA_PMREQ_P	K28.3	D21.5	D23.0	D23.0
SATA_PMREQ_S	K28.3	D21.4	D21.3	D21.3
SATA_R_ERR	K28.3	D21.5	D22.2	D22.2
SATA_R_IP	K28.3	D21.5	D21.2	D21.2
SATA_R_OK	K28.3	D21.5	D21.1	D21.1
SATA_R_RDY	K28.3	D21.4	D10.2	D10.2
SATA_SOF	K28.3	D21.5	D23.1	D23.1
SATA_SYNC	K28.3	D21.4	D21.5	D21.5
SATA_WTRM	K28.3	D21.5	D24.2	D24.2
SATA_X_RDY	K28.3	D21.5	D23.2	D23.2

[a] Except for SATA_ERROR, all values are defined by SATA.

7.1.3 Primitive sequences

7.1.3.1 Primitive sequence overview

Table 57 summarizes the types of primitive sequences.

Table 57 — Primitive sequences

Primitive sequence type	Number of times sent	Number of times received to detect
Single	1	1
Repeated	2	1
Triple	3	3
Redundant	6	3

~~ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements.~~ Rate matching ALIGNs shall be sent inside primitive sequences inside of connections if rate matching is enabled. Any number of ALIGNs may be inserted in primitive sequences.

7.1.3.2 Single primitive sequence

Primitives labeled as single primitive sequences are sent one time.

7.1.3.3 Repeated primitive sequence

Primitives that form repeated primitive sequences (e.g., SATA_HOLD) shall be sent two times, then be followed by SATA_CONT, then be followed by vendor-specific scrambled data dwords until ~~a response is received.~~

Figure 60 shows an example of a repeated primitive sequence.

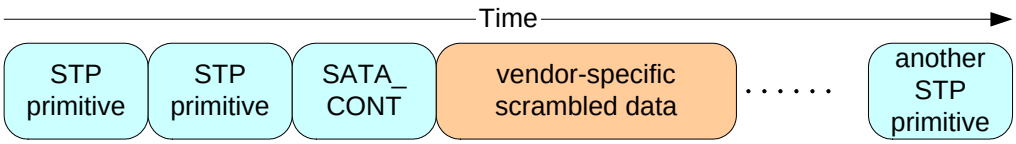


Figure 60 — Repeated primitive sequence

7.1.3.4 Triple primitive sequence

Primitives that form triple primitive sequences (e.g., CLOSE) shall be sent three times consecutively and followed by idle dwords until a response is received. ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements.

Receivers shall detect a triple primitive sequence by receiving the identical primitive in three consecutive dwords. A receiver shall not detect primitive sequences a second time until it has received three consecutive dwords that are not any of the following:

a) the original primitive; or
b) ALIGNs.

Figure 61 shows an example of a triple primitive sequence.

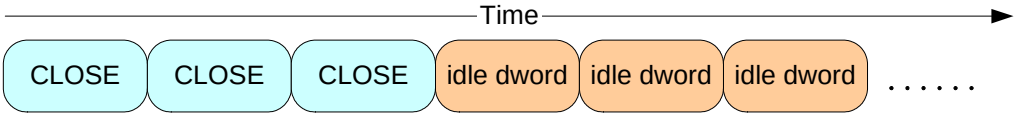


Figure 61 — Triple primitive sequence

7.1.3.5 Redundant primitive sequence

Primitives that form redundant primitive sequences (e.g., BROADCAST (CHANGE)) shall be sent six times consecutively. ALIGNs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements.

A receiver shall detect redundant primitive sequences by receiving an identical primitive for three consecutive dwords. Redundant primitive sequences shall only be detected outside of a connection. A receiver shall not detect a redundant primitive sequence a second time until it has received six consecutive dwords that are not any of the following:

a) the original primitive; or
b) an ALIGN.

Figure 62 shows an example of a redundant primitive sequence.

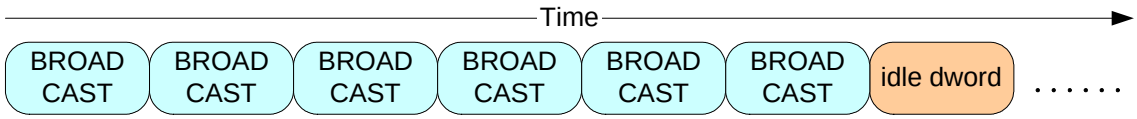


Figure 62 — Redundant primitive sequence

7.1.4 Primitives not specific to type of connections

7.1.4.1 AIP (Arbitration in progress)

AIP is sent by an expander device after a connection request to indicate that the connection request is being processed and indicate the status of the connection request.

The versions of AIP representing different statuses are defined in table 58.

Table 58 — AIP primitives

Primitive	Description
AIP (NORMAL)	Expander device has just accepted the connection request.
AIP (RESERVED 0)	Reserved. Processed the same as AIP (NORMAL).
AIP (RESERVED 1)	Reserved. Processed the same as AIP (NORMAL).
AIP (RESERVED 2)	Reserved. Processed the same as AIP (NORMAL).
AIP (WAITING ON CONNECTION)	Expander device has determined the routing for the connection request, but the destination phys are all busy with active connections.
AIP (WAITING ON DEVICE)	Expander device has determined the routing for the connection request and forwarded it to the output physical link.
AIP (WAITING ON PARTIAL)	Expander device has determined the routing for the connection request, but the destination phys are all busy with other connection requests that have also received AIP (WAITING ON PARTIAL).
AIP (RESERVED WAITING ON PARTIAL)	Reserved. Processed the same as AIP (WAITING ON PARTIAL).

See 7.12 for details on connections.

7.1.4.2 ALIGN

ALIGNs are used for:

a) character and dword alignment during the phy reset sequence; and
b) ~~are used for~~ rate matching and clock skew management after the phy reset sequence.

ALIGN shall be used to construct OOB signals and during the speed negotiation sequence. Phys shall rotate through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3) for all ALIGNs sent after the speed negotiation sequence. SAS phys receiving ALIGNs shall not verify the rotation and shall accept any of the ALIGNs at any time.

SAS devices shall only detect an ALIGN after decoding all four characters in the primitive.

~~NOTE 14 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.~~

See 7.2 for requirements for ALIGN insertion and other details on clock skew management.

7.1.4.3 BREAK

BREAK is used to abandon a connection request or break a connection.

See 7.12.5 and 7.12.6 for details on breaking connections.

7.1.4.4 BROADCAST

BROADCASTs are used to notify all initiator ports and target ports in a domain of an event.

The versions of BROADCAST representing different reasons are defined in table 59.

Table 59 — BROADCAST primitives

Primitive	Description
BROADCAST (CHANGE)	Notification of a configuration change.
BROADCAST (RESERVED CHANGE)	Reserved. End devices process the same as BROADCAST (CHANGE).
BROADCAST (RESERVED 0)	Reserved.
BROADCAST (RESERVED 1)	Reserved.

When an expander port receives a BROADCAST it shall transmit the same BROADCAST on at least one phy in all other expander ports. BROADCAST shall only be sent outside of connections after the initialization sequence has completed.

An expander device is not required to queue multiple identical BROADCAST indications for the same expander port. If a second identical BROADCAST indication is requested before the first indication has been transmitted, the second indication may be dropped.

BROADCAST (CHANGE) is sent by an expander device to notify initiator ports and other expander devices that a configuration change has occurred. BROADCAST (CHANGE) may also be sent by initiator ports. BROADCAST (CHANGE) shall be ignored by target ports.

BROADCAST (RESERVED CHANGE) shall be processed the same as BROADCAST (CHANGE) by end devices. BROADCAST (RESERVED 0) and BROADCAST (RESERVED 1) shall be ignored by initiator ports and target ports.

See 7.11 for details on domain changes. See 10.3.1.2 for details on counting BROADCAST (CHANGE) generation in an expander device.

7.1.4.5 CLOSE

CLOSE is used to close an open connection. This primitive may be originated by an initiator port, SSP target port, or by an expander device on behalf of an SATA target device.

The versions of CLOSE representing different reasons are defined in table 60.

Table 60 — CLOSE primitives

Primitive	Description
CLOSE (CLEAR AFFILIATION)	Close an open STP connection and clear the initiator affiliation.
CLOSE (NORMAL)	Close an open connection.
CLOSE (RESERVED 0)	Reserved. Processed the same as CLOSE (NORMAL).
CLOSE (RESERVED 1)	Reserved. Processed the same as CLOSE (NORMAL).

See 7.12.7 for details on closing connections.

7.1.4.6 EOAF (End of address frame)

EOAF indicates the end of an address frame.

See 7.4 for details on address frames.

7.1.4.7 ERROR

ERROR is sent by an expander device when it is forwarding dwords from a SAS or SATA link to a SAS link and it receives an invalid dword.

See 6.9 for details on error handling by expander devices.

7.1.4.8 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 and before the identification sequence (see 4.4).

7.1.4.9 NOTIFY

NOTIFY may be sent in place of an ALIGN. It may or may not affect the ALIGN sequencing (i.e., rotation through ALIGN (0), ALIGN (1), ALIGN (2), or ALIGN (3)). NOTIFY shall not be transmitted until at least three ALIGNs have been transmitted since the previous NOTIFY.

NOTIFY shall not be forwarded through expander devices.

NOTIFY (ENABLE_SPINUP) is transmitted by an initiator port or expander port and is used to specify to an SSP 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 SSP 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-3) and/or the START STOP UNIT command (see SBC-2 ~~and RBC),~~ ~~as described in~~ ~~TBD.~~

Initiator devices and expander devices shall transmit NOTIFY (ENABLE_SPINUP) while attached to SSP target devices (i.e., devices that report SSP target support in their IDENTIFY address frames). They shall transmit one NOTIFY (ENABLE_SPINUP) after power on when the enclosure is ready for initial target device 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 15 The initiator device or expander device uses NOTIFY (ENABLE_SPINUP) to avoid exceeding enclosure power supply capabilities during spin-up of multiple 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 target devices spinning up at a time. An expander device may allow this timing to be configured by a NVROM programming with enclosure-specific sequencing patterns, or may employ more complex, dynamic interaction with the enclosure power supply.

NOTE 16 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 target device to spin-up automatically on receipt of NOTIFY (ENABLE_SPINUP).

Target devices with multiple target ports shall accept NOTIFY (ENABLE_SPINUP) from all target ports (e.g., NOTIFY (ENABLE_SPINUP) received on target port A serves as a wakeup for a START STOP UNIT command received through target port B).

NOTIFY (RESERVED 0), NOTIFY (RESERVED 1), and NOTIFY (RESERVED 2) shall be ignored by all devices.

7.1.4.10 OPEN_ACCEPT

OPEN_ACCEPT indicates the acceptance of a connection request.

See 7.12 for details on connection requests.

7.1.4.11 OPEN_REJECT

OPEN_REJECT indicates that a connection request has been rejected and indicates the reason for the rejection. The response to some OPEN_REJECTs is to abandon the connection request and the response to other OPEN_REJECTs is to retry the connection request.

All of the OPEN_REJECT versions defined in table 61 shall result in the originating device abandoning the connection request.

Table 61 — OPEN_REJECT abandon primitives

Primitive	Originator	Description
OPEN_REJECT (BAD DESTINATION)	Expander phy	The destination SAS address equals the source SAS address or the expander device determines the connection request needs to be routed to the same expander port as the expander port through which the connection request arrived.
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	Any device	Requested connection rate is not supported on some physical link on the pathway between the source phy and destination phy. When an initiator phy is directly attached to a target phy, reques[illegible] connection rate is not supported by the destination phy.
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	Destination phy	Device with destination SAS address exists but the destination device does not support the requested initiator/ target role and/or protocol.
OPEN_REJECT (RESERVED ABANDON 0)		Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 1)		Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 2)		Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 3)		Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (STP RESOURCES BUSY)	Expander phy	Device with destination SAS address exists but the STP target port has an active affiliation with another STP initiator port.
OPEN_REJECT (WRONG DESTINATION)	Destination phy	The destination SAS address does not match the SAS address of the end device to which the connection request was delivered.

All of the OPEN_REJECT versions defined in table 62 shall result in the originating device retrying the connection request.

Table 62 — OPEN_REJECT retry primitives

Primitive	Originator	Description
OPEN_REJECT (NO DESTINATION) [c]	Expander device	No such destination device. For an otherwise valid SAS address of a SATA target devices attached to the expander device, indicates the initial Register - Device to Host FIS has not been successfully received (see 10.3.1.6).
OPEN_REJECT (PATHWAY BLOCKED) [b]	Expander device	An expander device determined the pathway was blocked by higher priority connection requests.
OPEN_REJECT (RESERVED CONTINUE 0) [a]		Reserved. Process the same as OPEN_REJECT (RETRY).
OPEN_REJECT (RESERVED CONTINUE 1) [a]		Reserved. Process the same as OPEN_REJECT (RETRY).
OPEN_REJECT (RESERVED INITIALIZE 0) [c]		Reserved. Process the same as OPEN_REJECT (NO DESTINATION).
OPEN_REJECT (RESERVED INITIALIZE 1) [c]		Reserved. Process the same as OPEN_REJECT (NO DESTINATION).
OPEN_REJECT (RESERVED STOP 0) [b]		Reserved. Process the same as OPEN_REJECT (PATHWAY BLOCKED)
OPEN_REJECT (RESERVED STOP 1) [b]		Reserved. Process the same as OPEN_REJECT (PATHWAY BLOCKED).
OPEN_REJECT (RETRY) [a]	Destination phy	Device with destination SAS address exists but is not able to accept connections.

[a] If the I_T nexus loss timer is already running, it is stopped.
[b] If the I_T nexus loss timer is already running, it continues running. Abandon connection request rather than retry if the I_T nexus loss timer expires.
[c] If the I_T nexus loss timer is already running, it continues running; if it is not already running, it is initialized and started. Abandon connection request rather than retry if the I_T nexus loss timer expires.

When a destination device detects more than one reason to transmit an OPEN_REJECT, the device shall transmit only one OPEN_REJECT and shall select the primitive using the following priority:

1) OPEN_REJECT (WRONG DESTINATION) (highest priority selection);
2) OPEN_REJECT (PROTOCOL NOT SUPPORTED);
3) OPEN_REJECT (CONNECTION RATE NOT SUPPORTED); or
4) OPEN_REJECT (RETRY) (lowest priority selection).

When an expander device detects more than one reason to transmit an OPEN_REJECT, the expander shall transmit only one OPEN_REJECT primitive and shall select that primitive using the following priority:

1) OPEN_REJECT (BAD DESTINATION) or OPEN_REJECT (NO DESTINATION) (highest priority selection);
2) OPEN_REJECT (CONNECTION RATE NOT SUPPORTED); or
3) OPEN_REJECT (STP RESOURCES BUSY) or OPEN_REJECT (PATHWAY BLOCKED) (lowest priority selection).

See 7.12 for details on connection requests.

7.1.4.12 SOAF (Start of address frame)

SOAF indicates the start of an address frame.

See 7.4 for details on address frames.

7.1.5 Primitives used only inside SSP and SMP connections

7.1.5.1 ACK (Acknowledge)

ACK indicates the positive acknowledgement of an SSP frame.

See 7.16.3 for details on SSP frame transmission.

7.1.5.2 CREDIT_BLOCKED

CREDIT_BLOCKED indicates that no more credit is going to be sent during this connection.

See 7.16.4 for details on SSP flow control.

7.1.5.3 DONE

DONE is used to start closing an SSP connection and indicate 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 63. The SSP state machine describes when these are used (see 7.16.7).

Table 63 — DONE primitives

Primitive	Description
DONE (ACK/NAK TIMEOUT)	Timed out waiting for an ACK or NAK. The ACK/NAK count does not match the frame count. Transmitter is going to transmit BREAK in 1 ms unless DONE is received prior to that.
DONE (RESERVED TIMEOUT 0)	Reserved. Processed the same as DONE (ACK/NAK TIMEOUT).
DONE (RESERVED TIMEOUT 1)	Reserved. Processed the same as DONE (ACK/NAK TIMEOUT).
DONE (NORMAL)	Finished transmitting all frames.
DONE (RESERVED 0)	Reserved. Processed the same as DONE (NORMAL).
DONE (RESERVED 1)	Reserved. Processed the same as DONE (NORMAL).
DONE (CREDIT TIMEOUT)	Timed out waiting for an RRDY.

See 7.16.6 for details on closing SSP connections.

7.1.5.4 EOF (End of frame)

EOF indicates 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.

7.1.5.5 NAK (Negative acknowledgement)

NAK indicates the negative acknowledgement of an SSP frame and the reason for doing so.

The versions of NAK representing different reasons are defined in table 64.

Table 64 — NAK primitives

Primitive	Description
NAK (CRC ERROR)	The frame had a bad CRC.
NAK (RESERVED 0)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 1)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 2)	Reserved. Processed the same as NAK (CRC ERROR).

See 7.16.3 for details on SSP frame transmission.

7.1.5.6 RRDY (Receiver ready)

RRDY is used to increase SSP frame credit.

The versions of RRDY representing different reasons are defined in table 64.

Table 65 — 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)	Reserved. Processed the same as RRDY (NORMAL).
~~RRDY (RESERVED 2)~~	~~Reserved. Processed the same as RRDY (NORMAL).~~

See 7.16.4 for details on SSP flow control.

7.1.5.7 SOF (Start of frame)

SOF indicates 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.

7.1.6 Primitives used only inside STP connections and on SATA physical links

7.1.6.1 SATA_ERROR

SATA_ERROR is sent by an expander device when it is forwarding dwords from a SAS link to a SATA link and it receives an invalid dword.

See 6.9 for details on error handling by expander devices.

7.1.6.2 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 7.4 for rules on handling the power management primitives.

7.1.6.3 SATA_HOLD and SATA_HOLDA (Hold and hold acknowledge)

An expander device running SATA protocol shall transmit a SATA_HOLDA within 20 dwords of transmitting a SATA_HOLD. The expander device shall generate SATA_HOLDs to throttle data for any latency it adds.

NOTE 17 SATA assumes that once a SATA_HOLD is transmitted, SATA_HOLDA will arrive within 20 dwords. This limits the amount of buffering required in the SATA device transmitting SATA_HOLD.

7.1.6.4 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. Expander devices shall not transmit SATA_R_RDY until the STP connection is established.

7.1.6.5 Other primitives used inside STP connections and on SATA physical links

Other primitives used in STP connections and on SATA physical links are used as defined in SATA.

7.2 Clock skew management

The internal clock for a device is typically based on a PLL with its own clock generator. This is used when transmitting data onto the physical link. When receiving, however, data needs to be latched based on a clock derived from the input data itself. Although the input clock is nominally a fixed frequency, it may differ slightly from the internal clock frequency due to accepted manufacturing tolerance and, for SATA links, due to spread spectrum clocking. Over time, if the input clock is faster than the internal clock, the device may receive data 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 device may not have data when needed in an internal buffer; this is called an underrun.

To solve this, devices insert ALIGNs in the data stream. Receivers may pass the ALIGNs through to their internal buffers, or may strip it out when an overrun occurs. Receivers add ALIGNs when an underrun occurs. The internal logic shall ignore all ALIGNs that make it to the internal buffers.

Elasticity buffer circuitry, as shown in figure 63, is required to absorb the slight differences in frequencies between the initiator phy, target phy, and expander phys. The frequency tolerance for a SAS phy is specified in 5.7.2.

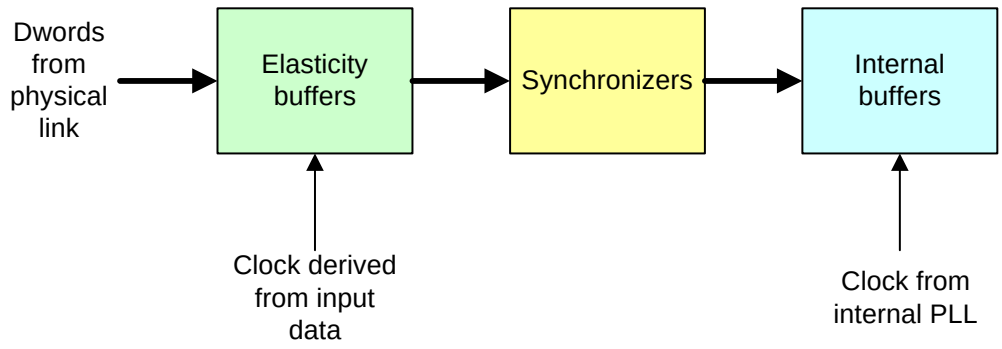


Figure 63 — Elasticity buffers

A SAS phy that is the original source for the data (i.e., ~~that is~~ not an expander phy forwarding dwords from another expander phy) shall periodically insert ALIGNs into the data stream as shown in table 66.

Table 66 — ALIGN insertion requirements

Original source of data	ALIGN requirement
SSP initiator phy or target phy in SSP connection, SMP initiator phy or SMP target phy in SMP connection, Any phy outside connections, or STP target phy in an STP connection	One ALIGN within every 2 048 dwords
STP initiator phy in an STP connection	Two consecutive ALIGNs within each 256 dwords plus one ALIGN within each 2 048 dwords

See 7.1.4.2 for details on rotating amongst ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3).

An expander device that is repeating data is allowed to insert or delete as many ALIGNs as required to match the transmit and receive connection rates.

NOTE 18 One possible implementation is for the expander device to delete all ALIGNs at the receive port and to insert ALIGNs at the transmit port whenever its elasticity buffer is empty.

An expander device that is attached to a SATA target device is allowed to insert or delete as many ALIGNs as required on the STP side to match the transmit and receive connection rates.

NOTE 19 One possible implementation is for the expander device to delete all ALIGNs at the receive port and to insert two consecutive ALIGNs at the transmit port when its elasticity buffer is empty or when 254 non-ALIGN dwords have been transmitted. An expander device is not required to insert ALIGNs in pairs when transmitting data to the initiator phy on the STP side.

7.3 Idle links

While no connection is open and a physical link is idle, or while an SSP or SMP connection is open and the physical link is idle, SAS phys shall transmit idle dwords.

While an STP connection is open, SAS STP devices transmit SATA_SYNC between frames (see SATA). After transmitting two SATA_SYNCs, SAS STP devices shall transmit SATA_CONT and start transmitting vendor-specific scrambled data dwords.

NOTE 20 SATA devices are allowed but not required to transmit SATA_CONT.

See 7.5 for details on scrambling.

7.4 CRC

7.4.1 CRC overview

All frames include cyclic redundancy check (CRC) values to help detect transmission errors.

Frames transmitted in a STP connection shall include a CRC as defined by SATA. Address frames, SSP frames, and SMP frames shall include a CRC as defined by this standard.

Annex B contains information on CRC generation/checker implementation.

Table 67 defines the CRC polynomials.

Table 67 — CRC polynomials

Function	Definition
F(x)	A polynomial of degree k-1 that is used to represent the k bits of the frame covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be the first bit transmitted.
L(x)	A degree 31 polynomial with all of the coefficients set to one: $L(x) = x^{31} + x^{30} + x^{29} + \ldots + x^2 + x^1 + 1$ (i.e., L(x) = FFFFFFFFh)
G(x)	The standard generator 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) = 1_04C11DB7h)
R(x)	The remainder polynomial, which is of degree less than 32.
P(x)	The remainder polynomial on the receive checking side, which is of degree less than 32.
Q(x)	The greatest multiple of G(x) in $(x^{32} \times F(x)) + (x^k \times L(x))$
Q'(x)	$x^{32} \times Q(x)$
M(x)	The sequence that is transmitted.
M'(x)	The sequence that is received.
C(x)	A unique polynomial remainder produced by the receiver upon reception of an error free sequence. This polynomial has the value: $C(x) = x^{32} \times L(x) / G(x)$ $C(x) = x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$ (i.e., C(x) = C704DD7Bh)

7.4.2 CRC generation

The equations that are used to generate the CRC from F(x) are as follows. All arithmetic is modulo 2.

CRC value in frame = L(x) + R(x) = one's complement of R(x)

NOTE 21 Adding L(x) (all ones) to R(x) ~~simply~~ produces the one's complement of R(x); this equation is specifying that the R(x) is inverted before it is transmitted.

The CRC is calculated by:

$(x^{32} \times F(x) + x^k \times L(x)) / G(x) = Q(x) + R(x) / G(x)$

The following equation specifies that the CRC is appended to the end of F(x):

$M(x) = x^{32} \times F(x) + CRC$

The bit order of F(x) presented to the CRC function is the same order as the bit transmission order - the bits within each byte of the data dword are transposed to match the implicit transposition in the 8b10b encoding process. This order is shown in figure 64.

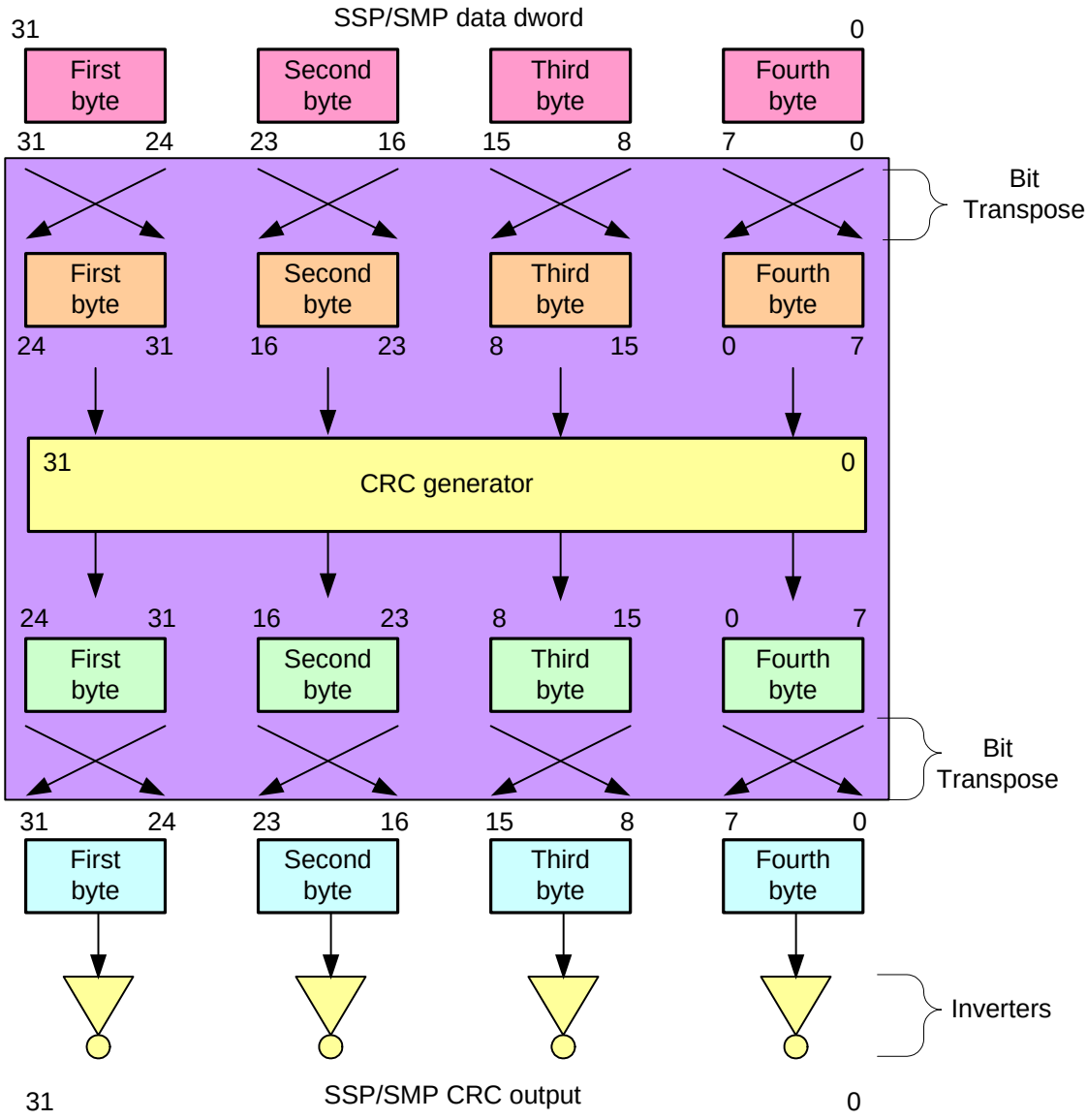


Figure 64 — CRC generator bit order

See 6.5 for details on how the CRC generator fits into the dword flow along with the scrambler.

7.4.3 CRC checking

The received sequence M'(x) may differ from the transmitted sequence M(x) if there are transmission errors. The process of checking the sequence for validity involves dividing the received sequence by G(x) and testing the remainder. Direct division, however, does not yield a unique remainder because of the possibility of leading zeros. Thus a term L(x) is prepended to M'(x) before it is divided. Mathematically, the received checking is shown in the following equation:

$x^{32} \times (M'(x) + x^{k} \times L(x)) / G(x) = Q'(x) + P(x) / G(x)$

In the absence of errors, the unique remainder is the remainder of the division:

$P(x) / G(x) = x^{32} \times L(x) / G(x) = C(x)$

The bit order of F(x) presented to the CRC checking function is the same order as the CRC generation bit order (see figure 64). See 6.5 for details on where the CRC checker fits into the dword flow along with the descrambler.

Annex B contains examples of CRC generation/checker implementations.

7.5 Scrambling

Scrambling is used to reduce the probability of long strings of zeros or ones appearing on the physical link. These patterns can cause issues in the physical layers.

There are several types of scrambling defined in table 68.

Table 68 — Scrambling types

Data	Description
Address frame scrambling	After an SOAF, all data dwords shall be scrambled until the EOAF.
SSP frame scrambling	During an SSP connection after an SOF, all data dwords shall be scrambled until the EOF.
SMP frame scrambling	During an SMP connection after an SOF, all data dwords shall be scrambled until the EOF.
STP frame scrambling	During an STP connection after a SATA_SOF, all data dwords shall be scrambled until the SATA_EOF.
Repeated SATA primitive scrambling	After a SATA_CONT, vendor-specific scrambled data dwords shall be sent until a primitive other than ALIGN is sent.
SAS idle dwords	When a connection is not open and the physical link is idle, vendor-specific scrambled data dwords shall be sent.
SSP idle dwords	When an SSP connection is open and the physical link is idle (i.e., between frames), vendor-specific scrambled data dwords shall be sent.

The scrambling polynomial is $x^{16} + x^{15} + x^{13} + x^4 + 1$. The data scrambling value shall be initialized to FFFFh at each SOF, SOAF, and SATA_SOF by both the transmitter and receiver. The data being transmitted shall be XORed with the data scrambling value by the transmitter, and the data being received shall be XORed with the data scrambling value by the receiver.

Table 69 shows when the scrambling logic shall treat data as big-endian and when it shall treat data as little-endian. See 6.5 for details on how the scrambler and descrambler fit into the dword flow.

Table 69 — Scrambling endianness

Data	Scrambling endianness
Address frame	Big-endian
SSP frame	Big-endian
SMP frame	Big-endian
STP frame	Little-endian
Repeated SATA primitive vendor-specific data dwords	Little-endian
SAS or SSP idle dwords	Vendor-specific

Annex E contains information on scrambling implementations.

7.6 Bit order of CRC and scrambler

Figure 65 shows how data dwords and primitives are routed to the bit transmission logic in figure 44. Data dwords go through the CRC generator and scrambler.

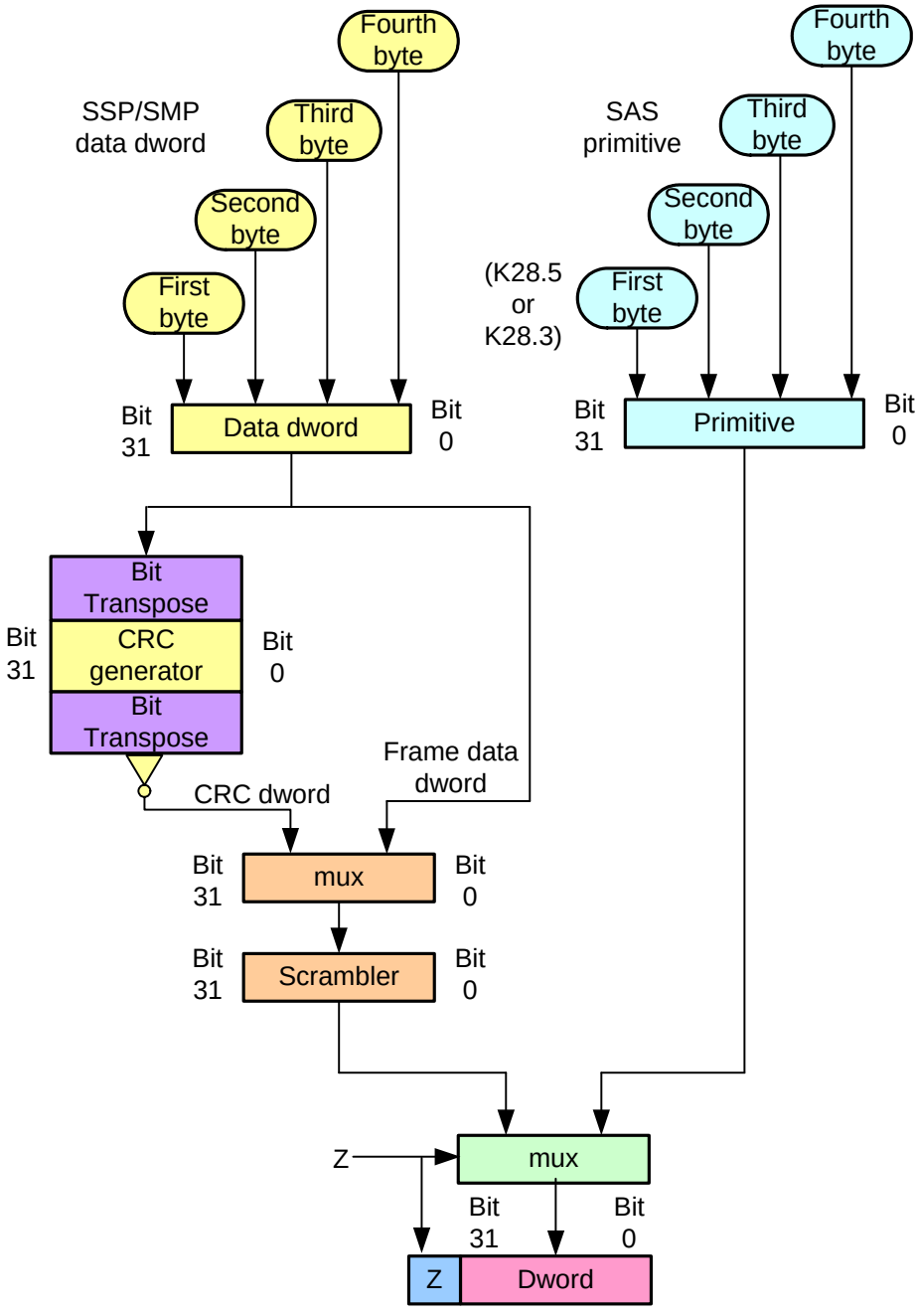


Figure 65 — Transmit path bit ordering

Figure 66 shows the routing of dwords received from the bit reception logic in figure 45.

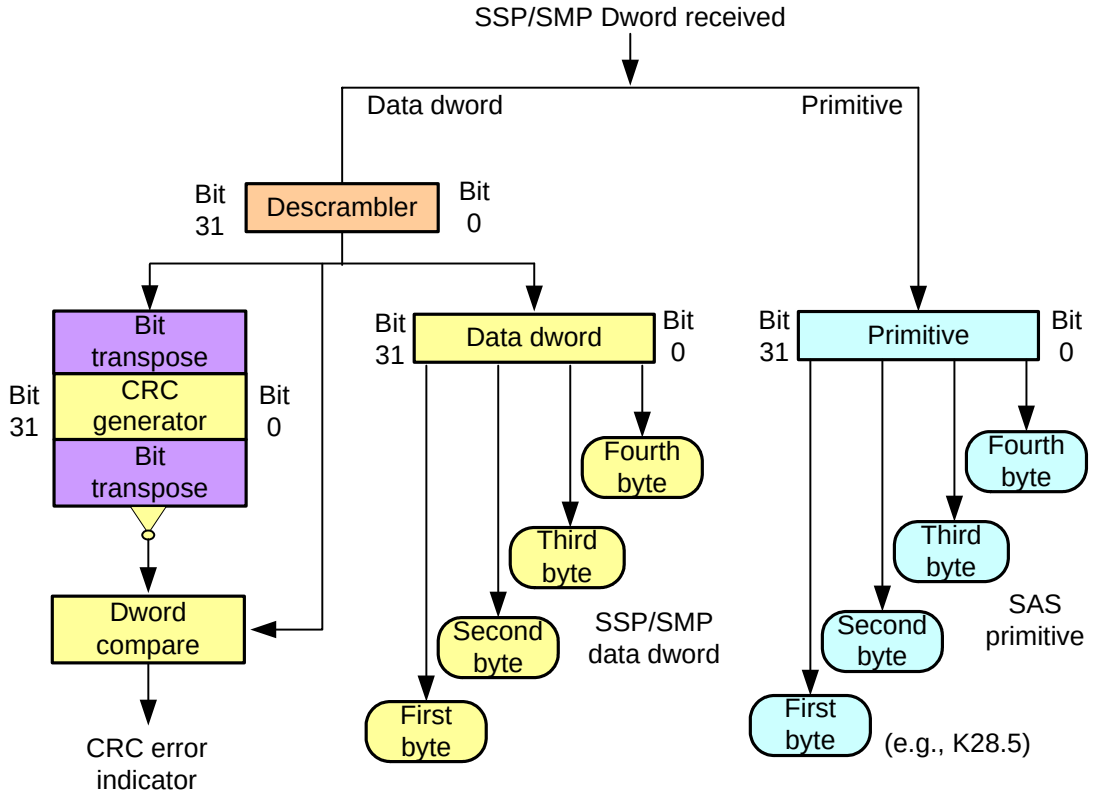


Figure 66 — Receive path bit ordering

7.7 Address frames

7.7.1 Address frames overview

Address frames are used for the identification sequence and for connection requests. The address frame follows an SOAF and ends with an EOAF. ~~Primitives may be inserted in the address frame.~~ 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 70 describes the address frame format.

Table 70 — Address frame format

Bit Byte	7	6	5	4	3	2	1	0
0					ADDRESS FRAME TYPE			
1								
				FRAME TYPE DEPENDENT BYTES				
27								
28	(MSB)							
				CRC				
31								(LSB)

The ADDRESS FRAME TYPE field indicates the type of address frame and is defined in table 71. This field determines the definition of the frame type dependent bytes.

Table 71 — Address frame types

Code	Frame type	Description
0h	Identify	Identification sequence
1h	Open	Connection request
All others	Reserved	

The CRC field is a CRC value (see 7.4) that is computed over the entire address frame.

Address frames with unknown address frame types, incorrect lengths, or CRC errors shall be ignored by the recipient.

7.7.2 IDENTIFY address frame

Table 72 defines the IDENTIFY address frame format used for the identification sequence. The IDENTIFY address frame is sent after the phy reset sequence completes and the physical link is a SAS physical link.

Table 72 — IDENTIFY address frame format

Bit Byte	7	6	5	4	3	2	1	0
0	Ignored	DEVICE TYPE			ADDRESS FRAME TYPE (0h)			
1	Ignored				Ignored			
2	Reserved				SSP INITIATOR	STP INITIATOR	SMP INITIATOR	Reserved
3	Reserved				SSP TARGET	STP TARGET	SMP TARGET	Ignored
4	Ignored							
11								
12	SAS ADDRESS							
19								
20	Ignored							
23								
24	Reserved							
27								
28	(MSB)							
31	CRC							(LSB)

~~The recipient shall ignore reserved and ignored fields in the IDENTIFY address frame.~~

The DEVICE TYPE field indicates the type of device containing the phy, and is defined in table 73. A device that is capable of being both an end device and an expander device shall only identify itself as an expander device in this field.

Table 73 — Device types

Code	Description
001b	End device ~~only~~
010b	Edge expander device
011b	Fanout expander device
All others	Reserved

An SMP INITIATOR bit of one indicates the presence of an SMP initiator device or SMP target/initiator device. An SMP INITIATOR bit of zero indicates neither an SMP initiator device nor an SMP target/initiator device are present.

An STP INITIATOR bit of one indicates the presence of an STP initiator device or STP target/initiator device. An STP INITIATOR bit of zero indicates neither an STP initiator device nor an STP target/initiator device are present.

An SSP INITIATOR bit of one indicates the presence of an SSP initiator device or SSP target/initiator device. An SSP INITIATOR bit of zero indicates neither an SSP initiator device nor an SSP target/initiator device are present.

An SMP TARGET bit of one indicates the presence of an SMP target device or SMP target/initiator device. An SMP TARGET bit of zero indicates neither an SMP target device nor an SMP target/initiator device are present.

An STP TARGET bit of one indicates the presence of an STP target device or STP target/initiator device. An STP TARGET bit of zero indicates neither an STP target device nor an STP target/initiator device are present.

An SSP TARGET bit of one indicates the presence of an SSP target device or SSP target/initiator device. An SSP TARGET bit of zero indicates neither an SSP target device nor an SSP target/initiator device are present.

The SAS ADDRESS field indicates the SAS address of the device transmitting the IDENTIFY address frame.

A wide port shall set the DEVICE TYPE field, SSP INITIATOR field, SMP INITIATOR field, STP INITIATOR field, SSP TARGET field, SMP TARGET field, STP TARGET field to the same set of values on each phy in the wide port. Recipient ports need not check the consistency of these fields across phys.

7.7.3 OPEN address frame

Table 74 defines the OPEN address frame format used for connection requests.

Table 74 — OPEN address frame format

Bit Byte	7	6	5	4	3	2	1	0
0	INITIATOR	PROTOCOL			ADDRESS FRAME TYPE (1h)			
1	FEATURES				CONNECTION RATE			
2	(MSB)	INITIATOR CONNECTION TAG						
3								(LSB)
4		DESTINATION SAS ADDRESS						
11								
12		SOURCE SAS ADDRESS						
19								
20	Reserved							
21	PATHWAY BLOCKED COUNT							
22	SCALE	(MSB)	ARBITRATION WAIT TIME					
23								(LSB)
24	(MSB)	Reserved						
27								(LSB)
28	(MSB)	CRC						
31								(LSB)

The PROTOCOL field indicates the protocol for the connection being requested and is defined in table 75.

Table 75 — Protocol

Code	Description
000b	SMP
001b	SSP
010b	STP
All others	Reserved

~~The destination device shall reject the connection request with OPEN_REJECT (PROTOCOL NOT SUPPORTED) if the PROTOCOL field is set to a value it does not support.~~

An INITIATOR bit set to one indicates the source device is acting as an initiator device. An INITIATOR bit set to zero indicates the source device is acting as a target device. If a target/initiator device sets the INITIATOR bit to

one, it shall operate only in its initiator role during the connection. If a target/initiator device sets the INITIATOR bit to zero, it shall operate only in its target role during the connection.

If a target/initiator device accepts an OPEN address frame with the INITIATOR bit set to one, it shall operate only in its target role during the connection. If a target/initiator device accepts an OPEN address frame with the INITIATOR bit set to zero, it shall operate only in its initiator role during the connection.

An initiator-only device shall reject connection requests from an initiator device (i.e., with the INITIATOR bit set to one) with OPEN_REJECT (PROTOCOL NOT SUPPORTED). A target-only device shall reject connection requests from a target device (i.e., with the INITIATOR bit set to zero) with OPEN_REJECT (PROTOCOL NOT SUPPORTED).

The CONNECTION RATE field indicates the connection rate being requested between the source and destination, and is defined in table 76. ~~The requested connection rate shall not exceed the slowest negotiated physical link rate along the pathway.~~

Table 76 — Connection rate

Code	Description
0h	1,5 Gbps
1h	3,0 Gbps
2h - Fh	Reserved

~~Every phy shall support the 1,5 Gbps connection rate at every physical link~~ rate.

When requesting a connection to a target port, an initiator port shall set the CONNECTION RATE field to a value supported by the pathway. For each wide link in the pathway, at least one physical link in the wide link shall support the connection rate. For each narrow link in the pathway, that physical link shall support the connection rate.

When requesting an SSP connection to an SSP initiator port, an SSP target port shall set the CONNECTION RATE field to the connection rate in effect when the command was received unless it has received an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED). See 7.12.2.2 for details on handling OPEN_REJECT (CONNECTION RATE NOT SUPPORTED).

The target port should send frames in an open connection to the initiator port regardless of whether the saved connection rate for that command matches the current connection rate; the target port should not close the connection just to reopen the connection at the saved connection rate.

When requesting an STP connection to an STP initiator port, an STP target port shall set the CONNECTION RATE field to the last value received in a connection request from the initiator port unless the STP target port has received an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED).

The FEATURES field shall be set to zero. The destination device shall reject the connection request with OPEN_REJECT (PROTOCOL NOT SUPPORTED) if the FEATURES field is set to a value it does not support.

~~The INITIATOR CONNECTION TAG field is used for SSP and STP connection requests to provide an initiator port an easier context lookup when the 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 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 target port. When requesting a connection to an initiator port, a target port shall set the INITIATOR CONNECTION TAG field to the value received in connection requests from the initiator port. An initiator port shall use the same INITIATOR CONNECTION TAG field value for all connection requests to the same target port, and shall only change the INITIATOR CONNECTION TAG field value when it has no commands outstanding to that target port. Targets are not required to check consistency of the INITIATOR CONNECTION TAG field in different connection requests from the same initiator port. SMP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh for SMP connection requests.

~~The destination device shall ignore the contents of reserved fields in the OPEN address frame.~~

The PATHWAY BLOCKED COUNT field indicates the number of times the port has retried this connection request due to receiving OPEN_REJECT (PATHWAY BLOCKED). The port shall not increment the PATHWAY BLOCKED COUNT value past FFh. If the port changes connection requests, it shall use a PATHWAY BLOCKED COUNT of 00h.

The SCALE bit and the ARBITRATION WAIT TIME field indicate how long the port transmitting the OPEN address frame has been waiting for a connection request to be accepted, and are defined in table 77. See 7.12.3 for details on arbitration fairness.

Table 77 — Arbitration wait time

SCALE	ARBITRATION WAIT TIME		
	Description	**Minimum value (0000h) meaning**	**Maximum value (7FFFh) meaning**
0b	Microseconds since the connection request was first made	0 µs	32 767 µs
1b	Milliseconds since the connection request was first made plus 32 768 µs	0 ms + 32 768 µs	32 767 ms + 32 768 µs

The DESTINATION SAS ADDRESS field indicates the SAS address of the port with which a connection is being requested.

The SOURCE SAS ADDRESS field indicates the SAS address of the port transmitting the OPEN address frame.

7.8 Identification and hard reset sequence

7.8.1 Overview

After the phy reset sequence has been completed indicating the physical link is using SAS rather than SATA, each phy shall transmit either:

a) an IDENTIFY address frame (see 7.7.2); or
b) a HARD_RESET.

Each phy shall also expect to receive an IDENTIFY address frame or a HARD_RESET from the phy to which it is attached. The combination of a phy reset sequence, an optional hard reset sequence, and an identification sequence is called a link reset sequence.

If a device supports more than one phy, it shall transmit the same SAS address on all phys for which it is capable of sharing within a port.

If a device detects the same SAS address incoming on different phys, it shall consider those phys part of the same wide port.

If a device detects different SAS addresses incoming on different physical links, it shall consider those physical links as independent physical links and consider those phys part of different ports.

If a device does not receive a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, it shall restart the phy reset sequence.

If a device receives an additional IDENTIFY address frame after receiving the first one, without an intervening phy reset sequence, it shall ignore it.

If a SAS phy receives a HARD_RESET, it shall be considered a reset event and cause a hard reset (see 4.4.2) of the port containing that phy.

7.8.2 Initiator device ~~specific~~ rules

After a link reset sequence, or after receiving a BROADCAST (CHANGE), the application client within an initiator device shall perform a discover process (see 4.6.11.5).

When this is done after a link reset sequence, this allows the application client within an initiator device to discover all the devices in the SAS domain. When this is done after a BROADCAST (CHANGE), this allows the application client within an initiator device to determine what has changed in the SAS domain.

The discover information may be used to select connection rates for connection requests.

If during the discover process (see 4.6.11.5) the application client within the initiator device detects a port with a SAS address it has already encountered, it has found a routing loop. It shall disable routing of destination SAS addresses through the expander port used to reach this previously encountered port to break the loop. The DISABLE EXPANDER ROUTE field in a SMP CONFIGURE ROUTE INFORMATION function request is used to disable the expander port of an expander device.

7.8.3 Fanout expander device specific rules

After completing the identify sequence, the expander connection manager within a fanout expander device shall be capable of routing connection requests to attached devices.

After a link reset sequence, or after receiving a BROADCAST (CHANGE), the application client within a fanout expander device that does not have a configurable expander route table shall follow the initiator device ~~specific~~ rules (see 7.8.2) to perform a discover process.

The expander connection manager of a fanout expander device that has a configurable expander route table is dependent on the completion of the discover process (see 4.6.11.5) for routing connection requests using the table routing method.

7.8.4 Edge expander device specific rules

After completing the identify sequence, the expander connection manager within an edge expander device shall be capable of routing connection requests to attached devices.

The expander connection manager of an edge expander device that has a configurable expander route table is dependent on the completion of the discover process (see 4.6.11.5) for routing connection requests using the table routing method.

7.8.5 Identification and hard reset (SL_IR) state machines

7.8.5.1 Overview

The identification and hard reset (SL_IR) state machines control the flow of dwords on a link that are associated with the identification and hard reset sequences ~~(see 7.8).~~ The state machines are as follows:

a) Transmit IDENTIFY or HARD_RESET (SL_IR_TIR) state machine (see 7.8.6.1);
b) Receive Identify Address Frame (SL_IR_RIF) state machine (see 7.8.6.2); and
c) Identification and hard reset control (SL_IR_IRC) state machine (see 7.8.6.3).

The SL_IR state machine sends the following parameters to the SAS link layer state machines (SL or XL):

a) Enable Disable SAS Link (Enable); and
b) Enable Disable SAS Link (Disable).

The SL_IR state machine sends the following parameters to the SL_IR transmitter:

a) Transmit IDENTIFY; and
b) Transmit HARD_RESET.

The SL_IR state machine receives the following parameters from the SL_IR receiver:

a) SOAF Received;
b) Data Dword Received;
c) EOAF Received; and
d) HARD_RESET Received.

Figure 67 shows the SL_IR state machines.

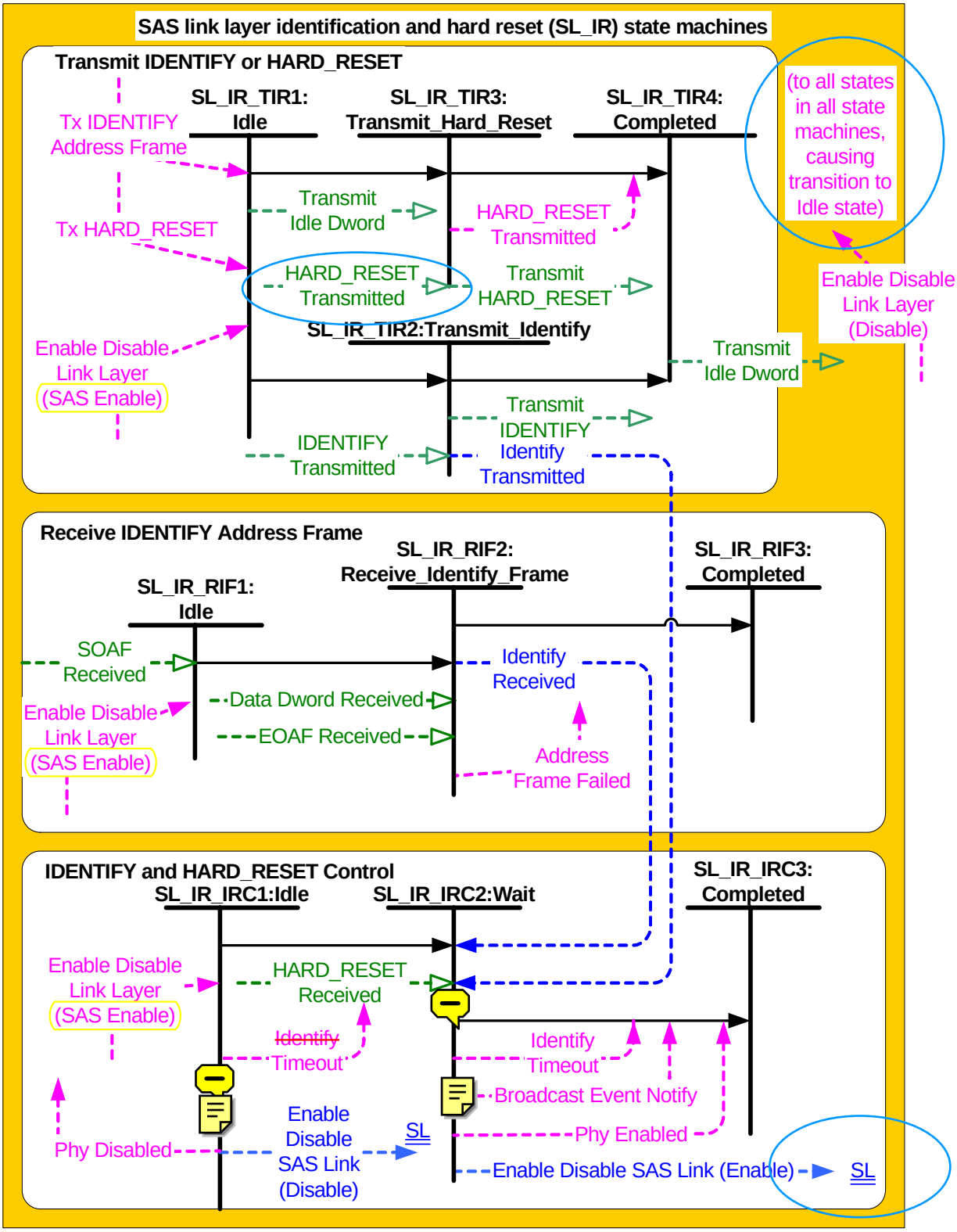


Figure 67 — SAS link layer identification and hard reset (SL_IR) state machines

7.8.6 SL_IR transmitter and receiver

When the SL_IR transmitter is requested to transmit a dword from any state within the SL_IR state machine, it shall transmit that dword. The following are dwords that may be transmitted by the SL_IR transmitter:

a) HARD_RESET;
b) SOAF/IDENTIFY address frame/EOAF; or
c) idle dword.

When the SL_IR transmitter has been requested to transmit an IDENTIFY address frame (i.e., SOAF, IDENTIFY address frame, and an EOAF) and a primitive is requested to be transmitted while the IDENTIFY address frame is being transmitted, the SL_IR transmitter shall not transmit the indicated primitive by inserting the primitive between the frames' dwords.

The SL_IR receiver shall ignore any primitive received inside an IDENTIFY address frame. In this case, a data dword shall be considered inside a frame when it is received after an SOAF and before an EOAF if the primitive is received after the 8th data dword following the SOAF.

7.8.6.1 Transmit IDENTIFY or HARD_RESET (SL_IR_TIR) state machine

7.8.6.1.1 Overview

The Transmit IDENTIFY or HARD_RESET (SL_IR_TIR) state machine's function is to transmit a single IDENTIFY address frame or HARD_RESET after the phy layer enables the link layer. The state machine contains the following states:

a) SL_IR_TIR1:Idle (see 7.8.6.1.2)(initial state);
b) SL_IR_TIR2:Transmit_Identify (see 7.8.6.1.3);
c) SL_IR_TIR3:Transmit_Hard_Reset (see 7.8.6.1.4); and
d) SL_IR_TIR4:Completed (see 7.8.6.1.5).

The state machine shall start in the SL_IR_TIR1:Idle state. The state machine shall transition to the SL_IR_TIR1:Idle state from any other state after receiving an Enable Disable Link Layer (Disable) confirmation.

~~This is the only state machine in the SL_IR state machines that transmits dwords on the physical link.~~

7.8.6.1.2 SL_IR_TIR1:Idle state

7.8.6.1.2.1 State description

This state shall repeatedly send Transmit Idle Dword to the SL_IR transmitter.

7.8.6.1.2.2 Transition SL_IR_TIR1:Idle to SL_IR_TIR2:Transmit_Identify

This transition shall occur when both:

a) an Enable Disable Link Layer (SAS Enable) confirmation is received; and
b) a Tx IDENTIFY Address Frame request is received.

7.8.6.1.2.3 Transition SL_IR_TIR1:Idle to SL_IR_TIR3:Transmit_Hard_Reset

This transition shall occur when both:

a) an Enable Disable Link Layer (SAS Enable) confirmation is received; and
b) a Tx HARD_RESET request is received.

7.8.6.1.3 SL_IR_TIR2:Transmit_Identify state

7.8.6.1.3.1 State description

This state shall send a Transmit IDENTIFY parameter to the SL_IR transmitter.

When this state receives an IDENTIFY Transmitted parameter, this state shall send an Identify Transmitted parameter to the IRC state machine.

7.8.6.1.3.2 Transition SL_IR_TIR2:Transmit_Identify to SL_IR_TIR4:Completed

This transition shall occur after this state has sent an Identify Transmitted parameter.

7.8.6.1.4 SL_IR_TIR3:Transmit_Hard_Reset state

7.8.6.1.4.1 State description

This state shall send a Transmit HARD_RESET parameter to the SL_IR transmitter.

When this state receives a HARD_RESET Transmitted parameter, this state shall send a HARD_RESET Transmitted confirmation to the management application layer.

7.8.6.1.4.2 Transition SL_IR_TIR3:Transmit_Hard_Reset to SL_IR_TIR3:Completed

This transition shall occur after sending a HARD_RESET Transmitted confirmation.

7.8.6.1.5 SL_IR_TIR4:Completed state

This state shall repeatedly send the Transmit Idle Dword parameter to the SL_IR transmitter.

7.8.6.2 Receive IDENTIFY Address Frame (SL_IR_RIF) state machine

7.8.6.2.1 Overview

The Receive Identify Address Frame (SL_IR_RIF) state machine receives an IDENTIFY address frame from the physical link and checks the IDENTIFY address frame to determine if the frame should be accepted or discarded by the link layer.

The state machine contains the following states:

a) SL_IR_RIF1:Idle (see 7.8.6.2.2)(initial state);
b) SL_IR_RIF2:Receive_Identify_Frame (see 7.8.6.2.3); and
c) SL_IR_RIF3:Completed (see 7.8.6.2.4).

The state machine shall start in the SL_IR_RIF1:Idle state. The state machine shall transition to the SL_IR_RIF1:Idle state from any other state after receiving an Enable Disable Link Layer (Disable) confirmation.

7.8.6.2.2 SL_IR_RIF1:Idle state

7.8.6.2.2.1 State description

This state waits for an SOAF to be received from the physical link, indicating an address frame is arriving.

7.8.6.2.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame

This transition shall occur when both:

a) an Enable Disable Link Layer (SAS Enable) confirmation is received; and
b) an SOAF Received parameter is received.

7.8.6.2.3 SL_IR_RIF2:Receive_Identify_Frame state

7.8.6.2.3.1 State description

This state receives the dwords of an address frame and the EOAF.

After receiving the frame, it shall check if it is a correct IDENTIFY address frame.

This state shall accept an IDENTIFY address frame and send an Identify Received parameter to the IRC state machine if:

a) the ADDRESS FRAME TYPE field is set to Identify;
b) the number of bytes between the SOAF and EOAF is 32; and
c) the CRC field contains a valid CRC.

Otherwise, this state shall discard the IDENTIFY address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an illegal IDENTIFY address frame was received.

7.8.6.2.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed

This transition shall occur after receiving an EOAF and sending the Identify Received parameter or Address Frame Failed confirmation.

7.8.6.2.4 SL_IR_RIF3:Completed state

This state does nothing except wait for the Enable Disable Link Layer (Disable) confirmation.

7.8.6.3 Identification and hard reset control (SL_IR_IRC) state machine

7.8.6.3.1 Overview

The SL_IR_IRC state machine's function is to ensure IDENTIFY address frames have been both received and transmitted before enabling the rest of the link layer, and notify the link layer if a HARD_RESET is received before an IDENTIFY address frame has been received.

The IRC state machine contains the following states:

a) SL_IR_IRC1:Idle (see 7.8.6.3.2)(initial state);
b) SL_IR_IRC2:Wait (see 7.8.6.3.3); and
c) SL_IR_IRC3:Completed (see 7.8.6.3.4).

The state machine shall start in the SL_IR_IRC1:Idle state. The state machine shall transition to the SL_IR_IRC1:Idle state from any other state after receiving an Enable Disable Link Layer (Disable) confirmation.

7.8.6.3.2 SL_IR_IRC1:Idle state

7.8.6.3.2.1 State description

This state waits for the link layer to be enabled. This state shall:

a) send an Enable Disable SAS Link (Disable) parameter to the SL state machine (see 7.13) or XL state machine (see 7.14) halting any link layer activity; and
b) send a Phy Disable confirmation to the port layer and the management application layer indicating that the phy is not ready for use.

7.8.6.3.2.2 Transition SL_IR_IRC1:Idle to SL_IR_IRC2:Wait

This transition shall occur when an Enable Disable Link Layer (SAS Enable) confirmation is received.

7.8.6.3.3 SL_IR_IRC2:Wait state

7.8.6.3.3.1 State description

This state ensures that an IDENTIFY address frame has been received by the Receive IDENTIFY Address Frame state machine and that a IDENTIFY address frame has been transmitted by the Transmit IDENTIFY or HARD_RESET state machine before enabling the rest of the link layer. The IDENTIFY address frames may be transmitted and received on the link in any order.

After this state receives an Identify Transmitted parameter ~~from the Transmit IDENTIFY or HARD_RESET state machine,~~ it shall initialize a receive identify timeout timer to 1 ms. If an Identify Received parameter is received ~~from the Receive IDENTIFY Address Frame state machine~~ before the identify timeout timer is exceeded, this state shall:

a) send an Identify Sequence Complete confirmation to the management application layer ~~indicating that the identify sequence has completed,~~ with arguments carrying the contents of the incoming IDENTIFY address frame;
b) in an expander device, send a Broadcast Event Notify (Identification Sequence Complete) confirmation to the expander function;
c) send an Enable Disable SAS Link (Enable) parameter to the SL state machine (see 7.13) in initiator devices and target devices or the XL state machine (see 7.14) in expander devices indicating that the rest of the link layer may start operation; and
d) send a Phy Enabled confirmation to the port layer and the management application layer indicating that the phy is ready for use.

If the identify timeout timer is exceeded before an Identify Received parameter 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 before an Identify Received parameter is received, this state shall send a HARD_RESET Received confirmation to the management application layer indicating that a HARD_RESET occurred.

If this state receives a HARD_RESET after an Identify Received parameter is received, the HARD_RESET shall be ignored.

7.8.6.3.3.2 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 Identify Sequence Complete confirmation to the management application layer.

7.8.6.3.4 SL_IR_IRC3:Completed state

This state does nothing except wait for the Enable Disable Link Layer (Disable) confirmation.

7.9 Power management

~~SATA interface power management is not supported in SAS.~~

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 target port while an STP connection is not open, it shall not forward it to any initiator port and shall reply with a SATA_PMNAK. If the primitives arrives while an STP connection is open, it may forward it to the STP initiator port.

~~An expander device may intercept SATA register FIS transfers and hide any SATA target device support for interface power management in the SCONTROL and SSTATUS registers.~~

SCSI idle and standby power conditions, implemented with the START STOP UNIT command (see SBC-2 ~~and RBC)~~ and the Power Condition mode page (see SPC-3), may be supported by SSP initiator ports and target ports as described in 10.1.8.

ATA idle and standby power modes, implemented with the IDLE, IDLE IMMEDIATE, STANDBY, STANDBY IMMEDIATE, and CHECK POWER MODE commands (see ATA/ATAPI V1), may be supported by STP initiator ports. The ATA sleep power mode, implemented with the SLEEP command, shall not be used.

7.10 Near-end analog loopback test

A phy performing the near end analog loopback test shall internally route its transmitter to its receiver. The phy shall not transmit anything on the link and shall ignore all incoming signals from the link, including OOB signals.

Figure 68 shows the near-end analog loopback test mode.

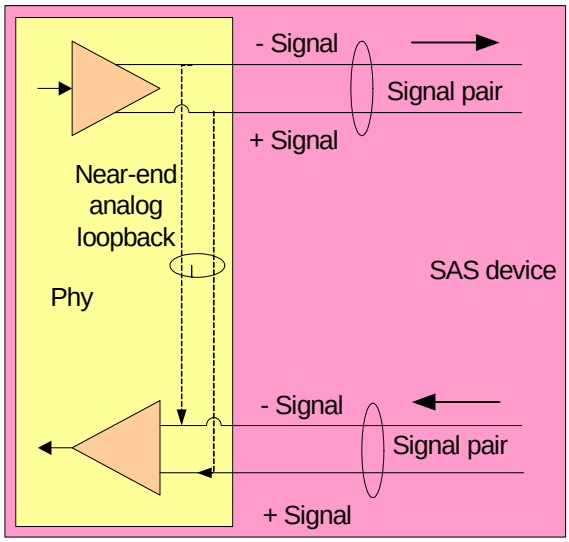


Figure 68 — Test modes

This test mode may be invoked in an initiator device using vendor-specific means.

An application client may request an expander device set one of its phys into near-end analog loopback test mode using the SMP PHY_CONTROL function (see 10.3.1.9). After enabled, all data sent to that expander device shall be routed to the expander phy to be looped back. Expander devices in the pathway treat this as an open SMP connection. The phy shall be operating at a valid physical link rate before being switched into near-end analog loopback test mode. If the phy is not operating at a valid physical link rate, the PHY_CONTROL function shall fail.

Once the test is completed, the application client shall transmit a BREAK or CLOSE or have the initiator phy start a phy reset sequence. The expander device shall disable the loopback mode and start a phy reset sequence on that phy. The phy device shall loopback each of the BREAK or CLOSE primitives before the loopback mode is disabled.

7.11 Domain changes

After power on or receiving BROADCAST (CHANGE), SAS initiator ports should scan the domain with a discover process (see 4.6.11.5) to search for initiator devices, expander devices and target devices.

The expander device shall transmit BROADCAST (CHANGE) from at least one phy in each expander port other than the expander port that is the cause for transmitting BROADCAST (CHANGE).

Expander devices shall transmit BROADCAST (CHANGE) for the following reasons:

a) after an expander phy has lost dword synchronization;
b) after the link reset sequence completes; and
c) after the expander device receives BROADCAST (CHANGE).

BROADCAST (CHANGE) may be sent by initiator ports to force other initiator ports and expander ports to re-run the discovery process, but should not be sent by target ports.

An initiator port that detects BROADCAST (CHANGE) shall follow the initiator device ~~specific~~ rules (see 7.8.2) to discover the topology.

A fanout expander device that detects BROADCAST (CHANGE) shall follow the fanout device ~~specific~~ rules (see 7.8.3) to discover the topology.

An edge expander device that detects BROADCAST (CHANGE) shall follow the edge device ~~specific~~ rules (see 7.8.4).

See 10.3.1.2 for details on counting BROADCAST (CHANGE) generation in an expander device.

7.12 Connections

7.12.1 Connection overview

A connection is opened between an initiator phy and a target phy before communication can begin.

SSP initiator ports open SSP connections to transmit SCSI commands, task management functions, or transfer data. SSP target ports open SSP connections to transfer data or transmit status.

SMP initiator ports open SMP connections to transmit SMP requests and receive SMP responses.

STP initiator ports open STP connections to transmit SATA frames. Expander devices open STP connections on behalf of SATA target ports to transmit SATA frames.

The OPEN address frame is used to request that a connection be opened. AIP, OPEN_ACCEPT and OPEN_REJECT are the responses to an OPEN address frame. BREAK is used to abandon connection requests and to unilaterally break a connection. CLOSE is used for orderly closing a connection.

Connections use a single pathway from the initiator phy to the target phy. While a connection is open, only one pathway shall be used for that connection.

For STP connections, connections may be between the initiator phy and the expander device attached to a SATA target device. The SATA target device 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 7.7.3) 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 decides to abandon the connection request with BREAK.

After transmitting an OPEN address frame, the source phy shall initialize an open timeout timer to 1 ms and start the timer. Whenever an AIP is received, the source phy shall reinitialize and restart the timer. Source phys are not required to enforce a limit on the number of AIPs received before abandoning the connection request, but they may do so. When any connection response is received, the source phy shall reinitialize the timer. If the timer expires before a connection response is received, the source phy may assume the destination port does not exist and shall transmit BREAK to abandon the connection request.

The OPEN address frame flows through expander devices onto intermediate physical links. If none of the prospective intermediate physical links does not support the requested connection rate, the expander device shall return OPEN_REJECT (CONNECTION RATE NOT SUPPORTED). If the OPEN address frame reaches the destination, it shall return either OPEN_ACCEPT or OPEN_REJECT. Rate matching is used on any physical links in the pathway with negotiated physical link rates that are faster than the requested connection rate (see clause 7.15).

7.12.2.2 Connection request responses

Table 78 shows the ~~possible~~ responses to an OPEN address frame.

Table 78 — Connection request responses

Response	Description
AIP	Arbitration in progress. While the expander devices are trying to open a connection to the selected destination port, it returns an AIP to the source port. The source port shall set its open timeout timer back to its initial value when it receives an AIP. AIP is sent by an expander device while it is internally arbitrating for access to an expander port.
OPEN_ACCEPT	Connection request accepted. This is sent by the destination port.
OPEN_REJECT	Connection request rejected. This is sent in response by the destination port or by an expander device. The different versions are described in 7.1.4.11.
OPEN address frame	If AIP has been previously detected, this indicates an overriding connection request. If AIP has not yet been detected, this indicates two connection requests crossing on the physical link.
BREAK	The destination port or expander port may reply with BREAK indicating the connection is not being established.
No response; timer expires	The source port shall abandon the connection request by transmitting BREAK.

After receiving an OPEN_REJECT that indicates a retry may be performed (see table 62), the source port shall wait a retry delay of 15 µs before issuing another connection request to the same destination port.

After an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) has been received, the target port shall set the connection rate for future requests for that I_T_L_Q to:

a) the last value received in a connection request from the initiator port;
b) 1,5 Gbps; or
c) the connection rate in effect when the command was received.

An initiator phy or target phy shall accept an incoming connection request and transmit OPEN_ACCEPT if the DESTINATION SAS ADDRESS field matches its SAS address and the PROTOCOL field matches one of its supported protocols.

7.12.3 Arbitration fairness

SAS supports least-recently used arbitration fairness.

Each initiator port, target port, and expander port may include an arbitration wait timer counting the time from when the port makes a connection request until its request is granted. The arbitration wait timer shall count in microseconds from 0 µs to 32 767 µs and in milliseconds from 32 768 µs to 32 767 ms + 32 768 µs. The timer shall stop incrementing when its value reaches 32 767 ms + 32 768 µs.

Initiator ports and target ports should implement arbitration wait timers. If implemented, they shall set the timer to zero and start the timer when they transmit the first OPEN address frame for the connection request. When the 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 timer.

An initiator port or target port may be unfair, setting the ARBITRATION WAIT TIME field in the OPEN address frame (see 7.7.3) to a higher value than its arbitration wait timer indicates. However, unfair ports shall not set the SCALE bit to one; this limits the amount of unfairness ~~and helps prevent livelocks.~~

Expander ports shall maintain arbitration wait timers. The expander port that receives an OPEN address frame shall set the timer to the value of the incoming ARBITRATION WAIT TIME field and start the 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 timer maintained by the incoming expander port.

A port shall stop the timer and set it to zero when it wins arbitration, receiving either OPEN_ACCEPT or OPEN_REJECT. A port shall stop the timer ~~and set it to zero~~ when it loses arbitration to a connection request that satisfies its arbitration request, receiving an OPEN address frame from the destination port with matching PROTOCOL and CONNECTION RATE fields.

When arbitrating for access to an outgoing expander port, the expander device shall select the connection request from the expander port with the largest arbitration wait time value. If the largest arbitration wait times are identical, then the connection request with the largest SOURCE SAS ADDRESS shall win arbitration.

If two connection requests pass on a physical link, the winner shall be determined by comparing OPEN address frame field values in this order:

1) largest ARBITRATION WAIT TIME field value; and
2) largest SOURCE SAS ADDRESS field value.

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 expander connection manager. An expander phy shall set the timer to the partial pathway timeout value (see 7.12.3.1.3) and run the timer whenever the expander connection manager provides confirmation to an expander phy that all phys within the requested destination port are blocked waiting on partial pathways.

NOTE 22 The partial pathway timeout value allows flexibility in specifying how long an expander device waits before attempting pathway recovery. The default value 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 must wait after receiving OPEN_REJECT (PATHWAY BLOCKED) before it may retry 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.

See 7.7.3 for details on the OPEN address frame and the ARBITRATION WAIT TIME field.

7.12.3.1 Arbitration and resource management in an expander device

7.12.3.1.1 Arbitration overview

The expander connection manager shall arbitrate and assign or deny path resources for connection attempts requested by each expander phy in response to receiving valid OPEN address frames.

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.

An expander phy requests path resources using the Request Path (attributes) expander primitive.

The expander connection manager uses the Arb Won, Arb Lost, and Arb Reject confirmations to provide arbitration confirmation back to the requesting phy.

Each path request contains the Arbitration Wait Time and the Source SAS Address arguments from the received OPEN address frame.

If two path requests contend, the winner shall be determined by comparing OPEN address frame field values in this order:

1) largest Arbitration Wait Time;
2) largest Source SAS Address; and
3) largest Connection Rate.

The expander connection shall generate the Arb Reject confirmation when any of the following conditions are met:

a) the Request Path (attributes) does not map to a valid phy;
b) the Request Path (attributes) specifies an unsupported connection rate; or
c) the Request Path (attributes) specifies a destination port which contains at least one partial pathway and pathway recovery rules require this connection request to release path resources.

The expander connection manager shall generate the Arb Lost confirmation when all of the following conditions are met:

a) the Request Path (attributes) maps to an available phy at a supported connection rate; and
b) the destination phy of this connection request has received a higher priority OPEN address frame with this phy as its destination (this case occurs when two phys both receive an OPEN address frame destined for each other, the expander connection manager shall provide arbitration lost confirmation to the phy that received the lowest priority OPEN address frame).

The expander connection manager shall generate the Arb Won confirmation when all of the following conditions are met:

a) the Request Path (attributes) maps to an available phy at a supported connection rate; and
b) no higher priority connection requests are present with this phy as the destination.

7.12.3.1.2 Arbitration status

Arbitration status shall be conveyed between expander devices and by expander devices to SAS endpoints using AIP primitives. This status is used to monitor the progress of connection attempts and to facilitate pathway recovery as part of deadlock avoidance.

The arbitration status of an expander phy is set to the last value of AIP received.

7.12.3.1.3 Partial Pathway Timer

The expander connection manager maintains a Partial Pathway Timeout timer (PPT timer) for each phy within the expander device. This timer is used for pathway recovery when potential deadlock conditions exist. The default value for the partial pathway timeout value shall be 7 µs with a minimum configurable value of 1 µs and a maximum configurable value of 15 µs.

The expander connection manager shall decrement the PPT timer once per microsecond, ~~until reaching zero,~~ when the following conditions are met:

a) there are no unallocated phys within a requested destination port available to complete the connection; and
b) at least one phy within the requested destination port contains a blocked partial pathway.

When either of the conditions above are not met, the expander connection manager shall hold the PPT timer at an initial value set to the partial pathway timeout value.

7.12.3.1.4 Pathway Recovery

Pathway recovery provides a means to abort connection requests in order to prevent deadlock using Pathway Recovery Priority comparisons. Pathway Recovery Priority is defined as the concatenation of the PATHWAY BLOCKED COUNT and the SOURCE SAS ADDRESS fields within the OPEN address frame of a blocked connection request. Table 79 defines the components of Pathway Recovery Priority.

Table 79 — Pathway Recovery Priority

Bits (MSB) 71-64	Bits 63-0 (LSB)
PATHWAY BLOCKED COUNT	SOURCE SAS ADDRESS

When the PPT timer for an arbitrating phy expires (i.e., reaches a value of zero), the expander connection manager shall determine whether to continue the connection request or to abort the connection request ~~as follows:~~

The expander connection manager shall instruct the arbitrating phy to reject the connection request by transmitting OPEN_REJECT (PATHWAY_BLOCKED) when the PPT timer expires and the Pathway Recovery Priority of the arbitrating phy (i.e., the phy requesting the connection) is less than the Pathway Recovery Priority of all phys within the destination port with an arbitration status of WAITING_ON_PARTIAL. ~~If the PATHWAY_BLOCKED COUNT fields match, then the comparison~~ is ~~effectively~~ done only with the SOURCE SAS ADDRESS fields.

7.12.4 Expander devices and connection requests

7.12.4.1 All expander devices

Before an expander device transmits AIP, it may have transmitted an OPEN address frame on the same physical link. Fairness dictates which OPEN address frame will win (see 7.12.3).

After an expander device transmits an AIP, it shall not transmit an OPEN address frame unless it has higher priority than the incoming connection request.

Expander devices shall transmit no more than three AIPs consecutively without transmitting an idle dword. Expander devices shall transmit at least one AIP every 128 dwords.

Expander devices shall transmit AIP immediately after receiving an OPEN address frame.

7.12.4.2 Edge expander devices

When an edge expander device receives a connection request, it shall compare the destination SAS address to the SAS addresses of the devices to which each of its phys is attached. For all phys which have table routing attributes and are attached to edge expander devices, it shall compare the destination SAS addresses to all the enabled SAS addresses in the expander route table.

If it finds a match in one or more phys, then the expander device shall arbitrate for access to one of the matching phys and forward the connection request.

If it does not find a match, but at least one phy has the subtractive routing attribute and is attached to an expander device (either edge or fanout), and the request did not come from that expander device, the connection request shall be forwarded to the expander device through any of the subtractive routing phys.

If it does not find a match and no subtractive routing phy is available, the edge expander device shall reply with OPEN_REJECT (NO DESTINATION).

If the destination phy is in the same expander port as the source phy, the edge expander device shall reply with OPEN_REJECT (BAD DESTINATION). When two edge expander device sets are attached (by subtractive ports), ~~this means~~ requests to non-existent devices return OPEN_REJECT (BAD DESTINATION) rather than OPEN_REJECT (NO DESTINATION). When a fanout expander device is in the domain, an OPEN_REJECT (NO DESTINATION) is returned.

Table 80 shows the routing table maintained by an edge expander device.

Table 80 — Edge expander device routing table

Phy	SAS address	Expander device attached
Internal	Edge expander device SAS address	Yes or no
Phy 0	Attached SAS address	Yes or no
Phy 1	Attached SAS address	Yes or no
…	…	…
Phy n	Attached SAS address	Yes or no

7.12.4.3 Fanout expander devices

When a fanout expander device receives a connection request, it shall compare the destination SAS address to the SAS addresses of the devices to which each of its phys is attached. For all phys which are attached to edge expander devices, it shall compare the destination SAS addresses to all the enabled SAS addresses in the expander route table.

If it finds a match in one or more phys, it shall arbitrate for access to one of the matching phys and forward the connection request.

If it does not find a match, it shall reply with OPEN_REJECT (NO DESTINATION). If the destination phy is in the same expander port as the source phy, it shall reply with OPEN_REJECT (BAD DESTINATION).

7.12.5 Abandoning a connection request

BREAK may be used to abandon a connection request. The source port shall transmit a BREAK after the open timer expires or if it chooses to abandon its request for any other reason.

~~After transmitting BREAK, the source port shall initialize a break timeout timer to 1 ms and start the timer. If the timer expires before a break response is received, the source port may assume the physical link is unusable.~~

Table 81 lists the ~~possible~~ responses to a BREAK sent before a connection response has been received.

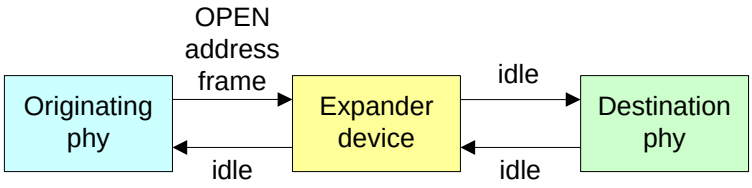
Table 81 — Abandon connection request responses

Response	Description
BREAK	This confirms that the connection request has been abandoned.
Open response listed in 7.12.2	The BREAK was too late and an open response arrived late. The originator shall honor this as a response to the open request it was attempting to abandon.
No response and timer expires	The originating port shall assume the connection request has been abandoned.

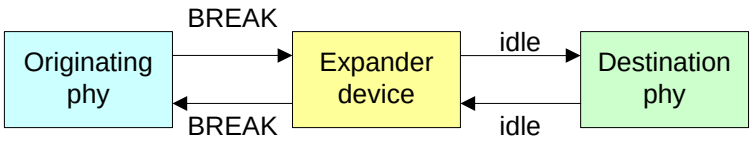
When a port sourcing a BREAK is attached to an expander device, the BREAK response to the source port is generated by the expander port to which the source port is attached, not the target port. If the expander device has transmitted a connection request to the destination, it shall also transmit BREAK to the destination. If the expander device has not transmitted a connection request to the destination, it shall not transmit BREAK to

the destination. The expander device shall transmit BREAK back to the originating port after it has ensured that an open response will not occur. Figure 69 shows an example of BREAK usage.

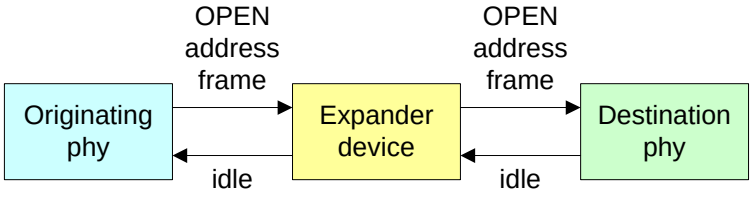
Case 1: OPEN address frame has not propogated through the expander device:



Case 1 result: BREAK only on originating device physical link



Case 2: OPEN address frame has propogated through the expander device:



Case 2 result: BREAK on originating device's physical link, then on destination device's physical link

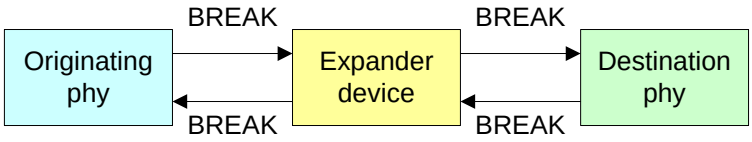


Figure 69 — BREAK usage

Figure 70 shows the sequence for a connection request that times out.

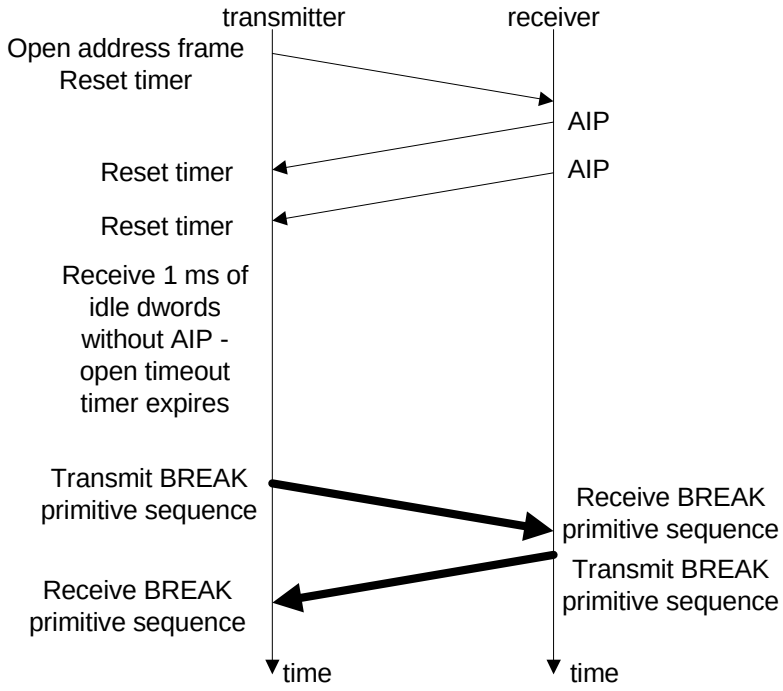


Figure 70 — Connection request timeout example

If a port detects a timeout while closing a connection (e.g., while exchanging DONEs in SSP, or while exchanging CLOSEs in any protocol) it may transmit a BREAK to break the connection.

7.12.6 Breaking a connection

A BREAK may also be used to break a connection, in cases where CLOSE is not available. After transmitting BREAK, the originating port shall ignore all incoming dwords except for BREAKs.

Table 82 lists the ~~possible~~ responses to a BREAK sent after a connection has been established.

Table 82 — Break connection responses

Response	Description
BREAK	This confirms that the connection has been broken.
No response and timer expires	The originating port shall assume the connection has been broken.

In addition to a BREAK, a connection may be broken as the result of an error or disconnection at the PHY layer.

In addition to the actions described in this subclause and in 7.12.5, the following shall be the responses by an SSP device to a broken connection:

a) 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;
b) Transmitted frames for which an ACK has been received prior to a broken connection shall be considered successfully transmitted; and
c) Transmitted frames for which an ACK or NAK has not been received prior to a broken connection shall be considered not successfully transmitted.

7.12.7 Closing a connection

CLOSE is used to close a connection of any protocol. The originating port for the close may have been either the source or destination port ~~when the connection was opened.~~

When an SSP device has both sent and received DONE, it shall transmit a CLOSE (NORMAL) (see 7.16.6).

When an STP initiator port or an expander device decides to close an STP connection, it shall transmit a CLOSE (NORMAL) or CLOSE (CLEAR AFFILIATION) (see 7.17.3).

An STP initiator port may issue CLOSE (CLEAR AFFILIATION) in place of a CLOSE (NORMAL) to cause the expander device on the far end of the STP connection to clear the affiliation between the STP initiator and the SATA phy at the other end of the STP connection, leaving the phy available for another STP initiator to establish a connection. This behavior is in addition to the normal effect of closing the connection. If an expander that supports attachment of a SATA target receives CLOSE (CLEAR AFFILIATION), the expander shall clear the affiliation between the STP initiator that sent the CLOSE (CLEAR AFFILIATION) and the phy at the other end of the connection being closed.

No additional primitives for the connection shall follow the CLOSE. Expander devices shall close the full-duplex connection upon detecting a CLOSE in each direction.

When a port has both transmitted and received CLOSE, it shall consider the connection closed.

Figure 71 shows the sequence for a closing an SSP connection.

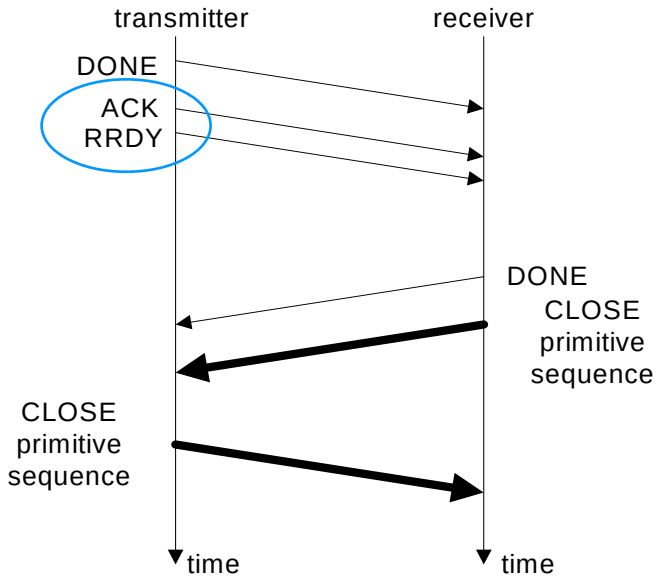


Figure 71 — Closing an SSP connection example

7.13 SAS link layer state machine for initiator phys and target phys (SL)

7.13.1 Overview

The SAS link layer (SL) state machine controls connections, handling both connection requests (OPEN address frames), CLOSEs, and BREAKs. The state machine contains the following states:

a) SL0:Idle (see 7.13.3)(initial state);
b) SL1:ArbSel (see 7.13.4);
c) SL2:Selected (see 7.13.5);
d) SL3:Connected (see 7.13.6);
e) SL4:DisconnectWait (see 7.13.7);
f) SL5:BreakWait (see 7.13.8); and
g) SL6:Break (see 7.13.9).

The state machine shall start in the SL0:Idle state. The state machine shall transition to the SL0:Idle state from any other state after receiving an Enable Disable SAS Link (Disable) parameter ~~from the SL_IR state machines~~ (see 7.8.5).

The SL state machine receives the following parameters ~~from the SSP, STP, and SMP link layer state machines:~~

a) Request Break; and
b) Request Close.

The SL state machine sends the following parameters to the SSP, STP, and SMP link layer state machines:

a) Enable Disable SSP;
b) Enable Disable STP; and
c) Enable Disable SMP.

The SL state machine receives the following parameters ~~from the SL_IR state machines~~ (see 7.8.5):

a) Enable Disable SAS Link (Enable); and
b) Enable Disable SAS Link (Disable).

Unless otherwise stated within the state description, all disparity errors, illegal characters, and unexpected primitives (i.e., any primitive not described in the description of the SL state) received within any SL state shall be ignored and idle dwords shall be transmitted.

Any detection of an internal error shall cause the SL state machine to transition to the SL5:BreakWait state.

Figure 72 describes the SAS link layer state machine.

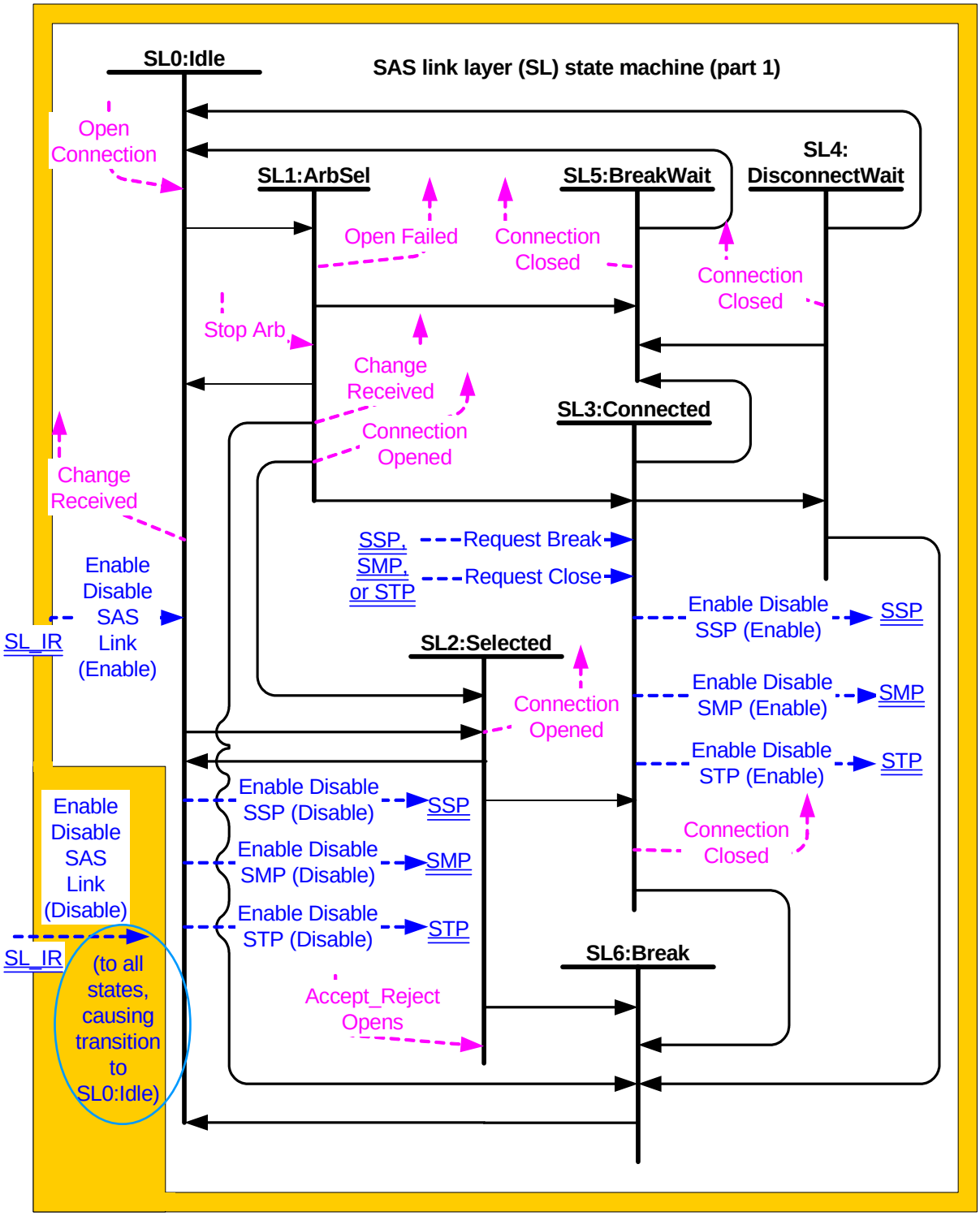


Figure 72 — SAS link layer (SL) state machine (part 1)

Figure 73 describes the parameters sent to the SL transmitter and received from the SL receiver.

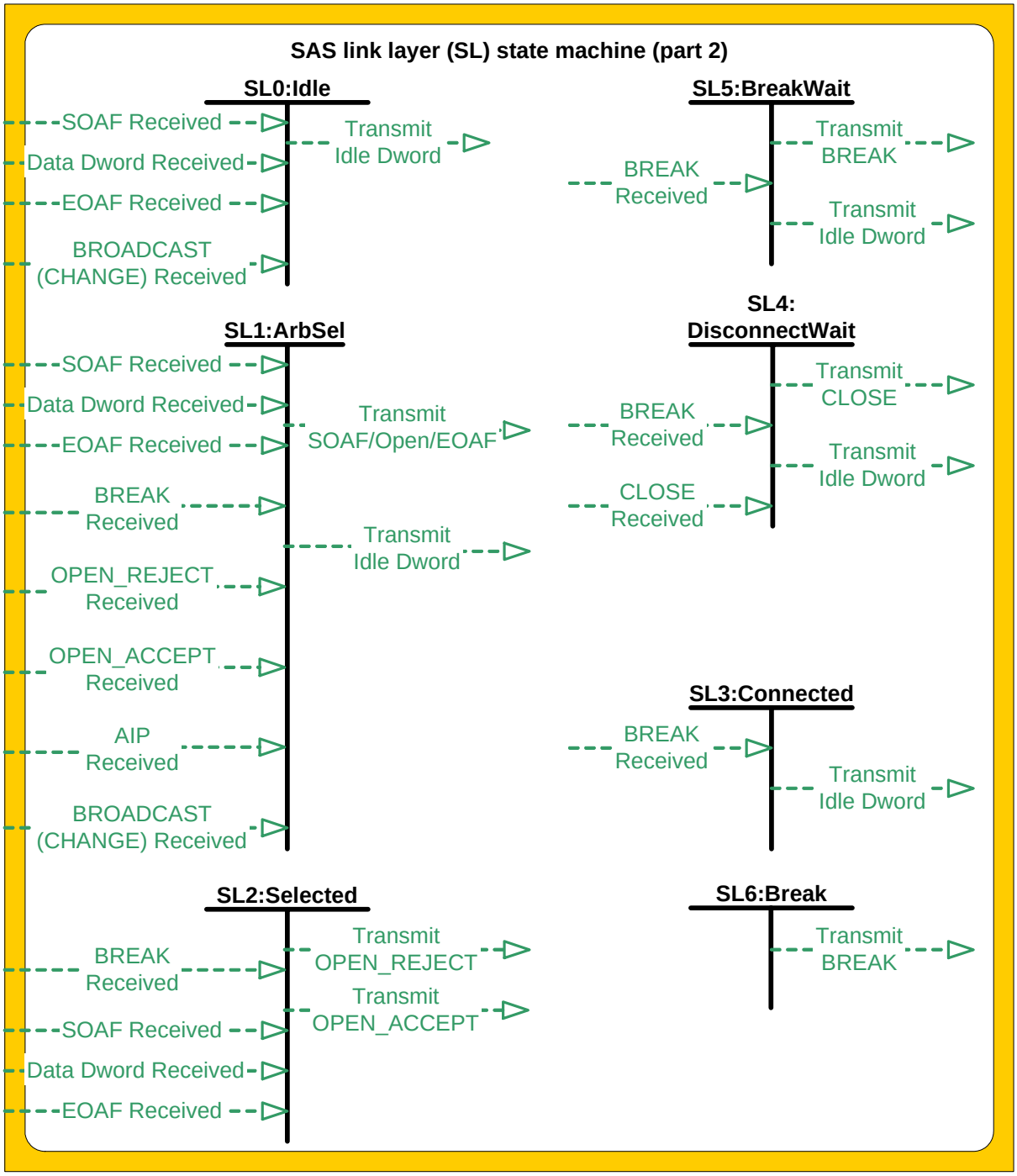


Figure 73 — SAS link layer (SL) state machine (part 2)

7.13.2 SL transmitter and receiver

When the SL transmitter is requested to transmit a dword from any state within the SAS link layer state machine, it shall transmit that dword. If there are multiple requests to transmit primitives, the following priority should be followed when selecting the primitives:

1) BREAK;
2) CLOSE;
3) OPEN_ACCEPT or OPEN_REJECT;
4) SOAF/OPEN address frame/EOAF; then

5) idle dword.

When the SL transmitter has been requested to transmit a OPEN address frame (i.e., SOAF, the data dwords of the OPEN address frame, and an EOAF) and a primitive is requested to be transmitted while the frame is being transmitted, the SL transmitter shall not transmit the indicated primitive by inserting the primitive between the frames' dwords.

The SL receiver shall ignore any primitive received inside an OPEN address frame. In this case, a dword shall be considered inside a frame when it is received after an SOAF and before an EOAF if the primitive is received after the 8th data dword following the SOAF.

7.13.3 SL0:Idle state

7.13.3.1 State description

The SL0:Idle state is the initial state and is the state that is used when the SL state machine is activated and there is no active connection.

Upon entry to this state, this state shall send Enable Disable SSP (Disable), Enable Disable SMP (Disable), and Enable Disable STP (Disable) parameters to the SSP, SMP, and STP link layer state machines.

After an Enable Disable SSP Link (Enable) confirmation is received this state shall send an Enable Disable SSP Link (Enable) confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SL transmitter (see 7.3).

If a BROADCAST (CHANGE) Received parameter is received, this state shall send a Change Received confirmation to the management layer.

An address frame (i.e., all the data dwords between an SOAF and EOAF) shall be discarded if any of the following conditions are true:

a) the number of data dwords between the SOAF and EOAF is not 8 data dwords; or
b) there is a CRC error.

If consecutive SOAF Received parameters are received without an intervening EOAF Received parameter (i.e., SOAF, data dwords, SOAF, data dwords, and EOAF instead of SOAF, data dwords, EOAF, SOAF, data dwords, and EOAF) then this state shall discard all data dwords between those SOAFs.

If the frame is discarded then no further action is taken by this state.

7.13.3.2 Transition SL0:Idle to SL1:ArbSel

This transition shall occur after both an Enable Disable SAS Link (Enable) confirmation is received and an Open Connection request is received. The Open Connection request includes these arguments:

a) source SAS address;
b) destination SAS address;
c) protocol;
d) arbitration wait time;
e) link rate;
f) initiator bit; and
g) initiator connection tag.

7.13.3.3 Transition SL0:Idle to SL2:Selected

This transition shall occur after both an Enable Disable SAS Link (Enable) confirmation is received and a valid OPEN address frame is received.

A valid OPEN address frame is 8 data dwords long and has no CRC error (see 7.7.3).

7.13.4 SL1:ArbSel state

7.13.4.1 State description

This state is used to make a connection request.

This state shall:

1) request an OPEN address frame be transmitted by sending an Transmit SOAF/open/EOAF parameter to the SL transmitter with the address frame fields set to the arguments received with the Open Connection request;
2) initialize an open timeout timer to 1 ms, start the open timeout timer; and
3) request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SL transmitter.

This state shall not respond to incoming BREAKs, OPEN_REJECTs, and OPEN_ACCEPTs until after the OPEN address frame has been transmitted.

If a BROADCAST (CHANGE) Received parameter is received this state shall send a Change Received confirmation to the management layer.

If an AIP Received parameter is received after requesting the OPEN address frame be transmitted, the open timeout timer shall be set to its initial value of 1 ms and resume counting. The state machine shall not enforce a limit on the number of AIPs received.

An address frame (i.e., all the data dwords between an SOAF and EOAF) shall be discarded if any of the following conditions are true:

a) the number of data dwords between the SOAF and EOAF is not 8 data dwords; or
b) there is a CRC error.

If consecutive SOAF Received parameters are received without an intervening EOAF Received parameter (i.e., SOAF, data dwords, SOAF, data dwords, and EOAF instead of SOAF, data dwords, EOAF, SOAF, data dwords, and EOAF) then this state shall discard all data dwords between those SOAFs.

c) If the frame is discarded then no further action is taken by this state relating to the invalid address frame.

7.13.4.2 Transition SL1:ArbSel to SL0:Idle

This transition shall occur after receiving an OPEN_REJECT Received parameter and after sending one of the following confirmations to the port layer:

a) If an OPEN_REJECT (NO DESTINATION) parameter is received this state shall send an Open Failed (No Destination) confirmation to the port layer;
b) If an OPEN_REJECT (BAD DESTINATION) parameter is received this state shall send an Open Failed (Bad Destination) confirmation to the port layer;
c) If an OPEN_REJECT (WRONG DESTINATION) parameter is received this state shall send an Open Failed (Wrong Destination) confirmation to the port layer;
d) If an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) parameter is received this state shall send an Open Failed (Connection Rate Not Supported) confirmation to the port layer;
e) If an OPEN_REJECT (PROTOCOL NOT SUPPORTED) parameter is received this state shall send an Open Failed (Protocol Not Supported) confirmation to the port layer;
f) If an OPEN_REJECT (RETRY) parameter is received this state shall send an Open Failed (Retry) confirmation to the port layer; or
g) If an OPEN_REJECT (PATHWAY BLOCKED) parameter is received this state shall send an Open Failed (Pathway Blocked) confirmation to the port layer.

7.13.4.3 Transition SL1:ArbSel to SL2:Selected

This transition shall occur:

a) If one or more AIP Received parameters have been received before a valid OPEN address frame is received. The incoming OPEN address frame shall then override the outgoing OPEN address frame; or
b) If no AIP Received parameters have been received before a valid OPEN address frame 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 use the value of the arbitration wait time argument to the Open Connection request for the outgoing OPEN address frame and the value of the ARBITRATION WAIT TIME field received in the incoming OPEN address frame.

A valid OPEN address frame is 8 data dwords long and has no CRC error.

7.13.4.4 Transition SL1:ArbSel to SL3:Connected

This transition shall occur if an OPEN_ACCEPT Received parameter is received.

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 pass the Open STP Connection argument to the SL3:Connected state. 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 pass the Open SSP Connection argument, Source Port Hashed Value argument (i.e., hashed value of the source port identifier), and the Destination Port Hashed Value argument (i.e., hashed value of the source destination identifier) to the SL3:Connected. 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 pass the Open SMP Connection argument, the Source Port Hashed Value argument (i.e., hashed value of the source port identifier), and the Destination Port Hashed Value argument (i.e., hashed value of the source destination identifier) to the SL3:Connected state. At this point an SMP connection has been opened between the source phy and the destination phy.

7.13.4.5 Transition SL1:ArbSel to SL5:BreakWait

This transition shall occur if a BREAK Received parameter 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; or
b) there is no response to the OPEN address frame within the open timeout and after sending an Open Failed (Open Timeout Occurred) confirmation to the port layer.

7.13.4.6 Transition SL1:ArbSel to SL6:Break

This transition shall occur after:

a) receiving a BREAK Received parameter; and
b) sending an Open Failed (Break Received) confirmation to the port layer.

7.13.5 SL2:Selected state

7.13.5.1 State description

This state completes the establishment of an SSP, SMP, or STP connection when an incoming connection request has won arbitration by requesting an OPEN_ACCEPT or an OPEN_REJECT be transmitted by

sending a Transmit OPEN_ACCEPT parameter or a Transmit OPEN_REJECT parameter to the SL transmitter.

While in this state, the SAS port accepts opening a connection between it and the destination SAS port by by requesting an OPEN_ACCEPT be transmitted by sending a Transmit OPEN_ACCEPT parameter to the SL transmitter.

This state may reject opening a connection by requesting an OPEN_REJECT be transmitted by sending a Transmit OPEN_REJECT parameter to the SL transmitter.

7.13.5.2 Transition SL2:Selected to SL0:Idle

This transition shall occur:

1) If the OPEN address frame DESTINATION SAS ADDRESS field does not match the SAS address of this device, then after this state requests an OPEN_REJECT (WRONG DESTINATION) be transmitted by sending a Transmit OPEN_REJECT (WRONG DESTINATION) parameter to the SL transmitter;
2) If the OPEN address frame PROTOCOL field is set to a protocol that is not supported, then after this state requests an OPEN_REJECT (PROTOCOL NOT SUPPORTED) be transmitted by sending a Transmit OPEN_REJECT (PROTOCOL NOT SUPPORTED) parameter the SL transmitter;
3) If the OPEN address frame CONNECTION RATE field is set to a connection rate that is not supported, then after this state requests an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) be transmitted by sending a Transmit OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) parameter the SL transmitter; or
4) If an Accept_Reject Opens (Reject SSP) request, Accept_Reject Opens (Reject SMP) request, or Accept_Reject Opens (Reject STP) request is received, then after this state requests an OPEN_REJECT (RETRY) be transmitted by sending a Transmit OPEN_REJECT (RETRY) parameter to the SL transmitter.

7.13.5.3 Transition SL2:Selected to SL3:Connected

If the requested protocol is SSP and the port layer has not sent an Accept_Reject Opens (Reject SSP) request then this transition shall occur after:

a) requesting an OPEN_ACCEPT be transmitted by sending a Transmit OPEN_ACCEPT parameter to the SL transmitter; and
b) sending a Connection Opened (SSP, Destination Opened) confirmation to the port layer.

This transition shall pass the Open SSP Connection argument to the SL3:Connected state. At this point an SSP connection has been opened between the source phy and the destination phy.

If the requested protocol is SMP and the port layer has not sent an Accept_Reject Opens (Reject SMP) request then this transition shall occur after:

a) requesting an OPEN_ACCEPT be transmitted by sending a Transmit OPEN_ACCEPT parameter to the SL transmitter; and
b) sending a Connection Opened (SMP, Destination Opened) confirmation to the port layer.

This transition shall pass the Open SMP Connection argument to the SL3:Connected state. At this point an SMP connection has been opened between the source phy and the destination phy.

If the requested protocol is STP and the port layer has not sent an Accept_Reject Opens (Reject STP) request then this transition shall occur after:

a) requesting an OPEN_ACCEPT be transmitted by sending a Transmit OPEN_ACCEPT parameter to the SL transmitter; and
b) sending a Connection Opened (STP, Destination Opened) confirmation to the port layer.

This transition shall pass the Open STP Connection argument to the SL3:Connected state. At this point an STP connection has been opened between the source phy and the destination phy.

7.13.5.4 Transition SL2:Selected to SL6:Break

This transition shall occur after a BREAK Received parameter is received.

7.13.6 SL3:Connected state

7.13.6.1 State description

This state enables the SSP, STP, or SMP link layer state machine to transmit dwords during a connection.

If this state is entered from SL1:ArbSel state or the SL2:Selected state with an argument of Open SMP Connection then this state shall send an Enable Disable SMP (Enable) parameter to the SMP link layer state machines (see 7.18.4).

If this state is entered from SL1:ArbSel state or the SL2:Selected state with an argument of Open SSP Connection then this state shall send an Enable Disable SSP (Enable) parameter to the SSP link layer state machines (see 7.16.7).

If this state is entered from SL1:ArbSel state or the SL2:Selected state with an argument of Open STP Connection then this state shall send an Enable Disable STP (Enable) parameter to the STP link layer state machines.

This state requests idle dwords be transmitted by repeatedly sending Transmit Idle Dword parameters to the SL transmitter until the SSP, SMP, or STP link layer state machine starts transmitting.

7.13.6.2 Transition SL3:Connected to SL4:DisconnectWait

This transition shall occur if a Request Close parameter is received.

7.13.6.3 Transition SL3:Connected to SL5:BreakWait

This transition shall occur if a Request Break parameter is received and a BREAK Received parameter has not been received.

7.13.6.4 Transition SL3:Connected to SL6:Break

This transition shall occur if a BREAK Received parameter is received and after this state has sent a Connection Closed (Break Received) confirmation to the port layer.

7.13.7 SL4:DisconnectWait state

7.13.7.1 State description

This state closes the connection and releases all resources associated with the connection.

This state:

1) shall request a CLOSE be transmitted by sending a Transmit CLOSE parameter to the SL transmitter;
2) requests three idle dwords be transmitted by sending at least three Transmit Idle Dword parameters to the SL transmitter; and
3) initialize a close timeout timer to 1 ms and start the timer.

A CLOSE Received parameter may be received at any time while in this state.

7.13.7.2 Transition SL4:DisconnectWait to SL0:Idle

This transition shall occur after:

a) requesting a CLOSE be transmitted by sending a Transmit CLOSE parameter to the SL transmitter;
b) receiving a CLOSE Received parameter; and
c) sending a Connection Closed (Normal) confirmation to the port layer.

7.13.7.3 Transition SL4:DisconnectWait to SL5:BreakWait

This transition shall occur if:

a) a BREAK Received parameter has not been received;
b) no CLOSE Received parameter is received in response to a Transmit CLOSE parameter within a close timeout; and

c) after sending a Connection Closed (Close Timeout) confirmation to the port layer.

7.13.7.4 Transition SL4:DisconnectWait to SL6:Break

This transition shall occur after receiving a BREAK Received parameter and after sending a Connection Closed (Break Received) confirmation to the port layer.

7.13.8 SL5:BreakWait state

7.13.8.1 State description

This state closes the connection if one is established and releases all resources associated with the connection.

This state shall:

1) shall request a BREAK be transmitted by sending a Transmit BREAK parameter to the SL transmitter;
2) request six idle dwords be transmitted by sending at least six Transmit Idle Dword parameters to the SL transmitter; and
3) initialize a break timeout timer to 1 ms and start the timer.

7.13.8.2 Transition SL5:BreakWait to SL0:Idle

This transition shall occur after receiving a BREAK Received parameter or if the break timeout is exceeded. If a BREAK Received parameter is not received before the timer is exceeded, this state shall send a Connection Closed (Link Broken) confirmation to the port layer before making this transition.

7.13.9 SL6:Break state

7.13.9.1 State description

This state closes any connection and releases all resources associated with this connection.

This state shall request a BREAK be transmitted by sending a Transmit BREAK parameter to the SL transmitter.

While in this state all primitives received shall be ignored.

7.13.9.2 Transition SL6:Break to SL0:Idle

This transition shall occur after sending a Transmit BREAK parameter to the SL transmitter.

7.14 SAS link layer state machine for expander phys (XL)

7.14.1 Overview

The expander link layer (XL) state machine controls the flow of dwords on the physical link and establishes and maintains connections with another XL state machine as facilitated by the expander function - specifically the expander connection manager and expander connection router.

The XL state machine contains the following states:

a) XL0:Idle (see 7.14.2)(initial state);
b) XL1:Request_Path (see 7.14.3);
c) XL2:Request_Open (see 7.14.4);
d) XL3:Open_Confirm_Wait (see 7.14.5);
e) XL4:Open_Reject (see 7.14.6);
f) XL5:Forward_Open (see 7.14.7);
g) XL6:Open_Response_Wait (see 7.14.8);
h) XL7:Connected (see 7.14.9);
i) XL8:Close_Wait(see 7.14.10);
j) XL9:Break (see 7.14.11); and

k) XL10:Break_Wait (see 7.14.12).

The XL state machine shall be activated after the completion of the phy reset sequence ~~by receiving an after receiving~~ an Enable Disable SAS Link (Disable) parameter from the SL_IR state machines (see 7.8.5).

The XL state machine sends the following requests to the expander connection manager:

a) Request Path;
b) Phy Status (Partial Pathway);
c) Phy Status (Blocked Partial Pathway); and
d) Phy Status (Connected).

The XL state machine receives the following confirmations ~~from the expander connection manager:~~

a) Arb Reject (No Destination);
b) Arb Reject (Bad Destination);
c) Arb Reject (Bad Connection Rate);
d) Arb Reject (Pathway Blocked);
e) Arbitrating (Waiting On Connection);
f) Arbitrating (Waiting On Partial);
g) Arbitrating (Blocked On Partial);
h) Arb Won; and
i) Arb Lost.

A pair of XL state machines communicate with each other using the following requests/indications which are routed through the expander connection router:

a) Transmit Open;
b) Transmit Break;
c) Transmit Close; and
d) Transmit Dword.

A pair of XL state machines communicate with each other using the following confirmations/responses which are routed through the expander connection router:

a) Arb Status (Normal);
b) Arb Status (Waiting On Partial);
c) Arb Status (Waiting On Connection);
d) Arb Status (Waiting On Device);
e) Open Accept;
f) Open Reject;
g) Backoff Retry; and
h) Backoff Reverse Path.

The XL state machine sends the following requests to the broadcast primitive processor:

a) Broadcast Event Notify.

The XL state machine receives the following requests ~~from the broadcast primitive processor:~~

a) Transmit Broadcast Primitive.

The XL state machine receives the following parameters ~~from the SL_IR state machine:~~

a) Enable Disable SAS Link (Enable); and
b) Enable Disable SAS Link (Disable).

Unless otherwise stated within a state description, all disparity errors, illegal characters, and unexpected primitives received within any XL state shall be ignored.

Figure 74 defines the expander link layer (XL) state machine.

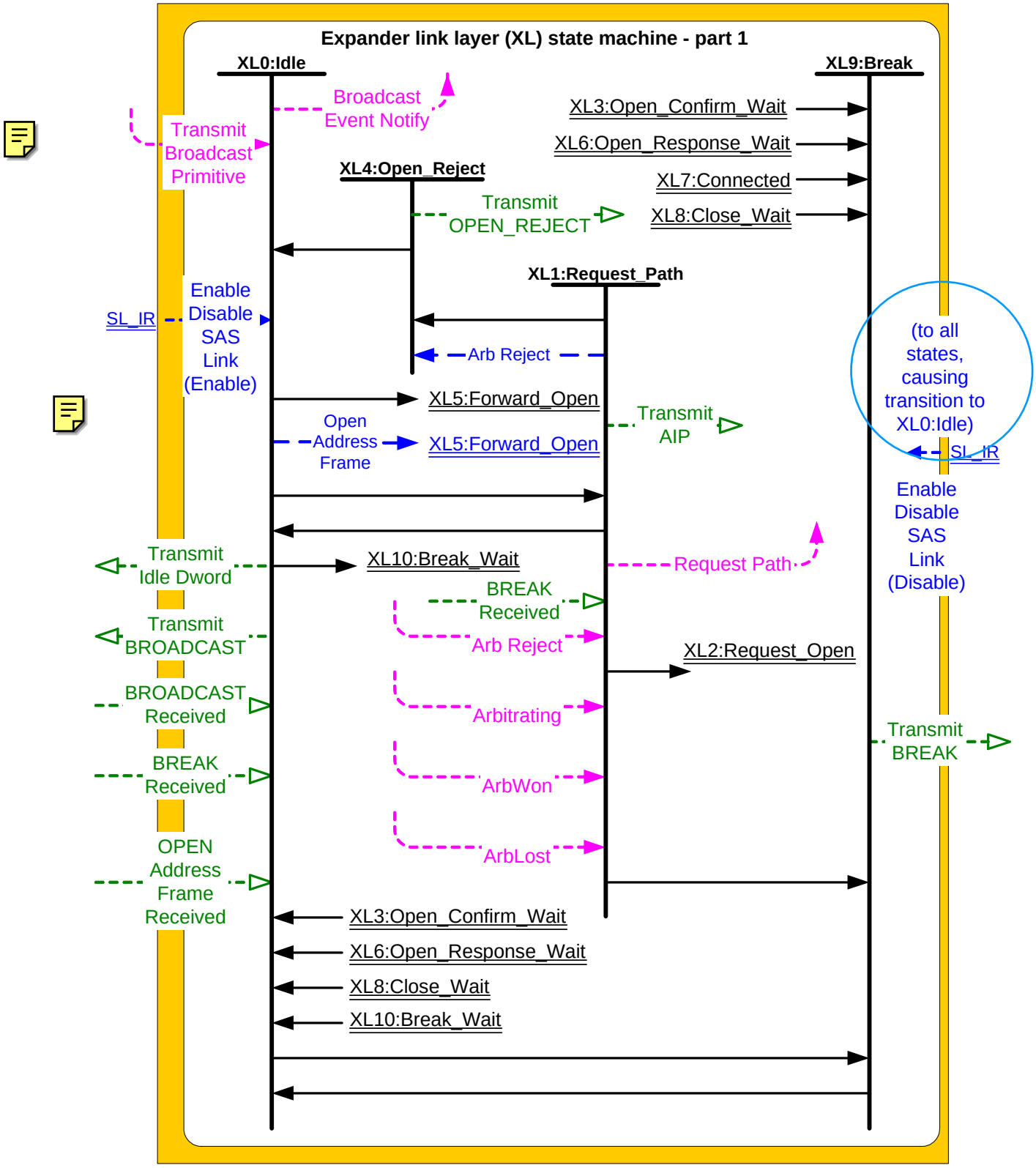


Figure 74 — Expander link layer (XL) state machine (part 1)

Figure 75 further defines the expander link layer (XL) state machine.

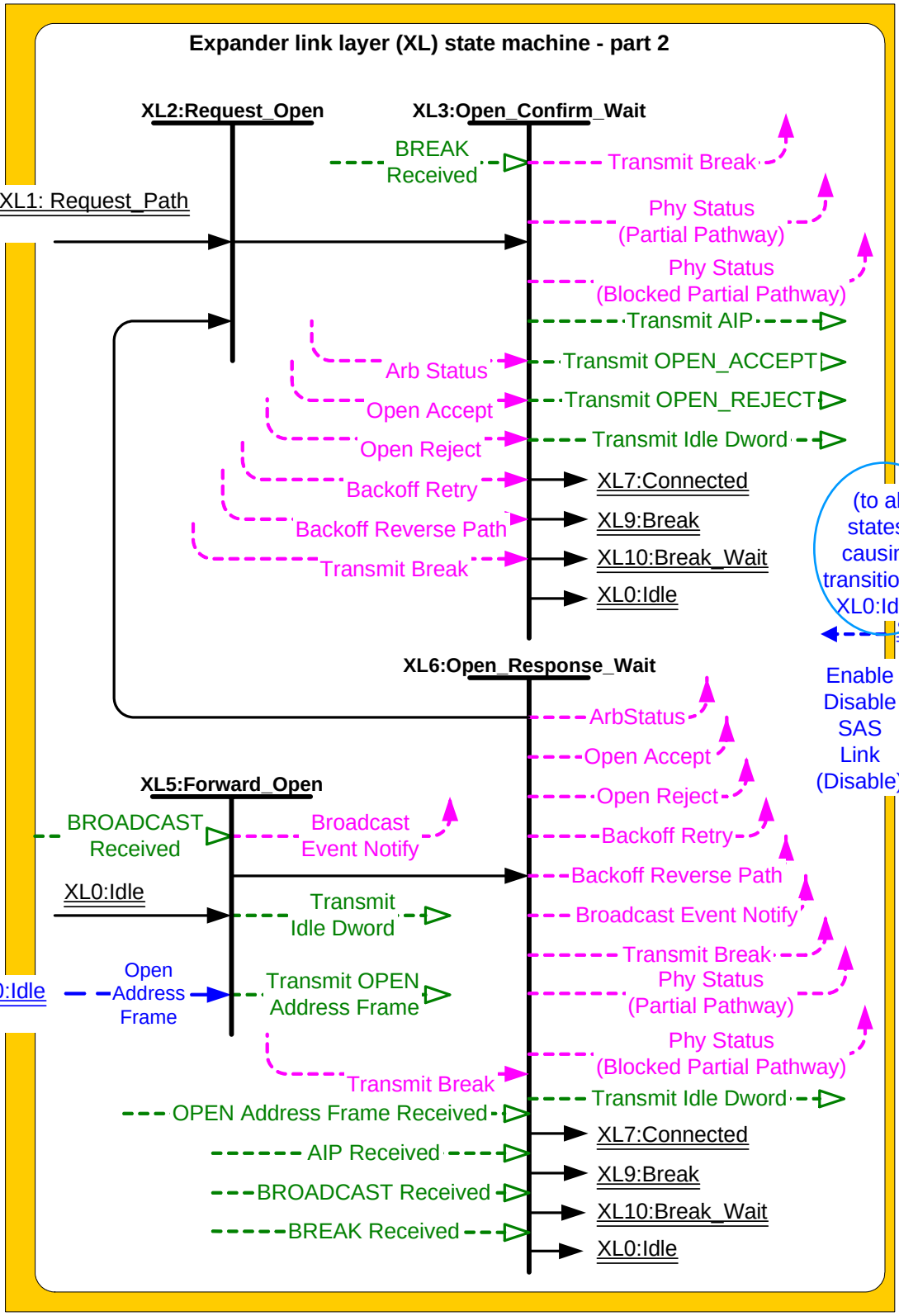


Figure 75 — Expander link layer (XL) state machine (part 2)

Figure 76 further defines the expander link layer (XL) state machine.

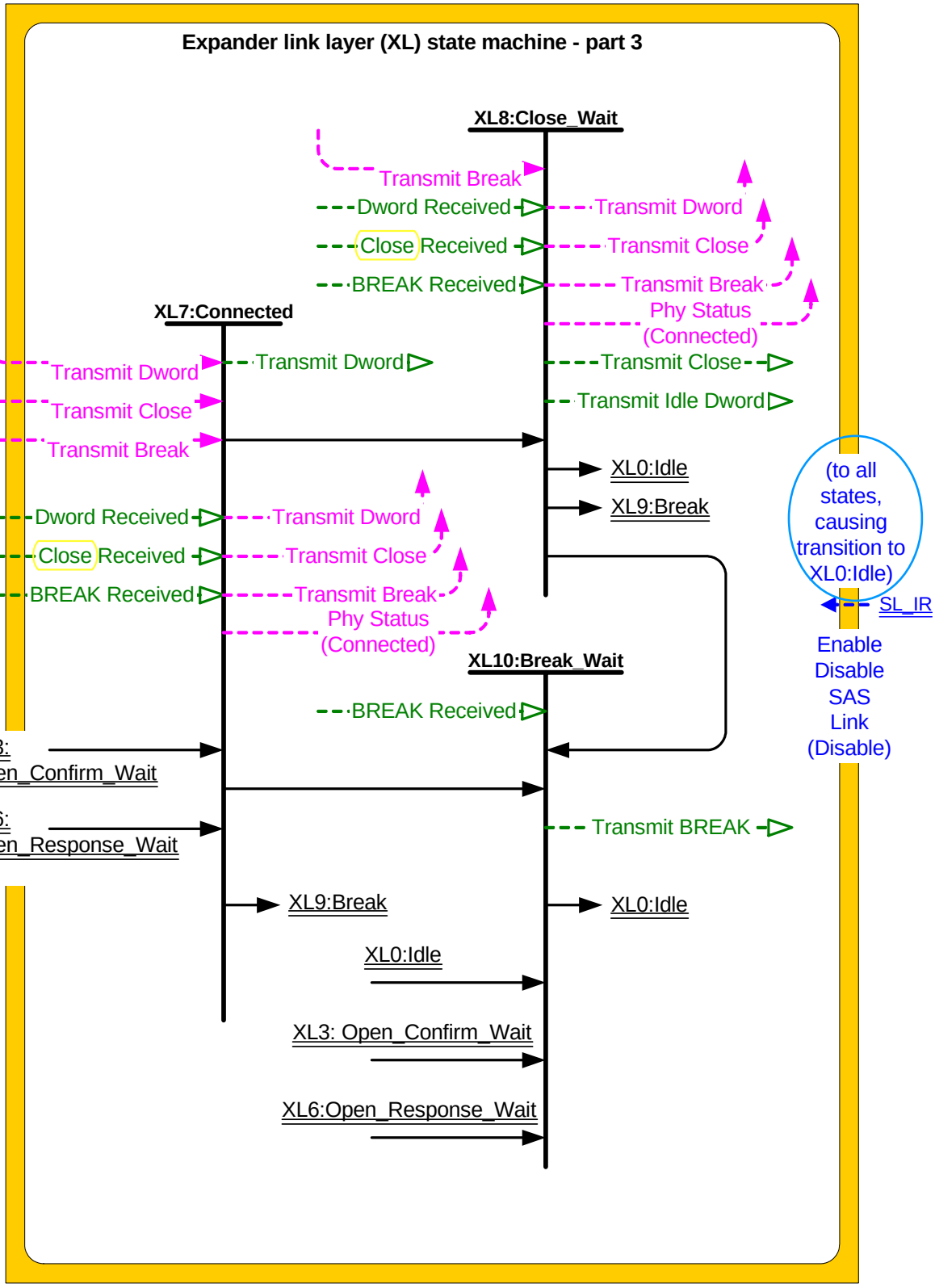


Figure 76 — Expander link layer (XL) state machine (part 3)

7.14.2 XL0:Idle state

7.14.2.1 State description

This state is the initial state and the state that occurs when there is no pending or active connection.

If a BROADCAST Received parameter is received, this state shall send a Broadcast Event Notify confirmation to the expander function with the argument indicating the specific BROADCAST primitive received (e.g., BROADCAST (CHANGE)).

If a Transmit Broadcast Primitive parameter is received, this state shall send a Transmit BROADCAST parameter to the XL transmitter. Otherwise, this state shall repeatedly send a Transmit Idle Dword parameter to the XL transmitter.

7.14.2.2 Transition XL0:Idle to XL1:Request_Path

This transition shall occur when the following conditions are met:

a) Transmit Open indication is not being received ~~from another phy via the expander connection router;~~
b) Transmit Break indication is not being received ~~from another phy via the expander connection router;~~ and
c) an OPEN Address Frame Received parameter is received.

7.14.2.3 Transition XL0:Idle to XL5:Forward_Open

This transition shall occur when the following conditions are met:

a) Transmit Break indication is not being received ~~from another phy via the expander connection router;~~ and
b) Transmit Open indication is received ~~from another XL state machine via the expander connection router.~~

7.14.2.4 Transition XL0:Idle to XL9:Break

This transition shall occur when a BREAK Received parameter is received.

7.14.2.5 Transition XL0:Idle to XL10:Break_Wait

This transition shall occur when a Transmit Break indication is received from another XL state machine via the expander connection router.

7.14.3 XL1:Request_Path state

7.14.3.1 State description

This state is used to arbitrate for connection resources and to specify the destination of the connection.

This state shall send the following parameters to the XL transmitter:

a) Transmit AIP (WAITING ON PARTIAL) when an Arbitrating (Waiting On Partial) confirmation is received from the expander connection manager;
b) Transmit AIP (WAITING ON PARTIAL) when an Arbitrating (Blocked On Partial) confirmation is received from the expander connection manager;
c) Transmit AIP (WAITING ON CONNECTION) when an Arbitrating (Waiting On Connection) confirmation is received from the expander connection manager; or
d) Transmit AIP (NORMAL).

This state shall send a Request Path request to the expander connection manager with the following arguments:

a) destination SAS address;
b) source SAS address;
c) protocol;
d) connection rate;

e) arbitration wait time;
f) pathway blocked count; and
g) partial pathway timeout status.

This state shall initialize a partial pathway timeout timer to the partial pathway timeout value (see 7.12.3.1.3) and shall decrement the timer until reaching zero, while the Arbitrating (Blocked On Partial) confirmation is being received ~~from the expander connection manager.~~

When the Arbitrating (Blocked On Partial) confirmation is not received, the partial pathway timeout timer shall be held at an initial value set to the partial pathway timeout value.

When the partial pathway timeout timer expires, i.e. reaches zero while receiving the Arbitrating (Blocked On Partial) confirmation, timeout status is conveyed to the expander connection manager via the partial pathway timeout status argument in the Request Path request.

7.14.3.2 Transition XL1:Request_Path to XL2:Request_Open

This transition shall occur when an Arb Won confirmation is received ~~from the expander connection manager.~~

7.14.3.3 Transition XL1:Request_Path to XL4:Open_Reject

This transition shall occur when an Arb Reject confirmation is received ~~from the expander connection manager.~~

7.14.3.4 Transition XL1:Request_Path to XL0:Idle

This transition shall occur when an Arb Lost confirmation is received ~~from the expander connection manager.~~

7.14.3.5 Transition XL1:Request_Path to XL9:Break

This transition shall occur when a BREAK Received parameter is received.

7.14.4 XL2:Request_Open state

7.14.4.1 State description

This state is used to forward an OPEN address frame through the expander connection router to a destination phy.

This state shall repeatedly send a Transmit Idle Dword parameter to the XL transmitter.

This state shall send a Transmit Open request through the expander connection router to a destination phy, ~~received by the destination phy as a Transmit Open indication.~~ The arguments to the Transmit Open request/ ~~indication~~ are:

a) destination SAS address;
b) source SAS address;
c) protocol;
d) connection rate;
e) arbitration wait time;
f) initiator bit;
g) initiator connection tag;
h) features; and
i) pathway blocked count.

7.14.4.2 Transition XL2:Request_Open to XL3:Open_Confirm_Wait

This transition shall occur after the OPEN address frame has been forwarded to a destination phy.

7.14.5 XL3:Open_Confirm_Wait state

7.14.5.1 State description

This state waits for confirmation to an OPEN address frame sent on a destination phy.

This state shall send the following parameters to the XL transmitter:

a) Transmit AIP (NORMAL) when an Arb Status (Normal) confirmation is received from a destination phy;
b) Transmit AIP (WAITING ON PARTIAL) when an Arb Status (Waiting On Partial) confirmation is received from a destination phy;
c) Transmit AIP (WAITING ON CONNECTION) when an Arb Status (Waiting On Connection) confirmation is received from a destination phy;
d) Transmit AIP (WAITING ON DEVICE) when an Arb Status (Waiting On Device) confirmation is received from a destination phy;
e) Transmit OPEN_ACCEPT when an Open Accept confirmation is received from a destination phy;
f) Transmit OPEN_REJECT when an Open Reject confirmation is received from a destination phy; or
g) Transmit Idle Dword when none of the previous conditions are present.

This state shall send a Transmit Break request to a destination phy when a BREAK Received parameter is received.

If an Arb Status (Waiting on Partial) is received, this state shall send a Phy Status (Blocked Partial Pathway) confirmation to the expander connection manager. Otherwise, this state shall send a Phy Status (Partial Pathway) confirmation to the expander connection manager.

7.14.5.2 Transition XL3:Open_Confirm_Wait to XL0:Idle

This transition shall occur after:

a) An Open Reject confirmation is received ~~from a destination phy,~~ OPEN_REJECT is transmitted, and path resources have been released;
b) A Backoff Retry confirmation is received ~~from a destination phy~~ and path resources have been released; or
c) A Backoff Reverse Path confirmation is received ~~from a destination phy.~~

7.14.5.3 Transition XL3:Open_Confirm_Wait to XL7:Connected

This transition shall occur after Open Accept confirmation is received ~~from a destination phy~~ and OPEN_ACCEPT is transmitted.

7.14.5.4 Transition XL3:Open_Confirm_Wait to XL9:Break

This transition shall occur after a BREAK Received parameter is received and a Transmit Break request has been sent to a destination phy.

7.14.5.5 Transition XL3:Open_Confirm_Wait to XL10:Break_Wait

This transition shall occur when a Transmit Break indication is received ~~from a destination phy.~~

7.14.6 XL4:Open_Reject state

7.14.6.1 State description

This state is used to reject a connection request.

Thist state shall send the following parameters to the XL transmitter:

a) Transmit OPEN_REJECT (NO DESTINATION) when an Arb Reject (No Destination) confirmation is received ~~from the expander connection manager;~~
b) Transmit OPEN_REJECT (BAD DESTINATION) when an Arb Reject (Bad Destination) confirmation is received ~~from the expander connection manager;~~

c) Transmit OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) when an Arb Reject (Bad Connection Rate) confirmation is received ~~from the expander connection manager;~~
d) Transmit OPEN_REJECT (PATHWAY BLOCKED) when an Arb Reject (Pathway Blocked) confirmation is received ~~from the expander connection manager.~~

7.14.6.2 Transition XL4:Open_Reject to XL0:Idle

This transition shall occur after OPEN_REJECT has been transmitted.

7.14.7 XL5:Forward_Open state

7.14.7.1 State description

This state is used to transmit an OPEN address frame indicated by the Transmit Open indication received from a source phy via the expander connection router.

If a BROADCAST Received parameter is received, this state shall send a Broadcast Event Notify confirmation to the expander function with the argument indicating the specific BROADCAST primitive received (e.g., BROADCAST (CHANGE)).

This state shall send a Transmit OPEN Address Frame parameter to the XL transmitter with the fields set to the values specified by the Transmit Open indication.

7.14.7.2 Transition XL5:Forward_Open to XL6:Open_Response_Wait

This transition shall occur after the OPEN address frame has been transmitted.

7.14.8 XL6:Open_Response_Wait state

7.14.8.1 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 transmit idle dwords.

If a BROADCAST Received parameter is received before an AIP Received parameter is received, this state shall send a Broadcast Event Notify confirmation to the expander function with the argument indicating the specific BROADCAST primitive received (e.g., BROADCAST (CHANGE)).

This state shall send the following responses through the expander connection router to a source phy, received by the source phy as confirmations:

a) Open Accept when OPEN_ACCEPT is received;
b) Open Reject when OPEN_REJECT is received;
c) Backoff Retry when a higher priority OPEN address frame is received (see 7.12.3) and 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; or
d) Backoff Reverse Path when a higher priority OPEN address frame is received (see 7.12.3) and 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.

This state shall send the following arbitration responses through the expander connection router to a source phy, received by the source phy as confirmations:

a) Arb Status (Waiting On Device) when an AIP Received parameter has not been received;
b) Arb Status (Normal) when an AIP (NORMAL) Received parameter is received;
c) Arb Status (Waiting On Partial) when an AIP (WAITING ON PARTIAL) Received parameter is received;
d) Arb Status (Waiting On Connection) when an AIP (WAITING ON CONNECTION) Received parameter is received; and
e) Arb Status (Waiting On Device) when an AIP (WAITING ON DEVICE) Received parameter is received.

This state shall send a Transmit Break request to a source phy when a BREAK Received parameter is received.

If an AIP (WAITING ON PARTIAL) Received parameter is received, this state shall send a Phy Status (Blocked Partial Pathway) confirmation to the expander connection manager. Otherwise, this state shall send a Phy Status (Partial Pathway) confirmation to the expander connection manager.

7.14.8.2 Transition XL6:Open_Response_Wait to XL0:Idle

The XL7:XL0 transition shall occur after one of the following conditions are met:

a) OPEN_REJECT is received, Open Reject response has been sent to a source phy, and path resources have been released; or
b) a higher priority OPEN address frame is received (see 7.12.3) and 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, a Backoff Retry response has been sent to a source phy, and path resources have been released.

7.14.8.3 Transition XL6:Open_Response_Wait to XL2:Request_Open

This transition shall occur after a higher priority OPEN address frame is received (see 7.12.3) and 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, and Backoff Reverse Path response has been sent to a source phy.

7.14.8.4 Transition XL6:Open_Response_Wait to XL7:Connected

This transition shall occur after OPEN_ACCEPT is received and Open Accept response is sent to a source phy.

7.14.8.5 Transition XL6:Open_Response_Wait to XL9:Break

This transition shall occur after a BREAK is received and Transmit Break response is sent to a source phy.

7.14.8.6 Transition XL6:Open_Response_Wait to XL10:Break_Wait

This transition shall occur when a Transmit Break indication is received ~~from a source phy.~~

7.14.9 XL7:Connected state

7.14.9.1 State description

This state provides a full-duplex circuit between two phys within an expander device.

This state shall transmit all dwords received by the Transmit Dword indication from a connected phy via the expander connection router.

This state shall send all valid dwords received by the SAS phy through the expander connection router to a connected phy using the Transmit Dword request with the exception of BREAK and CLOSEs.

This state shall replace invalid dwords received by the SAS phy as specified in section 7.12.4

This state shall send a Transmit Close request to a connected phy when a CLOSE Received parameter is received.

This state shall send a Transmit Break request to a connected phy when a BREAK Received parameter is received.

This state shall send a Phy Status (Connected) confirmation to the expander connection manager.

7.14.9.2 Transition XL7:Connected to XL8:Close_Wait

Tis transition shall occur when a Transmit Close indication is received ~~from a connected phy via the expander connection router.~~

7.14.9.3 Transition XL7:Connected to XL9:Break

This transition shall occur when a BREAK Received parameter is received.

7.14.9.4 Transition XL7:Connected to XL10:Break_Wait

This transition shall occur when a Transmit Break indication is received ~~from a connected phy via the expander connection router.~~

7.14.10 XL8:Close_Wait state

7.14.10.1 State description

This state closes a connection and releases path resources.

This state shall send a Transmit CLOSE parameter to the XL transmitter, then shall repeatedly send a Transmit Idle Dword parameter to the XL transmitter.

This state shall send all valid dwords received by the SAS phy through the expander connection router to a connected phy using the Transmit Dword request with the exception of BREAK and CLOSEs.

This state shall send a Transmit Close request to a connected phy when a CLOSE Received parameter is received. The expander device shall transmit the same CLOSE primitive that was received (e.g. CLOSE (NORMAL) forwarded as CLOSE (NORMAL)).

This state shall send a Transmit Break request to a connected phy when a BREAK Received parameter is received.

This state shall send a Phy Status (Connected) confirmation to the expander connection manager.

7.14.10.2 Transition XL8:Close_Wait to XL0:Idle

This transition shall occur after a CLOSE has been both transmitted and received and after path resources have been released for this connection.

7.14.10.3 Transition XL8:Close_Wait to XL9:Break

This transition shall occur when a BREAK Received parameter is received.

7.14.10.4 Transition XL8:Close_Wait to XL10:Break_Wait

This transition shall occur when a Transmit Break indication is received ~~from a connected phy via the expander connection router.~~

7.14.11 XL9:Break state

7.14.11.1 State description

This state closes any connection and releases any path resources.

This state shall send a Transmit BREAK parameter to the XL transmitter.

7.14.11.2 Transition XL9:Break to XL0:Idle

This transition shall occur after transmitting a BREAK.

7.14.12 XL10:Break_Wait state

7.14.12.1 State description

This state closes any connection and releases any path resources.

This state shall send a Transmit BREAK parameter to the XL transmitter. After transmitting the BREAK this state shall initialize a break timeout timer to 1 millisecond and start the timer.

7.14.12.2 Transition XL10:Break_Wait to XL0:Idle

This transition shall occur after a BREAK Received parameter is received or after the break timeout timer expires, ~~whichever occurs first.~~

7.15 Rate matching

Initiator ports shall use SMP to discover the negotiated physical link rate of the target phys and the negotiated physical link rates of every intermediate expander device-to-expander device physical link, and shall not request a connection rate that is faster than the slowest negotiated physical link rate on any potential intermediate physical link.

Each successful connection request contains the connection rate of the pathway.

Every phy in the physical link shall insert ALIGNs between dwords to match the connection rate.

The faster phy shall rotate between ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3) ~~to reduce EMI.~~

Figure 77 shows an example of rate matching between a 1,5 Gbps source phy and a 3,0 Gbps destination phy, with one ALIGN inserted between dwords on the faster physical link.

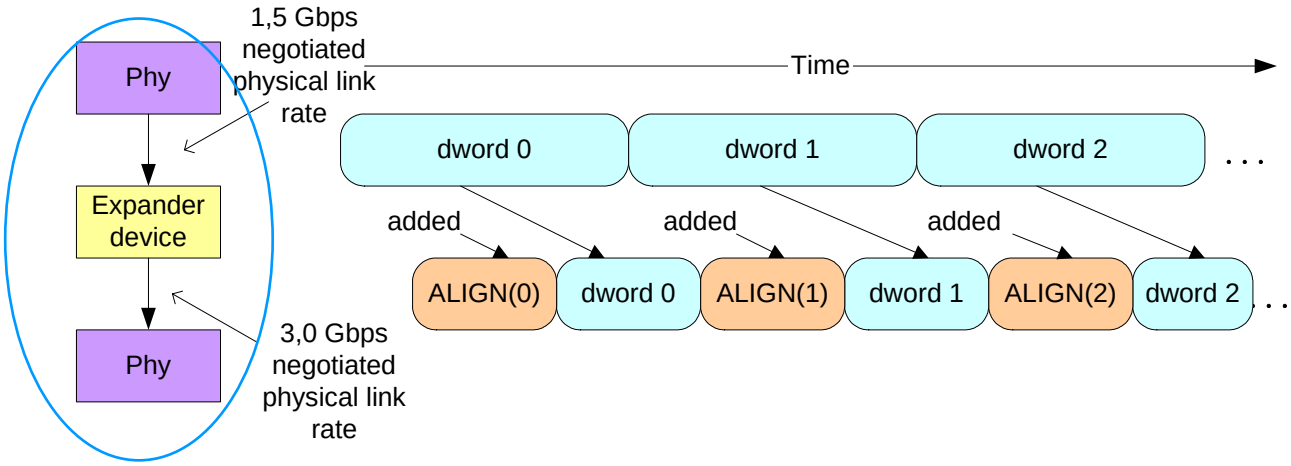


Figure 77 — Rate matching example

After transmitting the OPEN address frame, the source shall start transmitting idle dwords including ALIGNs at the selected connection rate while waiting for the connection response. This enables each expander device to start forwarding dwords from the source to the destination ~~immediately~~ after seeing an OPEN_ACCEPT. The OPEN address frame itself may or may not have ALIGNs interspersed.

If an STP initiator port discovers an SATA target device with a physical link rate greater than the maximum connection rate supported by the pathway from the STP initiator port, the STP initiator port should use the SMP PHY CONTROL function (see 10.3.1.9) to set the MAXIMUM PHYSICAL LINK RATE field of the phy attached to the SATA target device to the maximum connection rate supported by the pathway.

7.16 SSP link layer

7.16.1 Opening an SSP connection

A port that accepts an OPEN address frame shall transmit at least one RRDY within 1 ms of transmitting an OPEN_ACCEPT. If the port is not able to grant credit, it shall respond with OPEN_REJECT (RETRY).

7.16.2 Full duplex

SSP is a full duplex protocol. A port may receive a frame or primitive on a physical link the same time it is transmitting a frame or primitive on the same physical link.

When a connection is open and a port has no more frames to transmit, it shall transmit DONE (NORMAL) to start closing the connection. The other direction may still be active, so the DONE (NORMAL) may be followed by RRDYs, ACKs, and NAKs.

7.16.3 SSP frame transmission

During an SSP connection, frames are preceded by SOF and followed by EOF as shown in figure 78.

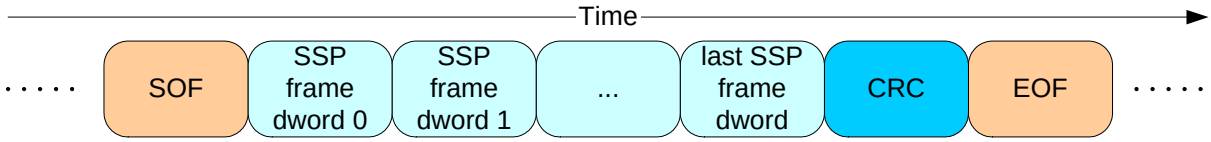


Figure 78 — SSP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.4). ~~The link layer shall check that the number of data dwords between the SOF and EOF is at least 28 bytes and that the CRC is valid.~~

Every frame shall be acknowledged within 1 ms with either a positive acknowledgement (ACK) or a negative acknowledgement (NAK). ACK means the frame was received into a frame buffer without errors. NAK means the frame was received with an error; NAK (CRC ERROR) means the frame was received with a CRC error.

Some interlocked frames which are NAKed may be retried by the transport layer (see 9.2.4). Non-interlocked frames which are NAKed shall not be retried by any layer. The SCSI command associated with a NAKed frame that is not successfully retried shall be terminated with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of DATA PHASE CRC ERROR DETECTED.

7.16.4 SSP flow control

SSP devices grant credit for permission to transmit frames with RRDY. 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 devices shall not increment credit past 255 frames.

An SSP initiator port or an SSP target/initiator port acting in its initiator role shall never refuse to provide credit by withholding RRDY because it needs to transmit a frame itself. It may refuse to provide credit for other reasons. An SSP target port or an SSP target/initiator port acting in its target role may refuse to provide credit for any reason, including because it needs to transmit a frame itself. This prevents deadlocks where both ports are waiting on the other to provide credit.

7.16.5 Interlocked frames

Table 83 shows which frames shall be interlocked.

Table 83 — Frame interlock requirements

Frame type	Interlock requirement
COMMAND	Interlocked
TASK	Interlocked
XFER_RDY	Interlocked
DATA	Non-interlocked
RESPONSE	Interlocked

Before transmitting an interlocked frame, an SSP port shall wait for all frames to be acknowledged with ACK or NAK, even if credit is available. After transmitting an interlocked frame, an SSP port shall not transmit another frame until it has been acknowledged with ACK or NAK, even if credit is available.

Before transmitting a non-interlocked frame, an SSP port 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 port may transmit another non-interlocked frame with the same tag if credit is available. The port shall not transmit:

a) a non-interlocked frame with a different tag; or
b) an interlocked frame;

until all 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 initiator port could be transmitting COMMAND while receiving RESPONSE).

A port may transmit primitives responding to traffic on the back channel (e.g. an ACK or NAK to acknowledge a frame, or an RRDY to grant more backchannel credit) while waiting for an interlocked frame it transmitted to be acknowledged. These primitives may also be interspersed within the frame.

Figure 79 shows an example of interlocked frame transmission.

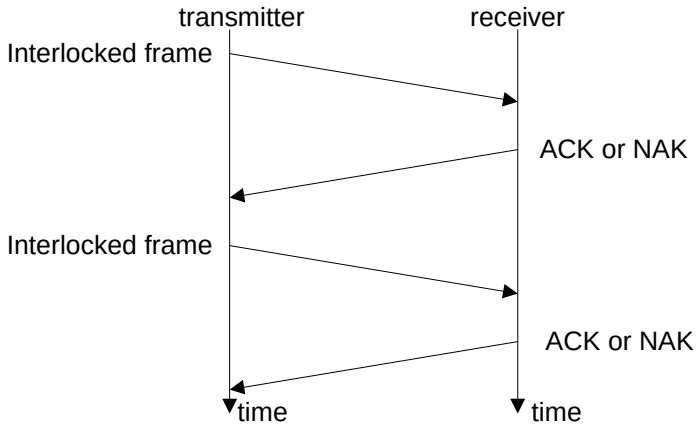


Figure 79 — Interlocked frames

Figure 80 shows an example of non-interlocked frame transmission with the same tags.

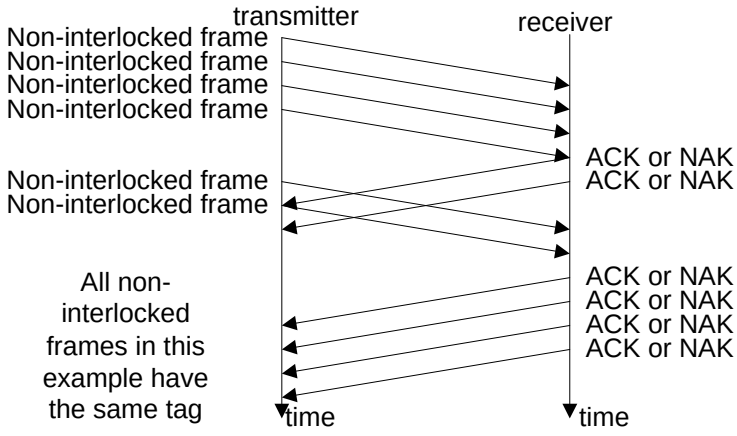


Figure 80 — Non-interlocked frames with the same tag

Figure 81 shows an example of non-interlocked frame transmission with different tags.

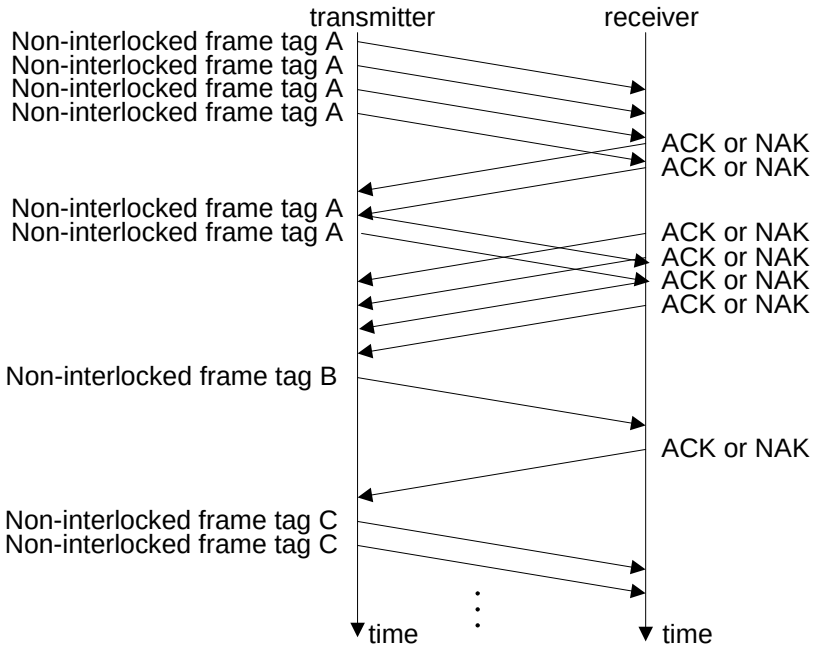


Figure 81 — Non-interlocked frames with different tags

7.16.6 Preparing to close an SSP connection

DONE is exchanged prior to closing an SSP connection.

When a source SSP device has no frames to transmit, it shall transmit DONE (NORMAL). When a destination SSP device has no frames to transmit, it may wait for a vendor-specific period of time, and then shall transmit DONE (NORMAL). There are several versions of the DONE primitive indicating additional information about why the SSP connection is being closed:

a) DONE (NORMAL) indicates normal completion; the transmitter has no more frames to transmit;
b) DONE (CREDIT TIMEOUT) indicates the transmitter still has frames to transmit, but did not receive an RRDY granting frame credit within 1 ms; and
c) DONE (ACK/NAK TIMEOUT) indicates the transmitter transmitted a frame but did not receive the corresponding ACK or NAK within 1 ms; the ACK/NAK count is imbalanced and the transmitter is going to transmit a BREAK in 1 ms unless the recipient replies with DONE and the connection is closed.

After transmission of DONE, a device shall not transmit any more frames during this connection. However, a device may transmit ACK, NAK, and RRDY after transmitting DONE if the other device is still transmitting frames on the back channel. Once a port has both transmitted and received DONE, it may close the connection by transmitting the CLOSE (see 7.12.7).

7.16.7 SSP link layer (SSP) state machines

7.16.7.1 Overview

The SSP link layer contains several state machines that run in parallel to control the flow of dwords on the link during an SSP connection. The SSP link state machines are as follows:

a) Transmit Interlocked Frame Monitor (SSP_TIM state machine);
b) Transmit Frame Credit Monitor (SSP_TCM state machine);
c) DONE Control (SSP_D state machine);
d) Transmit Frame Control (SSP_TF state machine);
e) Receive Frame Control (SSP_RF state machine);
f) Receive Frame Credit Monitor (SSP_RCM state machine);

g) Receive Interlocked Frame Monitor (SSP_RIM state machine);
h) Transmit Credit Control (SSP_TC state machine); and
i) Transmit ACK/NAK Control (SSP_TAN state machine).

All the SSP state machines shall begin after receiving an Enable Disable SSP (Enable) parameter ~~from the SL state machine~~ (see 7.13).

All the SSP state machines shall stop after:

a) receiving an Enable Disable SSP (Disable) parameter ~~from the SL state machine;~~
b) receiving a Request Close parameter ~~from the SSP_D1:DONE_Wait state~~ indicating that the connection shall be closed; or
c) receiving a Request Break parameter ~~from the SSP_D1:DONE_Wait state~~ indicating that a BREAK shall be transmitted.

If a state machine consists of multiple states the initial state is as indicated in the state machine description in this subclause.

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. The SSP_TIM state machine contains the SSP_TIM1:Tx_Interlock_Monitor state (see 7.16.7.2).

The SSP_TCM state machine's function is to ensure that credit is available from the originator before a frame is transmitted. The SSP_TCM state machine contains the SSP_TCM1:Tx_credit_monitor state (see 7.16.7.3).

The SSP_D state machine's function is to ensure a DONE has been received and transmitted before the SL state machine disables the SSP state machines. The SSP_D state machine contains the SSP_D1:Done_Wait state (see 7.16.7.4).

The SSP_TF state machine's function it to control when the SSP_T state machine transmits SOF, frame dwords, EOF, and DONE. The SSP_TF state machine contains the following states:

a) SSP_TF1:Connected_Idle (see 7.16.7.5)(initial state);
b) SSP_TF2:Tx_Wait (see 7.16.7.6);
c) SSP_TF3:Indicate_Frame_Tx (see 7.16.7.7); and
d) SSP_TF4:Indicate_DONE_Tx (see 7.16.7.8).

The SSP_RF state machine's function is to receive frames and to determine if those frames were successful or unsuccessful received. The SSP_RF state machine contains the SSP_RF1:Rcv_Frame state (see 7.16.7.9).

The SSP_RCM state machine's function is to ensure that there was credit given to the originator for every frame that is received. The SSP_RCM state machine contains the SSP_RCM1:Rcv_Credit_Monitor state (see 7.16.7.10).

The SSP_RIM state machine's function is to inform the SSP_RF1:Rcv_Frame state when the number of ACKs and NAKs transmitted equals the number of the EOFs received. The SSP_RIM state machine contains the SSP_RIM1:Rcv_Interlock_Monitor state (see 7.16.7.11).

The SSP_TC state machine's function it to control the sending of requests to transmit an RRDY or CREDIT_BLOCKED. The SSP_TC state machine contains the following states:

a) SSP_TC1:Idle (see 7.16.7.12)(initial state); and
b) SSP_TC2:Indicate_Credit_Tx (see 7.16.7.13).

The SSP_TAN state machine's function is to control the sending of requests to transmit an ACK or NAK. The SSP_TAN state machine contains the following states:

a) SSP_TAN1:Idle (see 7.16.7.14)(initial state); and
b) SSP_TAN2:Indicate_ACK/NAK_Tx (see 7.16.7.15).

Figure 82 shows the SSP states related to frame transmission.

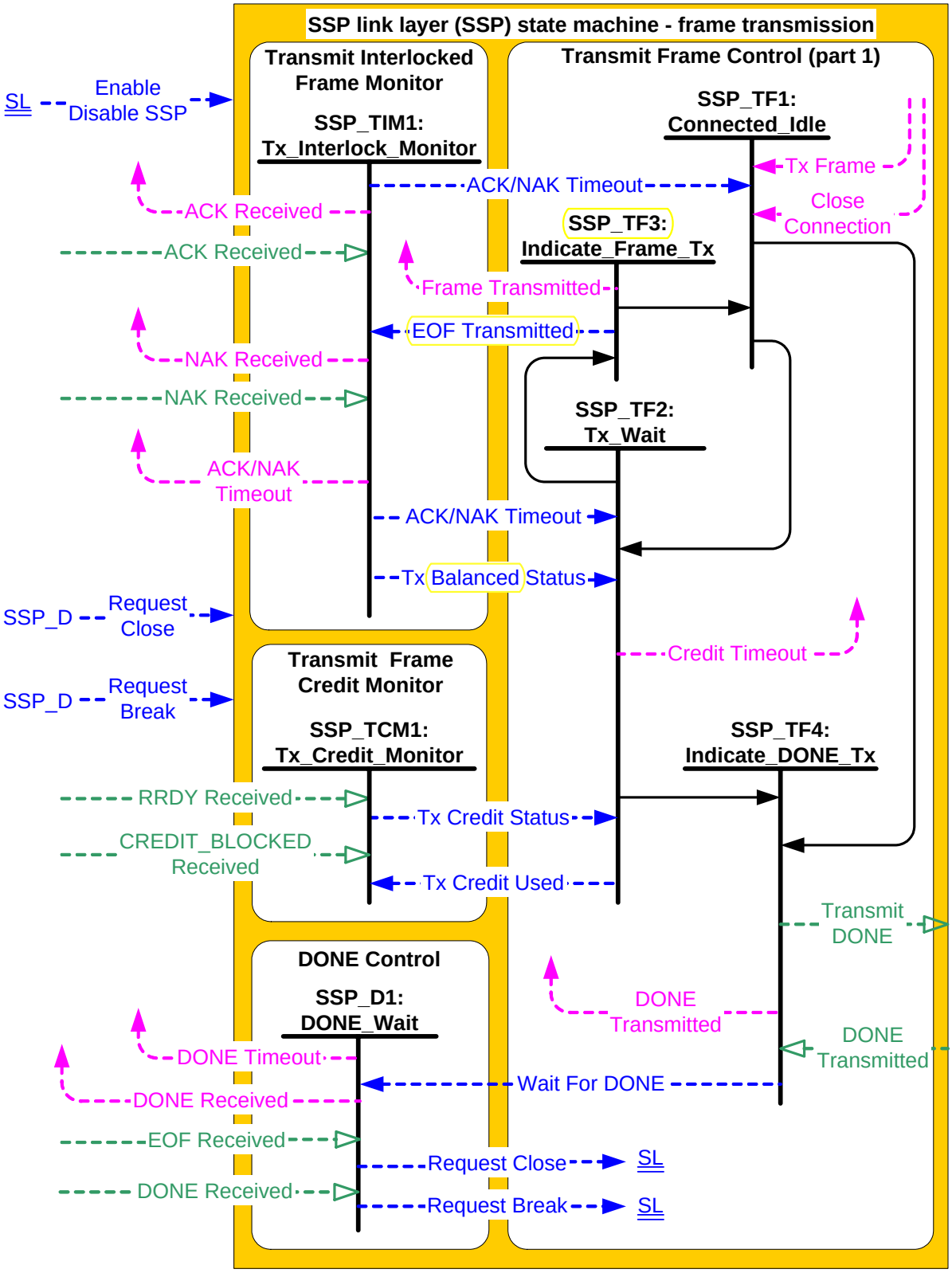


Figure 82 — SSP link layer (SSP) state machines (part 1 - frame transmission)

Figure 83 shows the SSP states related to frame reception.

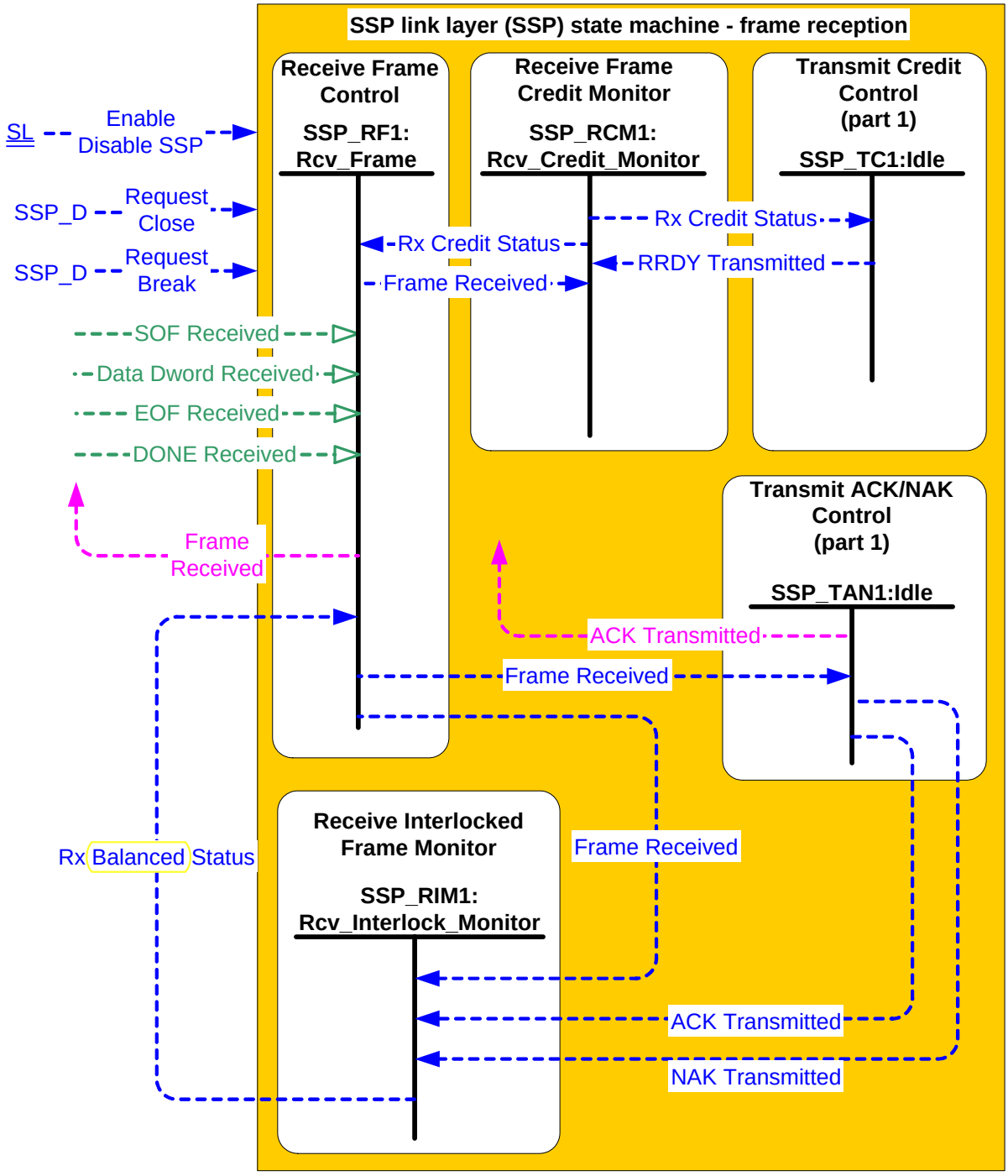


Figure 83 — SSP link layer (SSP) state machines (part 2 - frame reception)

Figure 84 shows the SSP states related to primitive transmission.

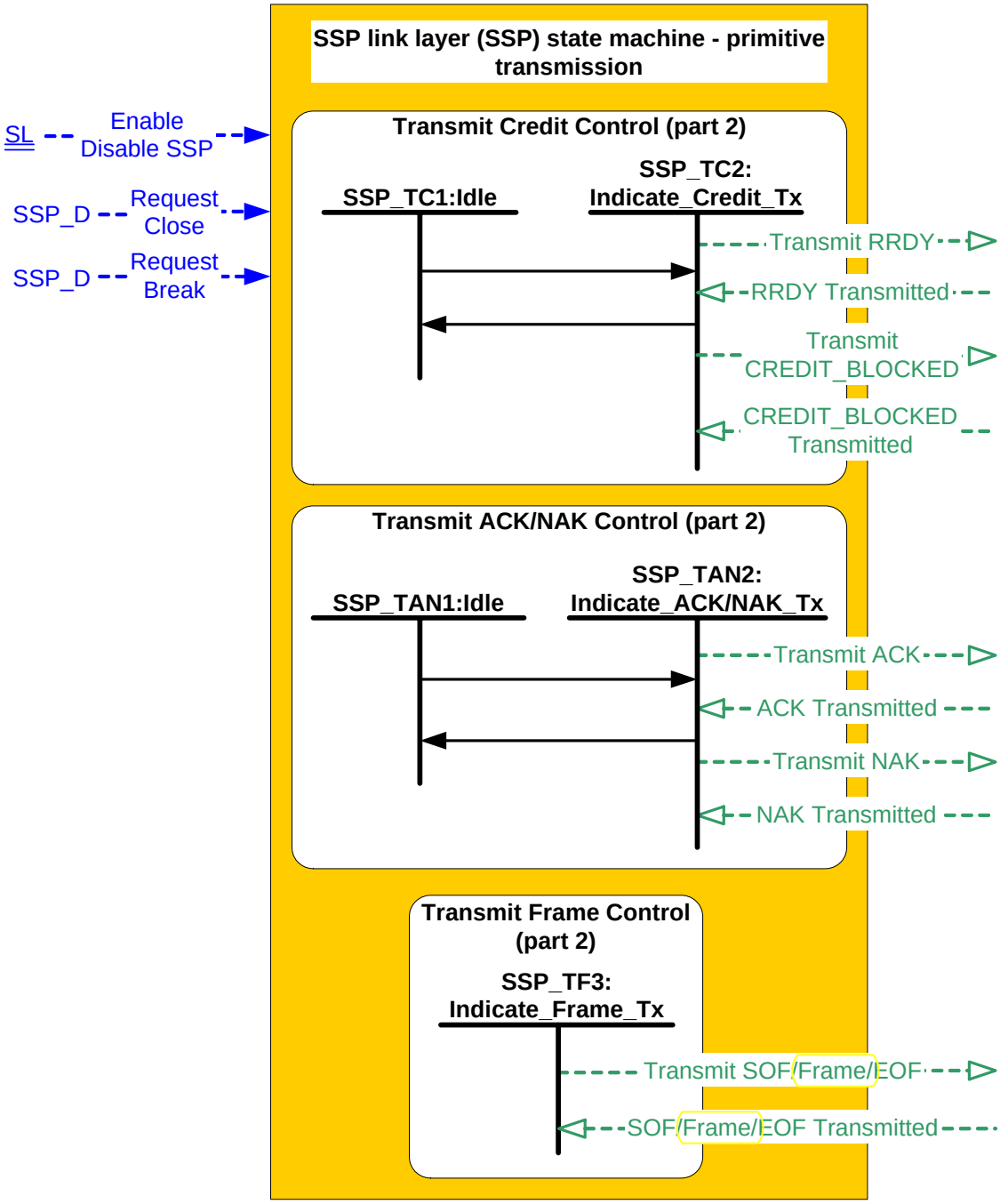


Figure 84 — SSP link layer (SSP) state machines (part 3 - primitive transmission)

7.16.7.2 SSP_TIM1:Tx_Interlock_Monitor state

This state monitors the number of frames transmitted and monitors the number of ACKs and NAKs received. This state ensures that an ACK or NAK is received for each frame transmitted and indicates a timeout if they are not.

The EOF Transmitted parameter shall be used by this state to count the number of frames transmitted.

When the number of EOF Transmitted parameters received equals the number of ACK Received and NAK Received parameters received then there is ACK/NAK balance and this state shall send the Tx Balanced Status (Balanced) parameter to the SSP_TF2:Tx_Wait state. When the number of EOF Transmitted

parameters received does not equal the number of ACK Received and NAK Received parameters received then this there is ACK/NAK unbalance and this state shall send the Tx Balanced Status (Unbalanced) parameter to the SSP_TF2:Tx_Wait state.

If there is an ACK/NAK unbalance and an ACK Received parameter is received this state shall:

a) use the ACK Received parameter to count the number of ACKs and NAKs received; and
b) send an ACK Received confirmation to the port layer each time the ACK Received parameter is received.

If there is an ACK/NAK unbalance and an NAK Received parameter is received this state shall:

a) use the NAK Received parameter to count the number of ACKs and NAKs received; and
b) send an NAK Received confirmation to the port layer each time the NAK Received parameter is received.

If there is ACK/NAK balance, the ACK Received parameter and NAK Received parameter shall be ignored.

If there is ACK/NAK balance the ACK/NAK timeout timer shall be disabled.

Each time an ACK/NAK unbalance occurs, the ACK/NAK timeout timer shall be initialized to 1 ms and shall start timing. The timer shall be re-initialized to 1 ms each time an ACK or NAK is counted. If the timer expires, this state 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 the SSP_TIM state machine receives an Enable Disable SSP (Enable) parameter, Request Close parameter, or Request Break parameter the number of frames transmitted shall be set to the number of ACKs and NAKs received.

7.16.7.3 SSP_TCM1:Tx_credit_monitor state

This state shall keep track of the number of transmit frame credits received versus the number of transmit frame credits used. This state adds transmit frame credit for each RRDY Received parameter received and subtracts transmit frame credit for each Tx Credit Used parameter received. This state shall remember any CREDIT_BLOCKED Received parameter that is received.

When transmit frame credit is available, this state shall send the Tx Credit Status (Credit Available) parameter to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and credit is not blocked, this state shall send the Tx Credit Status (Credit Not Available) parameter to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and credit is blocked, this state shall send the Tx Credit Status (Credit Blocked) parameter to the SSP_TF2:Tx_Wait state.

When the SSP_TCM state machine receives an Enable Disable SSP (Enable) parameter, Request Close parameter, or Request Break parameter transmit frame credit shall be set to not available and credit shall not be blocked.

7.16.7.4 SSP_D1:DONE_Wait state

7.16.7.4.1 State description

This state 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 parameter has been received before the Wait For DONE Parameter is received, this state shall send the Request Close parameter to the SL state machine (see 7.13) and all the SSP state machines.

If the DONE Received parameter has not been received when the Wait For DONE Parameter is received, this state shall initialize the DONE timeout timer to 1 ms. If the Wait For DONE (Close Connection) parameter or the Wait for DONE (Credit Timeout) parameter was received, the DONE timeout timer shall be re-initialized to

1 ms each time the EOF Received parameter is received. If the Wait For DONE (ACK/NAK Timeout) parameter was received the DONE timeout timer shall not be re-initialized.

If the DONE Received parameter is received before the DONE timeout timer expires, this state shall send the Request Close parameter to the SL state machine and all the SSP state machines.

If the DONE Received parameter is not received before the DONE timeout timer expires, this state shall:

a) send a DONE Timeout confirmation to the port layer; and
b) send a Request Break parameter to the SL state machine and all the SSP state machines.

Any time a DONE Received parameter is received this state shall send a DONE Received confirmation to the port layer. A DONE (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 may be used by the application layer to decide when to reuse tags.

7.16.7.5 SSP_TF1:Connected_idle state

7.16.7.5.1 State description

This state waits for a request ~~from the port layer~~ to transmit a frame or to close the connection.

7.16.7.5.2 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 (Balanced) request was received this transition shall pass a Transmit Balanced Frame argument to the Tx_Wait state.

If a Tx Frame (Nonbalanced) request was received this transition shall pass a Transmit Unbalanced Frame argument to the Tx_Wait state.

If a Close Connection request was received this transition shall pass a Close Connection argument to the Tx_Wait state.

7.16.7.5.3 Transition SSP_TF1:Connected_Idle to SSP_TF4:Indicate_Done_Tx

This transition shall occur if an ACK/NAK Timeout parameter is received. This transition shall pass an ACK/NAK Timeout argument to the Tx_Wait state.

7.16.7.6 SSP_TF2:Tx_Wait state

7.16.7.6.1 State description

This state monitors the Tx Balanced Status parameter and the Tx Credit Status parameter 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 an argument of Transmit Balanced Frame or Transmit Nonbalanced Frame, and:

a) if the last Tx Credit Status parameter received had an argument of Not Available this state shall initialize the Credit timeout timer to 1 ms; or
b) if the last Tx Credit Status parameter had an argument other than Not Available this state shall disable the Credit timeout timer.

7.16.7.6.2 Transition SSP_TF2:Tx_Wait to SSP_TF3:Indicate_Frame_Tx

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Balanced Frame if:

a) the last Tx Balanced Status parameter received had an argument of Balanced; and
b) the last Tx Credit Status parameter received had an argument of Credit Available.

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Nonbalanced Frame if the last Tx Credit Status parameter received had an argument of Credit Available.

This transition shall occur after sending a Tx Credit Used parameter to the SSP_TCM1:Tx_Credit_Monitor state.

7.16.7.6.3 Transition SSP_TF2:Tx_Wait to SSP_TF4:Indicate_Done_Tx

This transition shall pass a Connection Closed argument to the Indicate_Done_Tx state if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Close Connection and the last Tx Balanced Status parameter received had an argument of Balanced.

This transition shall pass a Credit Timeout argument to the Indicate_Done_Tx state if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Balanced Frame or Transmit Nonbalanced Frame and the last Tx Credit Status parameter received had an argument of Blocked.

This transition shall pass a Credit Timeout argument to the Indicate_Done_Tx state if:

a) this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Balanced Frame or Transmit Nonbalanced Frame;
b) the Credit timeout timer expires before a Tx Credit Status (Available) parameter is received; and
c) the last Tx Balanced Status parameter received had an argument of Balanced or a Tx Balanced Status (Balanced) is received before a ACK/NAK Timeout parameter is received.

This transition shall pass an ACK/NAK Timeout argument to the Indicate_Done_Tx state if an ACK/NAK Timeout parameter is received.

7.16.7.7 SSP_TF3:Indicate_Frame_Tx state

7.16.7.7.1 State description

This state shall request a frame transmission by sending a Transmit SOF/frame/EOF parameter to the SSP transmitter. Each time a Transmit SOF/frame/EOF parameter is sent to the SSP transmitter, one SOF/frame/EOF is transmitted.

In this state receiving an SOF/frame/EOF Transmitted parameter indicates that the frame has been transmitted.

7.16.7.7.2 Transition SSP_TF3:Indicate_Frame_Tx to SSP_TF1:Connected_idle

This transition shall occur after:

a) receiving an SOF/frame/EOF Transmitted parameter;
b) sending an EOF Transmitted parameter to the SSP_TIM1:Tx_Interlock_Monitor; and
c) sending a Frame Transmitted confirmation to the port layer.

7.16.7.8 SSP_TF4:Indicate_Done_Tx state

This state shall request a DONE be transmitted by sending one of the following Transmit DONE parameters to a SSP transmitter:

a) Transmit DONE (CLOSE CONNECTION) parameter if this state was entered from the Tx_Wait state with an argument of Close Connection;
b) Transmit DONE (ACK/NAK TIMEOUT) parameter if this state was entered from the Tx_Wait state or the Connected_Idle state with an argument of ACK/NAK Timeout; or
c) Transmit DONE (CREDIT TIMEOUT) parameter if this state was entered from the Tx_Wait state with an argument of Credit Timeout.

After a DONE transmitted parameter is received this state shall send the DONE Transmitted confirmation to the port layer and send the:

a) Wait For DONE (Close Connection) parameter if this state was entered from the Tx_Wait state with an argument of Close Connection;
b) Wait For DONE (ACK/NAK TIMEOUT) parameter if this state was entered from the Tx_Wait state or the Connected_Idle state with an argument of ACK/NAK Timeout; or
c) parameter if this state was entered from the Tx_Wait state with an argument of Credit Timeout.

7.16.7.9 SSP_RF1:Rcv_Frame state

This state:

a) checks the frame to determine if the frame should be accepted or discarded by the link:
b) checks the frame to determine if an ACK or NAK should be transmitted; and
c) sends a Received Frame confirmation to the port layer.

The frame (i.e., all the dwords between an SOF and EOF) shall be discarded if any of the following conditions are true:

a) the number of data dwords between the SOF and EOF is less than 7;
b) the number of data dwords after the SOF is greater than 263 data dwords;
c) the Rx Credit Status (Credit Exhausted) parameter is received; or
d) the DONE Received parameter is received.

If consecutive SOF Received parameters are received without an intervening EOF Received parameter (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 all dwords between those SOFs.

If the frame is discarded then no further action is taken by this state. If the frame is not discarded then this state shall:

a) send a Frame Received parameter to the SSP_RCM1:Rcv_Credit_Monitor state; and
b) send a Frame Received parameter to the SSP_RIM1:Rcv_Interlock_Monitor state;

If the frame CRC is good and the frame contained no invalid data dwords, this state shall send the Frame Received (Successful) parameter to the SSP_TAN1:Idle state and:

a) if the last Rx Credit Status parameter received had an argument of Credit Extended send the Frame Received (ACK/NAK Balanced) confirmation to the port layer; or
b) if the last Rx Credit Status parameter received had an argument of Credit Exhausted send a Frame Received (ACK/NAK Unbalanced) confirmation to the port layer.

If the frame CRC is bad or the frame contained invalid data dwords, this state shall send the Frame Received (Unsuccessful) parameter to the SSP_TAN1:Idle state.

7.16.7.10 SSP_RCM1:Rcv_Credit_Monitor state

This state 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 this state shall send the Rx Credit Status (Available) parameter to the SSP_TC1:Idle state. This state shall only send the Rx Credit Status (Available) parameter to the SSP_TC1:Idle state after frame receive resources become available. The specifications for when or how resources become available is outside the scope of this standard.

This state may send the Rx Credit Status (Blocked) parameter to the SSP_TC1:Idle state indicating that no more credit is going to be sent during this connection. After sending the Rx Credit Status (Blocked) parameter to the SSP_TC1:Idle state, this state shall not send the Rx Credit Status (Available) parameter to the SSP_TC1:Idle state for the duration of the current connection. The Rx Credit Status (Blocked) parameter should be sent to the SSP_TC1:Idle state when no further credit is going to become available within a credit timeout (i.e., less than 1 ms).

This state shall indicate through the Rx Credit Status parameter only the amount of resources available to handle received frames (e.g., if this state has resources for 5 frames the maximum number of Rx Credit Status requests with the Available argument outstanding is 5).

This state shall use the RRDY Transmitted parameter to keep track of the number of RRDYs transmitted. This state shall use the Frame Received parameter to keep a track of the number of frames received.

Any time the number of RRDY Transmitted parameters received exceeds the number of Frame Received parameters received this state shall send a Rx Credit Status (Credit Extended) parameter to the SSP_RF1:Rcv_Frame state.

Any time the number of RRDY Transmitted parameters received equals the number of Frame Received parameters received this state shall send a Rx Credit Status (Credit Exhausted) parameter to the SSP_RF1:Rcv_Frame state.

If this state receives an Enable Disable SSP (Enable) parameter, Request Close parameter, or Request Break parameter the frame receive resources shall be initialized to the no credit value for the current connection.

7.16.7.11 SSP_RIM1:Rcv_Interlock_Monitor state

This state monitors the number of frames received versus the number of ACKs and NAKs transmitted.

This state shall use the ACK Transmitted parameter and the NAK Transmitted parameter ~~from the SSP_TAN1:Idle state~~ to keep track of the number of ACKs and NAKs transmitted. This state shall use the Received Frame parameter ~~from the SSP_RF1:Rcv_Frame state~~ to keep a track of the number of frames received.

Any time the number of the ACK Transmitted parameters and the number of NAK Transmitted parameters received equals the number of Received Frames parameters received this state shall send the Rx Balanced Status (Balanced) parameter to the SSP_RF1:Rcv_Frame state.

Any time the number of the ACK Transmitted parameters and the number of NAK Transmitted parameters received does not equal the number of Received Frames parameters received this state shall send the Rx Balanced Status (Unbalanced) parameter to the SSP_RF1:Rcv_Frame state.

When the SL state machine sends the Enable Disable SSP (Enable) parameter the number of the ACKs and NAKs transmitted shall be set to the number of frames received.

7.16.7.12 SSP_TC1:Idle state

7.16.7.12.1 State description

This state waits for a Rx Credit Status parameter to be received.

When this state is entered from the SSP_TC2:Indicate_Credit_Tx state with an argument of RRDY Transmitted it shall send an RRDY Transmitted parameter to the SSP_RCM1:Rcv_Credit_Monitor.

7.16.7.12.2 Transition SSP_TC1:Idle to SSP_TC2:Indicate_Credit_Tx

This transition shall pass a Transmit RRDY argument to the Indicate_Credit_Tx state if a Rx Credit Status (Available) parameter was received.

7.16.7.13 SSP_TC2:Indicate_Credit_Tx state

7.16.7.13.1 State description

If this state is entered into from the SSP_TC1 state with an argument of Transmit RRDY, this state shall request a single RRDY be transmitted by sending a Transmit RRDY parameter to the SSP transmitter.

If this state is entered into from the SSP_TC1 state with an argument of CREDIT_BLOCKED, this state shall request a single CREDIT_BLOCKED by sending a Transmit CREDIT_BLOCKED parameter to the SSP transmitter.

7.16.7.13.2 Transition SSP_TC2:Indicate_Credit_Tx to SSP_TC1:Idle

This transition shall occur after receiving an RRDY Transmitted parameter or the CREDIT_BLOCKED Transmitted parameter.

7.16.7.14 SSP_TAN1:Idle state

7.16.7.14.1 State description

This state waits for a Frame Received parameter ~~from the SSP_RF1:Rcv_Frame state.~~

When this state is entered into from the SSP_TAN2:Indicate_ACK/NAK_Tx state with an ACK Transmitted argument this state shall:

a) send an ACK Transmitted parameter to the SSP_RIM1:Rcv_Interlock_Monitor state: and
b) send an ACK Transmitted confirmation to the port layer.

When this state is entered into from the SSP_TAN2:Indicate_ACK/NAK_Tx state with a NAK Transmitted argument it shall send a NAK Transmitted parameter to the SSP_RIM1:Rcv_Interlock_Monitor state.

7.16.7.14.2 Transition SSP_TAN1:Idle to SSP_TAN2:Indicate_ACK/NAK_Tx

This transition shall pass a Transmit ACK argument to the Indicate_ACK/NAK_Tx state if a Frame Received (Successful) parameter was received.

This transition shall pass a Transmit NAK argument to the Indicate_ACK/NAK_Tx state if a Frame Received (Unsuccessful) parameter was received.

If multiple Frame Received (Unsuccessful) parameters and Frame Received (Successful) parameters are received, then the order in which the Transmit ACK arguments and Transmit NAK arguments are passed to the Indicate_ACK/NAK_Tx state shall be the same order as the Frame Received (Unsuccessful) parameters and Frame Received (Successful) parameters were received.

7.16.7.15 SSP_TAN2:Indicate_ACK/NAK_Tx state

7.16.7.15.1 State description

If this state is entered into from the SSP_TAN1 state with an argument of Transmit ACK, this state shall request a single ACK be transmitted by sending a Transmit ACK parameter to the SSP transmitter.

If this state is entered into from the SSP_TAN1 state with an argument of Transmit NAK, this state shall request a single NAK be transmitted by sending a Transmit NAK parameter to the SSP transmitter.

7.16.7.15.2 Transition SSP_TAN2:Indicate_ACK/NAK_tx to SSP_TAN1:Idle

This transition shall occur after receiving an ACK Transmitted parameter or the NAK Transmitted parameter.

7.17 STP link layer

7.17.1 STP frame transmission

STP frame transmission is defined by SATA. During an STP connection, frames are preceded by SATA_SOF and followed by SATA_EOF as shown in figure 85.

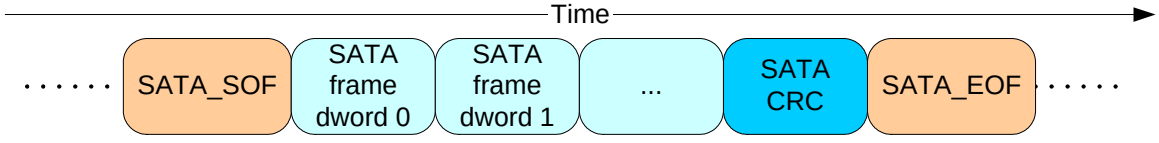


Figure 85 — STP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.4).

STP encapsulates the SATA protocol with connection management.

Table 84 shows a target port transmitting a SATA frame to an expander port. The expander device opens a connection to an STP initiator port or to an expander port on the path to the STP initiator port solely for the frame.

Table 84 — SATA target port transmitting a frame

SATA target port to expander port	Expander port to expander port or STP initiator port
SATA_SYNC	idle dword
SATA_X_RDY	OPEN address frame
<repeats>	SATA_X_RDY
<repeats>	<repeats>
<wait for SATA_R_RDY>	<wait for SATA_R_RDY>
SATA_SOF	SATA_SOF
FIS	FIS
CRC	CRC
SATA_EOF	SATA_EOF
SATA_WTRM	SATA_WTRM
SATA_SYNC	CLOSE

Table 85 shows an STP initiator port transmitting a frame, with the expander device attached to the SATA target port opening a connection solely for the frame.

Table 85 — STP initiator port transmitting a frame

STP initiator port to expander port	Expander port to SATA target port
idle dword	SATA_SYNC
OPEN address frame	SATA_SYNC
SATA_X_RDY	SATA_X_RDY
<repeats>	<repeats>
<wait for SATA_R_RDY>	<wait for SATA_R_RDY>
SATA_SOF	SATA_SOF
FIS	FIS
CRC	CRC
SATA_EOF	SATA_EOF
SATA_WTRM	SATA_WTRM SATA_SYNC (after expander chooses to close)
CLOSE	SATA_SYNC

Other primitives may be interspersed during the connection as defined by SATA.

The expander device adds the OPEN address frame and CLOSE on the expander port attached to the STP initiator. The expander device removes the OPEN address frame and CLOSE on the expander port attached to the SATA target port. While the connection is open, the expander device is not involved. Both initiator port and target port use SATA protocol, with SATA_HOLD/SATA_HOLDA flow control, while the connection is open.

7.17.2 STP flow control

Each expander device along the pathway of an STP connection between an STP initiator port and a SATA target device shall follow the SATA SATA_HOLD/SATA_HOLDA flow control protocol to implement flow control across every physical link in the pathway along the connection. The requirement applies to transfers in either direction. If an expander device receives a SATA_HOLD from the downstream phy receiving a stream of dwords from the expander device, the expander device shall stop transmitting data to the downstream phy and shall transmit SATA_HOLDA to the downstream phy. The number of dwords transmitted to the downstream phy between the time the downstream phy sends SATA_HOLD and the time the expander device sends SATA_HOLDA to the downstream phy shall be no more than 20.

Upon receiving a SATA_HOLD from the downstream phy, an expander device shall propagate the SATA_HOLD to the next upstream phy along the pathway of the connection, and shall accept up to 20 more dwords from the upstream phy before receiving SATA_HOLDA from the upstream phy. If the expander device issues SATA_HOLDA to the downstream phy before receiving SATA_HOLDA from the next upstream phy along the pathway of the connection, the expander device shall maintain dwords received from the upstream phy during that interval in an internal buffer until the downstream phy is again ready to receive dwords. The expander device may delay sending of SATA_HOLDA to the downstream phy to minimize the number of dwords it must store in an internal buffer if it can do so without exceeding the 20 dword SATA_HOLDA turnaround time on the link to the downstream phy.

Figure 86 shows the propagation of SATA_HOLD upstream through the pathway of an STP connection, the SATA_HOLDA reply from each expander device along the pathway, and the interval during which each expander device must accept incoming data dwords into a buffer. In this case the time to propagate SATA_HOLD to the next upstream phy is hidden behind the processing time to turnaround SATA_HOLDA to

the downstream phy, so the expander device need only accept up to 20 additional data dwords to account for the SATA_HOLDA turnaround time of the next upstream phy.

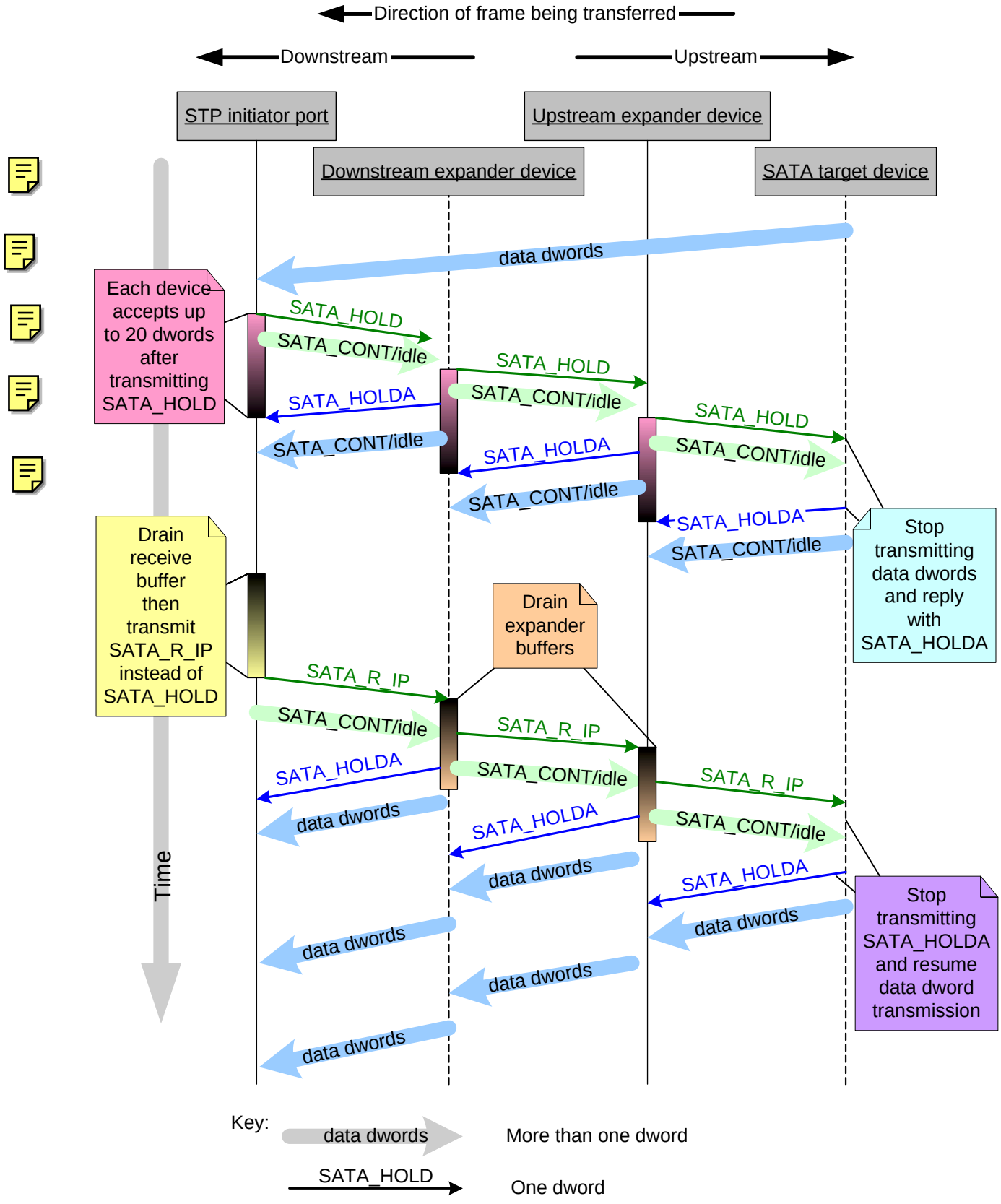


Figure 86 — STP expander device buffering and HOLD/HOLDA handshake

7.17.3 Preparing to close an STP connection

STP initiator ports or expander devices may originate closing an STP connection. An initiator or expander device shall not originate closing a STP connection after sending a SATA_X_RDY or SATA_R_RDY until after both sending and receiving SATA_SYNC. An initiator or expander device shall transmit CLOSE after receiving a CLOSE.

In a SCSI domain with a single initiator port, when a SATA target port transmits an SATA_X_RDY, the expander device may use the time between SATA_X_RDY and SATA_R_RDY to insert an OPEN address frame to open a connection to the initiator port. In a SAS domain with multiple initiator ports with command-tag queuing supported by the target ports, the expander device should accept the start of the frame including the tag, use SATA_HOLD to stall the rest of the frame while the expander device opens a connection to the proper initiator port, and release SATA_HOLD when the connection is ready and the start of the frame has been sent. Only data FISes are subject to flow control, so the expander device shall be capable of accepting a whole register FIS frame.

An expander device may issue CLOSE at the end of each frame, after a timeout waiting for another frame, after every n frames, after a certain time period, after a SATA_CONT is detected, after a SATA_HOLD is detected.

An expander device shall break an STP connection if the SATA physical link loses dword synchronization.

See 7.12.7 for details on closing connections.

7.17.4 STP link layer (STP) state machines

The STP link layer uses the link layer state machines defined in SATA.

7.18 SMP link layer

7.18.1 SMP frame transmission

Inside an SMP connection, the source device transmits a single SMP_REQUEST frame and the destination device responds with a single SMP_RESPONSE frame (see 9.4).

Frames are surrounded by SOF and EOF as shown in figure 87. There is no acknowledgement of SMP frames with ACK and NAK. There is no credit exchange with RRDY.

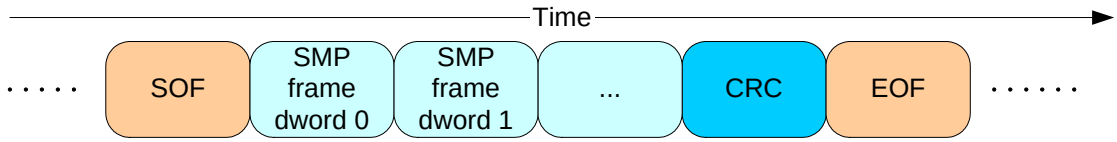


Figure 87 — SMP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.4). The link layer shall check that the number of data dwords between the SOF and the EOF is at least 8 bytes and that the CRC is valid.

7.18.2 SMP flow control

By accepting an SMP connection, the destination device indicates it is ready to receive one SMP_REQUEST frame.

When the source device transmits one SMP_REQUEST frame, it shall be ready to receive one SMP_RESPONSE frame.

7.18.3 Preparing to close an SMP connection

After receiving the SMP_RESPONSE frame, the source device shall transmit a CLOSE to close the connection.

After transmitting the SMP_RESPONSE frame, the destination device shall reply with a CLOSE.

~~The source device and destination device may leave the connection open to run loopback tests (see 7.10).~~

See 7.12.7 for details on closing connections.

7.18.4 SMP link layer (SMP) state machines

7.18.4.1 Overview

The SMP link layer state machines run in parallel to control the flow of dwords on the link during an SMP connection. The SMP link state machines are as follows:

a) SMP Initiator Link (SMP_IL) - in SMP initiator devices; and
b) SMP Target Link (SMP_TL) - in SMP target devices;

All the SMP state machines shall begin on receipt of an Enable Disable SMP (Enable) parameter ~~from the SL state machine~~ (see 7.13).

All the initiator device state machines within SMP shall stop after:

a) receiving an Enable Disable SSP (Disable) parameter ~~from the SL state machine;~~
b) receiving a Request Close parameter ~~from the SMP_IL3:Rcv_response_Frame state~~ indicating that the connection has been be closed; or
c) receiving a Request Break parameter ~~from the SMP_IL3:Rcv_response_Frame state~~ indicating that a BREAK has been transmitted.

All the target device state machines within SMP shall stop after:

a) receiving an Enable Disable SSP (Disable) parameter ~~from the SL state machine;~~
b) receiving a Request Close parameter ~~from the SMP_TL2:Wait_transmit_frame~~ state indicating that the connection has been be closed; or
c) receiving a Request Break parameter ~~from the SMP_TL1:Wait_originate_frame state~~ indicating that a BREAK has been transmitted.

The SMP_IL state machine's function is to transmit an SMP request frame and then receive the corresponding response frame. The SMP_IL state machine contains the following states:

a) SMP_IL1:Command_idle (see 7.18.4.2)(initial state);
b) SMP_IL2:Indicate_frame_tx (see 7.18.4.2.2);
c) SMP_IL3:Rcv_response_frame (see 7.18.4.2.3).

The SMP_TL state machine's function is to receive an SMP request frame and then transmit the corresponding SMP response frame. The SMP_TL state machine contains the following states:

a) SMP_TL1:Wait_orginate_frame (see 7.18.4.3.1)(initial state); and
b) SMP_TL2:Wait_transmit_frame (see 7.18.4.3.2).

Figure 88 shows the SMP state machines implemented by initiator devices.

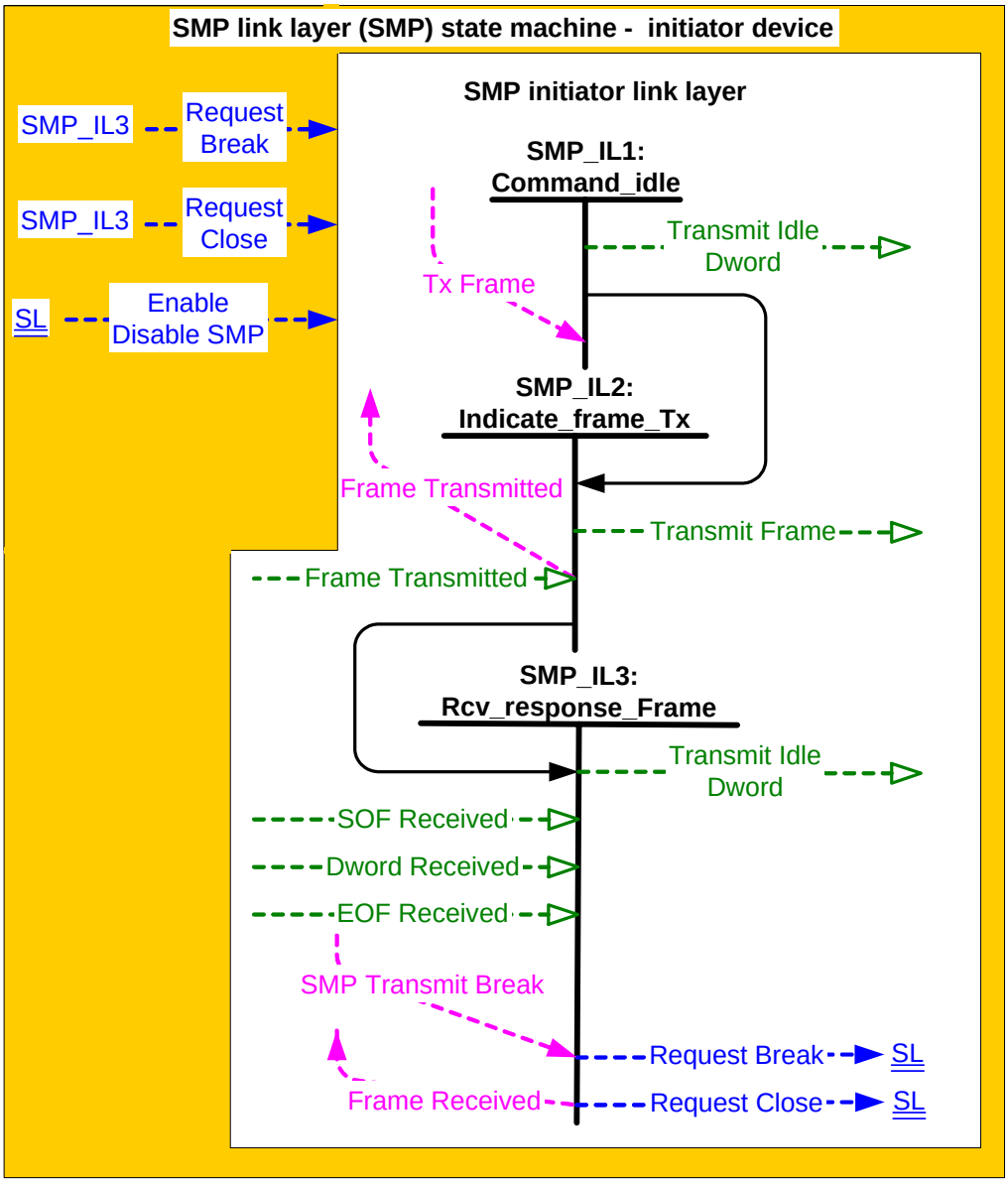


Figure 88 — SMP link layer (SMP) state machines – initiator device

Figure 89 shows the SMP state machines implemented by target devices.

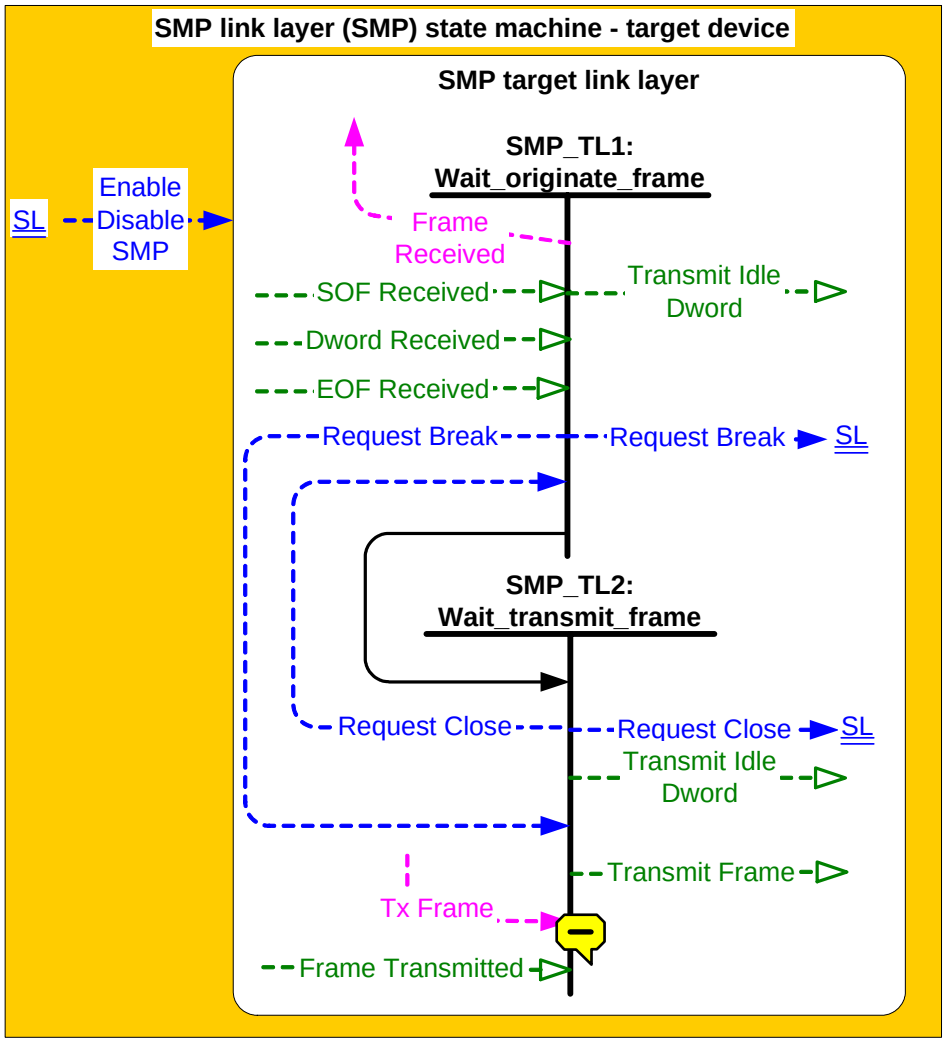


Figure 89 — SMP link layer (SMP) state machines – target device

7.18.4.2 SMP Initiator Link state machine

7.18.4.2.1 SMP_IL1:Command_idle state

7.18.4.2.1.1 State description

This state is the initial state and is the state that is used when the SMP state machine is activated ~~and there is no active connection.~~

This state shall request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SMP transmitter.

7.18.4.2.1.2 Transition SMP_IL1:Command_idle to SMP_IL2:Indicate_frame_tx

This transition shall occur after a Tx Frame (SMP) request is received.

7.18.4.2.2 SMP_IL2:Indicate_frame_tx state

7.18.4.2.2.1 State description

This state shall request an SMP frame be transmitted by sending a Transmit Frame parameter to the SMP transmitter.

After the Frame Transmitted parameter is received, this state shall send a Frame Transmitted confirmation to the port layer.

7.18.4.2.2.2 Transition SMP_IL2:Indicate_frame_tx to SMP_IL3:Rcv_response_frame

This transition shall occur after sending a Frame Transmitted confirmation to the port layer.

7.18.4.2.3 SMP_IL3:Rcv_response_frame state

7.18.4.2.3.1 State description

This state checks the SMP response frame and determines if the SMP response frame was successfully received (e.g., no CRC error).

If the SMP response frame is received with a CRC error, this state shall send a Frame Received (SMP Failure) confirmation to the port layer.

If the number of dwords between the SOF and EOF of the SMP response frame is less than 2, or the number of dwords after an SOF is greater than 258, this state shall send a Frame Received (SMP Failure) confirmation to the port layer. 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 (SMP) confirmation to the port layer
b) send a Request Close parameter to all the SMP_IL states; and
c) send a Request Close parameter to the SL state machine (see 7.13).

If a SMP Transmit Break request is received ~~from the port layer~~ this state shall send a Request Break parameter to the SL state machine and send a Request Break parameter to all the SMP_IL states.

This state shall request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SMP transmitter.

7.18.4.3 SMP Target Link state machine

7.18.4.3.1 SMP_TL1:Wait_originate_frame state

7.18.4.3.1.1 State description

This state waits for an SMP request frame and determines if the SMP request frame was successfully received (e.g., no CRC error).

If the SMP request frame is received with a CRC error this state shall send a Request Break parameter to the SL state machine (see 7.13) and the SMP_TL2:Wait_transmit_frame state.

If the number of data dwords between the SOF and EOF is less than 2, or the number of dword after the SOF is greater than 258, this state shall send a Request Break parameter to the SL state machine to indicate that a BREAK shall be transmitted.

This state shall request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SMP transmitter.

7.18.4.3.1.2 Transition SMP_TL1:Wait_originate_frame to SMP_TL2:Wait_transmit_frame

This transition shall occur after a valid SMP request frame is received and sending a Frame Received (SMP) confirmation to the port layer.

An SMP request frame shall be valid if:

a) the number of dwords between the SOF and EOF is greater than or equal to 2 and less than or equal to 258; and
b) the CRC is valid.

7.18.4.3.2 SMP_TL2:Wait_transmit_frame state

7.18.4.3.2.1 State description

This state shall:

1) After a Tx Frame (SMP) request is received, request an SMP response frame be transmitted by sending a Transmit Frame parameter to the SMP transmitter; then
2) After receiving a Frame Transmitted parameter, send a Request Close parameter to the SL state machine (see 7.13) and to the SMP_LT1:Wait_originate_frame state.

After sending Transmit Frame parameter to the SMP transmitter, this state shall request idle dwords be transmitted by repeatedly sending a Transmit Idle Dword parameter to the SMP transmitter.

8 Port layer

8.1 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.

Figure 90 shows the relationship of the port layer to the transport and link layers.

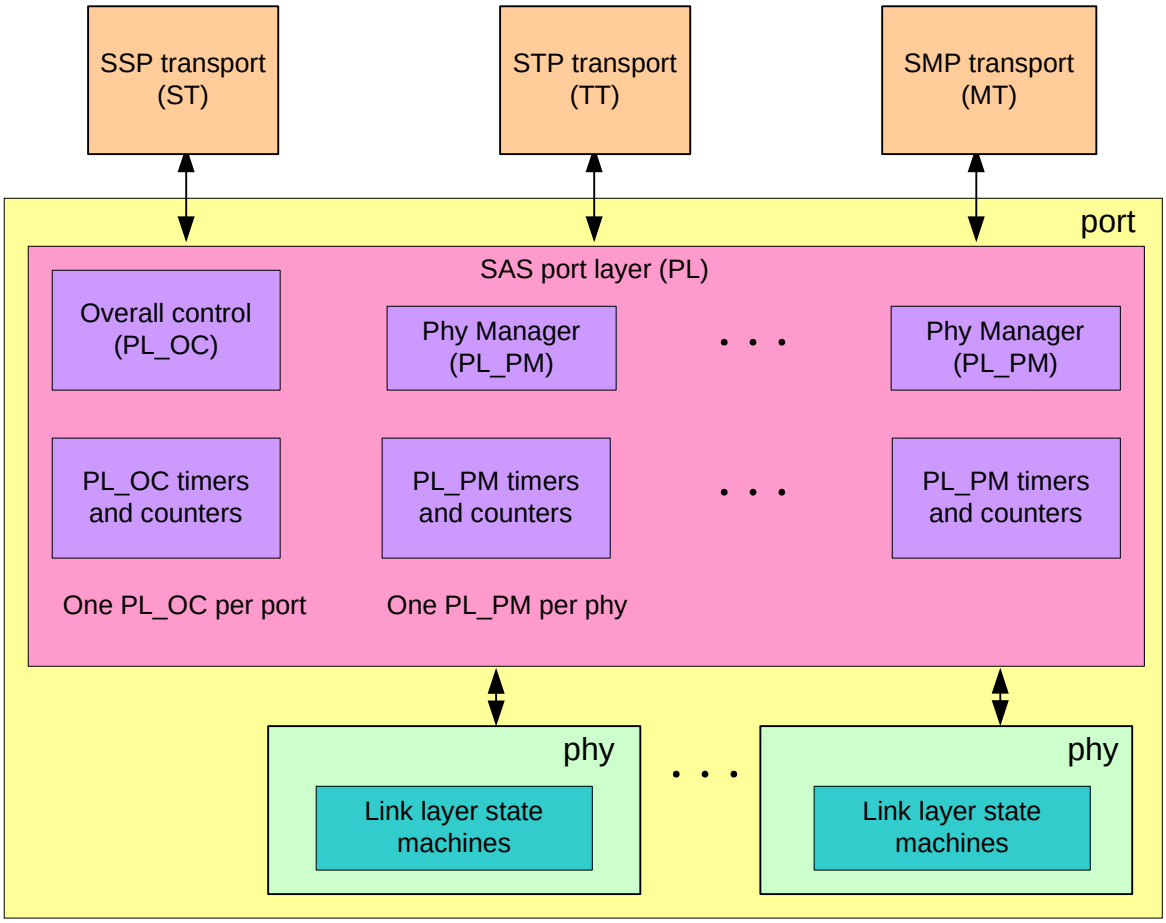


Figure 90 — Port layer state machines

Two state machines running in parallel plus several commonly accessible timers and counters form the port layer.

The PL state machines are:

a) Overall Control (PL_OC state machine); and
b) Phy Manager (PL_PM state machine).

There is one PC_OC state machine per port. There is one PL_PM state machine for each phy that is part of the port. The management application layer determines how phys are assigned to ports.

8.2 ~~Port layer~~ timers and counters

8.2.1 Timers and counters overview

The port layer maintains the following timers and counters for each phy (i.e., for each PL_PM state machine):

a) bus inactivity time limit timer (for SSP target ports only);
b) maximum connect timer (for SSP target ports only);
c) I_T nexus loss timer (for SSP target ports only);
d) arbitration wait time (AWT) timer; and
e) pathway blocked count (PBC) counter.

The port layer maintains the following overall timers (i.e., for the PL_OC state machine):

a) an AWT timer for each connection attempt to be retried; and
b) an I_T nexus loss timer for each destination for which a connection is requested.

8.2.2 Bus inactivity time limit timer

The bus inactivity time limit timer is initialized and monitored by the PL_PM state machine for SSP connections in SSP target ports. The bus inactivity time limit timer is initialized to the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.1.6.1). The timer shall count down. When the timer count has decremented to zero, the timer shall stop counting. A count of zero signifies that the timer has expired.

8.2.3 Maximum connect time limit timer

The maximum connect time limit timer is initialized and monitored by the PL_PM state machine for SSP connections in SSP target ports. The maximum connect time limit timer is initialized to the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.1.6.1). The timer shall count down. When the timer count has decremented to zero, the timer shall stop counting. A count of zero signifies that the timer has expired.

8.2.4 I_T nexus loss timer

The I_T nexus loss timer is implemented for SSP connections. It is either:

a) initialized to the I_T NEXUS LOSS TIME field in Protocol-Specific Port Control mode page (see 10.1.6.2), started counting, and assigned a running status. When running, the timer shall count down. When the timer count is decremented to zero, the timer shall stop counting and assigned an expired status;
b) initialized to a count of zero, stopped counting, and assigned a stopped status; or
c) initialized with received I_T nexus loss time and status arguments (i.e., stopped, running, or expired). If a I_T nexus loss timer is initialized with a stopped status, the I_T nexus loss timer shall remain stopped. If a I_T nexus loss timer is initialized with a running status, the I_T nexus loss timer shall start counting from the initialized I_T nexus loss time. If a I_T nexus loss timer is initialized with an expired status, the I_T nexus loss timer shall remain stopped and shall retain the expired status.

The I_T nexus loss timer provides the transport and application layers the information to enforce I_T nexus loss timeouts for:

a) connection attempts which fail for either:
   A) Open Failed (Connection Rate Not Supported);
   B) Open Failed (No Destination); or
   C) Open Failed (Open Timeout Occurred) confirmations;

   or

b) Open Failed (Pathway Blocked) confirmations received from the link layer if one of these confirmations is received from the link layer prior to receiving an Open Failed (Pathway Blocked) confirmation.
   A) Open Failed (Connection Rate Not Supported);
   B) Open Failed (No Destination); or
   C) Open Failed (Open Timeout Occurred) confirmation

If only Open Failed (Pathway Blocked) confirmations are received, the I_T nexus loss timer does not start running and the I_T nexus loss timeout limit is not applicable.

8.2.5 Arbitration wait time (AWT) timer

The AWT timer (see 7.12.3) value is passed to the link layer by the PL_PM state machine as an argument in its Open Connection request. The AWT timer is initialized to zero. The AWT timer shall count up.

8.2.6 Pathway blocked count (PBC) counter

The PBC counter (see 7.12.3) value is passed to the link layer by the PL_PM state machine as an argument in its Open Connection request. The PBC counter is initialized to zero. The PBC counter shall count up. The PBC counter shall not be incremented past FFh.

8.3 Port layer overall control (PL_OC) state machine

8.3.1 ~~Overview~~

A single PL_OC state machine exists for the port layer. The PL_OC state machine's purpose is to:

a) Select a Transmit Frame request from the SSP, SMP and STP transport layers or a Retry Frame parameter from the PL_PM state machines and select a phy on which to transmit the frame;
b) Close connections if needed;
c) Direct the PL_PM state machine to open if needed and transmit the frame;
d) Forward the response back to the transport layer;
e) Process the Cancel, SMP Transmit Break, and Accept_Reject Opens requests from the transport layer; and
f) Initialize and operate the I_T nexus loss timers and the AWT timers.

The PL_OC state machine contains these states:

a) PL_OC1:Idle; and
b) PL_OC2:Overall_Control.

The state machine shall start in the PL_OC1:Idle state. The state machine shall transition to the PL_PM1:Idle state from any other state after receiving a Phy Enabled confirmation from any phy assigned to the port.

Figure 91 shows the PL_OC state machine.

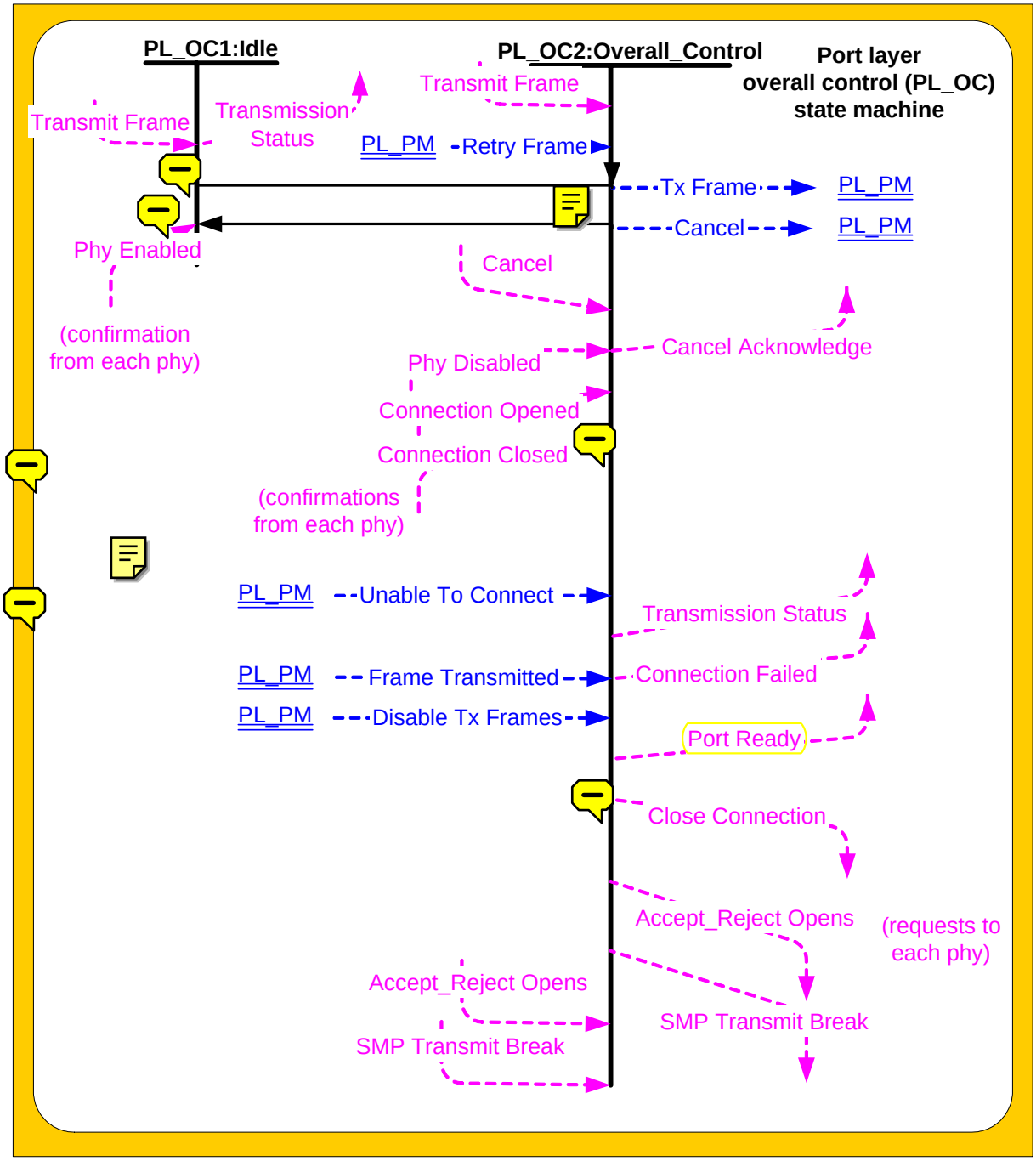


Figure 91 — Port layer overall control (PL_OC) state machine

8.3.2 ~~8.3.2~~ PL_OC1:Idle state

8.3.2.1 ~~8.3.2.1~~ State description

This state waits for the management application layer to notify it that the port is enabled.

This state shall send a Transmission Status (No Phys In Port) confirmation in response to each new Transmit Frame request.

Upon entry, this state shall send a Transmission Status (No Phys In Port) confirmation for each queued request.

8.3.2.2 ~~8.3.2.2~~ Transition PL_OC1:Idle to PC_OC2:Overall_Control

This transition shall occur when an Phy Enabled confirmation is received from at least one phy assigned to the port.

8.3.3 PL_OC2:Overall_Control state

8.3.3.1 State description

8.3.3.1.1 State description overview

This state receives Transmit Frame requests from one or more transport layers (e.g., SSP, SMP, and STP) and Retry Frame parameters from one or more PL_PM state machines. Each transport layer may have multiple Transmit Frame requests pending at once, and each PM_PM state machine may generate multiple Retry Frame parameters over time. This state chooses the order in which the requests are processed. This state machine shall service one Transmit Frame request or Retry Frame parameter at a time.

If multiple phys exist in the port, this state may establish multiple connections to multiple destination SAS addresses simultaneously (one per phy) and may transmit and receive via the different connections simultaneously.

The Transmit Frame request from the transport layer includes the following arguments:

a) I_T nexus loss time;
b) Interlocked or Not Interlocked (for SSP only);
c) destination SAS address;
d) source SAS address;
e) connection rate;
f) initiator bit;
g) initiator connection tag; and
h) frame.

The Retry Frame parameter from the PL_PM state machine includes the following arguments:

a) Phy identifier - the phy on which the frame was last attempted;
b) Balance Required (for SSP only);
c) arbitration wait time;
d) pathway blocked count;
e) destination SAS address;
f) source SAS address;
g) connection rate;
h) initiator bit;
i) initiator connection tag; and
j) frame.

The Tx Frame parameter sent to the PL_PM state machine includes the following arguments:

a) Balance Required or Balance Not Required (for SSP only);
b) arbitration wait time;
c) pathway blocked count;
d) I_T nexus loss time and status;
e) destination SAS address;
f) source SAS address;
g) connection rate;
h) initiator bit;
i) initiator connection tag; and
j) frame.

8.3.3.1.2 Keep track of connections/frame requests

This state machine shall keep track of which phys (represented by corresponding PL_PM state machines) are available. A phy is available if it is not processing a Tx Frame parameter.

This state shall consider a phy as having an active connection after receiving a Connection Opened confirmation from the link layer until receiving a Connection Closed confirmation from the link layer.

This state shall consider a phy as processing a Tx Frame parameter after sending the parameter to the PL_PM state machine until receiving a Transmission Status confirmation or an Unable to Connect confirmation from the link layer.

If this state has no pending Transmit Frame requests or Retry Frame parameters pending for a phy with an open connection, this state should send a Close Connection request to that phy.

8.3.3.1.3 Select a request to process and the phy on which to process it

This state may process a Transmit Frame request or Retry Frame parameter whenever a phy is available. The phy should be chosen following this order of precedence:

1) a phy that has a connection open to the same destination and is not processing a Tx Frame request;
2) a phy that does not have a connection open; or
3) a phy that has a connection open to a different destination and is not processing a Tx Frame request.

A destination is considered the same if it has the same SAS address and protocol.

Additional rules for wide SSP port selection of phys are described in 8.3.3.1.4.

If a phy is chosen with a connection open to a different destination, this state shall send a Close Connection to the phy. This state may then process the request on the selected phy when the connection is closed, or it may choose another request to process on the selected phy when the connection is closed, or it may choose another phy on which to process the request if one becomes available. This state shall send no more than one Close Connection on behalf of the same request.

When Retry Frame parameters are pending with non-zero arbitration wait times, this state shall increment those arbitration wait times until they are processed.

This state shall not process a Retry Frame parameter until a retry delay (see 7.12.2.2) has expired from when the parameter was received.

8.3.3.1.4 SSP wide port rules

~~An initiator port that is a wide port may transmit COMMAND frames on multiple links simultaneously.~~

~~An initiator port shall not transmit a TASK frame requesting a task management function that only affects a single I_T_L_Q nexus (e.g., ABORT TASK or QUERY TASK; see SAM-3) specifying an I_T_L_Q nexus for which the initiator port is transmitting a frame or is waiting for a link layer acknowledgement for a frame.~~

~~An initiator port shall not transmit a TASK frame requesting a function that only affects an I_T_L nexus (e.g., ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, or LOGICAL UNIT RESET; see SAM-3) specifying an I_T_L nexus for which the initiator port is transmitting a frame or is waiting for a link layer acknowledgement for a frame.~~

~~An initiator port shall not transmit a TASK frame requesting a function that only affects an I_T nexus (see SAM-3) specifying an I_T nexus for which the initiator port is transmitting a frame or is waiting for a link layer acknowledgement for a frame.~~

A wide port shall only transmit DATA frames for a given I_T_L_Q on one phy at a time. The wide port may switch phys for DATA frames once all of the previous DATA frames have been acknowledged by the link layer. Read DATA frames and write DATA frames for the same I_T_L_Q may be transmitted simultaneously and may be on the same or different phys.

A target port shall not transmit a RESPONSE frame for a given I_T_L_Q nexus if the target port is transmitting any frame for the same I_T_L_Q on another phy. The target port shall not transmit a RESPONSE frame for a given I_T_L_Q if the target port is waiting for a link layer acknowledgement for that I_T_L_Q on any phy.

8.3.3.1.5 Filling in the Tx Frame arguments

This state shall send a Tx Frame parameter to the PL_PM state machine corresponding to the selected phy carrying all the arguments from the selected Transmit Frame request or Retry Frame parameter unless otherwise specified.

The AWT and PBC arguments for the PL_OC Tx Frame parameter are transferred to the selected PL_PM's AWT timer and PBC counter, respectively. The AWT and PBC arguments for Transmit Frame requests shall be zero and are not received as arguments with the Transmit Frame request from the transport layer. The AWT and PBC arguments for the PL_OC Retry Frame parameters are received as arguments (which are read from the appropriate PL_PM AWT timer or PBC counter) in the Unable to Connect parameter received from the PL_PM state machine. The PL_OC Retry Frame AWT argument is stored in a corresponding PL_OC Retry Frame AWT timer which shall start incrementing if the AWT argument received from the PL_PM state machine is non zero. If the AWT argument received from the PL_PM state machine is zero, the PL_OC AWT timer shall not start incrementing.

The PL_OC Retry Frame parameter shall include a Balanced Required argument.

The I_T nexus loss arguments for the PL_OC Tx Frame parameter are transferred to the selected PL_PM's I_T nexus loss timer. The selected PL_PM timer shall be set to the time and state (i.e, either stopped, running, or expired) of these arguments received in the Tx Frame parameter. The I_T nexus loss time and status arguments for the Tx Frame parameter are read from the PL_OC I_T nexus loss timer for this destination. For each destination, the PL_OC I_T nexus loss timer is updated with the time and status arguments of the PL_PM I_T nexus loss timer received with an Unable To Connect parameter from the PL_PM state machine.

Some of the arguments are modified as follows:

a) For SSP requests, the Tx Frame parameter should include a Balance Not Required argument when:
   A) a Transmit Frame (Non Interlocked) request is being processed;
   B) the phy selected has a connection open to the same destination;
   C) the previous request processed for this phy was a Transmit Frame (Non Interlocked) request to the same destination;
   D) the connection has not been lost since the previous frame was transmitted; and
   E) the frame has the same tag value as the previous frame.

   Otherwise, it shall include a Balance Required argument.

b) If this state is processing either:
   A) a Transmit Frame request or Retry Frame parameter with the same destination as the last Retry Frame parameter received; or
   B) the last Retry Frame parameter received;

   and:

   A) the phy selected does not have a connection opened or has a connection open to a different destination; and
   B) this is the next request sent to any phy after the last Retry Frame parameter was received;

   then the pathway blocked count shall be set to the value of the pathway blocked count argument received with the last Retry Frame parameter.

   As a result, only the next request from the port uses that pathway blocked count value. If the port chooses a request to a different destination, it shall not transfer the pathway blocked count value to the different request; the pathway blocked count shall be set to zero.

c) The arbitration wait time argument shall be set to:
   A) the value of the arbitration wait time argument received with the last Retry Frame parameter; or
   B) zero or a vendor-specific value less than or equal to 7FFFh (see 7.7.3). The argument should be set to zero.

   If the arbitration wait time from the last Retry Frame parameter is used for a different frame, the arbitration wait time for the last Retry Frame parameter shall be set to zero.

As a result, only the next request from the port uses that arbitration wait time value. If the port chooses a request to a different destination, it is allowed to transfer the arbitration wait time value to the different request.

8.3.3.1.6 Confirmations

This state shall send a Transmission Status confirmation or an Connection Failed confirmation to the transport layer to finish servicing each Transmit Frame request.

This state shall send a Transmission Status (Frame Transmitted) confirmation to the transport layer after receiving a Frame Transmitted parameter from the PL_PM state machine. Since the transport layer responses for any sequence of frame transmissions (e.g. multiple non-interlocked frame transmissions in SSP) are instantaneous, the next Transmit Request from the transport layer for the same I_T_L_Q should be available when the Transmission Status (Frame Transmitted) confirmation is sent. This ensures that this state is able to choose the same phy and connection (e.g. and continue a non-interlocked sequence).

This state shall send a Transmission Status (Connection Lost) confirmation to the transport layer if a Connection Closed confirmation is received from the link layer after a Tx Frame parameter is sent but before a Frame Transmitted parameter is received.

For SSP connections, this state shall send a Connection Failed (Connection Lost Without ACK/NAK) confirmation to the transport layer if a Connection Closed confirmation is received from the link layer after a Frame Transmitted parameter is received but before an ACK Received, NAK Received or ACK/NAK Timeout confirmation is received.

After a Connection Closed confirmation is received from the link layer, the phy is available for a new connection.

This state shall send a Transmission Status confirmation or a Cancel Acknowledgement confirmation to the transport layer after receiving an Unable to Connect parameter from the PL_PM state machine based on the mapping in table 86.

Table 86 — Confirmations to transport layer from the PL_OC state machine

Unable To Connect parameter	Confirmation sent to transport layer
Unable To Connect (Wrong Destination)	Transmission Status (Open Failed - Wrong Destination)
Unable To Connect (Connection Rate Not Supported)	Transmission Status (Open Failed - Connection Rate Not Supported)
Unable To Connect (Protocol Not Supported)	Transmission Status (Open Failed - Protocol Not Supported)
Unable To Connect (Bad Destination)	Transmission Status (Open Failed - Bad Destination)
Unable To Connect (STP Resources Busy)	Transmission Status (Open Failed - STP Resources Busy)
Unable To Connect (Break Received)	Transmission Status (Open Failed - Break Received)
Unable To Connect (No Destination)	Transmission Status (Open Failed - No Destination)
Unable To Connect (Open Timeout Occurred)	Transmission Status (Open Failed - Open Timeout Occurred)
Unable To Connect (Pathway Blocked)	Transmission Status (Open Failed - Pathway Blocked)
Unable To Connect (Cancel Acknowledgement)	Cancel Acknowledgement

The stopped, running or expired I_T nexus loss timer status received with the Unable To Connect parameter is sent to the transport layer as an argument with the Open Failed Transmission Status confirmation.

8.3.3.1.7 Handling Cancel requests

If this state receives a Cancel request for a specific Transmit Frame request from the transport layer and this frame is not currently being processed (either from the Transmit Frame request itself or from a Retry Frame parameter), this state shall send a Cancel Acknowledge confirmation to the transport layer and terminate processing of the Transmit Frame request.

If this state receives a Cancel request for a specific Transmit Frame request from the transport layer, this frame is currently being processed, and a connection has been opened, this state shall send a Close Connection request to the link layer. When the connection has been closed, this state shall send a Cancel Acknowledge confirmation to the transport layer and terminate processing of the Transmit Frame request.

If this state receives a Cancel request for a specific Transmit Frame request from the transport layer, this frame is currently being processed, and a connection has not yet been opened, this state shall send a Cancel parameter to the PL_PM state machine. When the PL_PM state machine has terminated the connection and returned an Unable To Connect (Cancel Acknowledge) parameter, this state shall send a Cancel Acknowledge confirmation to the transport layer and terminate processing of the Transmit Frame request.

8.3.3.1.8 Handling other requests

If a Disable Tx Frames parameter is received from the PL_PM state machine, this state shall not send another Tx Frame parameter to the phy until after a Connection Closed confirmation has been received from the phy.

If this state receives an Accept_Reject Opens request from the transport layer, this state shall send an Accept_Reject Opens request to the link layer.

If this state receives an SMP Transmit Break request from the transport layer, this state shall send an SMP Transmit Break request to the link layer.

8.3.3.2 Transition PL_OC2:Overall_Control to PL_OC1:Idle

This transition shall occur when a Phy Disabled confirmation has been received from every phy assigned to the port.

8.4 Port layer phy manager (PL_PM) state machine

8.4.1 ~~Overview~~

A PL_PM state machine exists for each phy contained in the port. The PL_PM state machine's purpose is to:

a) Request a connection sequence via the link layer and handle exception conditions;
b) Send a Retry Frame parameter to the PL_OC state machine if a connection attempt fails and is to be retried;
c) Maintain a connected state for the phy;
d) Forward frame transmission requests from the PL_OC state machine to the link layer;
e) Pass confirmations from the link layer to the PL_OC state machine;
f) Update the PL_PM I_T nexus loss timer, the PL_PM arbitration wait time timer, and the PL_PM pathway blocked count counter; and
g) For target ports only, implement the bus inactivity time limit timer and the maximum connect time timer and close the connection if they have been exceeded.

The PL_PM state machine contains these states:

a) PL_PM1:Idle;
b) PL_PM2:ReqWait;
c) PL_PM3:Connected; and
d) PL_PM4:Wait_For_Close.

The state machine shall start in the PL_PM1:Idle state. The state machine shall transition to the PL_PM1 state from any other state after receiving an Phy Disabled confirmation from the link layer.

Figure 92 shows part of the PL_PM state machine.

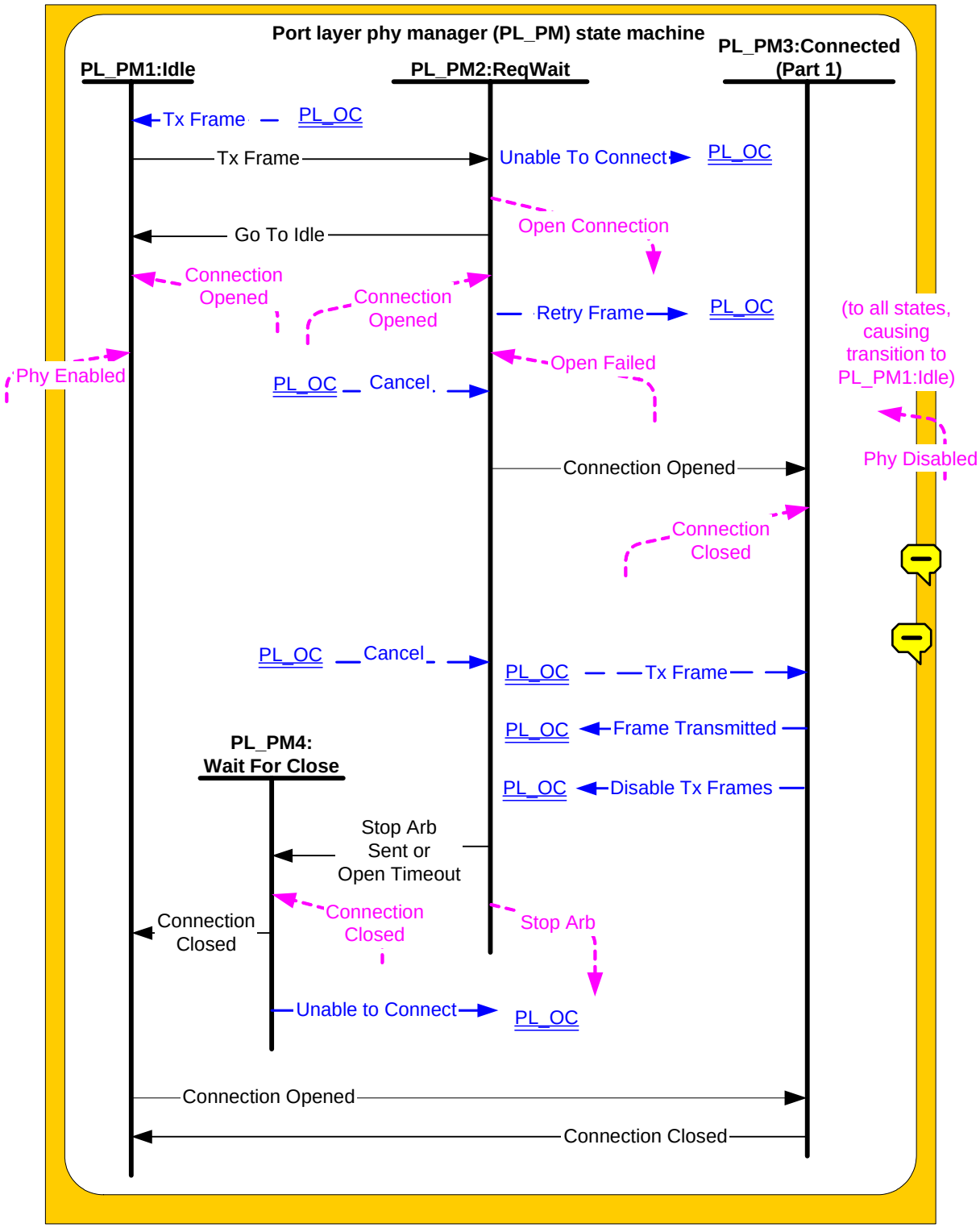


Figure 92 — Port layer phy manager (PL_PM) state machine (part 1)

Figure 93 shows more details of the PL_PM state machine.

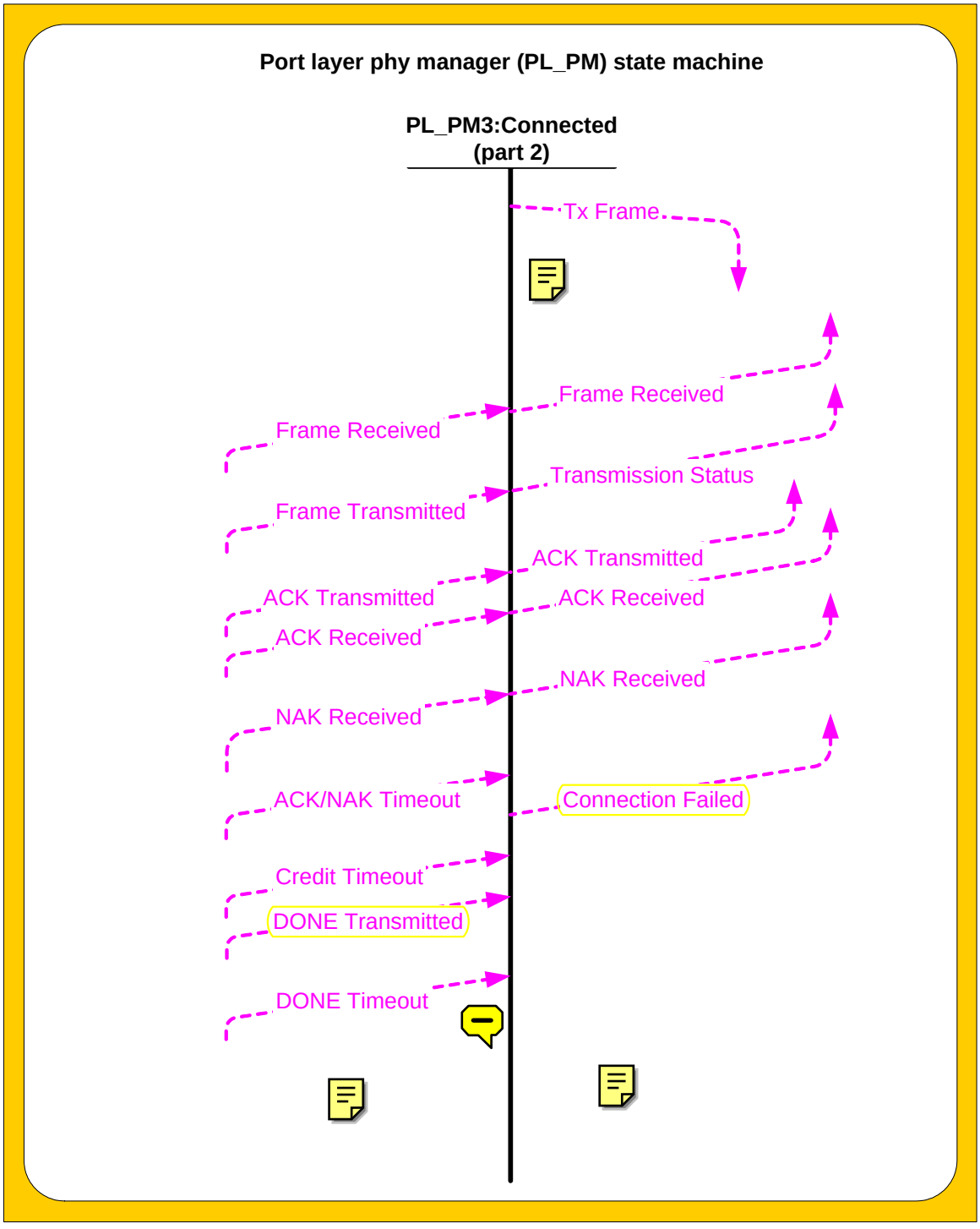


Figure 93 — Port layer phy manager (PL_PM) state machine (part 2)

8.4.2 PL_PM1:Idle state

8.4.2.1 State description

This is the initial state of the PL_PM state machine. This state is also entered when a connection is closed or a connection attempt is terminated.

8.4.2.2 Transition PL_PM1:Idle to PL_PM2:ReqWait

This transition shall occur after receiving a Tx Frame parameter from the PL_OC state machine.

8.4.2.3 Transition PL_PM1:Idle to PL_PM3:Connected

This transition shall occur after receiving a Connection Opened parameter from the link layer.

8.4.3 PL_PM2:ReqWait state

8.4.3.1 State description

8.4.3.1.1 State description overview

This state sends an Open Connection request to the selected phy and waits for a response.

The Open Connection request includes these arguments, filled in with values from the Tx Frame parameter:

a) destination SAS address;
b) protocol;
c) arbitration wait time;
d) pathway blocked count;
e) connection rate;
f) initiator bit; and
g) initiator connection tag.

This state shall start incrementing the arbitration wait time after sending the Open Connection request.

If a Cancel parameter is received from the PL_OC state machine, this state shall send a Stop Arb request to the link layer to cancel the connection request.

8.4.3.1.2 PL_PM I_T nexus loss timer

For SSP ports, the PL_PM I_T nexus loss timer for a destination shall be stopped, set to zero, and assigned a stopped status when one of the following confirmations is received:

a) Connection Opened;
b) Open Failed (Protocol Not Supported);
c) Open Failed (Retry); or
d) Open Failed (STP Resources Busy).

For SSP ports, if the PL_PM I_T nexus loss timer is not already running, the PL_PM I_T nexus loss timer shall be initialized, assigned a running status, and started decrementing when one of the following confirmations is received:

a) Open Failed (Connection Rate Not Supported);
b) Open Failed (No Destination); or
c) Open Failed (Open Time Out).

If the I_T nexus loss timer is already running, the I_T nexus loss timer shall continue running for those confirmations.

For SSP ports, the I_T Nexus Loss argument shall be included in any Open Failed confirmation if the I_T nexus loss timer expires while in this state.

8.4.3.1.3 Connection Opened handling

If a Connection Opened (Destination Opened) confirmation is received from the link layer and the destination does not match, this state shall send a Retry Frame parameter to the PL_OC state machine with arguments from the Tx Frame, including the updated arbitration wait time. The pathway blocked count shall be maintained. The arbitration wait time timer shall be stopped.

8.4.3.1.4 Open Failed handling

If a Cancel parameter has been received from the PL_OC state machine and an Open Failed confirmation is received, this state shall send an Unable To Connect (Cancel Acknowledgement) parameter to the PC_OC state machine. Otherwise, this state shall send either an Unable To Connect parameter with the same argument as that received with the Open Failure confirmation or a Retry Frame parameter. The Unable To Connect parameter shall include as arguments:

a) the counts of the AWT timer, PBC counter, and I_T Nexus loss timer; and
b) the status of the I_T Nexus loss timer.

The arbitration wait time timer shall be stopped if an Open Failed (Protocol Not Supported), Open Failed (Retry), or Open Failed (Wrong Destination) confirmation is received from the link layer.

This state shall send a Retry Frame parameter under the conditions shown in table 87. The pathway blocked argument shall be modified as specified; the other arguments shall be copied from the Tx Frame being processed.

Table 87 — Retry Frame conditions

Open Failed confirmation argument received from the link layer	Has the I_T nexus loss timer expired?	Pathway blocked count argument for Retry Frame
Open Failed (Pathway Blocked)	no	increment by one
Open Failed (Retry)	yes or no	set to zero

8.4.3.2 Transition PL_PM2:ReqWait to PL_PM1:Idle

This transition shall occur after an Open Failed parameter has been sent to the PC_OC state machine.

8.4.3.3 Transition PL_PM2:ReqWait to PL_PM3:Connected

This transition shall occur after a Connection Opened confirmation has been received from the link layer.

8.4.3.4 Transition PL_PM2:ReqWait to PL_PM4:Wait_For_Close

This transition shall occur after receiving an Open Failed (Port Layer Request) confirmation from the link layer.

If a Cancel parameter is received from the PL_OC state machine, this transition shall occur after receiving an Open Failed (Open Timeout Occurred) confirmation from the link layer.

8.4.4 PL_PM3:Connected state

8.4.4.1 State description

When in this state, frame transmissions and receptions may occur.

When in this state and a frame is to be transmitted, the PL_OC state machine may send the Tx Frame parameter to this state. For STP connections, the port layer connects the transport layer to the link layer and is transparent; Tx Frame is not used. For SSP connections, the Tx Frame parameter shall include Balance Required or Balance Not Required arguments. For SMP connections, the Tx Frame parameter shall include the SMP argument.

This state shall generate a Tx Frame request to the link layer when a Tx Frame parameter is received from the PL_OC state machine.

After a Tx Frame request has been sent to the link layer and a Frame Transmitted confirmation is received from the link layer, this state shall send a Frame Transmitted parameter to the PL_OC state machine to notify the PL_OC state that another frame may be transmitted.

This state shall also send the following confirmations regarding the previous frame transmission confirmations from the link layer to the transport layer:

a) ACK Transmitted;
b) ACK Received;
c) NAK Received;
d) Connection Failed (ACK/NAK Timeout): Mapped from the ACK/NAK Timeout confirmation from the link layer; or
e) Transmission Status (Credit Timeout): Mapped from the Credit Timeout confirmation from the link layer.

For SMP ports, this state shall send the Frame Received (SMP) or Frame Received (SMP Failure) confirmations from the link layer to the transport layer

For SSP ports, this state shall send the Frame Received (ACK/NAK Balanced) or Frame Received (ACK/NAK Not Balanced) confirmations from the link layer to the transport layer.

For SSP ports, this state shall transfer a Disable Tx Frames parameter to the PL_OC state machine if a DONE (ACK/NAK TIMEOUT) Received confirmation is received from the link layer.

For SSP ports, this state shall transfer a Disable Tx Frames parameter to the PL_OC state machine and a DONE Received confirmation to the application layer if a DONE Received confirmation is received from the link layer.

For SSP target ports, upon entry into this state, the bus inactivity time limit timer shall be initialized and started. It shall be re-initialized every time a Tx Frame request is processed. If the timer expires, this state shall send a Close Connection request to the link layer after the current Tx Frame is finished and shall ignore new Tx Frame requests. The timer stops running upon exit from this state.

For SSP target ports, upon entry into this state, the maximum connect time timer shall be initialized and started. If the timer expires, this state shall send a Close Connection request to the link layer after the current Tx Frame is finished and shall ignore new Tx Frame requests. The timer stops running upon exit from this state.

8.4.4.2 Transition PL_PM3:Connected to PL_PM1:Idle

This transition shall occur after a Connection Closed confirmation is received from the link layer.

8.4.5 PL_PM4:Wait_For_Close state

8.4.5.1 State description

This state shall wait for a Connection Closed confirmation to be received from the link layer and then send an Unable to Connect (Cancel Acknowledge) parameter to the PL_OC state machine.

8.4.5.2 Transition PL_PM4:Wait_For_Close to PL_PM1:Idle

This transition shall occur after a Connection Closed confirmation is received from the link layer.

9 Transport layer

9.1 Transport layer overview

The transport layer constructs and parses frame contents. For SSP, the transport layer only receives frames that are going to be ACKed by the link layer.

9.2 SSP transport layer

9.2.1 SSP frame format

Table 88 defines the SSP frame format.

Table 88 — SSP frame format

Bit Byte	7	6	5	4	3	2	1	0
0	FRAME TYPE							
1	(MSB)	HASHED DESTINATION SAS ADDRESS						
3								(LSB)
4	Reserved							
5	(MSB)	HASHED SOURCE SAS ADDRESS						
7								(LSB)
8	Reserved							
9	Reserved							
10	Reserved						TIMEOUT	Reserved
11	Reserved						NUMBER OF FILL BYTES	
12	Reserved							
13	Reserved							
15								
16	(MSB)	TAG						
17								(LSB)
18	(MSB)	TARGET PORT TRANSFER TAG						
19								(LSB)
20	(MSB)	RELATIVE OFFSET						
23								(LSB)
24	INFORMATION UNIT							
m								
	Fill bytes, if needed							
n - 3	(MSB)	CRC						
n								(LSB)

Table 89 defines the FRAME TYPE field.

Table 89 — FRAME TYPE field

Code	Name	Information unit	Originator	Information unit size (bytes)
01h	DATA	Data	Initiator port or target port	0 to 1 024
05h	XFER_RDY	Transfer ready	Target port	12
06h	COMMAND	Command	Initiator port	28 to 284
07h	RESPONSE	Response	Target port	24 to 1 024
16h	TASK	Task management function	Initiator port	28
F0h - FFh	Vendor-specific			
All others	Reserved			

An SSP frame containing a COMMAND information unit (IU) is called a COMMAND frame; an SSP frame containing a TASK IU is called a TASK frame; etc.

The HASHED DESTINATION SAS ADDRESS field contains the hashed value (see 4.2.3) of the destination SAS address. See 9.2.6.2.8 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 9.2.6.2.8 and 9.2.6.3.2 for transport layer requirements on checking this field.

The RETRANSMIT bit may be set to one for RESPONSE frames and shall be set to zero for all other frame types. This field indicates the frame is a retransmission after the target port timed out waiting for the ACK or NAK for its previous attempt to transmit the frame.

The NUMBER OF FILL BYTES field indicates the number of fill bytes between the INFORMATION UNIT field and the CRC field. The initiator port or target port transmitting a frame shall provide fill bytes to ensure that the CRC field is 4-byte aligned. The NUMBER OF FILL BYTES field shall be set to zero for all frame types except DATA frames. The contents of the fill bytes are vendor-specific.

The TAG field allows the initiator port to establish a context for commands and task management functions.

For COMMAND and TASK frames, the initiator port shall set the TAG field to a value that is unique for the I_T nexus. An initiator port shall not reuse the same tag when transmitting COMMAND or TASK frames to different LUNs in the same target port; it may reuse a tag when transmitting frames to different target ports. The TAG field in a COMMAND frame contains the tag defined in SAM-3; the TAG field in a TASK frame does not correspond to a SAM-3 tag, but does take the place of a SAM-3 tag. The tag space used in the TAG fields is shared across COMMAND and TASK frames.

For DATA, XFER_RDY, and RESPONSE frames, the target port shall set the TAG field to that of the command or task management function to which the frame pertains.

The TARGET PORT TRANSFER TAG field allows the target port to ~~quickly~~ establish a write data context when it has multiple outstanding XFER_RDY frames. This field shall be set by the target port in the XFER_RDY frame. Target ports that need this field shall set a value that is unique for the I_T nexus. Target ports that do not need this field shall set it to FFFFh. For each DATA frame that is sent in response to a XFER_RDY frame, the initiator port shall set the TARGET PORT TRANSFER TAG field to the value that was in the corresponding XFER_RDY frame. For each DATA frame that is not sent in response to a XFER_RDY frame (due to a non-zero FIRST BURST SIZE field in the Disconnect-Reconnect mode page), the initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh. The initiator port shall set this field to FFFFh for all frames other than DATA frames. The target port shall set this field to FFFFh for all frames other than XFER_RDY frames.

For DATA frames, the RELATIVE OFFSET field indicates the application client buffer offset as described by SAM-3. The relative offset shall be a multiple of four (i.e., each DATA frame shall begin on a dword boundary). ~~The RELATIVE OFFSET field shall be zero for the first read DATA frame and first write DATA frame of a command.~~ For all other frame types, the RELATIVE OFFSET field shall be ignored.

The INFORMATION UNIT field contains the information unit. The maximum size of the INFORMATION UNIT field is 1 024 bytes, making the maximum size of the frame 1 052 bytes (1 024 byte data + 24 byte header + 4 byte CRC).

The CRC field is a CRC value (see 7.4) that is computed over the entire frame 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 information unit

Table 90 defines the command IU. The COMMAND frame is sent by an initiator port to request a command be performed by a device server in a logical unit.

Table 90 — COMMAND information unit

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7	LOGICAL UNIT NUMBER							(LSB)
8	Reserved							
9	Reserved					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 specifies the address of the logical unit. The structure of the logical unit number field shall be as defined in SAM-3. If the addressed logical unit does not exist, the task manager shall follow the ~~SCSI~~ rules for selection of invalid logical units defined in SPC-2.

The TASK ATTRIBUTE field is defined in table 91.

Table 91 — TASK ATTRIBUTE field

Code	Task attribute	Description
000b	SIMPLE	Requests that the task be managed according to the rules for a simple task attribute (see SAM-3).
001b	HEAD OF QUEUE	Requests that the task be managed according to the rules for a head of queue task attribute (see SAM-3).
010b	ORDERED	Requests that the task be managed according to the rules for an ordered attribute (see SAM-3).
011b		Reserved
100b	ACA	Requests that the task be managed according to the rules for an automatic contingent allegiance task attribute (see SAM-3).
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 are reserved. For example, a six-byte CDB occupies the first six bytes of the CDB field; the remaining ten bytes are reserved and the ADDITIONAL CDB BYTES field is not present.

The contents of the CDB are defined in the SCSI command standards (e.g., SPC-3).

9.2.2.2 TASK information unit

Table 92 defines the task management function IU. The TASK frame is sent by an initiator port to request a task management function be performed by a task manager in a logical unit.

Table 92 — TASK information unit

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7				LOGICAL UNIT NUMBER				(LSB)
8	Reserved							
9	Reserved							
10	TASK MANAGEMENT FUNCTION							
11	Reserved							
12	(MSB)							
13				TAG OF TASK TO BE MANAGED				(LSB)
14								
27	Reserved							

The LOGICAL UNIT NUMBER field specifies the address of the logical unit. The structure of the logical unit number field shall be as defined in SAM-3. If the addressed logical unit does not exist, the task manager shall follow the ~~SCSI~~ rules for selection of invalid logical units defined in SPC-2.

Table 93 defines the TASK MANAGEMENT FUNCTION field.

Table 93 — Task management functions

Code	Task management function	Description
01h	ABORT TASK	The task manager shall perform the ABORT TASK task management function with L set to LOGICAL UNIT NUMBER and Q set to TAG OF TASK TO BE MANAGED (see SAM-3).
02h	ABORT TASK SET	The task manager shall perform the ABORT TASK SET task management function with L set to LOGICAL UNIT NUMBER (see SAM-3).
04h	CLEAR TASK SET	The task manager shall perform the CLEAR TASK SET task management function with L set to LOGICAL UNIT NUMBER (see SAM-3).
08h	LOGICAL UNIT RESET	The task manager shall perform the LOGICAL UNIT RESET task management function with L set to LOGICAL UNIT NUMBER (see SAM-3).
20h	Reserved.	
40h	CLEAR ACA	The task manager shall perform the CLEAR ACA task management function with L set to LOGICAL UNIT NUMBER (see SAM-3).
80h	QUERY TASK	The task manager shall perform the QUERY TASK task management function with L set to LOGICAL UNIT NUMBER and Q set to TAG OF TASK TO BE MANAGED (see SAM-3).
All others	Reserved.	

If TASK MANAGEMENT FUNCTION contains a reserved value, the task manager shall return a RESPONSE frame with its STATUS field set to GOOD and its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED. The TARGET RESET task management function defined in SAM-3 is not supported.

If TASK MANAGEMENT FUNCTION 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.

For ABORT TASK, if the TAG OF TASK TO BE MANAGED field does not contain the tag of a task currently in the task set, the task manager shall return a RESPONSE frame with its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION COMPLETE.

For QUERY TASK, if the TAG OF TASK TO BE MANAGED field contains the tag of a task currently in the task set, the task manager shall return a RESPONSE frame with its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION SUCCEEDED. If the TAG OF TASK TO BE MANAGED field does not contain the tag of a task currently in the task set, the task manager shall return a RESPONSE frame with its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION COMPLETE.

9.2.2.3 XFER_RDY information unit

Table 94 defines the transfer ready IU. The XFER_RDY frame is sent by a target port to request write data from the initiator port.

Table 94 — XFER_RDY information unit

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)			RELATIVE OFFSET				
3								(LSB)
4	(MSB)			WRITE DATA LENGTH				
7								(LSB)
8				Reserved				
11								

The RELATIVE OFFSET field indicates the initial application client buffer offset of the write data the initiator port may transmit to the logical unit ~~(using DATA frames).~~ The relative offset shall be a multiple of four (i.e., each DATA frame shall begin on a dword boundary). The RELATIVE OFFSET field shall be zero for the first XFER_RDY frame of a command unless the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.1.1.1.5) is not set to zero.

The WRITE DATA LENGTH field indicates how many bytes of write data the initiator port may transmit to the logical unit ~~(using DATA frames)~~ from the application client buffer starting at the relative offset. The 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 target port shall set the WRITE DATA LENGTH field to less than the value in the MAXIMUM BURST SIZE field (see 10.1.6.1.4).

If a target port transmits a XFER_RDY frame containing a WRITE DATA LENGTH field that is not divisible by four, the 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 aligned write data length).

The initial XFER_RDY frame for a given command shall set the relative offset to the value of the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.1.1.1.5). If any additional XFER_RDY frames are required, the RELATIVE OFFSET field shall be set to the value of the previous XFER_RDY frame's relative offset plus the previous XFER_RDY frame's write data length.

9.2.2.4 DATA information unit

Table 95 defines the data IU. The DATA frame is sent by an initiator port to deliver write data and is sent by a target port to deliver read data.

Table 95 — DATA information unit

Bit Byte	7	6	5	4	3	2	1	0
0				DATA				
n-1								

The DATA field contains the read or write data. The maximum size of the data IU is the maximum size of any IU in an SSP frame (see 9.2.1). The minimum size of the data IU is one byte.

An initiator port shall only transmit a DATA frame:

a) in response to an XFER_RDY frame; or
b) after transmitting a COMMAND frame if the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is not zero (see 10.1.6.1.5).

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 a target port per I_T_L_Q nexus is constrained by the value in the MAXIMUM BURST SIZE field (see 10.1.6.1.4).

The DATA frame shall only contain write data for a single XFER_RDY frame.

If a target port transmits a DATA frame containing a non-zero value in the NUMBER OF FILL BYTES field in the frame header (see 9.2.1), the target port shall not transmit any subsequent DATA frames for that command (i.e., only the last read DATA frame for a command may have data with a length that is not a multiple of four).

An 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 a XFER_RDY. An initiator port shall set the NUMBER OF FILL BYTES field in the frame header (see 9.2.1) to zero in all other DATA frames that it transmits.

NOTE 23 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 initiator port shall only transmit DATA frames in response to a XFER_RDY frame or an implied XFER_RDY frame as a result of the FIRST BURST SIZE field.~~

An initiator port shall not transmit a DATA frame for a given I_T_L_Q after it has sent a TASK frame that terminates that task (e.g., an ABORT TASK).

The initial read DATA frame for a given command shall set the relative offset to zero. If any additional read DATA frames are required, the RELATIVE OFFSET field shall be set to the value of the previous read DATA frame's relative offset plus the previous read DATA frame's data length.

The initial write DATA frame for a given command shall set the relative offset to zero. If any additional write DATA frames are required, the RELATIVE OFFSET field shall be set to the value of the previous write DATA frame's relative offset plus the previous write DATA frame's data length.

9.2.2.5 RESPONSE information unit

9.2.2.5.1 RESPONSE information unit overview

Table 96 defines the response IU. The RESPONSE frame is sent by a target port to deliver SCSI status (e.g., GOOD or CHECK CONDITION) and sense data, or to deliver SSP-specific status (e.g., illegal frame format).

Table 96 — RESPONSE information unit

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
9								
10	Reserved						DATAPRES	
11	STATUS							
12	Reserved							
15								
16	(MSB)							
19	SENSE DATA ~~LIST~~ LENGTH (n bytes)							(LSB)
20	(MSB)							
23	RESPONSE DATA LIST LENGTH (m bytes)							(LSB)
24	RESPONSE DATA							
23+m								
24+m	SENSE DATA							
23+m+n								

Table 97 defines the DATAPRES field, ~~which defines the format and content of the response IU.~~

Table 97 — DATAPRES field

Code	Name	Description	Reference
00b	NO DATA	No 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 target device shall return a RESPONSE frame with the DATAPRES field set to NO_DATA if a command completes without sense data to return.

The target device shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA in response to every TASK frame and if an error occurs while the transport layer is processing a COMMAND frame.

The target device 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 initiator device 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 STATUS field shall contain the status code for a command that has ended (see SAM-3 for a list of status codes).

The SENSE DATA LIST LENGTH field and the RESPONSE DATA LIST LENGTH field shall be set to zero and shall be ignored by the initiator device.

The SENSE DATA field and the RESPONSE DATA field shall not be present.

9.2.2.5.3 RESPONSE information unit RESPONSE_DATA format

If the DATAPRES field is set to RESPONSE_DATA, then the STATUS field and the SENSE DATA LIST LENGTH field shall be set to zero and shall be ignored by the initiator device.

The SENSE DATA field shall not be present.

The RESPONSE DATA LIST LENGTH field shall be set to four. Other lengths are reserved for future standardization.

The RESPONSE DATA field shall be present. Table 98 defines the RESPONSE DATA field, which contains information describing ~~certain~~ protocol failures detected during processing of a request received by the target port. The RESPONSE DATA field shall be present if the target port detects any of the conditions described by a non-zero RESPONSE CODE value and shall be present for a RESPONSE frame sent in response to a TASK frame.

Table 98 — RESPONSE DATA field

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	Reserved							
2	Reserved							
3	RESPONSE CODE							

Table 99 defines the RESPONSE CODE field, which indicates the error condition or the completion status of a task management function. See 10.1.1.5 and 10.1.1.13 for the mapping of these response codes to SCSI service responses.

Table 99 — RESPONSE CODE field

Code	Description
00h	Either: a) NO FAILURE, when responding to a COMMAND frame; or b) TASK MANAGEMENT FUNCTION COMPLETE, when responding to a TASK frame.
02h	TASK FRAME FIELDS INVALID
04h	TASK MANAGEMENT FUNCTION NOT SUPPORTED
05h	TASK MANAGEMENT FUNCTION FAILED
08h	TASK MANAGEMENT FUNCTION SUCCEEDED
All others	Reserved

9.2.2.5.4 RESPONSE information unit SENSE_DATA format

If the DATAPRES field is set to SENSE_DATA, then the STATUS field shall contain the status code for a command that has ended (see SAM-3 for a list of status codes).

The RESPONSE DATA LIST LENGTH field shall be set to zero and shall be ignored by the initiator.

The RESPONSE DATA field shall not be present.

The SENSE DATA LIST LENGTH field shall be set to a non-zero value indicating the number of bytes in the SENSE DATA field. The SENSE DATA LIST LENGTH field shall not be larger than 1 000 and shall be a multiple of four. If the SENSE DATA field size is not a multiple of four, it shall be padded to make its size a multiple of four. The value of the pad bytes are vendor-specific.

The SENSE DATA field contains sense data (see SAM-3).

9.2.3 Frame sequences

Figure 94 shows a task management function sequence. The transport protocol services (see 10.1.1) invoked by the application layer are also shown.

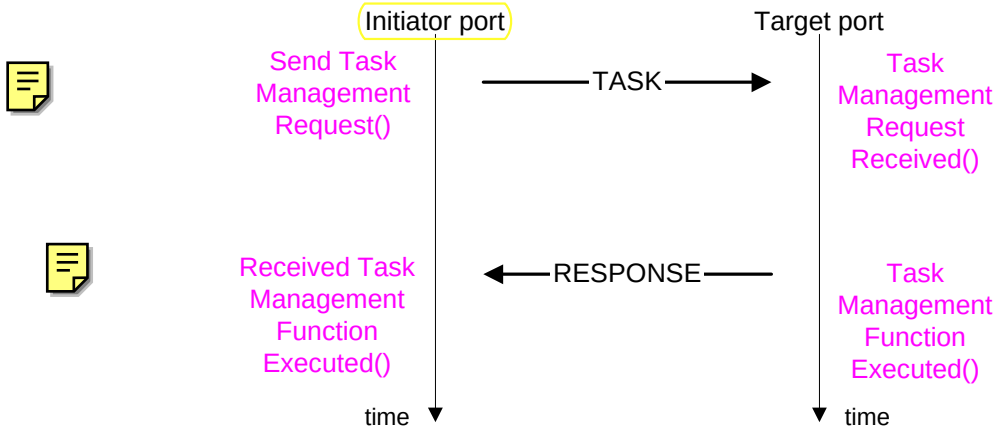


Figure 94 — Task management function sequence

Figure 95 shows a write command sequence. The transport protocol services (see 10.1.1) invoked by the application layer are also shown.

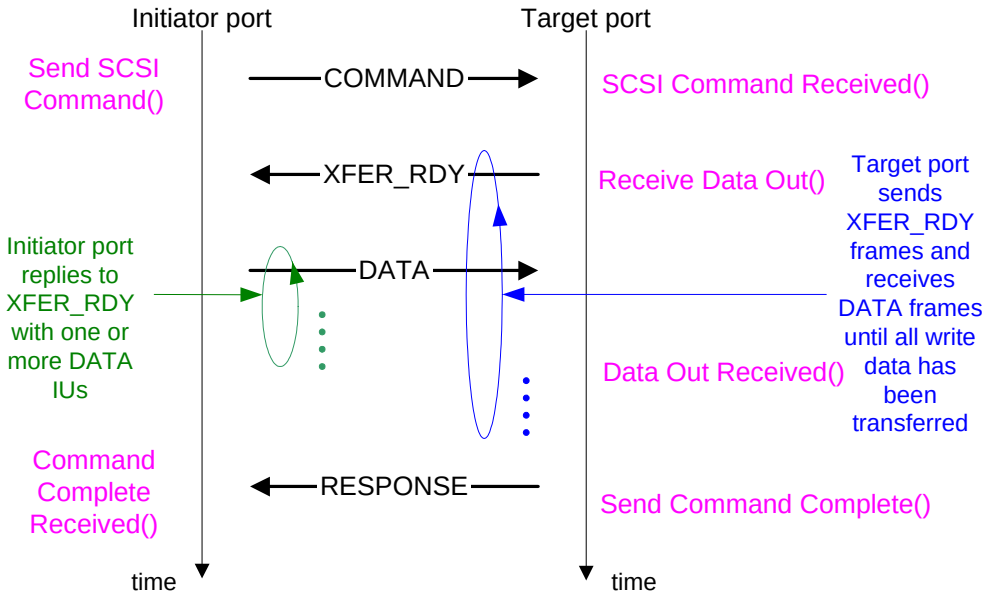


Figure 95 — Write command sequence

Figure 96 shows a read command sequence. The transport protocol services (see 10.1.1) invoked by the application layer are also shown.

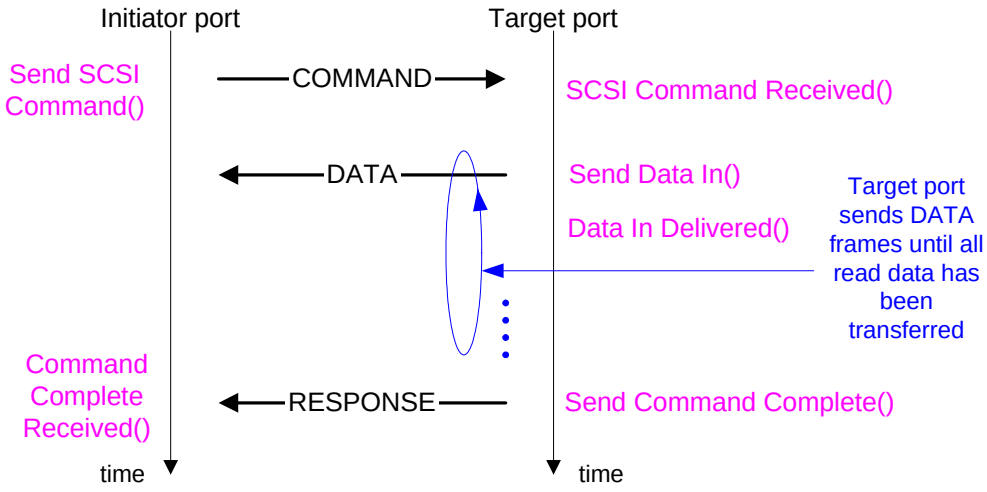


Figure 96 — Read command sequence

Figure 97 shows a bidirectional command sequence. The transport protocol services (see 10.1.1) invoked by the application layer are also shown.

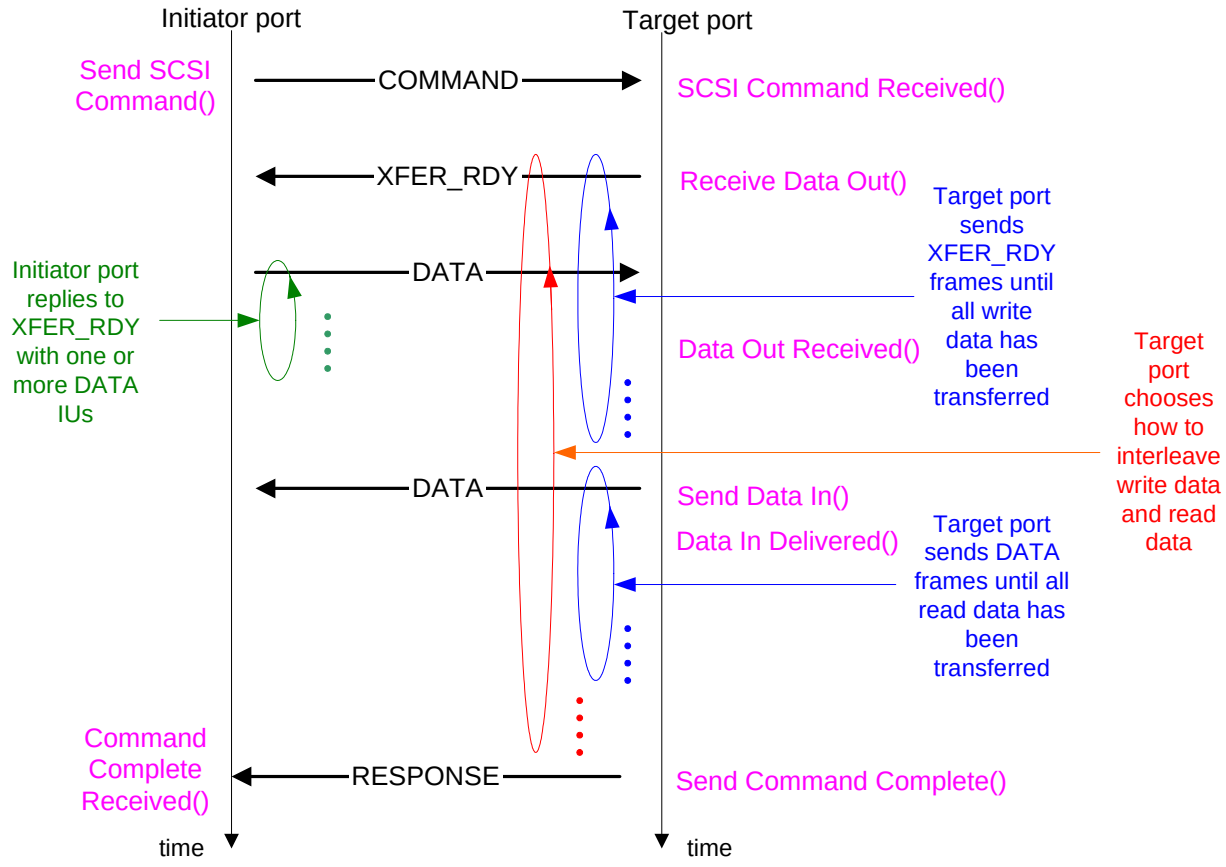


Figure 97 — Bidirectional command sequence

9.2.4 SSP transport layer handling of link layer errors

9.2.4.1 COMMAND frame

If an initiator port transmits a COMMAND frame and times out waiting for ACK or NAK it shall transmit a QUERY TASK task management function in the next connection to determine whether the command was received or not. If QUERY TASK returns a TASK MANAGEMENT FUNCTION SUCCEEDED response, the initiator port shall assume the command was ACKed. If QUERY TASK returns a TASK MANAGEMENT FUNCTION COMPLETE response, and a RESPONSE frame has not yet been received for that I_T_L_Q, the initiator port shall assume the command was NAKed or lost and may reuse the tag (see 10.1.3).

9.2.4.2 TASK frame

If an initiator port transmits a TASK frame and times out waiting for ACK or NAK it shall shall retransmit the TASK frame in a new connection (see 10.1.3).

9.2.4.3 XFER_RDY frame

If a target port transmits an XFER_RDY frame and does not receive an ACK or NAK, it shall close the connection with DONE (ACK/NAK TIMEOUT) and return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of ACK/NAK TIMEOUT (see 10.1.2).

If a target port transmits an XFER_RDY frame and receives a NAK, it shall return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of NAK RECEIVED (see 10.1.2).

9.2.4.4 DATA frame

If a target port transmits a DATA frame and does not receive an ACK or NAK, it shall close the connection with DONE (ACK/NAK TIMEOUT) and return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of ACK/NAK TIMEOUT (see 10.1.2).

If a target port transmits a DATA frame and receives a NAK, it shall return a CHECK CONDITION status for that command with a sense key of ABORTED COMMAND and an additional sense code of NAK RECEIVED (see 10.1.2).

If an initiator port transmits a DATA frame and does not receive an ACK or NAK, it shall abort the command with ABORT TASK (see 10.1.3).

If an initiator port transmits a DATA frame and receives a NAK, it shall abort the command with ABORT TASK (see 10.1.3).

9.2.4.5 RESPONSE frame

If a target port transmits a RESPONSE frame and does not receive an ACK or NAK, it shall try transmitting the RESPONSE frame again in a new connection. It shall do this at least one time. The RETRANSMIT bit shall be set to one on each of the retries (see 9.2.6.3.9).

If a target port transmits a RESPONSE frame and receives a NAK, it shall retry transmitting the RESPONSE frame at least one time (see 9.2.6.3.9).

Editor's Note 2: Perhaps for SAS-2: if the destination does not provide RRDY credit as required when opened => abort all IOs for that destination. If sending a RESPONSE frame results in NAKs or ACK/NAK timeouts for an I_T nexus loss time (rather than vendor-specific number of retries) => abort all IOs for that initiator port. If the destination replies with OPEN_REJECT (RETRY) for an I_T nexus loss time => abort all IOs for that initiator port.

If an initiator port receives a RESPONSE frame with a RETRANSMIT bit of one, and it has previously received a RESPONSE frame for the same I_T_L_Q nexus, it shall discard the extra RESPONSE frame. If it has not previously received the RESPONSE frame, it shall treat it as the valid RESPONSE frame (see 10.1.3).

9.2.5 SSP transport layer error handling

9.2.5.1 Target port error handling

If a target port receives an XFER_RDY frame or an unsupported frame type, it shall discard the frame (see 9.2.6.3.2).

If a target port receives a COMMAND frame that is too short to contain a LUN field, it shall discard the frame.

If a target port receives a COMMAND frame that contains a LUN field but is too small to contain a CDB, or the ADDITIONAL CDB LENGTH field indicates the frame should be longer than it is, the target port shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INFORMATION UNIT TOO SHORT (see 9.2.6.3.9).

If a target port receives a COMMAND frame and the ADDITIONAL CDB LENGTH field indicates the frame should be shorter than it is, the target port shall return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INFORMATION UNIT TOO LONG (see 9.2.6.3.9).

If a target port receives a TASK frame that is too short, it shall return a RESPONSE frame with a RESPONSE CODE set to TASK FRAME FIELDS INVALID (see 9.2.6.3.9).

If a target port receives a COMMAND frame with a TAG that is already in use, it may return a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of OVERLAPPED COMMANDS DETECTED (see 10.1.1.3).

If a target port receives a TASK frame with a TAG that is already in use, it may return a RESPONSE frame with a RESPONSE CODE set to TASK FRAME FIELDS INVALID (see 9.2.6.3.2).

If a target port receives a DATA frame with an unknown TAG, it shall discard the frame (see 9.2.6.3.2).

If a target port receives a DATA frame with an unknown TARGET PORT TRANSFER TAG, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of ILLEGAL TARGET PORT TRANSFER TAG RECEIVED (see 10.1.2).

If a target port receives a DATA frame with more write data than expected, it shall discard the frame and terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of TOO MUCH WRITE DATA (see 10.1.2).

If a target port receives a zero length DATA frame, it shall discard the frame and terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of INFORMATION UNIT TOO SHORT (see 10.1.2).

If a target port receives a DATA frame with a relative offset that was not expected, it shall discard that frame and any subsequent DATA frames received for that command and, shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of RELATIVE OFFSET ERROR (see 10.1.2).

9.2.5.2 Initiator port error handling

If an initiator port receives a COMMAND or TASK frame or an unsupported frame type, it shall discard the frame (see 9.2.6.2.8).

If an initiator port receives a DATA, XFER_RDY, or RESPONSE frame with an unknown TAG (including a tag for which it has sent a COMMAND or TASK frame but not yet received an ACK), it shall discard the frame. It may transmit an ABORT TASK or ABORT TASK SET to abort the command with that tag (see 9.2.6.2.8).

If an initiator port receives an XFER_RDY frame that is not twelve bytes long, it shall discard the frame. It may then transmit an ABORT TASK request to abort the command (see 10.1.3).

If an initiator port receives an XFER_RDY frame in response to a command with no write data, it shall discard the frame. It may then transmit an ABORT TASK request to abort the command (see 10.1.3).

If an initiator port receives an XFER_RDY frame requesting more write data than expected, it shall transmit an ABORT TASK to abort the command (see 10.1.3).

If an initiator port receives an XFER_RDY frame requesting zero bytes, it shall transmit an ABORT TASK to abort the command (see 10.1.3).

If an initiator port receives an XFER_RDY frame with a relative offset that was not expected, it shall transmit an ABORT TASK to abort the command (see 10.1.3).

If an initiator port receives a DATA frame with more read data than expected, it shall discard the frame and transmit an ABORT TASK to abort the command (see 10.1.3). It may receive a RESPONSE for the command before being able to abort it.

If an initiator port receives a DATA frame with zero bytes, it shall discard the frame and transmit an ABORT TASK to abort the command (see 10.1.3). It may receive a RESPONSE for the command before being able to abort it.

If an initiator port receives a DATA frame with a relative offset that was not expected, it shall discard that frame and any subsequent DATA frames received for that command and transmit an ABORT TASK to abort the command (see 10.1.3). It may receive a RESPONSE for the command before being able to abort it.

9.2.6 SSP transport layer state machines

9.2.6.1 Overview

The SSP transport layer contains state machines that perform the following functions:

a) receive transport protocol service and other SAS connection management requests from the SCSI initiator device's application layer;
b) send transport protocol service indications to the SCSI target device's application layer;
c) receive data delivery service requests, other SAS connection management requests, and transport protocol service responses from the SCSI target device's application layer;
d) send data delivery service confirmations to the SCSI target device's application layer;
e) send transport protocol service confirmations to the SCSI initiator device's application layer;
f) send requests to the SSP port layer state machines to transmit frames and manage SAS connections; and
g) receive confirmations from the SSP port layer state machines;

The SSP transport state machines are as follows:

a) Initiator Send Frame (ST_ISF state machines);
b) Initiator Receive Data (ST_IRD state machines);
c) Initiator Process Response (ST_IPR state machine);
d) Initiator Frame Router (ST_IFR state machine);
e) Target Frame Router (ST_TFR state machine); and
f) Target Transport Server (ST_TTS state machine).

9.2.6.2 Initiator device state machines

9.2.6.2.1 ~~Overview~~

The ST_ISF ~~(initiator send frame)~~ state machine receives transport protocol service requests ~~from the SCSI initiator device's application layer,~~ receives XFER_RDY Arrived parameters ~~from the ST_IFR (initiator frame router) state machine,~~ and constructs COMMAND, TASK, or data-out DATA frames. The service request may be to process either a command or task management function. This state machine also communicates with the port layer state machine via ~~several~~ requests and confirmations regarding frame transmission, and may communicate to the ST_IPR ~~(initiator process response)~~ state machine regarding service delivery subsystem failures.

The ST_ISF state machine contains the following states:

a) ST_ISF1:Send_Frame (see 9.2.6.2.2)(initial state);
b) ST_ISF2:Prepare_Command_Request (see 9.2.6.2.3); and
c) ST_ISF3:Prepare_Send_Data_Out (see 9.2.6.2.4).

The ST_IRD ~~(initiator receive data)~~ state machine receives and processes a parameter ~~from the ST_IFR (initiator frame router) state machine~~ containing a DATA frame.

The ST_IRD state machine contains the following states:

a) ST_IRD1:Receive_Data_In (see 9.2.6.2.5)(initial state); and
b) ST_IRD2:Process_Received_Data_In (see 9.2.6.2.6).

The ST_IPR ~~(initiator process response)~~ state machine receives a parameter ~~from the ST_IFR (initiator frame router) state machine~~ containing a RESPONSE frame or a parameter containing a service delivery subsystem failure ~~from the ST_ISF (initiator send frame) state machine.~~ The ST_IPR state machine processes the RESPONSE frame or the service delivery subsystem failure and sends a transport protocol service confirmation to the SCSI initiator device's application layer.

The ST_IPR state machine contains the following state:

a) ST_IPR1:Process_Received_Response (see 9.2.6.2.7)(initial state).

The ST_IFR ~~(initiator frame router)~~ state machine receives confirmations ~~from the port layer state machine~~ and, depending on the confirmation, may send a parameter to the ST_ISF ~~(initiator send frame),~~ ST_IRD

~~(initiator receive data),~~ or the ST_IPR ~~(initiator process response)~~ state machines. The ST_IFR state machine receives connection information ~~from the port layer state machine.~~ The ST_IFR state machine also receives Accept_Reject OPENs requests ~~from the SCSI initiator device's application layer~~ and sends these requests to the port layer state machine.

The ST_IFR state machine contains the following state:

a) ST_IFR1:Initiator_Frame_Router (see 9.2.6.2.8) (initial state).

Figure 98 describes the SSP transport layer (ST) initiator device state machines.

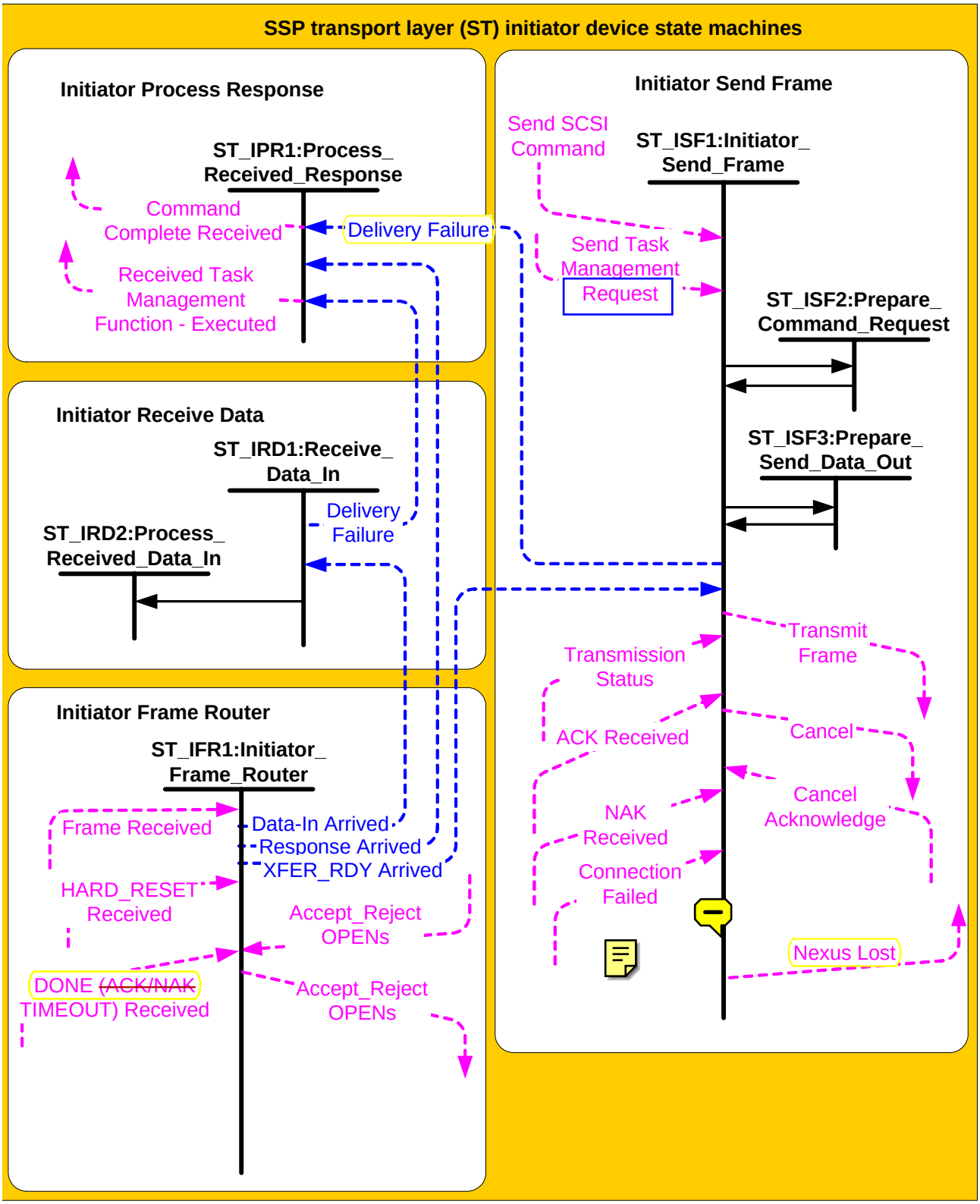


Figure 98 — SSP transport layer (ST) state machines - initiator device

9.2.6.2.2 ST_ISF1:Send_Frame state

9.2.6.2.2.1 State description

The ST_ISF state machine shall be initiated when a Send SCSI Command or a Send Task Management Request transport protocol service request is received ~~from the SCSI initiator device's application layer~~ or when an XFER_RDY Arrived parameter is received ~~from the ST_IFR (initiator frame router) state machine.~~

This state shall be entered when either a COMMAND or TASK frame is received ~~from the ST_ISF2:Prepare_Command_Request state,~~ or when a DATA frame is received ~~from the ST_ISF3:Prepare_Send_Data_Out state.~~

A Send SCSI Command or a Send Task Management Request transport protocol service request includes the following to be used in any OPEN address frames required to service the request:

a) connection rate;
b) initiator connection tag;
c) destination SAS address; and
~~d) source SAS address.~~

~~The request may also contain the initiator connection tag to be used in any OPEN address frame.~~

A Send SCSI Command transport protocol service request also includes the following to be used in any SSP frame for the request:

a) tag;
b) logical unit number;
c) task attribute;
d) additional CDB length;
e) CDB; and
f) additional CDB bytes.

If the request is for a data-out command, then the request also includes the number of bytes for the first burst size for the logical unit.

A Send Task Management Request transport protocol service request includes the following to be used in the TASK frame:

a) tag;
b) task management function; and
c) tag of ~~the~~ task to be managed.

If the ST_ISF state machine was initiated as the result of receiving a transport protocol service request, then this state shall transition to the ST_ISF2:Prepare_Command_Request state.

If the ST_ISF state machine was initiated as the result of receiving an XFER_RDY Arrived parameter, then:

a) If an XFER_RDY is not expected for the command (e.g., for a read command), then this state shall discard the frame and may send a Delivery Failure (XFER_RDY Not Expected) parameter to the ST_IPR:Process_Received_Response state. This state machine shall terminate if it sends the parameter;
b) If the length of the information unit is not 12 bytes, then this state shall discard the frame and may send a Delivery Failure (XFER_RDY Information Unit Too Short) parameter or Delivery Failure (XFER_RDY Information Unit Too Long) to the ST_IPR:Process_Received_Response state. This state machine shall terminate after sending the parameter;
c) If the length of the XFER_RDY information unit is 12 bytes and the write data length is zero or exceeds the amount of data remaining to be transferred for the data-out command, then this state shall send a Delivery Failure (XFER_RDY Incorrect Write Data Length) parameter to the ST_IPR:Process Received_Response state. This state machine shall terminate after sending the parameter;
d) If the length of the XFER_RDY information unit is 12 bytes and the relative offset is not expected, then this state shall send a Delivery Failure (XFER_RDY Relative Offset Error) parameter to the

ST_IPR:Process Received_Response state. This state machine shall terminate after sending the parameter; and

e) If the length of the XFER_RDY frame is 12 bytes, the write data length is correct, and an ACK Transmitted confirmation has been received, then this state shall transition to the ST_ISF3:Prepare_Send_Data_Out state.

If this state is entered from the ST_ISF2:Prepare_Command_Request state, then this state shall send a Transmit Frame (Interlocked) request to the port layer state machine.

If this state is entered from the ST_ISF3:Prepare_Send_Data_Out state, then this state shall send a Transmit Frame (Non-interlocked) request to the port layer state machine.

A Transmit Frame request ~~from this state~~ (either interlocked or non-interlocked) shall include the SSP frame and the following to be used for any OPEN address frame:

a) the initiator bit set to one;
b) connection rate;
c) initiator connection tag;
d) destination SAS address; and
e) source SAS address.

After sending a Transmit Frame request this state shall wait for a Transmission Status confirmation. If the confirmation is not Transmission Status (Frame Transmitted), then this state shall send a Delivery Failure (Service Delivery Subsystem Failure) parameter to the ST_IPR ~~(initiator process response)~~ state machine. The Delivery Failure attribute shall include:

a) any argument received with the Transmission Status confirmation; and
b) I_T_L_x nexus information (i.e., destination SAS address and tag);

After sending a Delivery Failure parameter to the ST_IPR state machine, the ST_ISF state machine shall terminate.

If the transmitted frame was a DATA frame, then this state may transition to the ST_ISF3:Prepare_Send_Data_Out state after receiving a Transmission Status (Frame Transmitted) confirmation.

After receiving a Transmission Status (Frame Transmitted) confirmation, this state shall then wait for one of the following confirmations ~~from the port layer state machine.~~

a) ACK Received;
b) NAK Received; or
c) Connection Failed.

If the transmitted frame was a COMMAND frame or TASK frame requiring a data-out operation, then this state shall wait to receive an ACK Received, NAK Received, or Connection Failed confirmation before transitioning from this state.

If a NAK Received confirmation is received, then this state shall send a Delivery Failure (Service Delivery Subsystem Failure - NAK Received) parameter to the ST_IPR state machine.

If a Connection Failed confirmation is received, then this state shall send a Delivery Failure (Service Delivery Subsystem Failure - Connection Failed) parameter to the ST_IPR state machine.

After sending a Delivery Failure parameter to the ST_IPR state machine, the ST_ISF state machine shall terminate.

This state may also send a Cancel request to the port layer state machine to cancel a previous Transmit Frame request. The ST_ISF state machine shall terminate upon receipt of a Cancel Acknowledge confirmation.

9.2.6.2.2.2 Transition ST_ISF1:Send_Frame to ST_ISF2:Prepare_Command_Request

This transition shall occur after a Send SCSI Command or Send Task Management Request transport protocol service request has been received.

9.2.6.2.2.3 Transition ST_ISF1:Send_Frame to ST_ISF3:Prepare_Send_Data_Out

This transition shall occur after:

a) an ACK Received confirmation has been received for a COMMAND frame for a data-out operation and the first burst size is not zero;
b) an XFER_RDY Arrived parameter has been received, all required values are present and correct, and an ACK Transmitted confirmation has been received; or
c) a Transmission Status (Frame Transmitted) confirmation for a Transmit Frame (Non-interlocked) request has been received and the number of data bytes that has been transmitted for the request is less than the first burst size or the write data length.

9.2.6.2.3 ST_ISF2:Prepare_Command_Request state

9.2.6.2.3.1 State description

This state shall construct either a COMMAND or TASK frame.

If the frame to be constructed is a COMMAND frame, then this state shall include the following ~~received from the SCSI initiator device's application layer~~ in the frame:

a) tag;
b) logical unit number;
c) task attribute;
d) additional CDB length;
e) CDB; and
f) additional CDB bytes.

If the frame to be constructed is a TASK frame, then this state shall include the following ~~received from the SCSI initiator device's application layer~~ in the frame:

a) tag;
b) task management function; and
c) tag of task to be managed.

This state shall generate and include the following in either a COMMAND or TASK frame:

a) frame type;
b) hashed destination SAS address;
c) hashed source SAS address;
d) retransmit bit set to zero; and
e) number of fill bytes.

9.2.6.2.3.2 Transition ST_ISF2:Prepare_Command_Request to ST_ISF1:Send_Frame

This transition shall occur after the ST_ISF2:Prepare_Command_Request state has constructed a COMMAND or TASK frame.

9.2.6.2.4 ST_ISF3:Prepare_Send_Data_Out state

9.2.6.2.4.1 State description

This state shall construct a DATA frame. This state shall include the following in the frame ~~(these were received either from the SCSI initiator device's application layer or included in an XFER_RDY Arrived parameter):~~

a) tag;
b) target port transfer tag; and
c) relative offset.

This state shall generate and include the following in the frame:

a) frame type;
b) hashed destination SAS address;

c) hashed source SAS address;
d) retransmit bit set to zero;
e) number of fill bytes; and
f) fill bytes.

This state shall include requested data in the frame. If all of the data for the request is not included in the frame, the number of data bytes in the frame shall be a multiple of four, and the number of fill bytes shall be zero.

9.2.6.2.4.2 Transition ST_ISF3:Prepare_Send_Data_Out to ST_ISF1:Send_Frame

This transition shall occur after the ST_ISF3:Prepare_Send_Data_Out state has constructed a DATA frame.

9.2.6.2.5 ST_IRD1:Receive_Data_In state

9.2.6.2.5.1 State description

The ST_IRD state machine shall be initiated when a Data-In Arrived parameter is received.

This state shall check the length and relative offset of the DATA information unit.

If the length of the information unit is zero, then this state shall send a Delivery Failure (DATA Incorrect Read Data Length) parameter to the ST_IPR:Process_Received_Response state.

If the length of the information unit is zero or exceeds the amount of data remaining to be transferred for the data-in command, then this state shall send a Delivery Failure (DATA Too Much Read Data) parameter to the ST_IPR:Process_Received_Response state.

If the relative offset is not the expected offset, then this state shall send a Delivery Failure (DATA Relative Offset Error) parameter to the ST_IPR:Process_Received_Response state.

This state machine shall terminate after sending the parameter.

9.2.6.2.5.2 Transition ST_IRD1:Receive_Data_In to ST_IRD2:Process_Received_Data_In

This transition shall occur after any value to be checked by the ST_IRD1:Receive_Data_In has been checked.

9.2.6.2.6 ST_IRD2:Process_Received_Data_In state

This state shall process the data-in data received by the ST_IRD1:Receive_Data_In state.

The ST_IRD state machine shall terminate after the data-in data is processed.

9.2.6.2.7 ST_IPR1:Process_Received_Response state

The ST_IPR state machine shall be initiated when a Response Arrived parameter is received or a Delivery Failure parameter is received.

If a Delivery Failure parameter is received, this state shall send a Command Complete Received or Received Task Management Function – Executed confirmation to the SCSI initiator device's application layer with the Service Response argument set as indicated by table 100.

Table 100 — Delivery Failure to Command Complete Received mapping

Delivery Failure argument	Command Complete Received (Service Response)
XFER_RDY Information Unit Too Short	Service Delivery or Target Failure - XFER_RDY Information Unit Too Short
XFER_RDY Information Unit Too Long	Service Delivery or Target Failure - XFER_RDY Information Unit Too Long
XFER_RDY Incorrect Write Data Length	Service Delivery or Target Failure - XFER_RDY Incorrect Write Data Length
XFER_RDY Relative Offset Error	Service Delivery or Target Failure - XFER_RDY Relative Offset Error
XFER_RDY Not Expected	Service Delivery or Target Failure - XFER_RDY Not Expected
DATA Incorrect Data Length	Service Delivery or Target Failure - DATA Incorrect Data Length
DATA Too Much Read Data	Service Delivery or Target Failure - DATA Too Much Read Data
DATA Relative Offset Error	Service Delivery or Target Failure - DATA Relative Offset Error
NAK Received	Service Delivery or Target Failure - NAK Received
Connection Failed	Service Delivery or Target Failure - Connection Failed

If a Response Arrived parameter is received, this state shall check the length of the RESPONSE information unit.

If the length of the information unit is not correct, then this state shall send a Command Complete Received (Service Response = Service Delivery or Target Failure) or Received Task Management Function – Executed (Service Response = Service Delivery or Target Failure) confirmation to the SCSI initiator device's application layer. The confirmation shall include the tag.

If the length is correct, this state shall send a Command Complete Received (Service Response = Task Complete), Command Complete Received (Service Response = Linked Command Complete), or Received Task Management Function – Executed (Service Response = Function Complete) transport protocol service confirmation. The confirmation shall also include a Retransmit argument indicating the state of the RETRANSMIT bit.

The ST_IPR state machine shall terminate after sending a confirmation.

9.2.6.2.8 ST_IFR1:Initiator_Frame_Router state

The ST_IFR state machine shall be initiated when:

a) an Accept_Reject OPENs request is received;
b) a Frame Received confirmation is received;
c) ~~a DONE Received confirmation is received;~~ or
d) a hard reset occurs.

If the ST_IFR state machine was initiated as the result of receiving an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request, then this state shall send an Accept_Reject OPENs request along with the received argument to the port layer state machine. The ST_IFR state machine shall terminate after sending an Accept_Reject OPENs request to the port layer state machine.

If the ST_IFR state machine was initiated as the result of receiving a Frame Received (Frame Failed) or a hard reset, then the ST_IFR state machine shall terminate.

If the ST_IFR state machine was initiated as the result of receiving a DONE Received, then this state shall notify the application layer and shall terminate.

If the ST_IFR state machine was initiated as the result of a Frame Received (ACK/NAK balanced) or Frame Received (ACK/NAK Not Balanced) confirmation, then this state shall check the frame type in the received frame. If the confirmation was Frame Received (ACK/NAK Balanced) and the frame type is not XFER_RDY, RESPONSE, or DATA, then the ST_IFR state machine shall discard the frame and terminate. If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then the ST_IFR state machine shall discard the frame and terminate.

If the frame type is correct relative to the confirmation, then this state may check that the hashed SAS source address and the hashed destination address in the frame match the SAS source address and SAS destination address used to open the connection. If this state checks these addresses and they do not match, then the ST_IFR state machine shall terminate.

If the frame type is:

a) XFER_RDY, then this state shall send a XFER_RDY Arrived parameter to the ST_ISF1:Send_Frame state specified by the tag;
b) RESPONSE, then this state shall send a Response Arrived parameter to the ST_IPR1:Process_Received_Response state specified by the tag; or
c) DATA, then this state shall send a Data-in parameter to the ST_IRD1:Receive_Data_In state specified by the tag.

Each of these parameters shall contain the content of the SAS frame. If the tag does not specify an existing state machine, then this state shall discard the frame and may send a vendor-specific confirmation to the application layer to abort the command using that tag.

The ST_IFR state machine shall terminate after sending a parameter to another state machine.

9.2.6.3 Target device state machines

9.2.6.3.1 Overview

The ST_TFR ~~(target frame router)~~ state machine receives confirmations ~~from the port layer state machine~~ and sends a transport protocol service indication to the SCSI target device's application layer or a Data-Out Received parameter to the ST_TTS ~~(target transport server)~~ state machine. The ST_TFR state machine also receives Accept_Reject OPENs requests ~~from the SCSI target device's application layer~~ and sends these requests to the port layer state machine.

The ST_TFR state machine contains the following state:

a) ST_TFR1:Target_Frame_Router (see 9.2.6.3.2)(initial state).

The ST_TTS ~~(target transport server)~~ state machine performs the following functions:

b) processes and sends transport protocol service confirmations to the SCSI target device's application layer;
c) receives and processes transport protocol service responses ~~from the SCSI target device's application layer;~~ and
d) communicates with the port layer state machine via ~~several~~ requests and confirmations regarding frame transmission.

The ST_TTS state machine contains the following states:

a) ST_TTS1:Request_Response_Router (see 9.2.6.3.3)~~(initial state);~~
b) ST_TTS2:Send_Frame (see 9.2.6.3.4);
c) ST_TTS3:Prepare_Send_Data_In (see 9.2.6.3.5);
d) ST_TTS4:Receive_Data_Out (see 9.2.6.3.6);
e) ST_TTS5:Prepare_XFER_RDY (see 9.2.6.3.7);
f) ST_TTS6:Process_Received_Data_Out (see 9.2.6.3.8); and

g) ST_TTS7:Prepare_Response (see 9.2.6.3.9).

Figure 99 describes the SSP transport layer (ST) target device state machines.

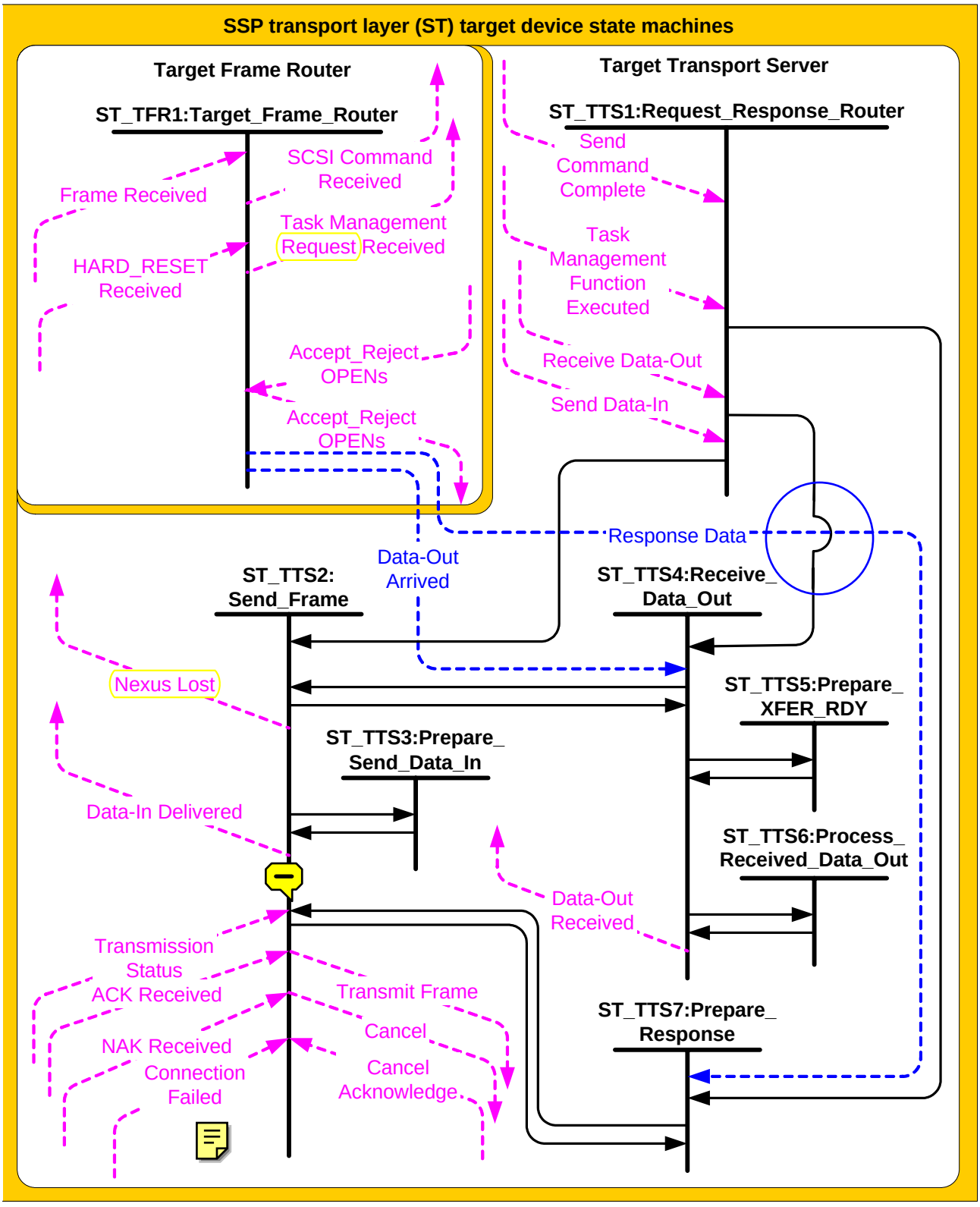


Figure 99 — SSP transport layer (ST) state machines - target device

9.2.6.3.2 ST_TFR1:Target_Frame_Router state

The ST_TFR state machine shall be initiated when:

a) an Accept_Reject OPENs request is received;
b) a Frame Received confirmation is received; or
c) a hard reset occurs.

If the ST_TFR state machine was initiated as the result of receiving an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request ~~from the SCSI target device's application layer,~~ then this state shall send an Accept_Reject OPENs request along with the received attribute to the port layer state machine. The ST_TFR state machine shall terminate after sending an Accept_Reject OPENs request to the port layer state machine.

If the ST_TFR state machine was initiated as the result of receiving a hard reset, then the ST_TFR state machine shall terminate.

If the ST_TFR state machine was initiated as the result of receiving a Frame Received (ACK/NAK Balanced) or Frame Received (ACK/NAK Not Balanced) confirmation ~~from the port layer state machine,~~ then this state shall check the frame type in the received frame. If the frame type is not COMMAND, TASK, or DATA, then this state shall discard the frame and terminate.

If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then this state shall discard the frame and terminate.

If the frame type is correct relative to the confirmation, then this state may check that the hashed SAS source address and the hashed destination address in the frame match the SAS source address and SAS destination address used to open the connection. If this state checks these addresses and they do not match, then the ST_TFR state machine shall discard the frame and terminate.

If the frame type is DATA, and the tag does not match a tag for an outstanding data-out command, then the ST_TFR state machine shall discard the frame and terminate.

If the frame type is COMMAND, then this state shall check that the length of the information unit is [28 + (4 x additional CDB length)] bytes. If the length of the information unit is less than the calculated length, then this state shall send a Response Data (Information Unit Too Short) parameter to the ST_TTS7:Prepare_Response state. If the length of the information unit is greater than the calculated length, then this state shall send a Response Data (Information Unit Too Long) parameter to the ST_TTS7:Prepare_Response state.

If the frame type is TASK, then this state shall check that the length of the information unit is 28 bytes. If the length of the information unit is not 28 bytes, then this state shall send a Response Data (Task Frame Fields Invalid) parameter to the ST_TTS7:Prepare_Response state.

If the frame type is COMMAND and the length of the information unit is correct, then this state shall send a SCSI Command Received transport protocol service indication to the SCSI target device's application layer.

If the frame type is TASK and the length of the information unit is correct, then this state may check if the tag conflicts with an existing tag (i.e., an existing command or task management function). If it conflicts, this state may send a Response Data (Task Frame Fields Invalid) parameter to the ST_TTS7:Prepare_Response state. If it does not check the tag or the tag does not conflict, this state shall send a Task Management Request Received transport protocol service indication to the SCSI target device's application layer. If the frame type is DATA, then this state shall send a Data-Out Arrived parameter to the ST_TTS4:Receive_Data_Out state. Each indication or parameter shall contain the content of the SAS frame.

The ST_TFR state machine shall terminate after sending a Data-Out Arrived parameter or transport protocol service indication.

9.2.6.3.3 ST_TTS1:Target_Request_Response_Router state

9.2.6.3.3.1 State description

The ST_TTS state machine shall be initiated when one of the following is received ~~from the SCSI target device's application layer:~~

a) a Send Data-In transport protocol service request;
b) a Receive Data-Out transport protocol service request;
c) a Task Management Function Executed transport protocol service response; or
d) a Send Command Complete transport protocol service response.

The request or response includes the following to be used in any OPEN address frames required to service the request or response:

a) connection rate;
b) initiator connection tag;
c) destination SAS address; and
d) source SAS address.

A Send Data-In transport protocol service request ~~also~~ includes the following:

a) I_T_L_x nexus (e.g., tag);
b) device server buffer (e.g., starting logical block address); and
c) request byte count (e.g., transfer length).

A Receive Data-Out transport protocol service request ~~also~~ includes the following:

a) I_T_L_x nexus (e.g., tag);
b) device server buffer (e.g., starting logical block address); and
c) request byte count (e.g., transfer length).

A Task Management Function Executed transport protocol service response or Send Command Complete transport protocol service response ~~also~~ includes the following:

a) I_T_L_x nexus (e.g., tag);
b) task management function; and
c) tag of task to be managed.

9.2.6.3.3.2 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS2:Send_Frame

This transition shall occur after the ST_TTS1:Target_Request_Response_Router state has received a Send Data-In transport protocol service request ~~from the SCSI target device's application layer.~~

9.2.6.3.3.3 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS4:Receive_Data_Out

This transition shall occur after the ST_TTS1:Target_Request_Response_Router state has received a Receive Data-Out transport protocol service request ~~from the SCSI target device's application layer.~~

9.2.6.3.3.4 Transition ST_TTS1:Target_Request_Response_Router to ST_TTS7:Prepare_Response

This transition shall occur after the ST_TTS1:Target_Request_Response_Router state has received a Task Management Function Executed transport protocol service response or a Send Command Complete transport protocol service response ~~from the SCSI target device's application layer.~~

9.2.6.3.4 ST_TTS2:Send_Frame state

9.2.6.3.4.1 State description

~~This state is entered when a DATA frame is received from the ST_TTS3:Prepare_Send_Data_In state, when an XFER_RDY frame is received from the ST_TTS4:Receive_Data_Out state, when a RESPONSE frame is received from the ST_TTS7:Prepare_Response state, or after the ST_TTS7:Prepare_Response state has determined that the vendor-specific number of retries for a RESPONSE frame has been exceeded.~~

If the TTS state machine was initiated as the result of ~~this state~~ receiving a Send Data-In transport protocol service request, the specified values are included with the request, and this state has received an ACK Transmitted confirmation, then this state shall transition to the ST_TTS3:Prepare_Send_Data_In state.

If this state is entered from the ST_TTS3:Prepare_Send_Data_In state for transmission of a DATA frame, then this state shall send a Transmit Frame (Non-interlocked) request to the port layer state machine.

If this state is entered from the ST_TTS4:Receive_Data_Out state for transmission of an XFER_RDY frame and this state has received an ACK Transmitted confirmation, then this state shall send a Transmit Frame (Interlocked) request to the port layer state machine.

If this state is entered from the ST_TTS7:Prepare_Response state for transmission of a RESPONSE frame and this state has received an ACK Transmitted confirmation, then this state shall send a Transmit Frame (Interlocked) request to the port layer state machine.

A Transmit Frame request from this state (either interlocked or non-interlocked) shall include the SSP frame and the following to be used for any OPEN address frame:

a) the initiator bit set to zero;
b) connection rate;
c) initiator connection tag;
d) destination SAS address; and
e) source SAS address.

After sending a Transmit Frame request this state shall receive a Transmission Status confirmation ~~from the port layer state machine.~~

If the frame transmitted was a DATA frame, then this state may transition to the ST_TTS3:Prepare_Send_Data_In state after receiving a Transmission Status (Frame Transmitted) confirmation.

If the confirmation is Transmission Status (Open Failed) and it includes an I_T Nexus Lost argument, this state shall send a Nexus Lost confirmation to the application layer.

If the confirmation is Transmission Status (Frame Transmitted) confirmation, then this state shall receive one of the following confirmations ~~from the port layer state machine.~~

a) ACK Received;
b) NAK Received; or
c) Connection Failed.

If the frame transmitted was an XFER_RDY frame or a RESPONSE frame, then this state shall wait to receive an ACK Received, NAK Received, or Connection Failed confirmation before transitioning from this state.

If the confirmation is ACK Received and the transmitted frame was an XFER_RDY frame, then this state shall transition to the ST_TTS4:Receive_Data_Out state.

This state shall send a Data-In Delivered (Delivery Result = DELIVERY SUCCESSFUL) transport protocol service confirmation to the SCSI target device's application layer after one of the following:

a) If a confirmation transmission status parameter for a DATA frame was Transmission Status (Frame Transmitted) followed by an ACK Received confirmation and the number of bytes moved for the Send Data-In transport protocol service request equals the Request Byte Count; or
b) If a confirmation transmission status parameter for a RESPONSE frame was Transmission Status (Frame Transmitted) followed by an ACK Received confirmation.

This state shall send a Data-In Delivered (Delivery Result = DELIVERY FAILURE - NAK RECEIVED) transport protocol service confirmation to the SCSI target device's application layer if the received transmission status confirmation parameter for a DATA or XFER_RDY frame was Transmission Status (Frame Transmitted) followed by a confirmation of NAK Received.

This state shall send a Data-In Delivered (Delivery Result = DELIVERY FAILURE - ACK/NAK TIMEOUT) transport protocol service confirmation to the SCSI target device's application layer if the received transmission status confirmation parameter for a DATA or XFER_RDY frame was Transmission Status (Frame

Transmitted) followed by a confirmation of Connection Failed (ACK/NAK Timeout) or Connection Failed (Connection Lost Without ACK/NAK).

A Data-In Delivered transport protocol service confirmation to the SCSI target device's application layer confirmation shall include the following:

a) any argument received from the port layer state machine (e.g., Transmission Status (Frame Transmitted) or Service Delivery Subsystem Failure; and
b) I_T_L_x nexus information (i.e., destination SAS address and tag).

The ST_TTS state machine shall terminate after sending the Data-In Delivered confirmation.

This state may also send a Cancel request to the port layer state machine to cancel a previous Transmit Frame request. The ST_TTS state machine terminates upon receipt of a Cancel Acknowledge confirmation ~~from the port layer state machine.~~

9.2.6.3.4.2 Transition ST_TTS2:Send_Frame to ST_TTS3:Prepare_Send_Data_In

This transition shall occur after either:

a) this state receives a Send Data-In transport protocol service request and an ACK Transmitted confirmation, or
b) this state receives a Transmission Status (Frame Transmitted) confirmation ~~and an ACK Received confirmation~~ for a DATA frame and the number of bytes moved for the Send Data-In transport protocol service request is less than the Request Byte Count.

9.2.6.3.4.3 Transition ST_TTS2:Send_Frame to ST_TTS4:Receive_Data_Out

This transition shall occur after this state has received a Transmission Status (Frame Transmitted) confirmation and an ACK Received confirmation for an XFER_RDY frame.

9.2.6.3.4.4 Transition ST_TTS2:Send_Frame to ST_TTS7:Prepare_Response

This transition shall occur if this state receives one of the following confirmations for a RESPONSE frame ~~from the port layer state machine:~~

a) Transmission Status with an attribute other than (Frame Transmitted);
b) NAK Received; or
c) Connection Failed.

9.2.6.3.5 ST_TTS3:Prepare_Send_Data_In state

9.2.6.3.5.1 State description

This state fetches the data from the Device Server Buffer and constructs a DATA frame. This state shall use the tag received from the ST_TTS2:Send_Frame state to construct the frame.

This state shall generate the following to be used in the frame:

a) frame type;
b) hashed destination SAS address;
c) hashed source SAS address;
d) retransmit bit set to zero;
e) number of fill bytes;
f) relative offset; and
g) fill bytes.

9.2.6.3.5.2 Transition ST_TTS3:Prepare_Send_Data_In to ST_TTS2:Send_Frame

This transition shall occur after the ST_TTS3:Prepare_Send_Data_In state has constructed a DATA frame.

9.2.6.3.6 ST_TTS4:Receive_Data_Out state

9.2.6.3.6.1 State description

~~This state is entered after one of the following occurs:~~

~~a) a Receive Data-Out service request is received from the ST_TS1:Request_Response_Router state;~~
~~b) a DATA frame is received from the ST_TFR (target frame router) state machine;~~
~~c) an ACK Received confirmation for an XFER_RDY frame was received from the ST_TTS2:Send_Frame state;~~
~~d) an XFER_RDY frame has been constructed by the ST_TTS5:Prepare_XFER_RDY state; or~~
~~e) data-out data has been processed by the ST_TTS6:Process_Received_Data_Out state.~~

If this state was entered as the result of receiving a Receive Data-Out service request from the ST_TS1:Request_Response_Router state then this state shall transition to the ST_TTS5:Prepare_XFER_RDY state.

If this state was entered as the result of receiving a DATA frame from the ST_TFR state machine, then this state shall check the target transport tag value in the DATA frame. If the value does not match, then this state shall send a Data-Out Received (Delivery Result = DELIVERY FAILURE - ILLEGAL TARGET PORT TRANSFER TAG RECEIVED) transport protocol service confirmation to the SCSI target device's application layer. This confirmation shall include the tag. The ST_TTS state machine shall terminate after sending the confirmation.

If this state was entered as the result of receiving a DATA frame from the ST_TFR state machine, then this state shall check the length of the data. If the length of the data exceeds that specified by the XFER_RDY frame that requested the data, then this state shall send a Data-Out Received (Delivery Result = DELIVERY FAILURE - TOO MUCH WRITE DATA) transport protocol service confirmation to the SCSI target device's application layer. This confirmation shall include the tag. The ST_TTS state machine shall terminate after sending the confirmation.

If this state was entered as the result of receiving a DATA frame from the ST_TFR state machine, then this state shall check the length of the data. If the length of the data is zero, then this state shall send a Data-Out Received (Delivery Result = DELIVERY FAILURE - INFORMATION UNIT TOO SHORT) transport protocol service confirmation to the SCSI target device's application layer. This confirmation shall include the tag. The ST_TTS state machine shall terminate after sending the confirmation.

If this state was entered as the result of receiving a DATA frame from the ST_TFR state machine, then this state shall check the relative offset. If the relative offset was not expected, then this state shall send a Data-Out Received (Delivery Result = DELIVERY FAILURE - RELATIVE OFFSET ERROR) transport protocol service confirmation to the SCSI target device's application layer. This confirmation shall include the tag. The ST_TTS state machine shall terminate after sending the confirmation.

If the target transport tag value matches the value sent with the corresponding XFER_RDY frame, and the length of the data does not exceed that specified by the XFER_RDY frame that requested the data, then this state shall transition to the ST_TTS6:Process_Received_Data_Out state.

If this state is entered as the result of the ST_TTS2:Send_Frame state receiving an ACK Received confirmation ~~from the port layer state machine~~ for an XFER_RDY frame, then this state shall wait for a Data-Out Arrived parameter ~~from the ST_TFR state machine.~~

If this state is entered from the ST_TTS5:Prepare_XFER_RDY state, then this state shall transition to the ST_TTS2:Send_Frame state.

If this state is entered from the ST_TTS6:Process_Received_Data_Out state and the number of bytes moved for the Receive Data-Out transport protocol service request is less than the Request Byte Count, then this state shall wait for a Data-Out Arrived parameter ~~from the ST_TFR1:Target_Frame_Router state.~~

If this state is entered from the ST_TTS6:Process_Received_Data_Out state and number of bytes moved for the Receive Data-Out transport protocol service request equals the Request Byte Count, then this state shall send a Data-Out Received (Delivery Result = DELIVERY SUCCESSFUL) transport protocol service confirmation to the SCSI target device's application layer. This confirmation shall include the tag. The ST_TTS state machine shall terminate after sending the confirmation.

9.2.6.3.6.2 Transition ST_TTS4:Receive_Data_Out to ST_TTS2:Send_Frame

This transition shall occur after this state has received an XFER_RDY frame from the ST_TTS5:Prepare_XFER_RDY state.

9.2.6.3.6.3 Transition ST_TTS4:Receive_Data_Out to ST_TTS5:Prepare_XFER_RDY

This transition shall occur after a Receive Data-Out transport protocol service request has been received by the ST_TTS4:Receive_Data_Out state from the ST_TTS1:Request_Response_Router state.

9.2.6.3.6.4 Transition ST_TTS4:Receive_Data_Out to ST_TTS6:Process_Received_Data_Out

This transition shall occur after the ST_TTS4:Receive_Data_Out state receives a Data-Out Arrived parameter ~~from the ST_TFR (target frame router) state machine.~~

9.2.6.3.7 ST_TTS5:Prepare_XFER_RDY state

9.2.6.3.7.1 State description

This state shall construct an XFER_RDY frame. This state shall use the following received from the Receive Data-Out transport protocol service request to construct the frame:

a) tag;
b) target port transfer tag;
c) relative offset; and
d) write data length.

This state shall generate the following to be used in the frame:

a) frame type;
b) hashed destination SAS address;
c) hashed source SAS address;
d) retransmit bit set to zero; and
e) number of fill bytes.

9.2.6.3.7.2 Transition ST_TTS5:Prepare_XFER_RDY to ST_TTS4:Receive_Data_Out

This transition shall occur after the ST_TTS5:Prepare_XFER_RDY state has constructed an XFER_RDY frame.

9.2.6.3.8 ST_TTS6:Process_Received_Data_Out state

9.2.6.3.8.1 State description

This state shall process the data received in a Data-Out parameter using the Device Server Buffer (e.g., logical block address) to which the data is to be transferred.

9.2.6.3.8.2 Transition ST_TTS6:Process_Received_Data_Out to ST_TTS4:Receive_Data_Out

This transition shall occur after data received in a Data-Out parameter has been processed.

9.2.6.3.9 ST_TTS7:Prepare_Response state

9.2.6.3.9.1 State description

If not already running, the ST_TTS state machine shall be initiated when a Response Data parameter is received ~~by this state from the ST_TFR state machine.~~

~~This state is entered after one of the following occurs:~~

~~a) a Response Data parameter is received by this state from the ST_TFR state machine;~~
~~b) a Task Management Function Executed transport protocol service response was received by the ST_TTS1:Target_Request_Response_Router state from the SCSI target device's application layer;~~

c) a Send Command Complete transport protocol service response was received by the ST_TTS1:Target_Request_Response_Router state from the SCSI target device's application layer; or
d) the ST_TTS2:Send_Frame state receives something other than a Transmission Status (Frame Transmitted) confirmation followed by an ACK Received confirmation for a RESPONSE frame ~~from the port layer state machine~~ (i.e., the frame transmission was unsuccessful).

If this state was entered as the result of receiving a Response Data parameter ~~from the ST_TFR state machine,~~ or a Task Management Function Executed transport protocol service response or a Send Command Complete transport protocol service response ~~from the ST_TTS1:Target_Request_Response_Router state,~~ then this state shall construct a RESPONSE frame.

If this state was entered as the result of receiving a Response Data parameter ~~from the ST_TFR state machine,~~ this state shall use the tag received in the parameter. If the argument for the Response Data parameter was Information Unit Too Short or Information Unit Too Long, then this state shall set the STATUS field in the RESPONSE frame to CHECK CONDITION and the additional sense code to INFORMATION UNIT TOO SHORT or INFORMATION UNIT TOO LONG respectively. If the argument was Task Frame Fields Invalid, then this state shall set the RESPONSE CODE field to TASK FRAME FIELDS INVALID.

Editor's Note 3: Requires SAM-3 changes: It would be cleaner if SCSI Command Received () and Task Management Function Request () each included Service Response arguments to carry this type of error report, so the application layer would generate all CHECK CONDITION responses. The transport layer still would have to generate all RESPONSE CODE responses.

This state shall use the following received from the SCSI target device's application layer to construct the frame:

a) tag;
b) status
c) response data; and
d) sense data.

This state shall generate the following to be used in the frame:

a) frame type;
b) hashed destination SAS address;
c) hashed source SAS address;
d) retransmit bit set to zero;
e) number of fill bytes;
f) fill bytes;
g) data present;
h) sense data list length; and
i) response data list length.

If this state was entered as the result of the ST_TTS2:Send_Frame state receiving something other than a Transmission Status (Frame Transmitted) confirmation followed by an ACK Received confirmation for a RESPONSE frame ~~from the port layer state machine~~ (i.e., the frame transmission was unsuccessful), then this state shall check to see if the vendor-specific number of retries for the RESPONSE frame has been exceeded.

If the vendor-specific number of retries has not been exceeded, the this state generate a RESPONSE frame using all of the values for the previous RESPONSE frame except that the retransmit bit shall be set to one.

Editor's Note 4: Requires SAM-3 changes: It would be nice if Send Command Complete () and Task Management Function Executed () each included Service Response outputs to notify the device server that the target port was unable to deliver the response. It might want to set a unit attention for that initiator or treat the situation as an I_T Nexus loss.

9.2.6.3.9.2 Transition ST_TTS7:Prepare_Response to ST_TTS2:Send_Frame

This transition shall occur after the ST_TTS7:Prepare_Response state has constructed a RESPONSE frame or if the vendor-specific number of retries for transmission of a RESPONSE frame has been exceeded.

9.3 STP transport layer

9.3.1 Initial FIS

A SATA target port transmits a Register - Device to Host FIS after completing the link reset sequence. The expander device shall update a set of shadow registers with these contents and shall not deliver them to any STP initiator port. The STP initiator ports may read the shadow register contents using the SMP REPORT SATA PORT function (see 10.3.1.6).

9.3.2 SATA tunneling for multiple STP initiator ports

After power-on reset or after a SMP PHY CONTROL request specifying a PHY OPERATION of HARD RESET, the expander device may accept a connection request to the SATA target port from any STP initiator port. Once the expander device has accepted a connection request for that SATA target port ~~from an STP initiator port,~~ the expander device shall reject all subsequent connection requests for that SATA target port from other STP initiator ports with OPEN REJECT (STP RESOURCES BUSY). The expander device shall continue to reject connection requests from other STP initiator ports to that SATA target port even if the connection with the original STP initiator port is closed. In this state, the expander device is said to maintain an affiliation between the STP initiator port and the SATA target port.

This affiliation shall persist until any of the following occurs:

a) Power on;
b) An SMP PHY CONTROL request specifying a PHY OPERATION of HARD RESET from any initiator specifying the phy with an affiliation is received;
c) An STP initiator port issues an SMP PHY CONTROL request specifying a PHY OPERATION of CLEAR AFFILIATION specifying the phy which has an affiliation with that STP initiator; or
d) An STP initiator closes a connection to the affected phy with CLOSE CLEAR AFFILIATION).

Any condition causing a link reset sequence other than power on or an SMP PHY CONTROL request specifying a PHY OPERATION of HARD RESET shall not clear an affiliation held between an expander phy connected to a SATA target port and an STP initiator port.

9.3.3 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).

9.3.4 STP transport layer (TT) state machines

The STP transport layer uses the transport layer state machines defined in SATA.

9.4 SMP transport layer

9.4.1 SMP overview

Inside an SMP connection, the source device transmits a single SMP_REQUEST frame and the destination device replies with a single SMP_RESPONSE frame.

Table 101 shows the types of frames.

Table 101 — SMP frame types

Name	Frame type	Originator
SMP_REQUEST	Management request	Initiator port
SMP_RESPONSE	Management response	Target port

Expander devices shall support the SMP_REQUEST and SMP_RESPONSE frames. Other target ports may support these frames.

Figure 100 shows an SMP request/response sequence.

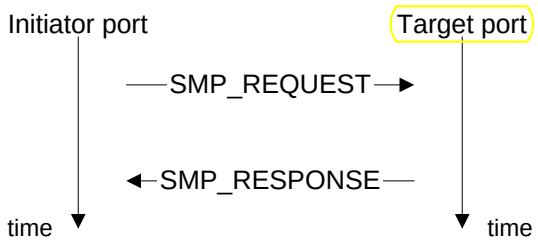


Figure 100 — SMP request/response sequence

9.4.2 SMP_REQUEST frame

Table 102 shows the SMP request frame format.

Table 102 — SMP_REQUEST frame format

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION							
2	Reserved							
3								
4	ADDITIONAL REQUEST BYTES							
n - 4								
n - 3	(MSB)	CRC						
n								(LSB)

The SMP FRAME TYPE field is set to 40h indicating this is an SMP_REQUEST frame.

The FUNCTION field indicates which function is being requested.

The ADDITIONAL REQUEST BYTES field definition and length is based on the function (see 10.3.1). The maximum size of the ADDITIONAL REQUEST BYTES field is 1 023 bytes, making the maximum size of the frame 1 032 bytes (1 024 byte data + 4 byte header + 4 byte CRC).

Fill bytes shall be included so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor-specific.

The CRC field is a CRC value (see 7.4) that is computed over the entire frame, and shall begin on a four-byte boundary.

9.4.3 SMP_RESPONSE frame

The SMP_RESPONSE frame is sent by the target port in response to an SMP_REQUEST frame.

Table 103 defines the SMP_RESPONSE frame format.

Table 103 — SMP_RESPONSE frame format

Bit / Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION							
2	FUNCTION RESULT							
3	Reserved							
4	ADDITIONAL RESPONSE BYTES							
n - 4								
n - 3	(MSB) CRC							
n								(LSB)

The SMP FRAME TYPE field is set to 41h indicating this is an SMP_RESPONSE frame.

The FUNCTION RESULT field is defined in table 104.

Table 104 — Function results

Code	Name	Description
00h	SMP FUNCTION ACCEPTED	The target port supports the SMP function; the ADDITIONAL RESPONSE BYTES contain the requested information.
01h	UNKNOWN SMP FUNCTION	The target port does not support the requested SMP function; the ADDITIONAL RESPONSE BYTES are not present.
02h	SMP FUNCTION FAILED	The target port supports the SMP function, but the requested function failed. The ADDITIONAL RESPONSE BYTES are not present.
All others	Function-specific	The target port supports the SMP function. These codes may indicate success, partial success, or failure. The ADDITIONAL RESPONSE BYTES contain information defined by the function.

The ADDITIONAL RESPONSE BYTES field definition depends on the function requested, and are described in the model section. The maximum size of the ADDITIONAL RESPONSE BYTES field is 1 023 bytes, making the maximum size of the frame 1 032 bytes (1 024 byte data + 4 byte header + 4 byte CRC).

Fill bytes shall be included so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor-specific.

The CRC field is a CRC value (see 7.4) that is computed over the entire frame, and shall begin on a four-byte boundary.

9.4.4 SMP transport layer state machines

9.4.4.1 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) Management Initiator Device (MT_ID state machine); and
b) Management Target Device (MT_TD state machine).

~~Each SAS expander device shall contain an MT_TD state machine. Other device types may contain MT_TD and/or MT_ID state machines.~~

9.4.4.2 Initiator device state machine

9.4.4.2.1 Overview

The MT_ID state machine processes requests from the management application layer. These management requests are communicated to the port layer and the resulting SMP frame or error condition is sent to the management application layer in a return confirmation.

The MT_ID state machine contains the following states:

a) MT_ID1:Idle (see 9.4.4.2.2)(initial state);
b) MT_ID2:Send (see 9.4.4.2.3); and
c) MT_ID3:Receive (see 9.4.4.2.4).

Figure 101 describes the MT_ID state machine.

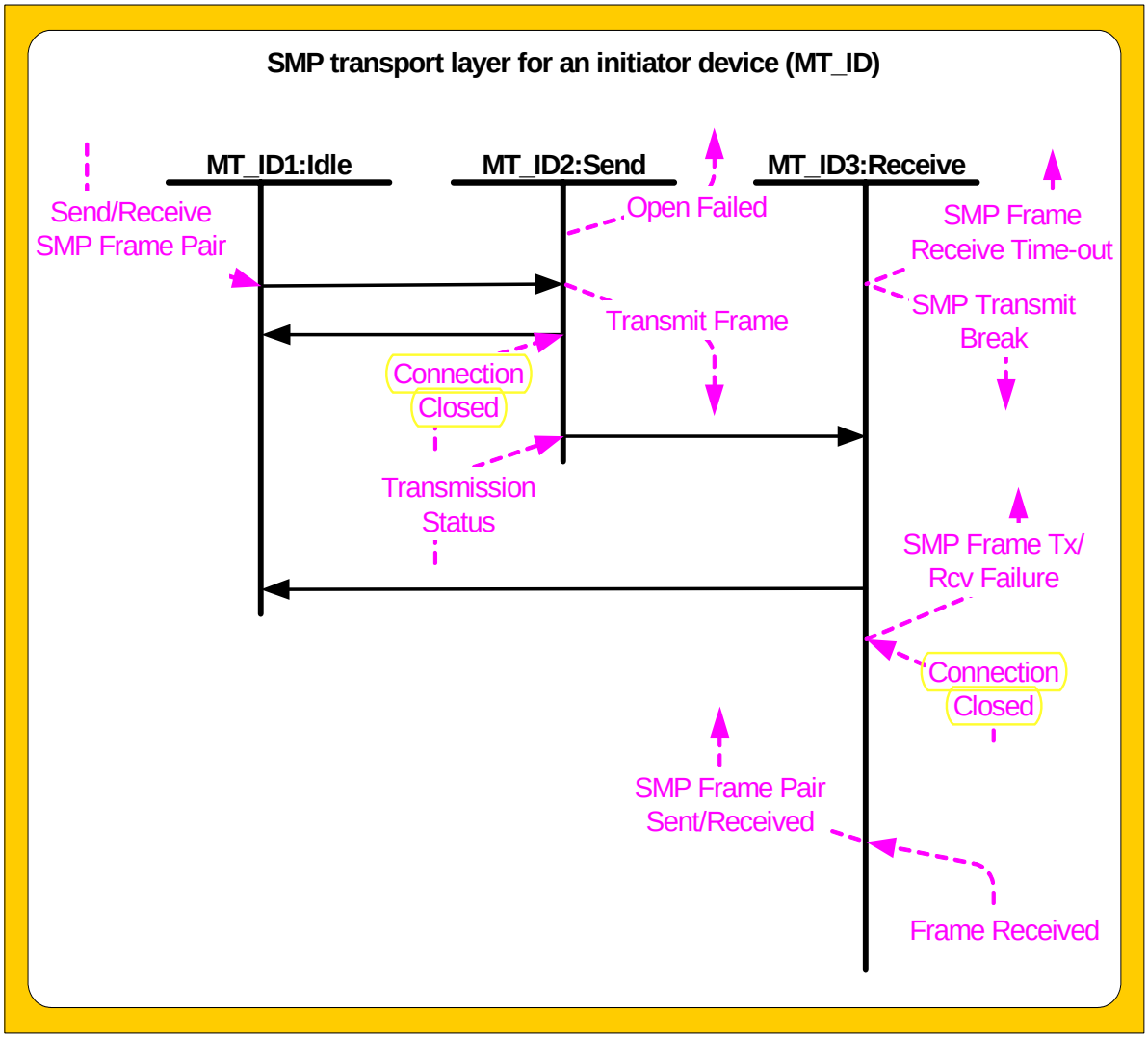


Figure 101 — SMP transport layer state machine - initiator device (MT_ID)

9.4.4.2.2 MT_ID1:Idle state

9.4.4.2.2.1 State description

This state waits for a Send/Receive SMP Frame Pair request ~~from the management application layer.~~ The request to transmit an SMP frame shall consist of values to be used in the CONNECTION RATE, INITIATOR CONNECTION TAG, DESTINATION SAS ADDRESS, and SOURCE SAS ADDRESS fields in the OPEN address frame, and the FUNCTION and ADDITIONAL REQUEST BYTES fields in the SMP_REQUEST frame.

9.4.4.2.2.2 Transition MT_ID1:Idle to MT_ID2:Send

This transition shall occur after a Send/Receive SMP Frame Pair request is received. This transition shall pass the following arguments to the MT_ID2 state:

a) connection rate;
b) initiator connection tag;
c) destination SAS address;
d) source SAS address;
e) function; and

f) additional request bytes.

9.4.4.2.3 MT_ID2:Send state

9.4.4.2.3.1 State description

This state constructs an SMP_REQUEST frame using the function and additional request bytes arguments ~~received in the MT_ID1:Idle to MT_ID2:Send transition,~~ and sends a Transmit Frame (SMP) request to the port layer. In addition to the SMP_REQUEST frame, the Transmit Frame (SMP) request shall contain the values to be used for the CONNECTION RATE, INITIATOR CONNECTION TAG, DESTINATION SAS ADDRESS, and SOURCE SAS ADDRESS fields in the OPEN address frame.

9.4.4.2.3.2 Transition MT_ID2:Send to MT_ID1:Idle

This transition shall occur after receiving either a Connection Closed confirmation or a Transmission Status confirmation other than a Transmission Status (SMP Frame Transmitted) confirmation, and sending an Open Failed confirmation to the management application layer.

9.4.4.2.3.3 Transition MT_ID2:Send to MT_ID3:Receive

This transition shall occur after a Transmission Status (SMP Frame Transmitted) confirmation is received.

9.4.4.2.4 MT_ID3:Receive state

9.4.4.2.4.1 State description

This state waits for a confirmation ~~from the port layer~~ that either an SMP frame has been received or a failure occurred.

This state shall initialize a SMP frame receive timeout timer to a vendor-specific time and start the timer upon entry into this state.

9.4.4.2.4.2 Transition MT_ID3:Receive to MT_ID1:Idle

This transition shall occur after one of the following:

a) an Frame Received (SMP) confirmation is received, and, as a result, this state has sent an SMP Frame Pair Sent/Received confirmation to the management application layer;
b) a Connection Closed or Frame Received (SMP Failure) confirmation is received, and, as a result, this state has sent an SMP Frame Tx/Rcv Failure confirmation to the management application layer; or
c) the SMP frame receive timeout timer is exceeded before a SMP Frame Pair Sent/Received confirmation is received, and, as a result, this state has sent an SMP Frame Receive Timeout confirmation to the management application layer and has sent an SMP Transmit Break request to the port layer.

9.4.4.3 Expander device and target device state machine

9.4.4.3.1 Overview

The MT_TD state machine informs the device server of the receipt of an SMP frame. Confirmation of the receipt of an SMP frame is communicated from the port layer and that confirmation is sent to the management application layer. The management application layer creates the corresponding SMP_RESPONSE frame and this state forwards it to the port layer.

The MT_TD state machine contains the following states:

a) MT_TD1:Idle (see 9.4.4.3.2)(initial state); and
b) MT_TD2:Send (see 9.4.4.3.3).

Figure 102 describes the MT_TD state machine.

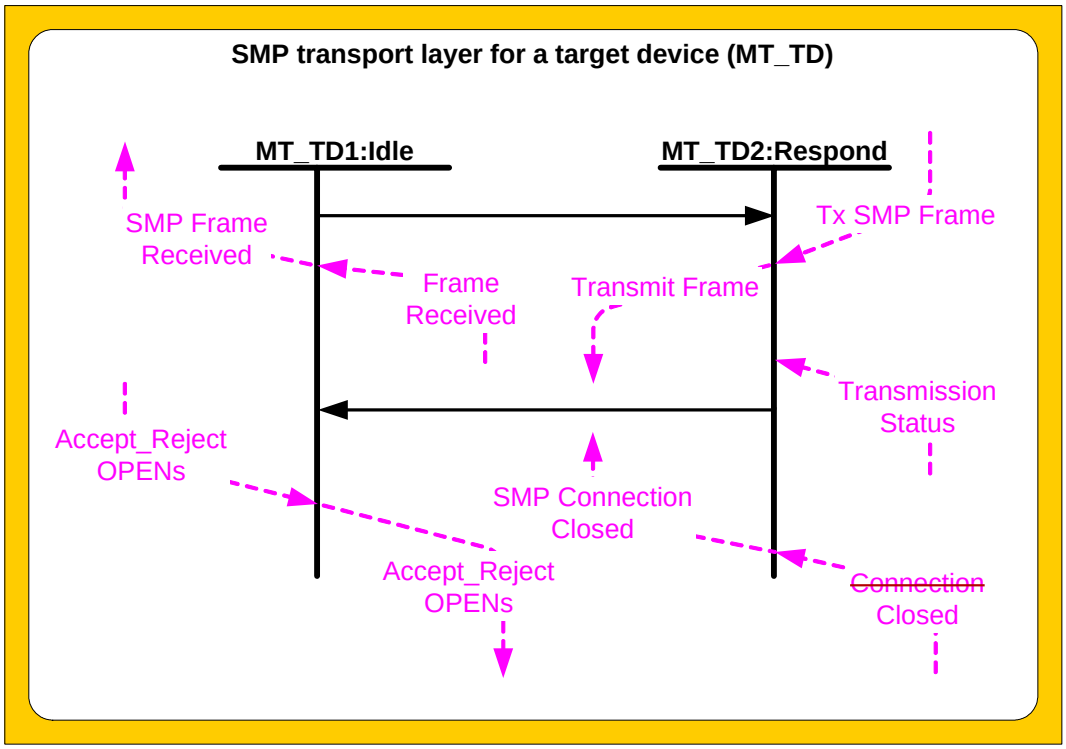


Figure 102 — SMP transport layer (MT) state machines - target device

9.4.4.3.2 MT_TD1:Idle state

9.4.4.3.2.1 State description

This state waits for a Frame Received (SMP) confirmation ~~from the port 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.

9.4.4.3.2.2 Transition MT_TD1:Idle to MT_TD2:Respond

This transition shall occur after an Frame Received (SMP) confirmation is received, and, as a result, this state has sent an SMP Frame Received confirmation to the management application layer.

9.4.4.3.3 MT_TD2:Respond state

9.4.4.3.3.1 State description

This state waits for a Tx SMP Frame request ~~from the management application layer.~~ Upon receipt, this state shall send a Transmit Frame (SMP) request to the port layer.

9.4.4.3.3.2 Transition MT_TD2:Respond to MT_TD1:Idle

This transition shall occur after one of the following:

a) a Transmission Status (SMP Frame Transmitted) confirmation is received; or
b) a Connection Closed confirmation is received, and, as a result, this state has sent an SMP Connection Closed confirmation to the management application layer.

10 Application layer

10.1 SCSI application layer

10.1.1 SCSI transport protocol services

10.1.1.1 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-3):

Service response = Execute Command (IN (I_T_L_x Nexus, CDB, [Task Attribute], [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Autosense Request]), OUT ([Data-In Buffer], [Sense Data], 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-3):

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 = LOGICAL UNIT RESET (IN (Nexus)); and
f) Service Response = QUERY TASK (IN (Nexus)).

SSP defines the transport protocol services required by SAM-3 in support of the these remote procedure calls.

Table 105 shows the mapping of the remote procedure calls to transport protocol services and how each transport protocol service is implemented in SSP.

Table 105 — 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 COMMAND frame
	Data Transfer [b]	Request	Send Data-In	T	DATA frames
		Confirmation	Data-In Delivered	T	Receipt of ACKs for the DATA frames
		Request	Receive Data-Out	T	XFER_RDY frame
		Confirmation	Data-Out Received	T	Receipt of DATA frames
ABORT TASK, ABORT TASK SET, CLEAR ACA, CLEAR TASK SET, LOGICAL UNIT RESET, and QUERY TASK	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 initiator device (I) or the target device (T) implements the transport protocol service.
[b] Initiator device Data Transfer transport protocol services are not specified by SAM-3.

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.

10.1.1.2 Send SCSI Command transport protocol service

An application client uses the Send SCSI Command transport protocol service request to have an initiator port transmit a COMMAND frame.

Send SCSI Command (IN (I_T_L_x Nexus, CDB, [Task Attribute], [Data-In Buffer Size], [Data-Out

Buffer], [Data-Out Buffer Size], [Autosense Request], [Command Reference Number]))

Table 106 shows how the arguments to the Send SCSI Command transport protocol service are used.

Table 106 — Send SCSI Command transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the frame header; and c) Q used to set the TAG field in the frame header.
CDB	Used to set the CDB field in the COMMAND frame.
[Task Attribute]	Used to set the TASK ATTRIBUTE field in the COMMAND frame.
[Data-In Buffer Size]	Maximum of 2^{32}
[Data-Out Buffer]	Internal to initiator port
[Data-Out Buffer Size]	Maximum of 2^{32}
[Autosense Request]	True

10.1.1.3 SCSI Command Received transport protocol service

A 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_x Nexus, CDB, [Task Attribute], [Autosense Request], [Command Reference Number]))

Table 107 shows how the arguments to the SCSI Command Received transport protocol service are determined.

Table 107 — SCSI Command Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the COMMAND frame header; and c) Q indicated by the TAG field in the COMMAND frame header.
CDB	From the CDB field in the COMMAND frame.
[Task Attribute]	From the TASK ATTRIBUTE field in the COMMAND frame.
[Autosense Request]	True
[Command Reference Number]	Ignored

~~If a target port calls SCSI Command Received () with a TAG already in use (i.e., an overlapped command), the device server responses are defined in SAM-3.~~

10.1.1.4 Send Command Complete transport protocol service

A device server uses the Send Command Complete transport protocol service response to have a target port transmit a RESPONSE frame.

Send Command Complete (IN (I_T_L_x Nexus, [Sense Data], Status, Service Response))

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 responsed to with Data-Out Received () and all its outstanding Send Data-In () calls have been responded to with Data-In Delivered ().

Table 108 shows how the arguments to the Send Command Complete transport protocol service are used.

Table 108 — Send Command Complete transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the frame header; and c) Q used to set the TAG field in the frame header.
[Sense Data]	Used to set the RESPONSE frame SENSE DATA field.
Status	Used to set the RESPONSE frame STATUS field.
Service Response	Used to set the RSPVALID bit and STATUS field in the RESPONSE frame: a) TASK COMPLETE: The RSPVALID bit is set to zero and the STATUS field is set to a value other than INTERMEDIATE or INTERME-DIATE-CONDITION MET; b) LINKED COMMAND COMPLETE: The RSPVALID bit is set to zero and the STATUS field is set to INTERMEDIATE or INTERMEDIATE-CONDITION MET; or c) SERVICE DELIVERY OR TARGET FAILURE: The RSPVALID bit is set to one.

10.1.1.5 Command Complete Received transport protocol service

An initiator port uses the Command Complete Received transport protocol service confirmation not notify an application client that it has received a response for its COMMAND frame (e.g., a RESPONSE frame or a NAK).

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 109 shows how the arguments to the Command Complete Received transport protocol service are determined.

Table 109 — Command Complete Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the RESPONSE frame or COMMAND frame header; and c) Q indicated by the TAG field in the RESPONSE frame or COMMAND frame header.
[Data-In Buffer]	Internal to the initiator port.
[Sense Data]	From the RESPONSE frame SENSE DATA field.
Status	From the RESPONSE frame STATUS field.
Service Response	From the RESPONSE frame RSPVALID bit and STATUS field, or from a NAK on the COMMAND frame: a) TASK COMPLETE: The RESPONSE frame contains an RSPVALID bit set to zero and a STATUS field set to a value other than INTERMEDIATE or INTERMEDIATE-CONDITION MET; b) LINKED COMMAND COMPLETE: The RESPONSE frame contains an RSPVALID bit set to zero and a STATUS field set to INTERMEDIATE or INTERMEDIATE-CONDITION MET; or c) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains an RSPVALID bit set to one, or the COMMAND frame was NAKed.

10.1.1.6 Send Data-In transport protocol service

A device server uses the Send Data-In transport protocol service request to have a target port transmit a DATA frame.

Send Data-In (IN (I_T_L_x 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 after it has called Send Command Complete () for that I_T_L_Q (e.g., a RESPONSE frame with for that I_T_L_Q) or called Task Management Function Executed for a task management function that terminates that task. (e.g., an ABORT TASK).

Table 110 shows how the arguments to the Send Data-In transport protocol service are used.

Table 110 — Send Data-In transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the DATA frame header; and c) Q used to set the TAG field in the DATA frame header.
Device Server Buffer	Internal to the device server.
Application Client Buffer Offset	Used to set the DATA frame RELATIVE OFFSET field.
Request Byte Count	Used to select the size of the DATA frame.

10.1.1.7 Data-In Delivered transport protocol service

A target port uses the Data-In Delivered transport protocol service indication to notify a device server of the results of transmitting a DATA frame.

Data-In Delivered (IN (I_T_L_x Nexus))

Table 111 shows how the arguments to the Data-In Delivered transport protocol service are determined.

Table 111 — Data-In Delivered transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the DATA frame header; and c) Q indicated by the TAG field in the DATA frame header.
Delivery Result	From the response to the outgoing DATA frame: a) DELIVERY SUCCESSFUL: The DATA frame received an ACK; or b) DELIVERY FAILURE: The DATA frame received a NAK or no response.

10.1.1.8 Receive Data-Out transport protocol service

A device server uses the Receive Data-Out transport protocol service request to have a target port transmit an XFER_RDY frame.

Receive Data-Out (IN (I_T_L_x 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 until Data Out Received () has returned for the previous Receive Data Out () call (i.e., no XFER_RDY 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).

A device server shall not call Receive Data Out () for a given I_T_L_Q after a Send Command Complete () has been called for that I_T_L_Q 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 112 shows how the arguments to the Receive Data-Out transport protocol service are used.

Table 112 — Receive Data-Out transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the XFER_RDY frame header; and c) Q usd to set the TAG field in the XFER_RDY frame header.
Application Client Buffer Offset	Used to set the XFER_RDY frame RELATIVE OFFSET field.
Request Byte Count	Used to set the XFER_RDY frame WRITE DATA LENGTH field.
Device Server Buffer	Internal to the device server.

10.1.1.9 Data-Out Received transport protocol service

A 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 DATA frames in response).

Data-Out Received (IN (I_T_L_x Nexus))

Table 113 shows how the arguments to the Data-Out Received transport protocol service are determined.

Table 113 — Data-Out Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the frame header; and c) Q indicated by the TAG field in the frame header.
Delivery Result	From the response to the XFER_RDY: a) DELIVERY SUCCESSFUL: The XFER_RDY was successfully transmitted and all the DATA frames for the requested write data were received; or b) DELIVERY FAILURE: The XFER_RDY received a NAK or no response.

10.1.1.10 Send Task Management Request transport protocol service

An application client uses the Send Task Management Request transport protocol service request to have an initiator port transmit a TASK frame.

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 114 shows how the arguments to the Send Task Management Request transport protocol service are used.

Table 114 — Send Task Management Request transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the TASK frame header; and c) Q used to set the TAG field in the TASK frame header.
Function Identifier	Used to set the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK; b) ABORT TASK SET; c) CLEAR ACA; d) CLEAR TASK SET; e) LOGICAL UNIT RESET; and f) QUERY TASK.

10.1.1.11 Task Management Request Received transport protocol service

A target port uses the Task Management Request Received transport protocol service indication to notify a device server that it has received a TASK frame.

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 115 shows how the arguments to the Task Management Request Received transport protocol service are determined.

Table 115 — Task Management Request Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus, where: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the TASK frame header; and c) Q indicated by the TAG field in the TASK frame header.
Function Identifier	From the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK; b) ABORT TASK SET; c) CLEAR ACA; d) CLEAR TASK SET; e) LOGICAL UNIT RESET; and f) QUERY TASK.

~~If a target port calls Task Management Request Received () with a TAG already in use, the device server responses are defined in SAM-3.~~

10.1.1.12 Task Management Function Executed transport protocol service

A device server uses the Task Management Function Executed transport protocol service response to have a target port transmit a RESPONSE frame.

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

A device server shall only call Task Management Function Executed () after receiving Task Management Request Received ().

Table 116 shows how the arguments to the Task Management Function Executed transport protocol service are used.

Table 116 — Task Management Function Executed transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T used to select an open connection; b) L used to set the LOGICAL UNIT NUMBER field in the RESPONSE frame header; and c) Q used to set the TAG field in the RESPONSE frame header.
Service Response	Used to set the SNSVALID bit, RSPVALID bit, and RESPONSE CODE field in the RESPONSE frame: a) FUNCTION REJECTED: The SNSVALID bit is set to zero, the RSPVALID bit is set to one, and the RESPONSE CODE field is set to: A) TASK FRAME FIELDS INVALID; B) TASK MANAGEMENT FUNCTION NOT SUPPORTED; or C) TASK MANAGEMENT FUNCTION FAILED; b) FUNCTION COMPLETE: The RESPONSE frame SNSVALID bit is set to zero, the RSPVALID bit is set to one, and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION COMPLETE; c) FUNCTION SUCCEEDED: The RESPONSE frame SNSVALID bit is set to zero, the RSPVALID bit is set to one, and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION SUCCEEDED; or d) SERVICE DELIVERY OR SUBSYSTEM FAILURE: The RESPONSE frame SNSVALID bit is set to one.

10.1.1.13 Received Task Management Function-Executed transport protocol service

An 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).

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 117 shows how the arguments to the Received Task Management Function-Executed transport protocol service are determined.

Table 117 — Received Task Management Function-Executed transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_x nexus	I_T_L_Q nexus: a) I_T indicated by the open connection; b) L indicated by the LOGICAL UNIT NUMBER field in the RESPONSE frame or TASK frame header; and c) Q indicated by the TAG field in the RESPONSE frame or TASK frame header.
Service Response	From the response to the TASK frame: a) FUNCTION REJECTED: The RESPONSE frame contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RESPONSE CODE field set to: A) TASK FRAME FIELDS INVALID; B) TASK MANAGEMENT FUNCTION NOT SUPPORTED; or C) TASK MANAGEMENT FUNCTION FAILED; b) FUNCTION COMPLETE: The RESPONSE frame contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RESPONSE CODE field set to TASK MANAGEMENT FUNCTION COMPLETE; c) FUNCTION SUCCEEDED: The RESPONSE frame contains a SNSVALID bit set to zero, an RSPVALID bit set to one, and an RESPONSE CODE field set to TASK MANAGEMENT FUNCTION SUCCEEDED; or d) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains a SNSVALID bit set to one, or the TASK frame was NAKed.

10.1.2 Device server error handling

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - ILLEGAL TARGET PORT TRANSFER TAG, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of ILLEGAL TARGET PORT TRANSFER TAG RECEIVED.

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - RELATIVE OFFSET ERROR, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of RELATIVE OFFSET ERROR.

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - TOO MUCH WRITE DATA, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of TOO MUCH WRITE DATA.

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - INFORMATION UNIT TOO SHORT, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of INFORMATION UNIT TOO SHORT.

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - ACK/NAK TIMEOUT, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of ACK/NAK TIMEOUT.

If a device server calls Receive Data-Out () and receives a Delivery Result set to DELIVERY FAILURE - NAK RECEIVED, it shall terminate the command with a CHECK CONDITION status with a sense key of ABORTED COMMAND and an additional sense code of NAK RECEIVED.

10.1.3 Application client error handling

If an initiator port calls Command Complete Received () and delivers a Service Response of:

a) Service Delivery of Target Failure - XFER_RDY Information Unit Too Short;
b) Service Delivery of Target Failure - XFER_RDY Information Unit Too Long;
c) Service Delivery of Target Failure - XFER_RDY Incorrect Write Data Length;
d) Service Delivery of Target Failure - XFER_RDY Relative Offset Error;
e) Service Delivery of Target Failure - XFER_RDY Not Expected;
f) Service Delivery of Target Failure - DATA Incorrect Data Length;
g) Service Delivery of Target Failure - DATA Too Much Read Data;
h) Service Delivery of Target Failure - DATA Relative Offset Error;
i) Service Delivery of Target Failure - NAK Received; or
j) Service Delivery of Target Failure - Connection Failed.

it shall send an ABORT TASK task management function to abort the command.

If an application client calls Send SCSI Command () and an initiator port calls Command Complete Received () and delivers a Service Response of Service Delivery of 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 it returns a Service Response of FUNCTION COMPLETE, and Command Complete Received () has not yet been called a second time for the command in question, the application client shall assume the command was not delivered successfully and may reuse the tag.

If an application client calls Send Task Management Request () and an initiator port calls Received Task Management Function Executed () and delivers a Service Response of Service Delivery of Target Failure - ACK/NAK Timeout, the application client shall call Send Task Management Request () again with identical arguments.

If an application client calls Send SCSI Command () and an initiator port calls Command Complete Received () a second time for the command, and the second call includes a Retransmit argument set to one, the application client shall ignore the second call.

If an application client calls Send Task Management Request () and an initiator port calls Received Task Management Function Executed () a second time for the task management function, and the second call includes a Retransmit argument set to one, the application client shall ignore the second call.

After a Command Complete Received () or Received Task Management Function Executed () call, an application client shall not reuse the tag until ~~it determines the ACK for the RESPONSE frame was seen by the target port. This is indicated by:~~

a) receiving another frame in the same connection;
b) receiving a DONE (NORMAL) Received or DONE (CREDIT TIMEOUT) Received confirmation; or
c) receiving a DONE (ACK/NAK TIMEOUT) Received confirmation and running a QUERY TASK task management function to confirm that the tag is no longer active in the logical unit.

10.1.4 SCSI transport protocol events

Table 118 shows the SCSI transport protocol events supported by this standard.

Table 118 — SCSI transport protocol events

Event	SAS SSP implementation
Transport Reset	Receipt of a hard reset sequence (see 4.4.2).
Nexus Lost	Recept of specific OPEN_REJECTs for a specific time period (see 4.5)

10.1.5 SCSI commands

10.1.5.1 INQUIRY command

The vital product data returned by the INQUIRY command (see SPC-3) is modified as described in 10.1.9.

10.1.5.2 LOG SELECT and LOG SENSE commands

SAS-specific log pages accessed with the LOG SELECT and LOG SENSE commands (see SPC-3) are described in 10.1.7.

10.1.5.3 MODE SELECT and MODE SENSE commands

SAS-specific mode pages accessed with the MODE SELECT and MODE SENSE commands (see SPC-3) are described in 10.1.6.

10.1.5.4 START STOP UNIT command

The power condition states controlled by the START STOP UNIT command (see SBC-2 ~~and RBC)~~ are modified as described in 10.1.8.

10.1.6 SCSI mode parameters

10.1.6.1 Disconnect-Reconnect mode page

10.1.6.1.1 Disconnect-Reconnect mode page overview

The Disconnect-Reconnect mode page (see SPC-3) provides the application client the means to tune the performance of the service delivery subsystem. Table 119 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 (e.g., as if the mode page is implemented and the field is set to zero).

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.

SSP 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 return CHECK CONDITION status for that MODE SELECT command. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to ILLEGAL FIELD IN PARAMETER LIST.

Table 119 — Disconnect-Reconnect mode page for SSP

Bit Byte	7	6	5	4	3	2	1	0
0	PS	Reserved	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-3.

The PAGE CODE (PS) field is 02h and the PAGE LENGTH field is 0Eh.

10.1.6.1.2 BUS INACTIVITY TIME LIMIT field

The value in the BUS INACTIVITY TIME LIMIT field specifies the maximum period that a target port is permitted to maintain a connection without transferring a frame to the initiator port. This value shall be the number of 100 µs increments between frames that the target port transmits during a connection. When this number is exceeded, the 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-3. A value of zero in this field shall specify that there is no bus inactivity time limit.

10.1.6.1.3 MAXIMUM CONNECT TIME LIMIT field

The value in the MAXIMUM CONNECT TIME LIMIT field specifies the maximum duration of a connection. This value shall be the number of 100 µs increments that a target port transmits during a connection after which the target port shall prepare to close the connection (i.e., a value of one in this field specifies that the time shall be less than or equal to 100 µs, a value of two in this field specifies that the time shall be less than or equal to 200 µs, etc.). If a target port is transferring a frame when the maximum connection time limit is exceeded, the target port shall complete transfer of the frame before preparing to close the connection. A value of zero in this field shall specify that there is no maximum connection time limit.

10.1.6.1.4 MAXIMUM BURST SIZE field

For read data, the value in the MAXIMUM BURST SIZE field shall specify the maximum amount of data that is transferred during a connection by a target port per I_T_L_Q nexus without transferring at least one frame for a different I_T_L_Q nexus. If the 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;

the target port shall prepare to close the connection after the amount of data specified by the MAXIMUM BURST SIZE field is transferred to the initiator port.

For write data, the value shall specify the maximum amount of data that a target port requests via a single XFER_RDY frame.

This value shall be specified in 512-byte increments (i.e., a value of one in this field specifies that the number of bytes transferred to the initiator port for the nexus shall be less than or equal to 512, a value of two in this field specifies that the number of bytes transferred to the initiator port for the nexus shall be less than or equal to 1 024, etc.). The device server may round this value down as defined in SPC-3. A value of zero in this field shall specify that there is no maximum burst size.

10.1.6.1.5 FIRST BURST SIZE field

The value in the FIRST BURST SIZE field specifies the maximum amount of write data that may be sent by the initiator port to the target port without having to receive an XFER_RDY frame from the target port. 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 transfer length is specified in the WRITE DATA LENGTH 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).

This value shall specify 512-byte increments of data (i.e., a value of one in this field specifies that the number of bytes transferred by the initiator port shall be less than or equal to 512, a value of two in this field specifies that the number of bytes transferred by the initiator port shall be less than or equal to 1 024, etc.).

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 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 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 initiator. All data for the command is not required to be transferred during this connection.

A value of zero in this field shall specify that there is no first burst size, i.e., an initiator port shall transmit no data frames to the target port before receiving an XFER_RDY frame.

10.1.6.2 Protocol-Specific Port mode page

10.1.6.2.1 Overview

The Protocol-Specific Port mode page (see SPC-3) contains parameters that affect SAS SSP target port operation. If the mode page is implemented, all logical units in SAS SSP target devices supporting the MODE

SELECT or MODE SENSE commands shall implement the page, and there shall be one copy of the mode page shared by all initiator ports.

If a target device has multiple target ports, changes in the short page parameters for one target port should not affect other target ports.

Table 121 shows the subpages of this mode page.

Table 120 — Protocol-Specific Port Control mode page subpages

Subpage	Description	Reference
Short page	Short format	10.1.6.2.2
Long page 00h	Not allowed	
Long page 01h	Phy Control And Discover subpage	10.1.6.2.3
Long page E0h - FEh	Vendor-specific	
Long page FFh	Return all subpages for the Port Control mode page	SPC-3
All others	Reserved	

10.1.6.2.2 Protocol-Specific Port mode page - short format

Parameters in this page shall affect all phys in the target port, and may affect all SAS SSP target ports in the target device.

Table 121 shows the format of the page for SAS SSP.

Table 121 — Protocol-Specific Port Control mode page for SAS SSP - short format

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (06h)							
2	Reserved				PROTOCOL IDENTIFIER (6h)			
3	Reserved							
4	(MSB)	I_T NEXUS LOSS TIME						
5								(LSB)
6	Reserved							
7								

The SPF field shall be set to zero for access to the short format mode page. 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 indicates how long in milliseconds the target port shall retry connection requests to an initiator port that are rejected with OPEN_REJECT (NO DESTINATION), OPEN_REJECT

(CONNECTION RATE NOT SUPPORTED), or connection timeouts before treating it as an I_T nexus loss (see 4.5). An I_T NEXUS LOSS TIME of zero indicates the target port shall never consider rejections an I_T nexus loss. If the mode page is implemented, the default setting shall be 2 000 ms. If the mode page is not implemented, the logical unit shall not implement an I_T nexus loss timer.

10.1.6.2.3 Protocol-Specific Port mode page - Phy Control And Discover subpage

The Phy Control And Discover subpage contains phy-specific parameters. Parameters in this page shall affect only the referenced phy.

Table 122 shows the format of the subpage for SAS SSP.

Table 122 — Protocol-Specific Port Control mode page for SAS SSP - Phy Control And Discover subpage

Bit Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h)							
4	(MSB)	PAGE LENGTH (n - 3)						
5								(LSB)
6	Reserved							
7	NUMBER OF PHYS							
SAS phy mode descriptors								
8	First SAS phy mode descriptor							
...	...							
n	Last SAS phy mode descriptor							

The SPF field is set to one to access the long format mode pages.

The NUMBER OF PHYS field contains the number of phys in the target device ~~and indicates the number of SAS phy mode descriptors that follow.~~ This field shall not be changeable.

A SAS phy mode descriptor shall be included for each phy in the target device, starting with the lowest numbered phy and ending with the highest numbered phy.

Table 123 shows the SAS phy mode descriptor.

Table 123 — SAS phy mode descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	PHY OPERATION							
3	Reserved							
4	Reserved	ATTACHED DEVICE TYPE			Reserved			
5	Reserved				NEGOTIATED PHYSICAL LINK RATE			
6	Reserved				ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	Reserved
7	Reserved				ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	Reserved
8	ATTACHED SAS ADDRESS							
15								
16	SAS ADDRESS							
23								
24	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
25	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
26	Vendor-specific							
27								
28	Reserved							
31								

The PHY IDENTIFIER field, ATTACHED DEVICE TYPE field, NEGOTIATED PHYSICAL LINK RATE field, ATTACHED SSP INITIATOR bit, ATTACHED STP INITIATOR bit, ATTACHED SMP INITIATOR bit, ATTACHED SSP TARGET bit, ATTACHED STP TARGET bit, ATTACHED SMP TARGET bit, ATTACHED SAS ADDRESS field, SAS ADDRESS field, HARDWARE MINIMUM PHYSICAL LINK RATE field, and HARDWARE MAXIMUM PHYSICAL LINK RATE field are defined in the SMP DISCOVER function (see 10.3.1.4). These fields shall not be changeable.

The PHY OPERATION field, PROGRAMMED MINIMUM PHYSICAL LINK RATE field, and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field are defined in the SMP PHY CONTROL function (see 10.3.1.9).

10.1.6.3 Protocol-Specific Logical Unit mode page

SSP logical units shall not include the Protocol-Specific Logical Unit mode page (see SPC-3).

10.1.7 SCSI log parameters

10.1.7.1 Protocol-Specific log page for SAS

The Protocol Specific log page for SAS defined in table 124 is used to report errors that have occurred on the target device's phy(s).

Table 124 — Protocol-Specific log page for SAS

Bit Byte	7	6	5	4	3	2	1	0
0	PAGE CODE (18h)							
1	Reserved							
2	(MSB)			PAGE LENGTH (m - 3)				
3								(LSB)
Protocol-specific log parameters								
4				First protocol-specific log parameter				
...	...							
				n^{th} protocol-specific log parameter				
m								

The PAGE CODE field is set to 18h.

The PAGE LENGTH field is set to the length of the log page minus three.

Table 125 defines the format for a SAS log parameter.

Table 125 — Protocol-Specific log parameter format for SAS

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1	PARAMETER CODE (relative target port identifier)							(LSB)
2	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (y - 3)							
4	Reserved				PROTOCOL IDENTIFIER (6h)			
5								
6	Reserved							
7	NUMBER OF PHYS							
	SAS phy log descriptors							
8								
	First SAS phy log descriptor							
	...							
	Last SAS phy log descriptor							
y								

The PARAMETER CODE field contains the relative target port identifier (see SPC-3) of the target port that this log parameter describes.

Table 126 defines the values for the log parameter control bits for this log parameter.

Table 126 — Parameter control bits for SAS log parameters

Bit	Value	Description
DU	0	The value is provided by the device server.
DS	0	The device server supports saving of the parameter.
TSD	0	The device server manages saving of the parameter.
ETC	0	No threshold comparison is made on this value.
TMC	any	This field is ignored when ETC is 0.
LBIN	1	The parameter is in binary format.
LP	1	The parameter is a list parameter.

The PARAMETER LENGTH field is set to the length of the log parameter minus three.

The PROTOCOL IDENTIFIER field is set to 6h.

The NUMBER OF PHYS field contains the number of SAS phy log descriptors that follow.

Table 127 defines the SAS phy log descriptor. Each SAS phy log descriptor is the same length.

Table 127 — SAS phy log descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	Reserved							
3								
4	Reserved	ATTACHED DEVICE TYPE			Reserved			
5	Reserved				NEGOTIATED PHYSICAL LINK RATE			
6	Reserved				ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	Reserved
7	Reserved				ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	Reserved
8	ATTACHED SAS ADDRESS							
15								
16	SAS ADDRESS							
23								
24	(MSB)	INVALID DWORD COUNT						
27								(LSB)
28	(MSB)	DISPARITY ERROR COUNT						
31								(LSB)
32	(MSB)	LOSS OF DWORD SYNCHRONIZATION						
35								(LSB)
36	(MSB)	PHY RESET PROBLEM						
39								(LSB)

The PHY IDENTIFIER field, ATTACHED DEVICE TYPE field, NEGOTIATED PHYSICAL LINK RATE field, ATTACHED SSP INITIATOR bit, ATTACHED STP INITIATOR bit, ATTACHED SMP INITIATOR bit, ATTACHED SSP TARGET bit, ATTACHED STP

TARGET bit, ATTACHED SMP TARGET bit, ATTACHED SAS ADDRESS field, and SAS ADDRESS field are defined in the SMP DISCOVER function (see 10.3.1.4).

The INVALID DWORD COUNT field, 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 data (see 10.3.1.5).

10.1.8 SCSI power condition states

The logical unit power condition states from the Power Condition mode page (see SPC-3) and START STOP UNIT command (see SBC-2 ~~and RBC),~~ if implemented, shall interact with the NOTIFY (ENABLE_SPINUP) primitive (see 7.1.4.9) to control temporary consumption of additional power (e.g., spin-up of rotating media) as described in this subclause.

a) after power on, if the target device has not received a START STOP UNIT command with the START bit set to zero, transition to the active power condition state after receiving NOTIFY (ENABLE_SPINUP). The target device automatically transitions after power on without waiting for the application client; and
b) after power on, if the target device has previously received a START STOP UNIT command with the START bit set to zero when it receives a START STOP UNIT command with the START bit set to one, spin-up after receiving the next NOTIFY (ENABLE_SPINUP). The application client's request is effectively delayed until NOTIFY (ENABLE_SPINUP) arrives.

The SCSI application layer Power Condition (SA_PC) state machine describes how the target device processes logical unit power condition state change requests and NOTIFY (ENABLE_SPINUP). ~~The SA_PC state machine is an enhanced version of the logical unit power condition state machines described in SPC-3, SBC-2, and RBC.~~ The SA_PC states are as follows:

a) SA_PC_0:Powered_On;
b) SA_PC_1:Active;
c) SA_PC_2:Idle;
d) SA_PC_3:Standby;
e) SA_PC_4:Stopped (specific to SBC-2 ~~and RBC~~ devices);
f) SA_PC_5:Active_Wait (specific to SAS devices); and
g) SA_PC_6:Idle_Wait (specific to SAS devices);

The SA_PC state machine stall start in the SA_PC_0:Powered_On state after power on.

Figure 103 describes the SA_PC state machine.

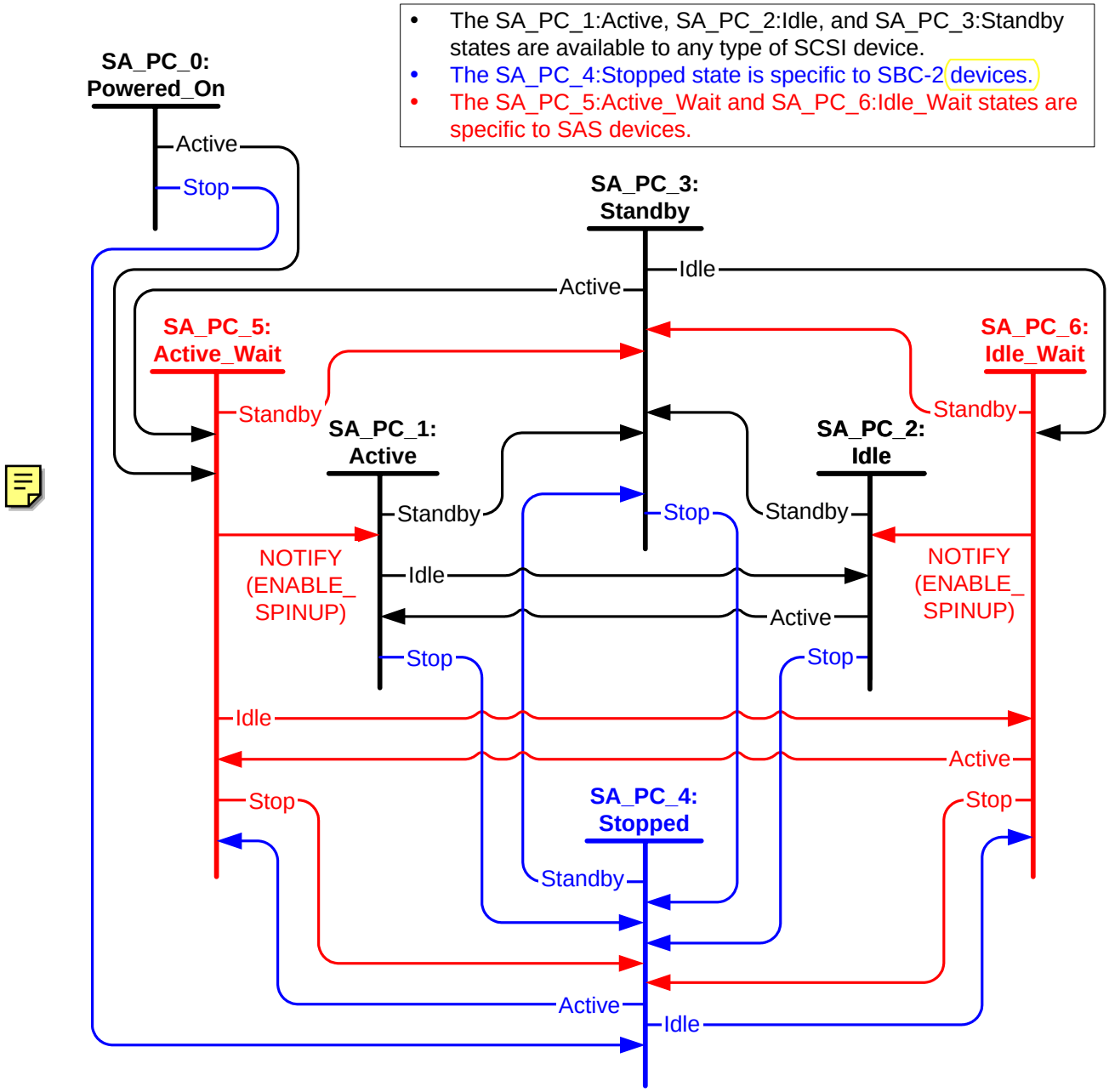


Figure 103 — SCSI application layer power condition (SA_PC) state machine for SAS

10.1.8.1 SA_PC_0:Powered_On state

10.1.8.1.1 State description

This state shall be entered upon power on. This state consumes zero time.

10.1.8.1.2 Transition SA_PC_0:Powered_On to SA_PC_4:Stopped

This transition shall occur if the device has been configured to start in the SA_PC_4:Stopped state.

10.1.8.1.3 Transition SA_PC_0:Powered_On to SA_PC_5:Active_Wait

This transition shall occur if the device has been configured to start in the SA_PC_5:Active state.

10.1.8.2 SA_PC_1:Active state

10.1.8.2.1 State description

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-3 for more details about this state.

10.1.8.2.2 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE IDLE is received; or
c) the Power Condition mode page idle timer expires.

10.1.8.2.3 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE STANDBY is received; or
c) the Power Condition mode page standby timer expires.

10.1.8.2.4 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 received.

10.1.8.3 SA_PC_2:Idle state

10.1.8.3.1 State description

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-3 for more details about this state.

10.1.8.3.2 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is received; or
c) a command is received which requires the active power condition.

10.1.8.3.3 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE STANDBY is received; or
c) the Power Condition mode page standby timer expires.

10.1.8.3.4 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 received.

10.1.8.4 SA_PC_3:Standby state

10.1.8.4.1 State description

While in this state, rotating media in block devices shall be stopped.

See SPC-3 for more details about this state.

10.1.8.4.2 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 received.

10.1.8.4.3 Transition SA_PC_3:Standby to SA_PC_5:Active_Wait

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is received; or
b) a command is received which requires the active power condition.

10.1.8.4.4 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE IDLE is received; or
c) a command is received which requires the idle power condition.

10.1.8.5 SA_PC_4:Stopped state

10.1.8.5.1 State description

This state is only implemented in block devices.

While in this state, rotating media shall be stopped.

See SBC-2 for more details about this state.

10.1.8.5.2 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 received.

10.1.8.5.3 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 received; or
b) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is received.

10.1.8.5.4 Transition SA_PC_4:Stopped to SA_PC_6:Idle_Wait

This transition shall occur if:

a START STOP UNIT command with the POWER CONDITION field set to IDLE is received.

10.1.8.6 SA_PC_5:Active_Wait state

10.1.8.6.1 State description

This state is only implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped.

10.1.8.6.2 Transition SA_PC_5:Active_Wait to SA_PC_1:Active

This transition shall occur if:

a) a NOTIFY (ENABLE_SPINUP) is received; or
b) the device does not temporarily consume additional power during the transition to SA_PC_1:Active.

10.1.8.6.3 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE STANDBY is received; or

the Power Condition mode page standby timer expires.

10.1.8.6.4 Transition SA_PC_5:Active_Wait to SA_PC_4:Stopped

This transition shall occur if a START STOP UNIT command with the START bit set to zero is received.

10.1.8.6.5 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE IDLE is received; or

the Power Condition mode page idle timer expires.

10.1.8.7 SA_PC_6:Idle_Wait state

10.1.8.7.1 State description

This state is only implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped.

10.1.8.7.2 Transition SA_PC_6:Idle_Wait to SA_PC_2:Idle

This transition shall occur if:

a) a NOTIFY (ENABLE_SPINUP) is received; or
b) the device does not temporarily consume additional power during the transition to SA_PC_2:Idle.

10.1.8.7.3 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 received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE STANDBY is received; or

the Power Condition mode page standby timer expires.

10.1.8.7.4 Transition SA_PC_6:Idle_Wait to SA_PC_4:Stopped

This transition shall occur if a START STOP UNIT command with the START bit set to zero is received.

10.1.8.7.5 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 received; or
b) ~~a START STOP UNIT command with the POWER CONDITION field set to FORCE ACTIVE is received; or~~
c) a command is received which requires the active power condition.

10.1.9 SCSI vital product data

Each logical unit in a SAS SSP target device shall include the identification descriptors listed in table 128 in the Device Identification vital product data (VPD) page (83h) returned by the INQUIRY command (see SPC-3).

Table 128 — Device Identification VPD page required identification descriptors

Field in identification descriptor	Identification descriptor			
	Logical unit name	**Target port identifier**	**Relative target port identifier**	**Target device name**
IDENTIFIER TYPE	NAA (3h)	NAA (3h)	Relative target port identifier (4h)	NAA (3h)
ASSOCIATION	Logical unit (0h)	Target port (1h)	Target port (1h)	Target device (2h)
CODE SET	Binary (1h)	Binary (1h)	Binary (1h)	Binary (1h)
IDENTIFIER LENGTH	8 [a] or 16 [b]	8	4	8
PIV (PROTOCOL IDENTIFIER VALID)	0	1	0	1
PROTOCOL IDENTIFIER	Any	SAS (6h)	Any	SAS (6h)
IDENTIFIER	NAA IEEE Registered format [a] or NAA IEEE Registered Extended format [b]	NAA IEEE Registered format [a] [c]	Target ports shall be numbered sequentially starting with 00000001h [d]	NAA IEEE Registered format [a]

[a] The IDENTIFIER field contains an NAA field set to IEEE Registered (5h); the IDENTIFIER LENGTH field is set to 8.
[b] The IDENTIFIER field contains an NAA field set to IEEE Registered Extended (6h); the IDENTIFIER LENGTH field is set to 16.
[c] The IDENTIFIER field contains the SAS address of the target port being used to run the INQUIRY command.
[d] The IDENTIFIER field contains the relative target port identifier of the target port being used to run the INQUIRY command.

The target device shall use different identifiers for each logical unit name, each target port identifier, and the target device name.

Logical units may include additional identification descriptors than those required by this standard.

10.2 ATA application layer

No SAS-specific ATA features are defined.

10.3 Management application layer

10.3.1 SMP functions

10.3.1.1 Function overview

The FUNCTION field in the SMP_REQUEST frame (see 9.4.2) and SMP_RESPONSE frame (see 9.4.3) is defined in table 129.

Table 129 — Management functions

Code	Function	Description	Request frame size (in bytes)	Response frame size (in bytes)	Reference
00h	REPORT GENERAL	Return general information about the device.	8	32	10.3.1.2
01h	REPORT MANUFACTURER INFORMATION	Return vendor and product identification.	8	64	10.3.1.3
02h - 0Fh	Reserved for general input functions.				
10h	DISCOVER	Return information about the specified phy.	16	44	10.3.1.4
11h	REPORT PHY ERROR LOG	Return error logging information about the specified phy.	16	32	10.3.1.5
12h	REPORT PHY SATA	Return information about a phy currently attached to a SATA target port.	16	60	10.3.1.6
13h	REPORT ROUTE INFORMATION	Return route table information.	16	44	10.3.1.7
14h - 1Fh	Reserved for phy input functions.				
20h - 3Fh	Reserved for input functions.				
40h - 7Fh	Vendor-specific.				
80h - 8Fh	Reserved for general output functions.				
90h	CONFIGURE ROUTE INFORMATION	Change route table information.	44	8	10.3.1.8
91h	PHY CONTROL	Request actions by the specified phy.	44	8	10.3.1.9
92h - 9Fh	Reserved for phy output functions.				
A0h - BFh	Reserved for output functions.				
C0h - FFh	Vendor-specific.				

~~The CRC field is included in each frame, although that field is parsed by the link layer.~~

10.3.1.2 REPORT GENERAL function

The REPORT GENERAL function returns general information about the device. This function may implemented by any type of device and should be implemented by expander devices.

Table 130 defines the request format.

Table 130 — REPORT GENERAL request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (00h)							
2	Reserved							
3								
4	(MSB)							
7	CRC							(LSB)

The CRC field is defined in 9.4.2.

1) Table 131 defines the response format.

Table 131 — REPORT GENERAL response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (00h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)			EXPANDER CHANGE COUNT				
5								(LSB)
6	(MSB)			EXPANDER ROUTE INDEXES				
7								(LSB)
8	Reserved							
9	NUMBER OF PHYS							
10	Reserved							CONFIGURABLE ROUTE TABLE
28				Reserved				
31								
28	(MSB)			CRC				
31								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. There are no FUNCTION RESULT values specific to this function.

The EXPANDER CHANGE COUNT field counts the number of BROADCAST (CHANGE)s originated by an expander device. 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 it transmits a BROADCAST (CHANGE) for either of the following reasons:

a) after an expander phy has lost dword synchronization; or
b) after the link reset sequence completes.

The expander device need not increment this field again unless a REPORT GENERAL response is transmitted. This field shall not be incremented when forwarding a BROADCAST (CHANGE) from another expander device. The EXPANDER CHANGE COUNT field shall wrap to zero after the maximum value (i.e., FFFFh) has been reached.

NOTE 24 Application clients that use the EXPANDER CHANGE COUNT field should read it often enough to ensure that it does not increment a multiple of 65 536 times between reading the field.

The EXPANDER ROUTE INDEXES field contains the maximum number of route indexes for the expander device. Expander devices shall support this field. Other device types shall not support this field.

~~If an edge expander device supports an expander route table, then the number of expander route indexes for each phy identifier shall be greater than or equal to the number of phys downstream from the edge expander phy.~~

~~If a fanout expander device supports an expander route table, then the number of expander route indexes shall be 64.~~

The NUMBER OF PHYS field contains the number of phys in the device.

The CONFIGURABLE ROUTE TABLE bit indicates whether the expander device has an expander route table that shall be configured. An expander device with a configurable route table shall have the CONFIGURABLE ROUTE TABLE bit set to one. An expander device without a configurable route table shall have the CONFIGURABLE ROUTE TABLE bit set to zero.

The CRC field is defined in 9.4.3.

10.3.1.3 REPORT MANUFACTURER INFORMATION function

The REPORT MANUFACTURER INFORMATION function returns vendor and product identification. This function may be implemented by any type of device.

Table 132 defines the request format.

Table 132 — REPORT MANUFACTURER INFORMATION request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (01h)							
2	Reserved							
3								
4	(MSB)							
7	CRC							(LSB)

The CRC field is defined in 9.4.2.

Table 133 defines the response format.

Table 133 — REPORT MANUFACTURER INFORMATION response

Bit / Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (01h)							
2	FUNCTION RESULT							
3	Reserved							
4	Ignored							
7								
8	Reserved							
9	Ignored							
10								
11	Reserved							
12	(MSB)	VENDOR IDENTIFICATION						
19								(LSB)
20	(MSB)	PRODUCT IDENTIFICATION						
35								(LSB)
36	(MSB)	PRODUCT REVISION LEVEL						
39								(LSB)
40	Vendor-specific							
59								
60	(MSB)	CRC						
63								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. There are no FUNCTION RESULT values specific to this function.

The ADDITIONAL LENGTH field indicates the length in bytes of the parameters, including the ADDITIONAL LENGTH field. If the ADDITIONAL REQUEST BYTES of the SMP_REQUEST is too small to transfer all of the parameters, the ADDITIONAL LENGTH shall not be adjusted to reflect the truncation.

The VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the product. The data shall be left aligned within the field. The vendor identification string should be one defined in SPC-3 for the Standard INQUIRY data VENDOR IDENTIFICATION field.

The PRODUCT IDENTIFICATION field contains sixteen bytes of ASCII data as defined by the vendor. The data shall be left aligned within the field.

The PRODUCT REVISION LEVEL field contains four bytes of ASCII data as defined by the vendor. The data shall be left-aligned within the field.

ASCII data fields 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 (highest offset) and the unused bytes shall be filled with space characters (20h).

The CRC field is defined in 9.4.3.

10.3.1.4 DISCOVER function

The DISCOVER function returns the physical link configuration information for the specified expander phy. This function provides information from the IDENTIFY address frame received by the phy, as well as the routing attribute supported by the phy. This function shall be implemented by all expander devices and shall not be implemented by other types of devices.

Table 134 defines the request format.

Table 134 — DISCOVER request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (10h)							
2	Reserved							
3								
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	(MSB)							
15	CRC							(LSB)

The PHY IDENTIFIER field indicates the phy (see 4.2.6) for the link configuration information being requested.

If the value is not within the range of zero to NUMBER OF PHYS (see 9.4.4.2) then the target port shall return a FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.

The CRC field is defined in 9.4.2.

Table 135 defines the response format.

Table 135 — DISCOVER response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (10h)							
2	FUNCTION RESULT							
3	Reserved							
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	Ignored	ATTACHED DEVICE TYPE			ROUTING ATTRIBUTE			
13	Reserved				NEGOTIATED PHYSICAL LINK RATE			
14	Reserved				ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	Reserved
15	Reserved				ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	ATTACHED SATA TARGET
16	ATTACHED SAS ADDRESS							
23								
24	SAS ADDRESS							
31								
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
34	Vendor-specific							
35								
36	Reserved				PARTIAL PATHWAY TIMEOUT VALUE			
37	Reserved							
39								
40	(MSB) CRC							
43								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 136.

Table 136 — Function results for DISCOVER

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy specified by the PHY IDENTIFIER field does not exist; rest of data is invalid.
All others	Reserved.	

The PHY IDENTIFIER field indicates the phy for which physical configuration link information is being returned.

The ROUTING ATTRIBUTE field indicates the routing attribute and routing method supported by the phy and is defined in table 137.

Table 137 — Routing attributes

Code	Name	Description
0h	Direct routing	Direct routing for attached end devices.
1h	Subtractive routing	Subtractive routing for attached expander devices, direct routing for attached end devices.
2h	Table routing	Table routing for attached expander devices, direct routing for attached end devices.
All others	Reserved.	

The ROUTING ATTRIBUTE field shall not change based on the attached device type. The routing method used by the expander connection manager shall change based on the attached device type as described in table 137.

The ATTACHED DEVICE TYPE field indicates the DEVICE TYPE value received during the link reset sequence and is defined in table 138. The ATTACHED DEVICE TYPE field shall only be set to a value other than 000b after the identification sequence is complete (when a SAS device is attached) or after the initial Register - Device to Host FIS has been received (when a SATA device is attached).

Table 138 — Attached device types

Code	Description
000b	No device attached
001b	End device ~~only~~
010b	Edge expander device
011b	Fanout expander device
All others	Reserved

The NEGOTIATED PHYSICAL LINK RATE field is defined in table 139 and indicates the physical link rate negotiated during the link reset sequence. The negotiated physical link rate may be outside the programmed minimum

physical link rate and programmed maximum physical link rate if they have been changed since the link reset sequence.

Table 139 — Negotiated physical link rate

Code	Description
0h	Phy is enabled; unknown physical link rate. [a]
1h	Phy is disabled.
2h	Phy is enabled; the phy obtained dword synchronization for at least one physical link rate during speed negotiation (either SAS or SATA), but the speed negotiation sequence failed. These failures may be logged in the SMP REPORT PHY ERROR LOG function (see 10.3.1.5) and/or the Protocol-Specific log page (see 10.1.7.1).
3h	Phy is enabled; detected a SATA target device and entered the SATA spinup hold state. The LINK RESET and HARD RESET operations in the SMP PHY CONTROL function (see 10.3.1.9) may be used to release the phy. This field shall be updated to this value after the SATA OOB sequence completes if SATA spinup hold is supported.
4h	Phy is enabled; This field shall be updated to this value after the speed negotiation sequence completes indicating a negotiated physical link rate of 1,5 Gbps.
5h	Phy is enabled; This field shall be updated to this value after the speed negotiation sequence completes indicating a negotiated physical link rate of 3,0 Gbps.
All others	Reserved.

[a] This code may be used by an application client ~~in its local data structures~~ to indicate an unknown negotiated physical link rate (e.g., before the discovery process has queried the phy).

The ATTACHED SSP INITIATOR bit indicates the SSP INITIATOR value received during the link reset sequence.

The ATTACHED STP INITIATOR bit indicates the STP INITIATOR value received during the link reset sequence.

The ATTACHED SMP INITIATOR bit indicates the SMP INITIATOR value received during the link reset sequence.

The ATTACHED SSP TARGET bit indicates the SSP TARGET value received during the link reset sequence.

The ATTACHED STP TARGET bit indicates the STP TARGET value received during the link reset sequence.

The ATTACHED SMP TARGET bit indicates the SMP TARGET value received during the link reset sequence.

The ATTACHED SSP INITIATOR bit, ATTACHED STP INITIATOR bit, ATTACHED SMP INITIATOR bit, ATTACHED SSP TARGET bit, ATTACHED STP TARGET bit, ATTACHED SMP TARGET bit, and ATTACHED SMP TARGET bit shall be updated at the end of the identification sequence.

An ATTACHED SATA TARGET bit set to one indicates a SATA target device is attached. An ATTACHED SATA TARGET bit set to zero indicates a SATA target device is not attached. This bit shall be updated after the SATA OOB sequence completes and before the SATA speed negotiation sequence begins (i.e., at SATA spinup hold time (see 6.10)).

The ATTACHED SAS ADDRESS field contains the SAS address of the attached phy. If the ATTACHED DEVICE TYPE field is set to 000b, the ATTACHED SAS ADDRESS field is invalid.

The SAS ADDRESS field contains the SAS address of this phy. This field shall be updated:

a) after the identification sequence completes, when a SAS device is attached; or
b) after the SATA OOB sequence completes, when a SATA device is attached.

The HARDWARE MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate supported by the phy. The values are defined in table 140.

The PROGRAMMED MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate set by the PHY CONTROL function (see 10.3.1.9). The values are defined in table 140.

The HARDWARE MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate supported by the phy. The values are defined in table 140.

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate set by the PHY CONTROL function (see 10.3.1.9). The values are defined in table 140.

Table 140 — Hardware and programmed physical link rates

Code	Description
5h	1,5 Gbps
6h	3,0 Gbps
All others	Reserved

The PARTIAL PATHWAY TIMEOUT VALUE field indicates the partial pathway timeout value set by the PHY CONTROL function (see 10.3.1.9). The default value for PARTIAL PATHWAY TIMEOUT VALUE shall be 7 µs.

The CRC field is defined in 9.4.3.

10.3.1.5 REPORT PHY ERROR LOG function

The REPORT PHY ERROR LOG returns error logging information about the specified phy. This function may implemented by any type of device.

Table 141 defines the request format.

Table 141 — REPORT PHY ERROR LOG request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (11h)							
2	Reserved							
3								
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	(MSB)							
15	CRC							(LSB)

The PHY IDENTIFIER field indicates the phy (see 4.2.6) for which information shall be reported.

The CRC field is defined in 9.4.2.

Table 142 defines the response format.

Table 142 — REPORT PHY ERROR LOG response

Bit / Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (11h)							
2	FUNCTION RESULT							
3	Reserved							
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	(MSB)	INVALID DWORD COUNT						
15								(LSB)
16	(MSB)	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 FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 143.

Table 143 — Function results for REPORT PHY ERROR LOG

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy specified by the PHY IDENTIFIER field does not exist; rest of data is invalid.
All others	Reserved.	

The INVALID DWORD COUNT field indicates the number of invalid dwords (see 3.1.67) that have been received (outside of phy reset sequences).

The DISPARITY ERROR COUNT field indicates the number of dwords containing disparity errors (see 6.2) that have been received (outside of phy reset sequences).

The LOSS OF DWORD SYNCHRONIZATION COUNT field indicates the number of times dword synchronization (see 6.9) has been lost (outside of phy reset sequences).

The PHY RESET PROBLEM COUNT field indicates the number of times the phy reset sequence has failed.

The CRC field is defined in 9.4.3.

10.3.1.6 REPORT PHY SATA function

The REPORT PHY SATA function returns information about the SATA state for a specified phy. This function shall be implemented by expander devices supporting attachment to SATA target devices. This function shall not be implemented by any other type of device.

Table 144 defines the request format.

Table 144 — REPORT PHY SATA request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (12h)							
2	Reserved							
3								
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	(MSB)							
15	CRC							(LSB)

The PHY IDENTIFIER field indicates the phy (see 4.2.6) for which information shall be reported.

The CRC field is defined in 9.4.2.

Table 145 defines the response format.

Table 145 — REPORT PHY SATA response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (12h)							
2	FUNCTION RESULT							
3	Reserved							
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							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	(MSB) CRC							
59								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 146.

Table 146 — Function results for REPORT PHY SATA

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy specified by the PHY IDENTIFIER field does not exist; rest of data is invalid
11h	PHY DOES NOT SUPPORT SATA	Phy does not support being attached to SATA target devices.
12h	SATA RESET HALTED	The attached device indicated SATA protocol during the phy reset sequence, and the phy reset sequence has not been completed.
13h	PHY NOT SATA	Phy is not currently using SATA protocol.
All others	Reserved.	

The STP SAS ADDRESS field contains the SAS address that is used when a SATA target device is attached to the specified phy.

The REGISTER DEVICE TO HOST FIS field contains the contents of the initial Register - Device to Host FIS delivered by the attached SATA target device after a link reset sequence (see SATA).

The contents of the REGISTER DEVICE TO HOST FIS field shall remain constant until a link reset sequence causes the attached SATA target to deliver another initial Register – Device to Host FIS. The FIS contents shall be stored with little-endian byte ordering. The first byte of the field (i.e., the FIS Type) shall be initialized to zero on power on and whenever dword synchronization is lost to indicate the field is invalid and the attached SATA target 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 target device has delivered the initial Register – Device to Host FIS.

An expander device that receives a connection request for a SATA target device that has not successfully delivered the initial Register – Device to Host FIS shall return an OPEN_REJECT (NO DESTINATION).

NOTE 25 If there is a problem receiving the expected initial Register - Device to Host FIS, the expander device should use SATA_R_ERR to retry until it succeeds. In the DISCOVER response, the ATTACHED SATA TARGET bit is set to one and the ATTACHED SAS ADDRESS field is valid, but the ATTACHED DEVICE TYPE field is set to 00b during this time.

The AFFILIATION VALID bit shall be set to one when the AFFILIATED STP INITIATOR SAS ADDRESS field is valid and the expander device is maintaining an active affiliation between an STP initiator port and the specified phy. The AFFILIATION VALID bit shall be set to zero when no STP initiator has an affiliation with the specified phy.

The AFFILIATED STP INITIATOR SAS ADDRESS field contains the SAS address of the STP initiator port that currently has an affiliation.

The CRC field is defined in 9.4.3.

10.3.1.7 REPORT ROUTE INFORMATION function

The REPORT ROUTE INFORMATION function returns an expander route entry from the expander route table within an expander device. Expander devices shall support this function if the EXPANDER ROUTE INDEXES field is non-zero in the REPORT GENERAL function. This function is used ~~primarily~~ as a diagnostic tool to resolve topology issues.

Table 147 defines the request format.

Table 147 — REPORT ROUTE INFORMATION request

Bit / Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (13h)							
2	Reserved							
3								
4	Reserved							
5								
6	(MSB) EXPANDER ROUTE INDEX							
7								(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	(MSB) CRC							
15								(LSB)

The EXPANDER ROUTE INDEX field indicates the expander route index for the route table entry being requested. If the value is not in the range of 0000h to the value of the EXPANDER ROUTE INDEXES field in the REPORT GENERAL function, the target port shall return a FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.

The PHY IDENTIFIER field indicates the phy identifier for the expander route entry being requested. If the value is not within the range of zero to NUMBER OF PHYS (see 9.4.4.2) the expander device shall return FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.

If the phy with the indicated PHY IDENTIFIER does not have the table routing attribute (see 4.x.x.x) the expander device shall return an INDEX DOES NOT EXIST function result for any EXPANDER ROUTE INDEX with the indicated phy identifier.

The CRC field is defined in 9.4.2.

Table y defines the response format.

Table 148 — REPORT ROUTE INFORMATION response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (13h)							
2	FUNCTION RESULT							
3	Reserved							
5								
6	(MSB)							
7	EXPANDER ROUTE INDEX							(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Ignored							
11	Reserved							
12	ROUTE ENTRY DISABLED	Ignored						
13	Ignored							
15								
16	ROUTED SAS ADDRESS							
23								
24	Ignored							
35								
36	Reserved							
39								
40	(MSB)							
43	CRC							(LSB)

The FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 149.

Table 149 — Function results for REPORT ROUTE INFORMATION

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy specified by the PHY IDENTIFIER field does not exist; rest of data is invalid.
11h	INDEX DOES NOT EXIST	Expander route index does not exist; rest of data is invalid.
All others	Reserved.	

The EXPANDER ROUTE INDEX field contains the expander route index for the expander route entry being returned.

The PHY IDENTIFIER field contains the phy identifier for the expander route entry being returned.

The ROUTE ENTRY DISABLED bit indicates whether the expander connection manager shall use the expander route entry to route connection requests. If the ROUTE ENTRY DISABLED bit is set to zero, then the expander connection manager shall use the expander route entry to route connection requests. If the ROUTE ENTRY DISABLED bit is set to one, the expander route entry shall not be used by the expander connection manager to route connection requests.The ROUTED SAS ADDRESS field contains the SAS address routed by this table entry.

The ROUTED SAS ADDRESS field contains the routed SAS address in the expander route entry.

The CRC field is defined in 9.4.3.

10.3.1.8 CONFIGURE ROUTE INFORMATION function

The CONFIGURE ROUTE INFORMATION function sets an expander route entry within the expander route table of a configurable expander device. Expander devices that do not have a configurable route table or end devices shall not support this function. Expander devices shall support this function if the CONFIGURABLE ROUTE TABLE field is set to one in the REPORT GENERAL response data.

Table 150 defines the request format.

Table 150 — CONFIGURE ROUTE INFORMATION request

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (90h)							
2	Reserved							
5								
6	(MSB)	EXPANDER ROUTE INDEX						
7								(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11								
12	DISABLE ROUTE ENTRY	Ignored						
13	Ignored							
15								
16	ROUTED SAS ADDRESS							
23								
24	Ignored							
35								
36	Reserved							
39								
40	(MSB)	CRC						
43								(LSB)

The EXPANDER ROUTE INDEX field indicates the expander route index for the expander route entry being configured. If the value is not in the range of zero to EXPANDER ROUTE INDEXES the target port shall return a FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.

The PHY IDENTIFIER field indicates the phy identifier for the route table entry being configured. If the value is not within the range of zero to NUMBER OF PHYS (see 9.4.4.2), the target port shall return a FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.

If the phy with the indicated PHY IDENTIFIER does not have the table routing attribute (see 4.x.x.x) the expander device shall return a FUNCTION RESULT of INDEX DOES NOT EXIST in the response frame.

The DISABLE ROUTE ENTRY bit indicates whether the expander connection manager shall use the expander route entry to route connection requests. If the DISABLE ROUTE ENTRY bit is set to zero, then the expander connection manager shall use the route table entry to route connection requests. If the DISABLE ROUTE ENTRY bit is set to one, the route table entry shall not be used by the expander connection manager to route connection requests.

The ROUTED SAS ADDRESS field contains the routed SAS address for the expander route entry being configured.

Table 151 defines the response format.

Table 151 — CONFIGURE ROUTE INFORMATION response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (90h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)			CRC				
7								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 149.

Table 152 — Function results for CONFIGURE ROUTE INFORMATION

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy specified by the PHY IDENTIFIER field does not exist.
11h	INDEX DOES NOT EXIST	Expander route index specified by the EXPANDER ROUTE INDEX field does not exist.
All others	Reserved.	

The CRC field is defined in 9.4.3.

10.3.1.9 PHY CONTROL function

The PHY CONTROL function requests actions by the specified phy. This function may implemented by any type of device.

Table 153 defines the request format.

Table 153 — PHY CONTROL request

Bit / Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (91h)							
2	Reserved							
3								
4	Ignored							
7								
8	Reserved							
9	PHY IDENTIFIER							
10	PHY OPERATION							
11	Reserved							
12	Ignored							
31								
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				Ignored			
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				Ignored			
34	Ignored							
35								
36	Reserved				PARTIAL PATHWAY TIMEOUT VALUE			
39	Reserved							
40	(MSB) CRC							
43								(LSB)

The PHY IDENTIFIER field indicates the phy (see 4.2.6) to which the PHY CONTROL request applies.

The CRC field is defined in 9.4.2.

Table 154 defines the PHY OPERATION field.

Table 154 — Phy operation

Code	Operation	Description
00h	NOP	No operation.
01h	LINK RESET	Perform a link reset sequence (see 4.4) on the specified phy and enable the specified phy. Any active affiliation (see 9.3.2) shall continue to be active. The phy shall bypass the SATA spinup hold state.
02h	HARD RESET	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, the link reset sequence shall include a hard reset sequence (see 4.4.2). Any active affiliation (see 9.3.2) shall be cleared. The phy shall bypass the SATA spinup hold state.
03h	DISABLE	Disable the specified phy (i.e., stop transmitting and receiving on the specified phy). The LINK RESET and HARD RESET operations may be used to enable the phy.
04h	NEA LOOPBACK	Set the specified phy to near-end analog loopback mode (see 7.10) if it is not disabled and it is currently operating at a valid physical link rate.
05h	CLEAR ERROR LOG	Clear the error log counters (see 10.3.1.5) for the specified phy.
06h	CLEAR AFFILIATION	Clear an active affiliation (see 9.3.2) 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 target port shall return a FUNCTION RESULT of SMP FUNCTION FAILED in the response frame.
All others	Reserved.	

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field indicate which physical link rates the phy shall support during speed negotiation during a link reset sequence. These fields may be set in the same request in which the LINK RESET operation is requested, or may be set

beforehand. These fields are updated regardless of the value in the PHY OPERATION field. Table 155 defines the values for these fields.

Table 155 — Programmed physical link rate

Code	Description
0h	Reserved
1h	Reserved
2h	Reserved
3h	Reserved
4h	Reserved
5h	1,5 Gbps
6h	3,0 Gbps
All others	Reserved

The PARTIAL PATHWAY TIMEOUT VALUE field specifies the amount of time in microseconds the expander phy shall wait after receiving Arbitrating (Blocked On Partial) confirmation from the expander connection manager before requesting that the expander connection manager resolve pathway blockage (see 7.12.3.1.4). A PARTIAL PATHWAY TIMEOUT VALUE value of zero indicates that partial pathway resolution shall be requested by the expander phy immediately upon reception of Arbitrating (Blocked On Partial) confirmation from the expander connection manager.

Table 156 defines the response format.

Table 156 — PHY CONTROL response

Bit Byte	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (91h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)			CRC				
7								(LSB)

The FUNCTION RESULT field is defined in 9.4.3. The FUNCTION RESULT field values specific to this function are defined in table 157.

Table 157 — Function results for PHY CONTROL

Code	Name	Description
10h	PHY DOES NOT EXIST	Phy does not exist; rest of data is invalid.
11h	PHY OPERATION FAILED	Operation specified by PHY OPERATION failed (e.g., reset attempted but failed to achieve dword synchronization).
12h	UNKNOWN PHY OPERATION	Unknown PHY OPERATION value.
13h	OTHER ERROR	Other problems occurred.
All others	Reserved.	

The CRC field is defined in 9.4.3.

Annex A
(normative)

Compliant jitter test pattern (CJTPAT)

A.1 Compliant jitter test pattern (CJTPAT)

The CJTPAT consists of a long run of low-density pattern, followed by a long run of high transition density pattern, followed by another short run of low-density pattern. The transitions between the pattern segments stress the receiver. The test pattern 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.

Table A.1 shows the CJTPAT when there is positive running disparity (RD+) at the beginning of the pattern. The 8b and 10b values of each character are shown.

Table A.1 — CJTPAT for RD+

Running disparity	First character	Second character	Third character	Fourth character	Running disparity
RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD+
	1000011100b	0111100011b	1000011100b	0111100011b	
The above dword of low density pattern is sent a total of 41 times					
RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D20.3 (74h)	RD-
	1000011100b	0111100011b	1000011100b	0010111100b	
Phase shift 11100001011					
RD-	D30.3 (7Eh)	D11.5 (ABh)	D21.5 (B5h)	D21.5 (B5h)	RD+
	0111100011b	1101001010b	1010101010b	1010101010b	
Phase shift 00011110100					
RD+	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	RD+
	1010101010b	1010101010b	1010101010b	1010101010b	
The above dword of high density pattern is sent a total of 12 times					
RD+	D21.5 (B5h)	D30.2 (5Eh)	D10.2 (4Ah)	D30.3 (7Eh)	RD+
	1010101010b	1000010101b	0101010101b	0111100011b	
Phase shift 01010000b and 10101111b					

If the same 8b characters are used when there is negative running disparity (RD-), 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 CJTPAT 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 — CJTPAT for RD-

Running disparity	First character	Second character	Third character	Fourth character	Running disparity
RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD-
	0111100011b	1000011100b	0111100011b	1000011100b	
The above dword of low density pattern is sent a total of 41 times					
RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D11.3 (6Bh)	RD+
	0111100011b	1000011100b	0111100011b	1101000011b	
Phase shift 00011110100b					
RD+	D30.3 (7Eh)	D20.2 (54h)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
	1000011100b	0010110101b	0101010101b	0101010101b	
Phase shift 11100001011b					
RD-	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
	0101010101b	0101010101b	0101010101b	0101010101b	
The above dword of high density pattern is sent a total of 12 times					
RD-	D10.2 (4Ah)	D30.5 (BEh)	D21.5 (B5h)	D30.3 (7Eh)	RD-
	0101010101b	0111101010b	10101010b	1000011100b	
Phase shift 10101111b and 01010000b					

To use CJTPAT as the payload in an SSP DATA frame, the 8b patterns for both RD+ and RD- shall be included.

Table A.3 shows a pattern containing both CJTPAT for RD+ and CJTPAT 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 — CJTPAT for RD+ and RD-

First character	Second character	Third character	Fourth character	Notes
D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	This dword is sent a total of 41 times.
...	...	...	...	
D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D20.3(74h)	This dword is sent once.
D30.3(7Eh)	D11.5(ABh)	D21.5(B5h)	D21.5(B5h)	This dword is sent once.
D21.5(B5h)	D21.5(B5h)	D21.5(B5h)	D21.5(B5h)	This dword is sent a total of 12 times.
...	...	...	...	
D21.5(B5h)	D30.2(5Eh)	D10.2(4Ah)	D30.3(7Eh)	This dword is sent once.
D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	This dword is sent a total of 41 times.
...	...	...	...	
D30.3(7Eh)	D30.3(7Eh)	D30.3(7Eh)	D11.3(6Bh)	This dword is sent once.
D30.3(7Eh)	D20.2(54h)	D10.2(4Ah)	D10.2(4Ah)	This dword is sent once.
D10.2(4Ah)	D10.2(4Ah)	D10.2(4Ah)	D10.2.(4Ah)	This dword is sent a total of 12 times.
...	...	...	...	
D10.2(4Ah)	D30.5(BEh)	D21.5(B5h)	D30.3(7Eh)	This dword is sent once.

When the CJTPAT is encapsulated in an SSP frame, the scrambler needs to be considered. By scrambling the desired 8b pattern prior to submitting it to the transmitter scrambler, the scrambling in the transmitter scrambler reverses the prior scrambling of the 8b pattern and the desired 10b pattern results. 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 protocol frame.

Table A.4 shows CJTPAT embedded in a SSP DATA frame with a 24-byte header following the SOF.

The second column (8b data dword) lists the desired 8b pattern data that is to be 8b10b encoded.

The third column (Scrambler output dword) lists the output, in rd format, of the transmitter scrambler.

The fourth column (Scrambled 8b data dword) shows the result of XORing the 8b data with the scrambler output. The data in this column, if supplied to the transmitter scrambler, results in the desired 10b test pattern on the physical link.

The scrambler is re-initialized at the beginning of each frame (SOF) and the scrambler output is independent of the scrambled data. The insertion of ALIGNs within the frame should be avoided because of the possible disruption of the pattern on the physical link.

Table A.4 — CJTPAT scrambled in an SSP DATA frame (part 1 of 4)

Frame contents	8b data dword	Scrambler output dword	Scrambled 8b data dword = 8b data dword XOR scrambler output dword
SOF	<primitive>	n/a	n/a
SSP DATA frame header	unknown	C2D2768Dh	unknown
	unknown	1F26B368h	unknown
	unknown	A508436Ch	unknown
	unknown	3452D354h	unknown
	unknown	8A559502h	unknown
	unknown	BB1ABE1Bh	unknown
SSP frame data	7E7E7E7Eh	FA56B73Dh	8428C943h
	7E7E7E7Eh	53F60B1Bh	2D887565h
	7E7E7E7Eh	F0809C41h	8EFEE23Fh
	7E7E7E7Eh	747FC34Ah	0A01BD34h
	7E7E7E7Eh	BE865291h	C0F82CEFh
	7E7E7E7Eh	7A6FA7B6h	0411D9C8h
	7E7E7E7Eh	3163E6D6h	4F1D98A8h
	7E7E7E7Eh	F036FE0Ch	8E488072h
SSP frame data	7E7E7E7Eh	1EF3EA29h	608D9457h
	7E7E7E7Eh	EB342694h	954A58EAh
	7E7E7E7Eh	53853B17h	2DFB4569h
	7E7E7E7Eh	E94ADC4Dh	9734A233h
	7E7E7E7Eh	5D200E88h	235E70F6h
	7E7E7E7Eh	6901EDD0h	177F93AEh
	7E7E7E7Eh	FA9E38DEh	84E046A0h
	7E7E7E7Eh	68DB4B07h	16A53579h
SSP frame data	7E7E7E7Eh	450A437Bh	3B743D05h
	7E7E7E7Eh	960DD708h	E873A976h
	7E7E7E7Eh	3F35E698h	414B98E6h
	7E7E7E7Eh	FE7698A5h	8008E6DBh
	7E7E7E7Eh	C80EF715h	B670896Bh
	7E7E7E7Eh	666090AFh	181EEED1h
	7E7E7E7Eh	FAF0D5CBh	848EABB5h
	7E7E7E7Eh	2B82009Fh	55FC7EE1h

Table A.4 — CJTPAT scrambled in an SSP DATA frame (part 2 of 4)

Frame contents	8b data dword	Scrambler output dword	Scrambled 8b data dword = 8b data dword XOR scrambler output dword
SSP frame data	7E7E7E7Eh	0E317491h	704F0AEFh
	7E7E7E7Eh	76F46A1Eh	088A1460h
	7E7E7E7Eh	F46D6948h	8A131736h
	7E7E7E7Eh	7BCD8A93h	05B3F4Edh
	7E7E7E7Eh	1513AD7Eh	6B6DD300h
	7E7E7E7Eh	1E72FEEEh	600C8090h
	7E7E7E7Eh	A014AA3Bh	DE6AD445h
	7E7E7E7Eh	23AAD4E7h	5DD4AA99h
SSP frame data	7E7E7E7Eh	B0DC9E67h	CEA2E019h
	7E7E7E7Eh	E0A573FBh	9EDB0D85h
	7E7E7E7Eh	06CA944Fh	78B4EA31h
	7E7E7E7Eh	63E29212h	1D9CEC6Ch
	7E7E7E7Eh	4578626Dh	3B061C13h
	7E7E7E7Eh	53260C93h	2D5872EDh
	7E7E7E7Eh	3E592202h	40275C7Ch
	7E7E7E7Eh	2B6ECA63h	5510B41Dh
SSP frame data	7E7E7E7Eh	636A1F1Fh	1D146161h
	7E7E7E74h	35B5A9Edh	4BCBD799h
	7EABB5B5h	4AA2A0FDh	34091548h
	B5B5B5B5h	71AFE196h	C41A5423h
	B5B5B5B5h	E1D57B62h	5460CED7h
	B5B5B5B5h	55A0568Ah	E015E33Fh
	B5B5B5B5h	82D18968h	37643CDDh
	B5B5B5B5h	234CB4FFh	96F9014Ah
SSP frame data	B5B5B5B5h	83481E7Fh	36FDABCAh
	B5B5B5B5h	B21AE87Fh	07AF5DCAh
	B5B5B5B5h	A9C5EACDh	1C705F78h
	B5B5B5B5h	6201ACC3h	D7B41976h
	B5B5B5B5h	F60939CEh	43BC8C7Bh
	B5B5B5B5h	395F767Dh	8CEAC3C8h
	B5B5B5B5h	2FA55841h	9A10EDF4h
	B55E4A7Eh	836D4A7Ah	36330004h

Table A.4 — CJTPAT scrambled in an SSP DATA frame (part 3 of 4)

Frame contents	8b data dword	Scrambler output dword	Scrambled 8b data dword = 8b data dword XOR scrambler output dword
SSP frame data	7E7E7E7Eh	388D587Ah	46F32604h
	7E7E7E7Eh	773DFF5Ch	09438122h
	7E7E7E7Eh	3C239CB3h	425DE2CDh
	7E7E7E7Eh	564D91A0h	2833EFDEh
	7E7E7E7Eh	43ED0BE1h	3D93759Fh
	7E7E7E7Eh	987429A7h	E60A57D9h
	7E7E7E7Eh	E52DDBA2h	9B53A5DCh
	7E7E7E7Eh	E78DC87Fh	99F3B601h
SSP frame data	7E7E7E7Eh	0AB8C669h	74C6B817h
	7E7E7E7Eh	64D083C9h	1AAEFDB7h
	7E7E7E7Eh	053DF93Ah	7B438744h
	7E7E7E7Eh	EEE9D9Eah	9097A794h
	7E7E7E7Eh	44BD3B97h	3AC345E9h
	7E7E7E7Eh	0FE24B8Ch	719C35F2h
	7E7E7E7Eh	F28D5694h	8CF328Eah
	7E7E7E7Eh	6310B6D9h	1D6EC8A7h
SSP frame data	7E7E7E7Eh	1792AECEh	69ECD0B0h
	7E7E7E7Eh	0A562EA1h	742850DFh
	7E7E7E7Eh	B048DF69h	CE36A117h
	7E7E7E7Eh	161A2878h	68645606h
	7E7E7E7Eh	5519CB51h	2B67B52Fh
	7E7E7E7Eh	19F5BE56h	678BC028h
	7E7E7E7Eh	EFFFB4B6h	9181CAC8h
	7E7E7E7Eh	B3826E72h	CDFC100Ch
SSP frame data	7E7E7E7Eh	E4722DDAh	9A0C53A4h
	7E7E7E7Eh	60BF5129h	1EC12F57h
	7E7E7E7Eh	248D90F5h	5AF3EE8Bh
	7E7E7E7Eh	4D06D21Ch	3378AC62h
	7E7E7E7Eh	7E96166Ch	00E86812h
	7E7E7E7Eh	5FAFE3B4h	21D19DCAh
	7E7E7E7Eh	506CB855h	2E12C62Bh
	7E7E7E7Eh	5BF03098h	258E4EE6h

Table A.4 — CJTPAT scrambled in an SSP DATA frame (part 4 of 4)

Frame contents	8b data dword	Scrambler output dword	Scrambled 8b data dword = 8b data dword XOR scrambler output dword
SSP frame data	7E7E7E7Eh	46D4B6B3h	38AAC8CDh
	7E7E7E7Eh	051B9E11h	7B65E06Fh
	7E7E7E7Eh	015CC556h	7F22BB28h
	7E7E7E7Eh	E21035Efh	9C6E4B91h
	7E7E7E7Eh	56604D75h	281E330Bh
	7E7E7E7Eh	2E76675Ch	50081922h
	7E7E7E7Eh	071476F0h	796A088Eh
	7E7E7E7Eh	AFF087Ebh	D18EF995h
SSP frame data	7E7E7E7Eh	1B62DB01h	651CA57Fh
	7E7E7E6Bh	23661F6Ch	5D186107h
	7E544A4Ah	F877B027h	8623FA6Dh
	4A4A4A4Ah	F5E389A2h	BFA9C3E8h
	4A4A4A4Ah	EEC73611h	A48D7C5Bh
	4A4A4A4Ah	4C04FB93h	064EB1D9h
	4A4A4A4Ah	E8D70F32h	A29D4578h
	4A4A4A4Ah	BFF03C54h	F5BA761Eh
SSP frame data	4A4A4A4Ah	E3403C01h	A90A764Bh
	4A4A4A4Ah	20FACA7Eh	6AB08034h
	4A4A4A4Ah	9942458Ch	D3080FC6h
	4A4A4A4Ah	37E2CB89h	7DA881C3h
	4A4A4A4Ah	5A1A9783h	1050DDC9h
	4A4A4A4Ah	CE48AA3Fh	8402E075h
	4A4A4A4Ah	06C9A761h	4C83ED2Bh
	4ABEB57Eh	06C03EABh	4C7E8BD5h
CRC	unknown	don’t care	unknown
EOF	<primitive>	N/A	N/A

Annex B
(informative)

SAS phy reset sequence examples

B.1 SAS phy reset sequence examples

Figure A.1 shows a speed negotiation between a SAS phy A that supports only the G1 physical link rate attached to a SAS phy B that only supports the G1 physical link rate. Both phys run the G1 speed negotiation window (valid) and the G2 speed negotiation window (invalid), that phy then selects G1 for the final speed negotiation window used to establish the negotiated physical link rate.

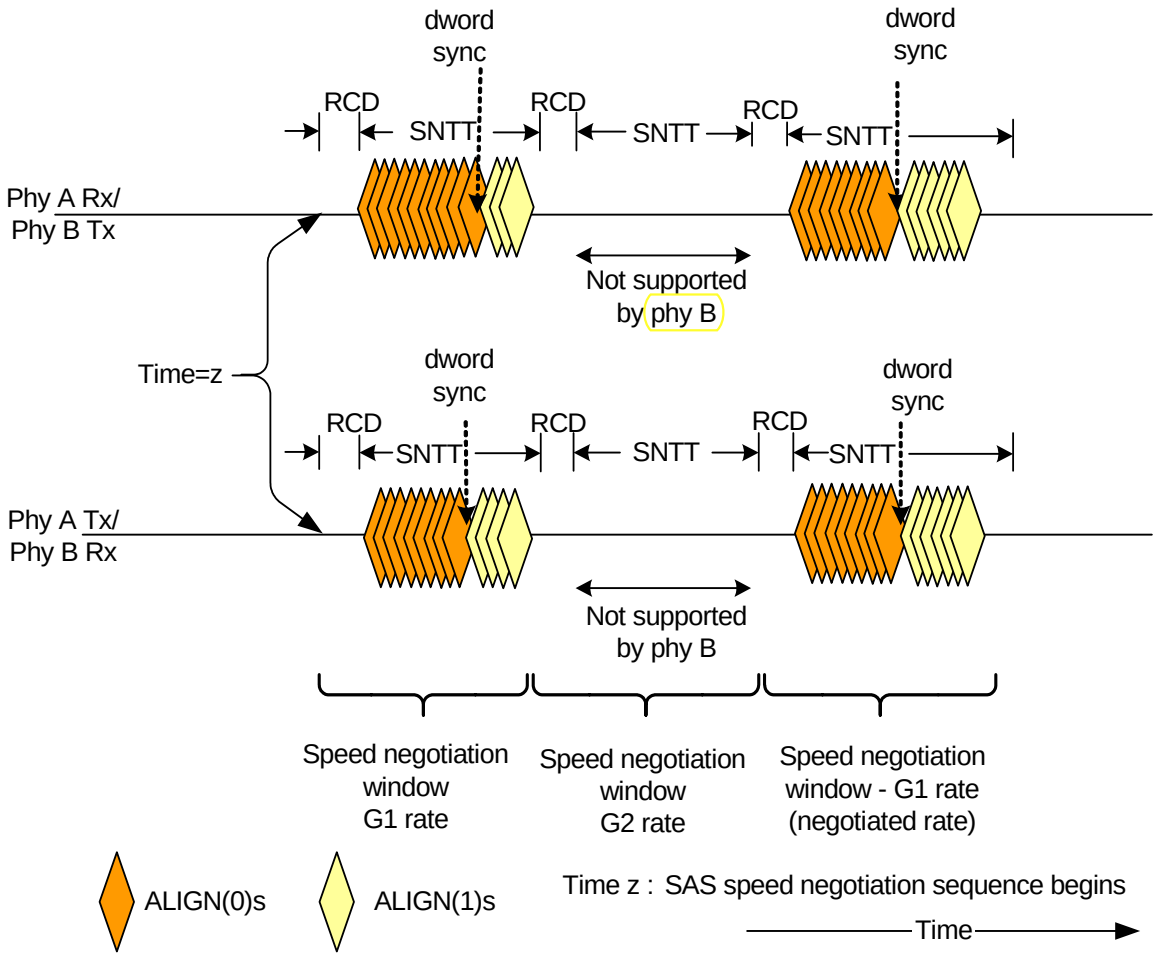


Figure B.1 — SAS speed negotiation sequence (phy A: G1 only, phy B: G1 only)

Figure A.2 shows a speed negotiation between a SAS phy A that supports G1 through G3 physical link rates attached to a SAS phy B that only supports G1 and G2 physical link rates. Both phys run the G1 speed negotiation window (valid), the G2 speed negotiation window (valid), and the G3 speed negotiation window

(supported by phy A but not by phy B, so invalid), that phy then selects G2 for the final speed negotiation window used to establish the negotiated physical link rate.

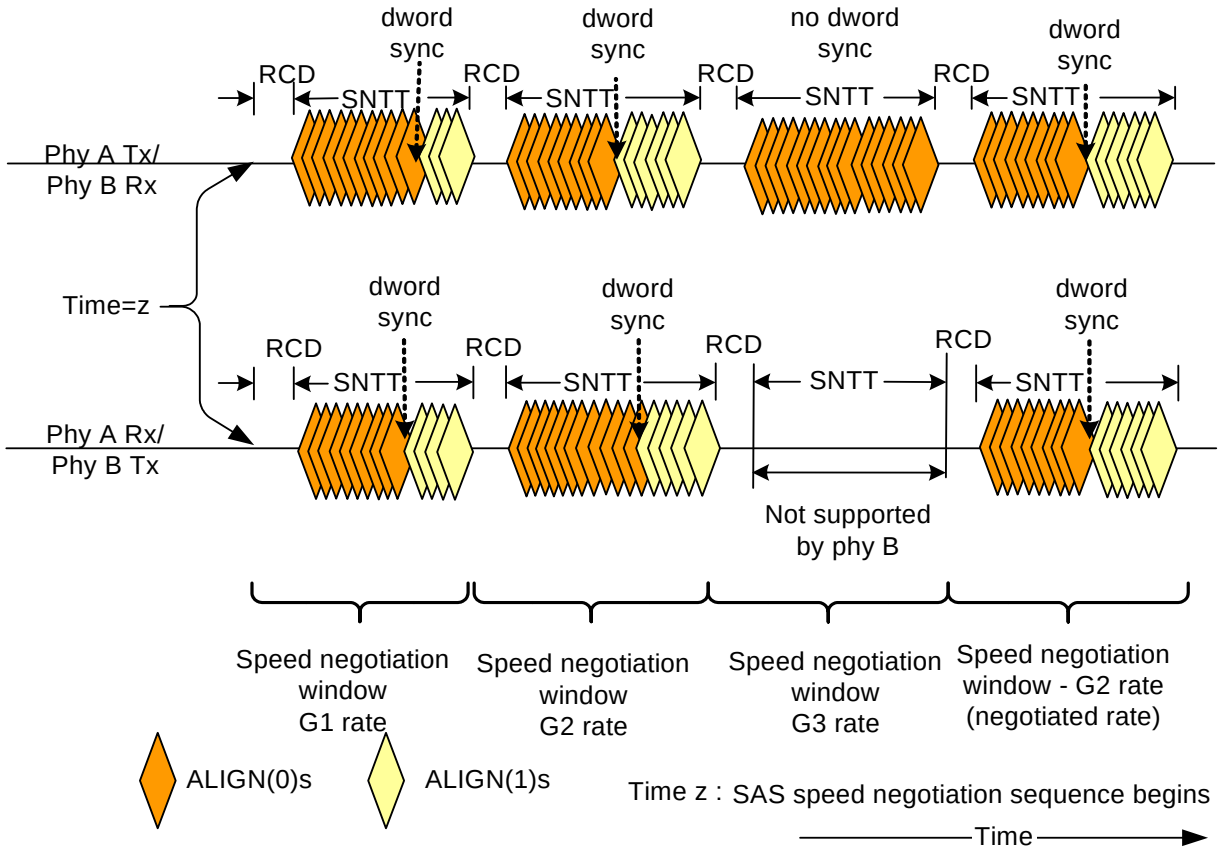


Figure B.2 — SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G1, G2)

Annex C
(informative)

CRC

C.1 CRC generator and checker implementation examples

Figure C.1 shows an example of a CRC generator.

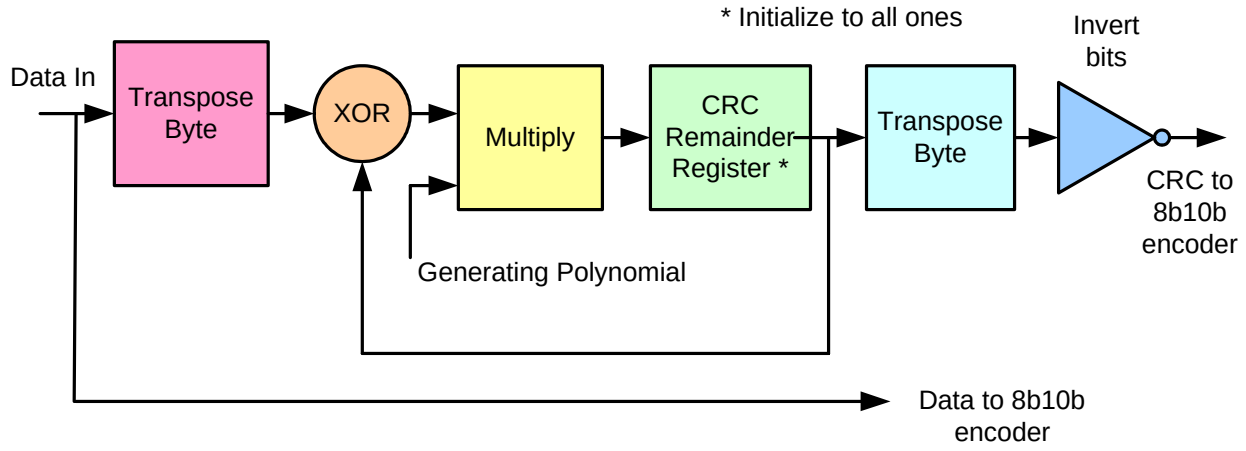


Figure C.1 — CRC generator example

Figure C.2 shows an example of a CRC checker.

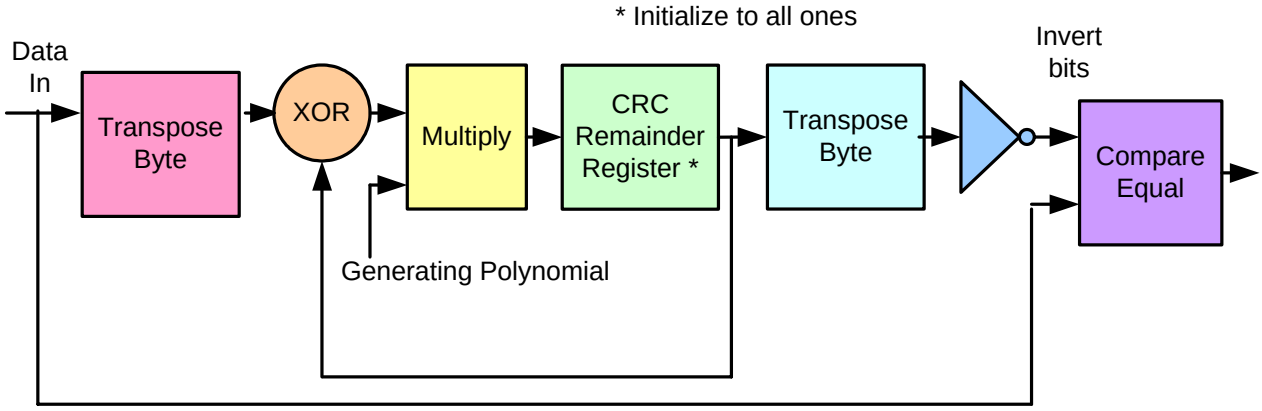


Figure C.2 — CRC checker example

The resulting CRC for test patterns 1, 2, and 3 below are included to provide a validation of CRC generating and check implementations.

C.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 */
}
```

C.3 CRC implementation with XORs

These equations generate the 32 bit CRC for frame transmission. 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;
```

C.4 CRC examples

Table C.1 shows several CRC examples. Data is shown in dwords, from first to last.

Table C.1 — CRC examples

Data	CRC
00010203h 04050607h 08090A0Bh 0C0D0E0Fh 10111213h 14151617h 18191A1Bh 1C1D1E1Fh	8A7E2691h
00000001h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h	898C0D7Ah
00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000001h	3B650D6Eh
06D0B992h 00B5DF59h 00000000h 00000000h 1234FFFFh 00000000h 00000000h 00000000h 00000000h 08000012h 01000000h 00000000h 00000000h	3F4F1C26h

Annex D
(informative)

SAS address hashing

D.1 Hashing overview

4.2.2 describes hashed SAS addresses and the algorithm used to create them.

D.2 Hash collision probability

The following are Monte-Carlo simulations evaluating the probability of collision in a system containing 128 devices. Four models were used for the models for the simulations.

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 initiator port 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 (representing a replacement unit) with another unique company ID.

The three lots model uses:

a) Two sequentially assigned initiator port 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 (representing a replacement unit) with another unique company ID.

Table D.1 shows the results of Monte-Carlo simulation.

Table D.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

D.3 Hash generation

One way to implement the hashing encoder in hardware is to use serial shift registers as show in figure D.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 (bit 63) × L + k added to (bit k modulo 2) for k = 0, 1, …, 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 can be treated as an equivalent binary sequence of 63 bits, which, in turn, may be 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.

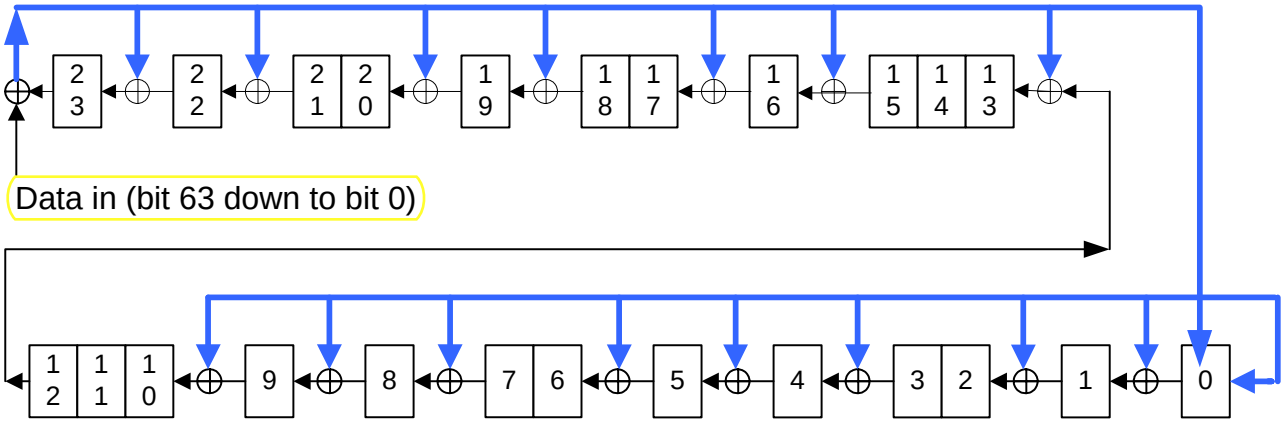


Figure D.1 — BCH(69, 39, 9) code generator

D.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;
}
```

D.5 Hash implementation with XORs

These equations generate the 24-bit HASHED SAS ADDRESS field 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;
```

D.6 Hash examples

Table D.2 shows examples using realistic SAS addresses as input values.

Table D.2 — 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 D.3 shows examples using a walking ones pattern to generate the input values.

Table D.3 — 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 D.4 shows examples using a walking zeros pattern to generate the input values.

Table D.4 — Hash results for a walking zeros pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
FFFFFFFFFFFFFFFEh	000000h	FFFFFFFEFFFFFFFFh	5915B5h
FFFFFFFFFFFFFFFDh	B64EEEh	FFFFFFFDFFFFFFFFh	046584h
FFFFFFFFFFFFFFFBh	01F445h	FFFFFFFBFFFFFFFFh	BE85E6h
FFFFFFFFFFFFFFF7h	B5A664h	FFFFFFF7FFFFFFFFh	106255h
FFFFFFFFFFFFFFEFh	062551h	FFFFFFEFFFFFFFFFh	968A44h
FFFFFFFFFFFFFFDFh	BA044Ch	FFFFFFDFFFFFFFFFh	407D11h
FFFFFFFFFFFFFFBFh	196101h	FFFFFFBFFFFFFFFFh	36B4CCh
FFFFFFFFFFFFFF7Fh	848CECh	FFFFFF7FFFFFFFFFh	DB2776h
FFFFFFFFFFFFEFFFh	647041h	FFFFFEFFFFFFFFFFh	DB2775h
FFFFFFFFFFFFDFFFh	7EAE6Ch	FFFFFDFFFFFFFFFFh	DB2773h
FFFFFFFFFFFFBFFFh	4B1236h	FFFFFBFFFFFFFFFFh	DB277Fh
FFFFFFFFFFFF7FFFh	206A82h	FFFFF7FFFFFFFFFFh	DB2767h
FFFFFFFFFFFEFFFFh	F69BEAh	FFFFEFFFFFFFFFFFh	DB2757h
FFFFFFFFFFFDFFFFh	805E4Dh	FFFFDFFFFFFFFFFFh	DB2737h
FFFFFFFFFFFBFFFFh	6DD503h	FFFFBFFFFFFFFFFFh	DB27F7h
FFFFFFFFFFF7FFFFh	6DE4E8h	FFFF7FFFFFFFFFFFh	DB2677h
FFFFFFFFFFEFFFFFh	6D873Eh	FFFEFFFFFFFFFFFFh	DB2577h
FFFFFFFFFFDFFFFFh	6D4092h	FFFDFFFFFFFFFFFFh	DB2377h
FFFFFFFFFFBFFFFFh	6CCFCAh	FFFBFFFFFFFFFFFFh	DB2F77h
FFFFFFFFFF7FFFFFh	6FD17Ah	FFF7FFFFFFFFFFFFh	DB3777h
FFFFFFFFFEFFFFFFh	69EC1Ah	FFEFFFFFFFFFFFFFh	DB0777h
FFFFFFFFFDFFFFFFh	6596DAh	FFDFFFFFFFFFFFFFh	DB6777h
FFFFFFFFFBFFFFFFh	7D635Ah	FFBFFFFFFFFFFFFFh	DBA777h
FFFFFFFFF7FFFFFFh	4C885Ah	FF7FFFFFFFFFFFFFh	DA2777h
FFFFFFFFEFFFFFFFh	2F5E5Ah	FEFFFFFFFFFFFFFFh	D92777h
FFFFFFFFDFFFFFFFh	E8F25Ah	FDFFFFFFFFFFFFFFh	DF2777h
FFFFFFFFBFFFFFFFh	BC8D2Dh	FBFFFFFFFFFFFFFFh	D32777h
FFFFFFFF7FFFFFFFh	1473C3h	F7FFFFFFFFFFFFFFh	CB2777h
FFFFFFFEFFFFFFFFh	9EA968h	EFFFFFFFFFFFFFFFh	FB2777h
FFFFFFFDFFFFFFFFh	503B49h	DFFFFFFFFFFFFFFFh	9B2777h
FFFFFFFBFFFFFFFFh	16387Ch	BFFFFFFFFFFFFFFFh	5B2777h
FFFFFFF7FFFFFFFFh	9A3E16h	7FFFFFFFFFFFFFFFh	000000h

Annex E
(informative)

Scrambling

Figure E.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.

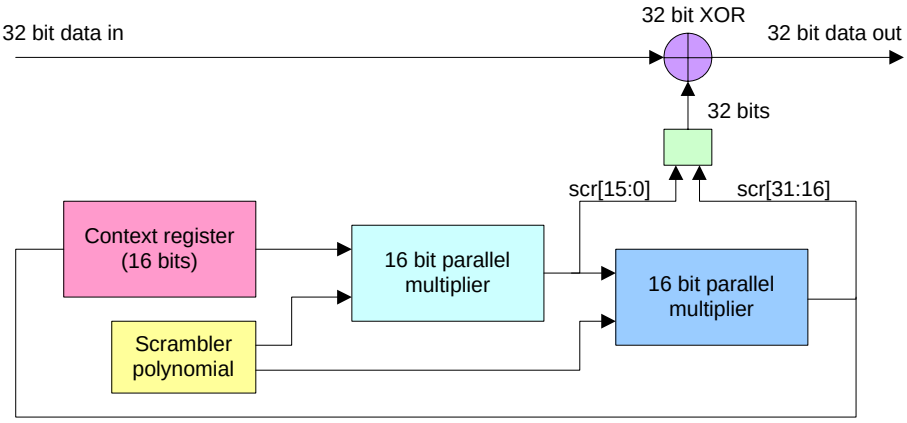


Figure E.1 — Scrambler

E.1 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 following an SOF.

The generator polynomial ~~specified~~ is:

$G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$

For parallelized versions of the scrambler, the initialized value is selected to produce a first dword output of C2D2768Dh for a dword input of all zeros.

```
#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;
    }
```

E.2 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;
```

E.3 Scrambler examples

Table E.1 shows several scrambler examples. Data is shown in dwords, from first to last.

Table E.1 — Scrambler examples

Data	Scrambled output
06D0B992h	C402CF1Fh
00B5DF59h	1F936C31h
00000000h	A508436Ch
00000000h	3452D354h
1234FFFFh	98616AFDh
00000000h	BB1ABE1Bh
00000000h	FA56B73Dh
00000000h	53F60B1Bh
00000000h	F0809C41h
08000012h	7C7FC358h
01000000h	BF865291h
00000000h	7A6FA7B6h
00000000h	3163E6D6h
3F4F1C26h	CF79E22Ah
00000000h	C2D2768Dh
00000000h	1F26B368h
00000000h	A508436Ch
00000000h	3452D354h
00000000h	8A559502h
00000000h	BB1ABE1Bh
00000000h	FA56B73Dh
00000000h	53F60B1Bh
00000000h	F0809C41h
00000000h	747FC34Ah
00000000h	BE865291h
00000000h	7A6FA7B6h
00000000h	3163E6D6h
B00F2BCCh [a]	4039D5C0h

[a] The last dword represents a CRC dword.

Annex F
(informative)

ATA architectural notes

F.1 STP differences from SATA

Some of the differences of STP compared with SATA are:

a) STP adds addressing of multiple SATA targets. Each SATA target is assigned a SAS address by its attached expander device. The STP initiator port understands addressing more than one target;
b) STP allows multiple initiators to share access to a SATA target, but only in an active/standby mode;
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) target ports are not sent to sleep during near-end analog loopback testing; and
g) use of SATA_CONT is required.

F.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 device spinup;
d) drive activity indication;
e) drive hot-plug improvements; and
f) power management improvements.

F.3 Byte and bit ordering

Dwords in STP frames are little-endian and feed into the STP CRC generator without swapping bits within each byte and inverting the output like the SAS CRC generator.

Figure F.1 shows the STP CRC bit ordering.

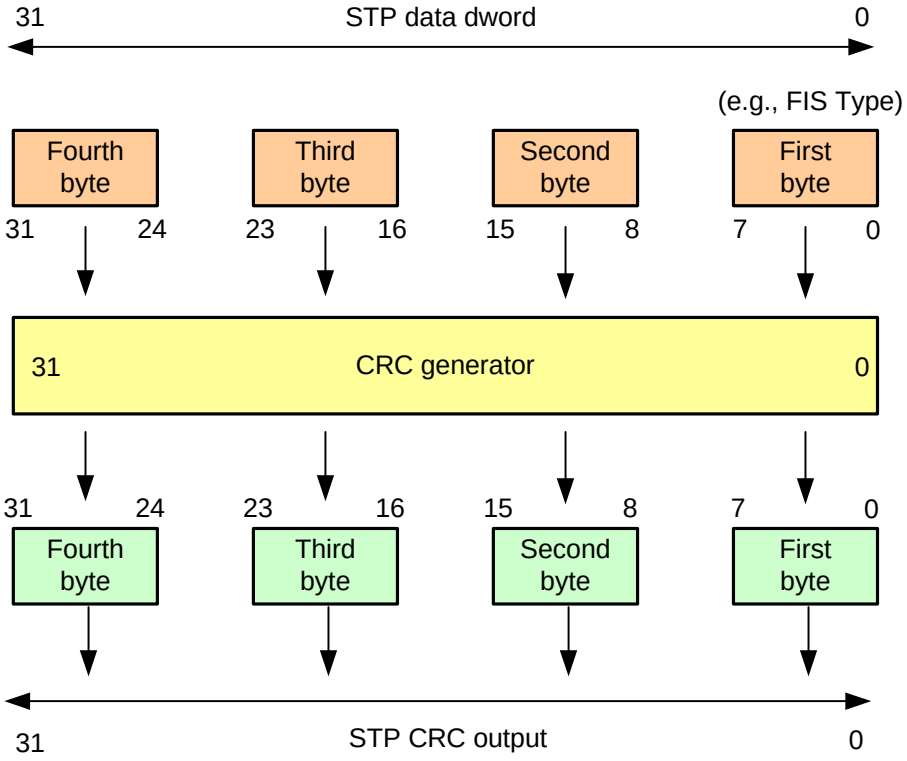


Figure F.1 — STP CRC bit ordering

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. Thus, the first byte contains the least-significant bit. In SSP and SMP, the first byte contains the most-significant bit.

Figure F.2 shows the STP transmit path bit ordering.

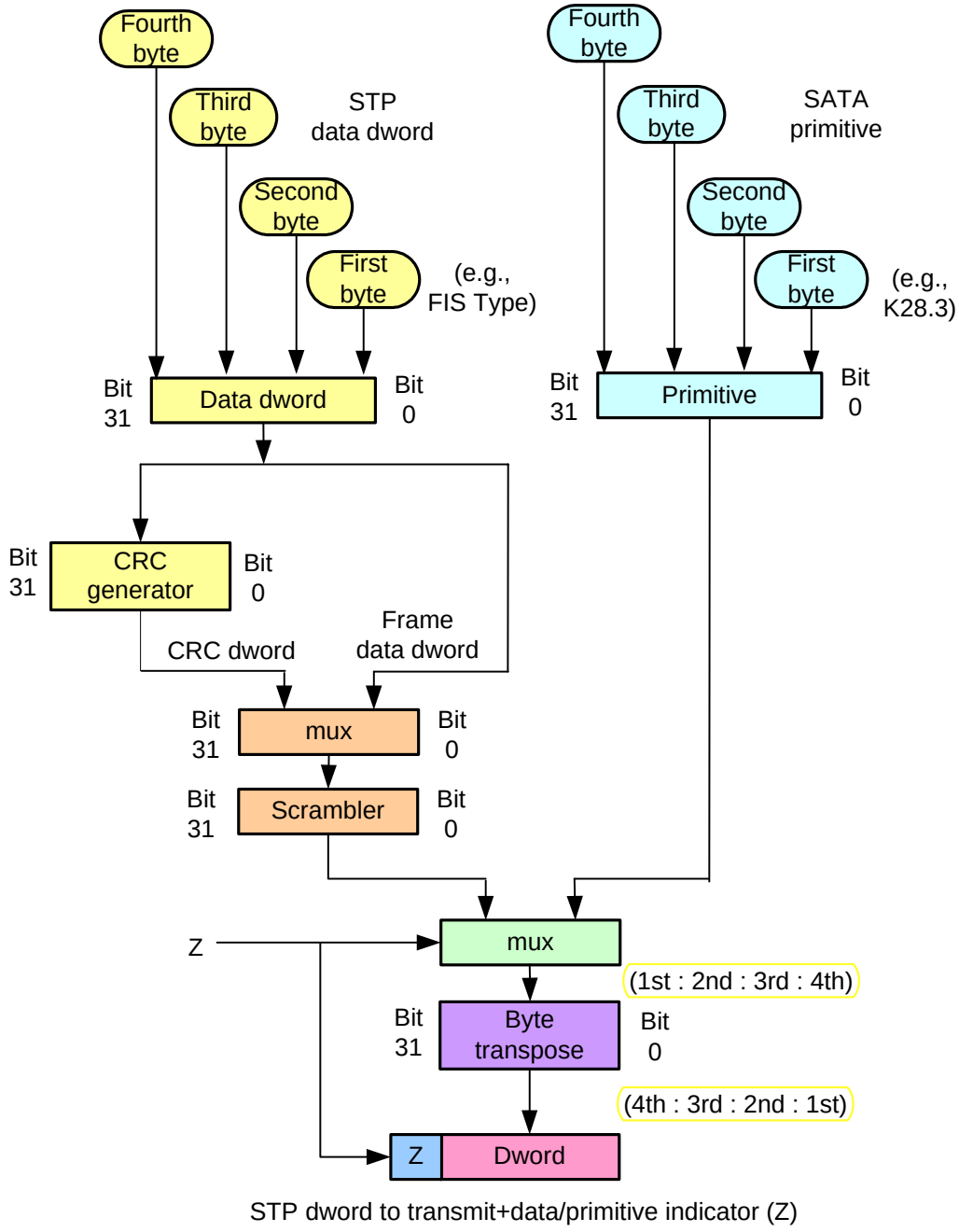


Figure F.2 — STP transmit path bit ordering

Figure F.3 shows the STP receive path bit ordering.

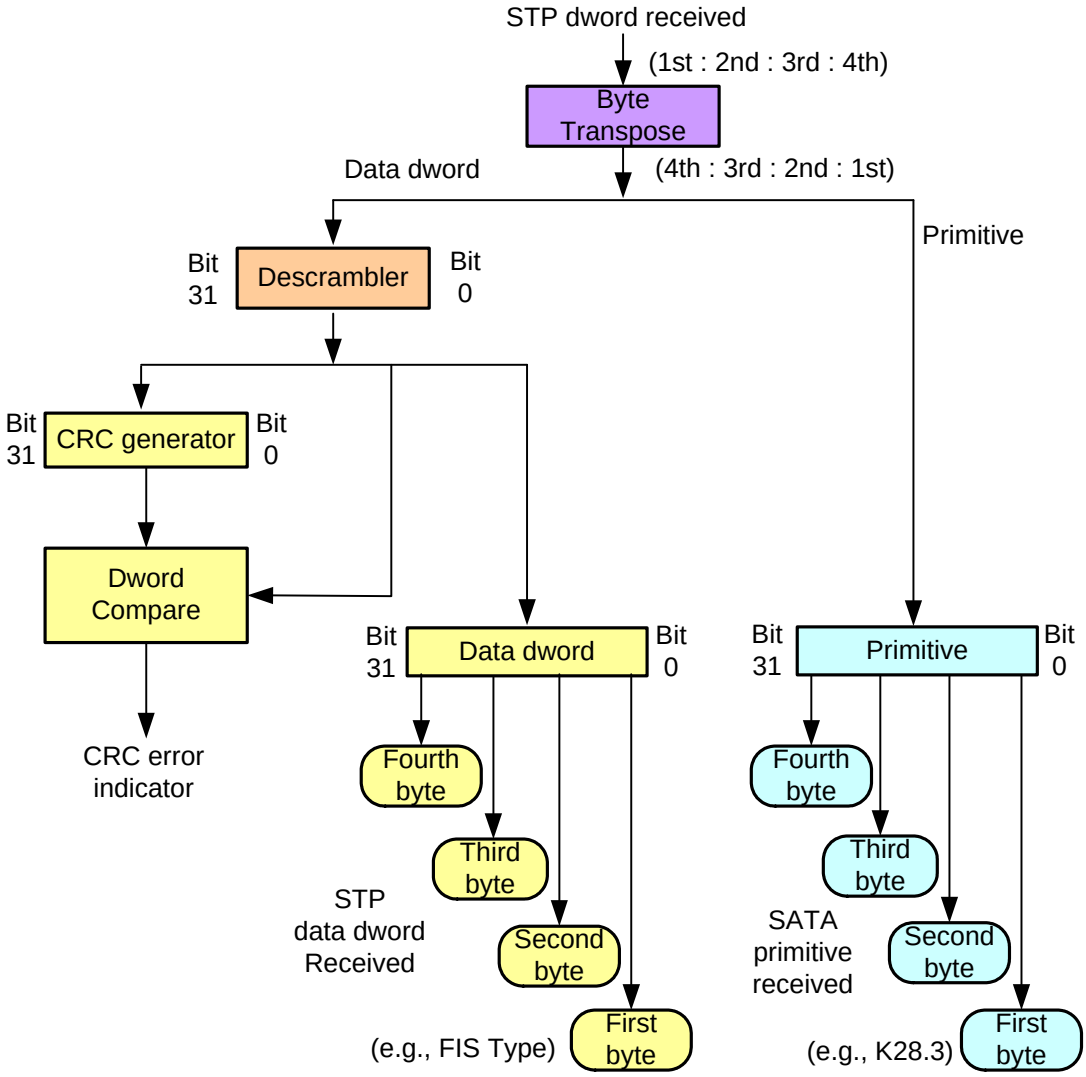


Figure F.3 — STP receive path bit ordering

Annex G
(informative)

Expander handling of connections

G.1 Overview

This annex provides examples of how SAS expander devices process connection requests.

Figure G.1 shows the topology used by examples in this annex.

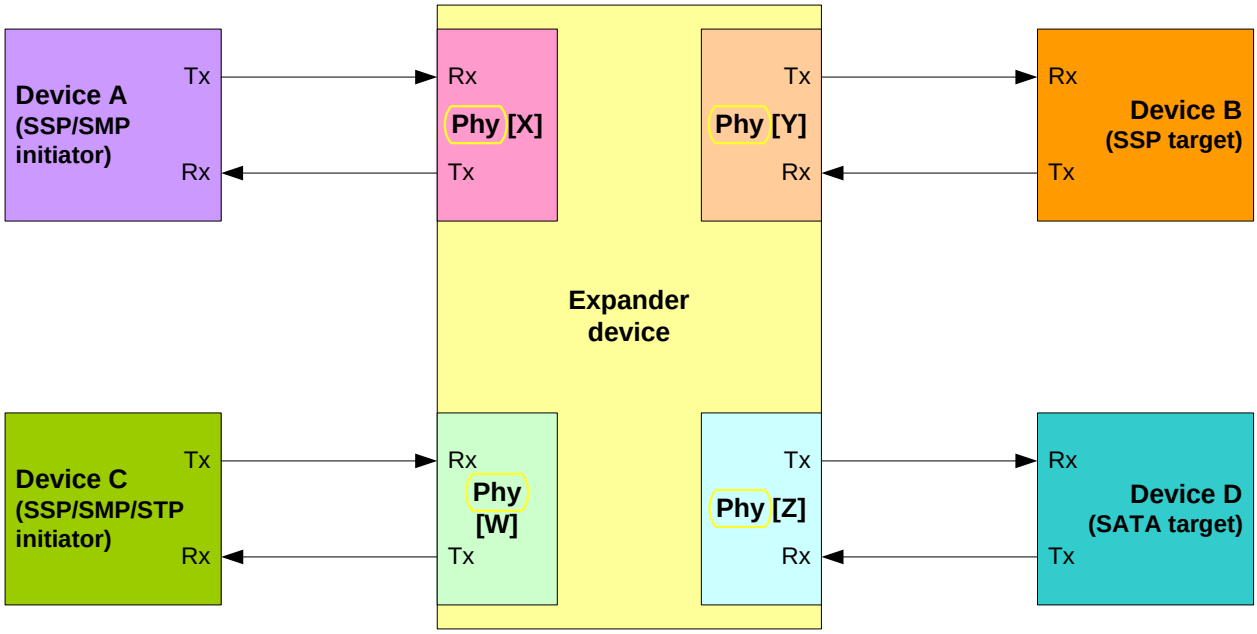


Figure G.1 — Example topology

Table G.1 defines the column headers used within the figures contained within this annex.

Table G.1 — Column descriptions for connection examples

Column header	Description
Phy [X] RX	Phy [X] Receive from device A
Phy [X] TX	Phy [X] Transmit to device A
Phy [X] XL state	Phy [X] XL state
Phy [X] XL req/rsp	Phy [X] XL request and response signals
Phy [X] XL cnf/ind	Phy [X] XL confirmation and indication signals
Phy [Y] XL cnf/ind	Phy [Y] XL confirmation and indication signals
Phy [Y] XL req/rsp	Phy [Y] XL request and response signals
Phy [Y] XL state	Phy [Y] XL state
Phy [Y] TX	Phy [Y] Transmit to device B
Phy [Y] RX	Phy [Y] Receive from device B

G.2 Connection request - Open accept

Figure G.2 shows the establishment of a successful connection between two end devices.

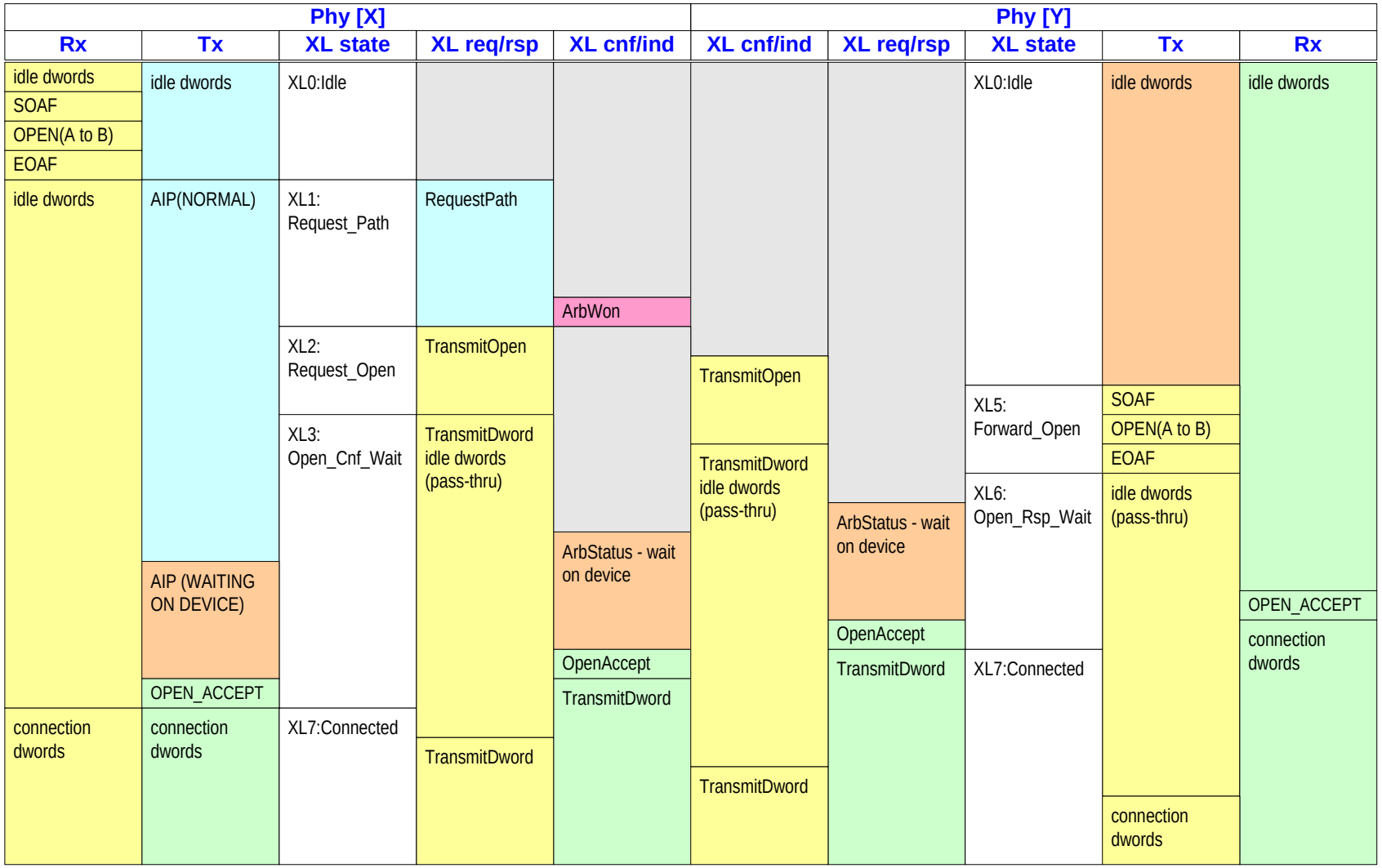


Figure G.2 — Open accept

G.3 Connection request - Open reject by end device

Figure G.3 shows failure to establish a connection due to rejection of the connection request by an end device.

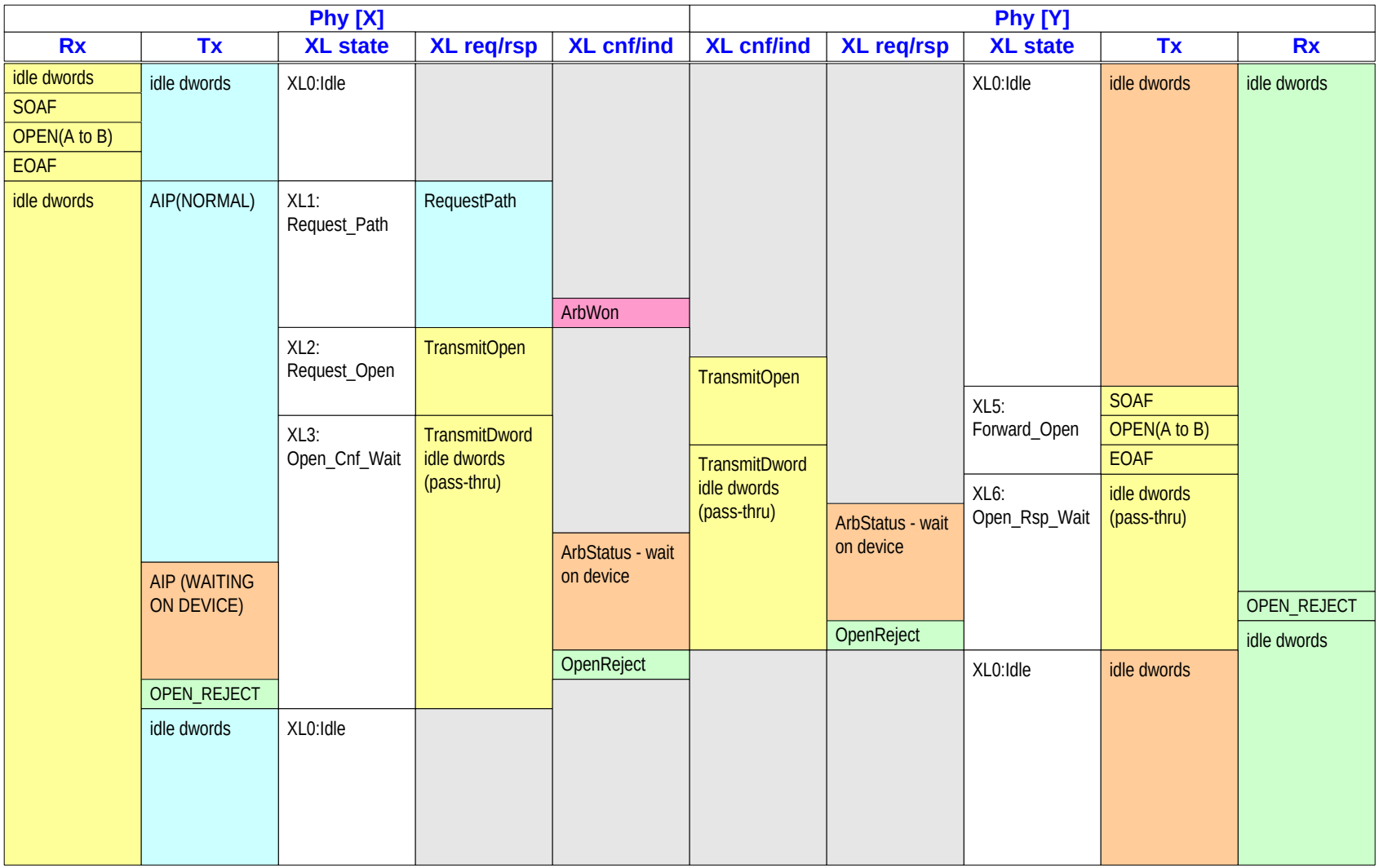


Figure G.3 — Open reject by end device

G.4 Connection request - Open reject by expander device

Figure G.4 shows failure to establish a connection due to rejection of the connection request by an expander device.

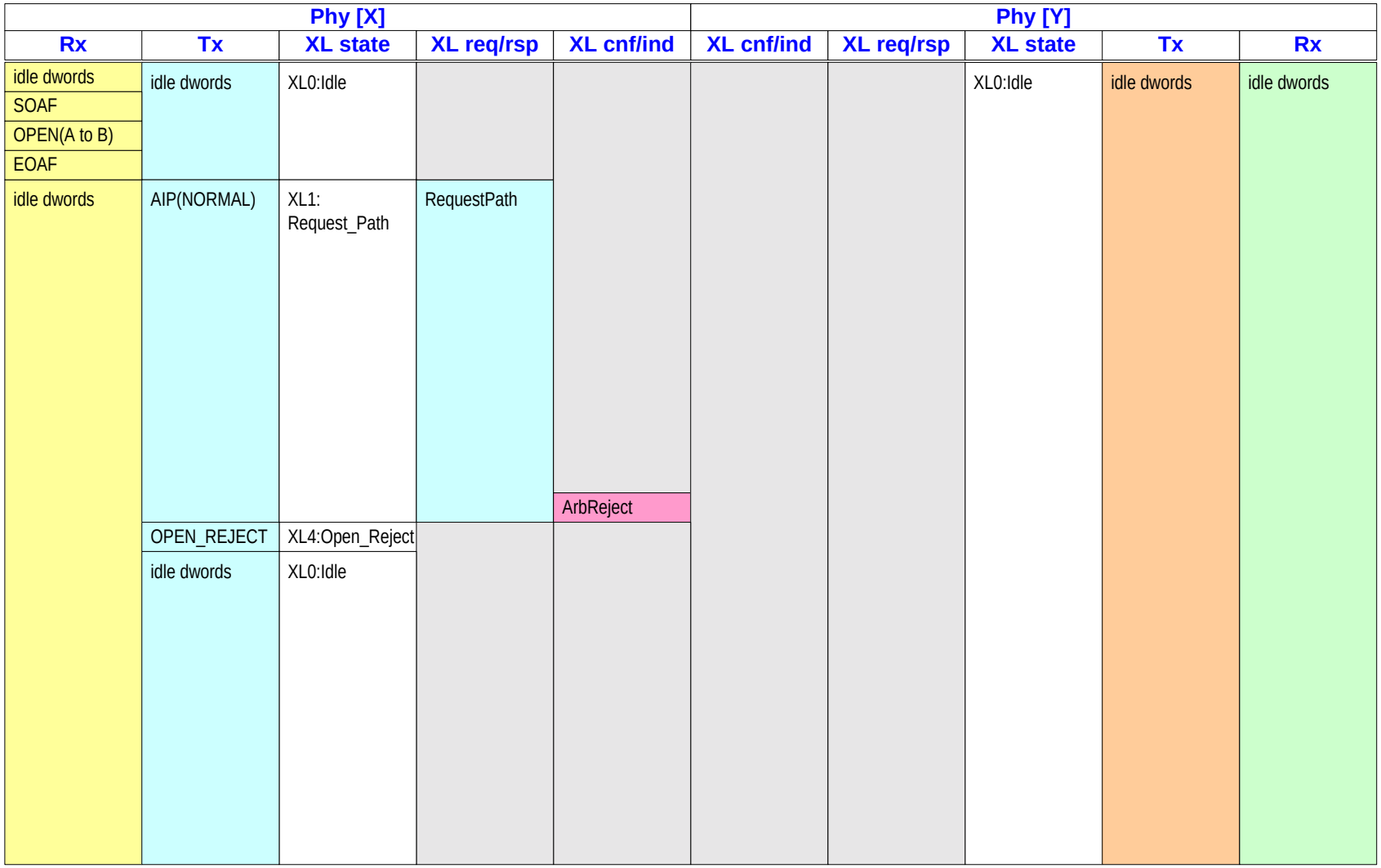


Figure G.4 — Open reject by expander device

G.5 Connection request - Arbitration lost

Figure G.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.

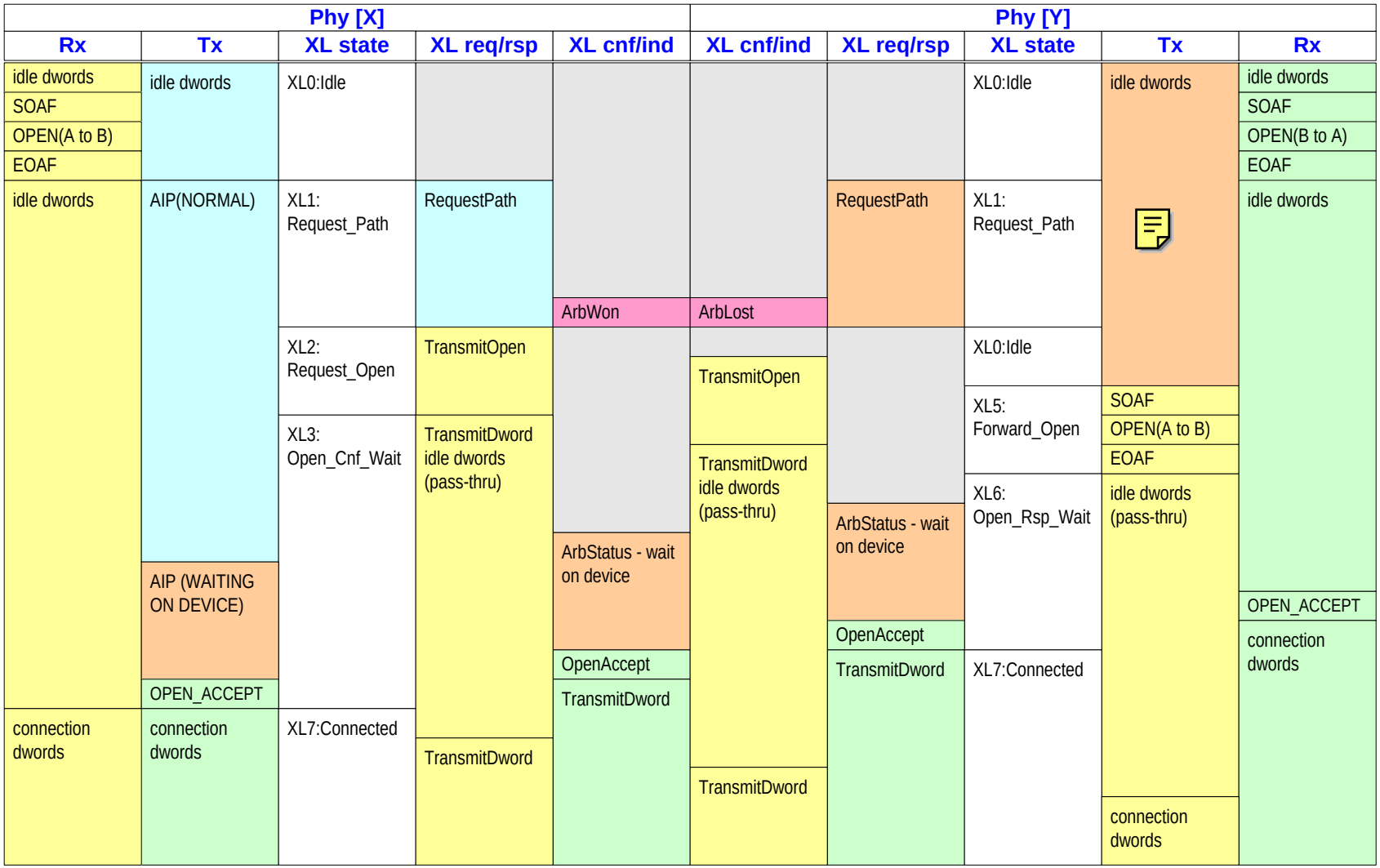


Figure G.5 — Arbitration lost

G.6 Connection request - Backoff and retry

Figure G.6 shows the condition which occurs when a higher priority OPEN address frame (B to C) is received by a phy which has previously forwarded an OPEN address frame to a different destination (A to B). In this case Phy[X] is required to back off and retry path arbitration.

Figure G.6 — Backoff and retry

G.7 Connection request - Backoff and reverse path

Figure G.7 shows the condition which occurs when a higher priority OPEN address frame (B to A) is received by a phy which has previously forwarded an OPEN address frame to the same destination (A to B). In this case Phy[Y] forwards the higher priority OPEN to Phy[X].

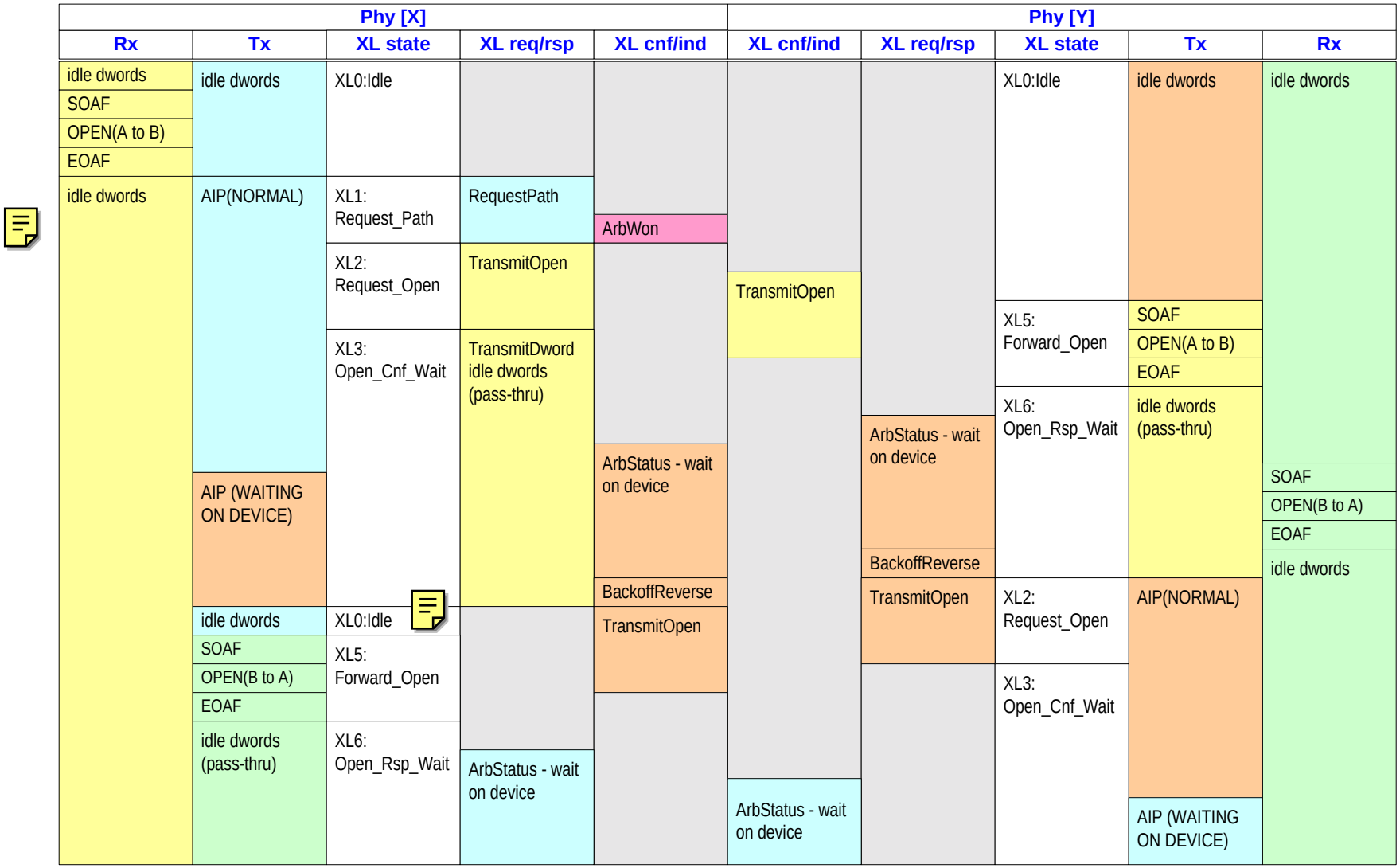


Figure G.7 — Backoff and reverse path

G.8 Connection close - single step

Figure G.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.

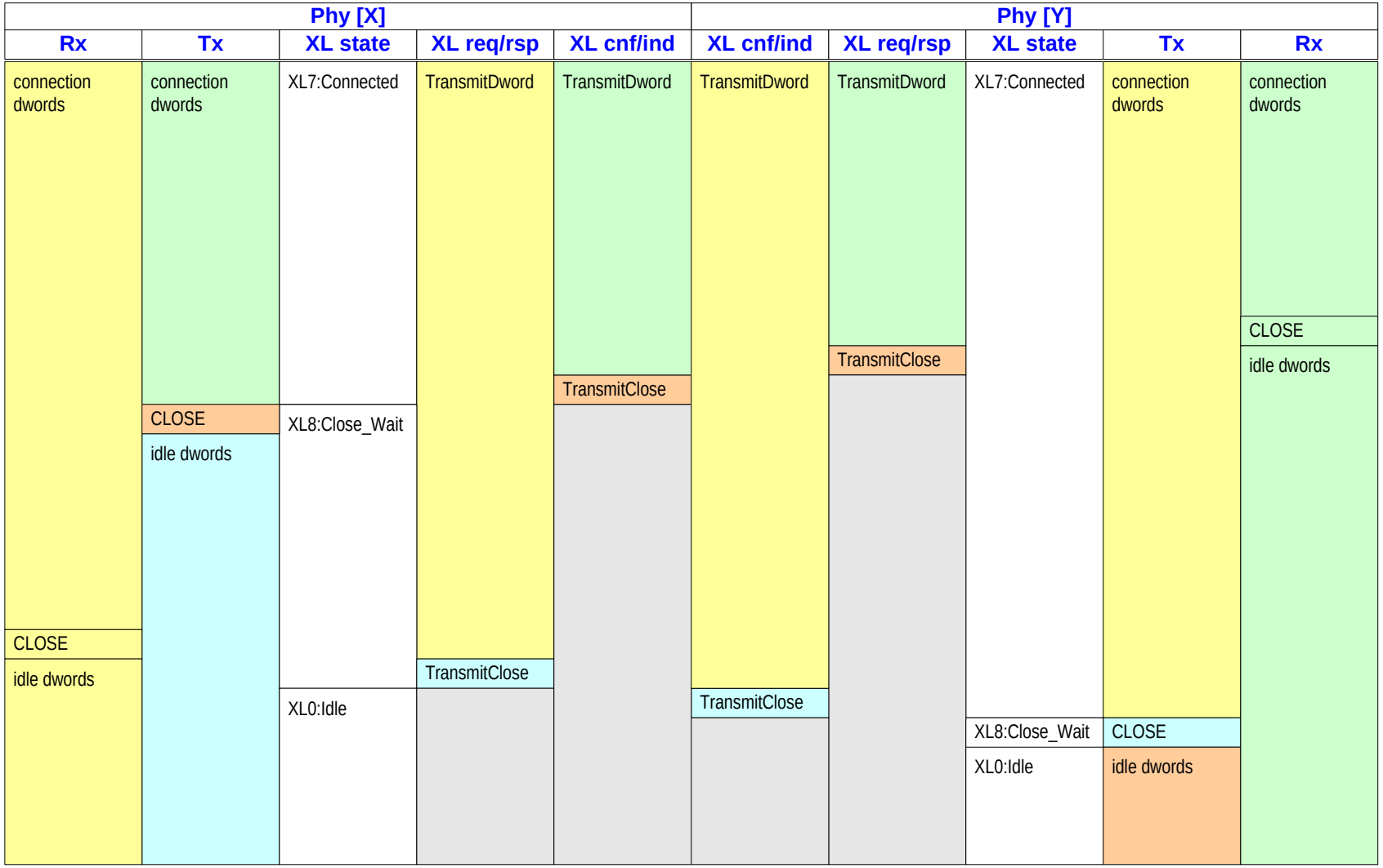


Figure G.8 — Connection close - single step

G.9 Connection close - simultaneous

Figure G.9 shows two end devices simultaneously transmitting CLOSE to each other.

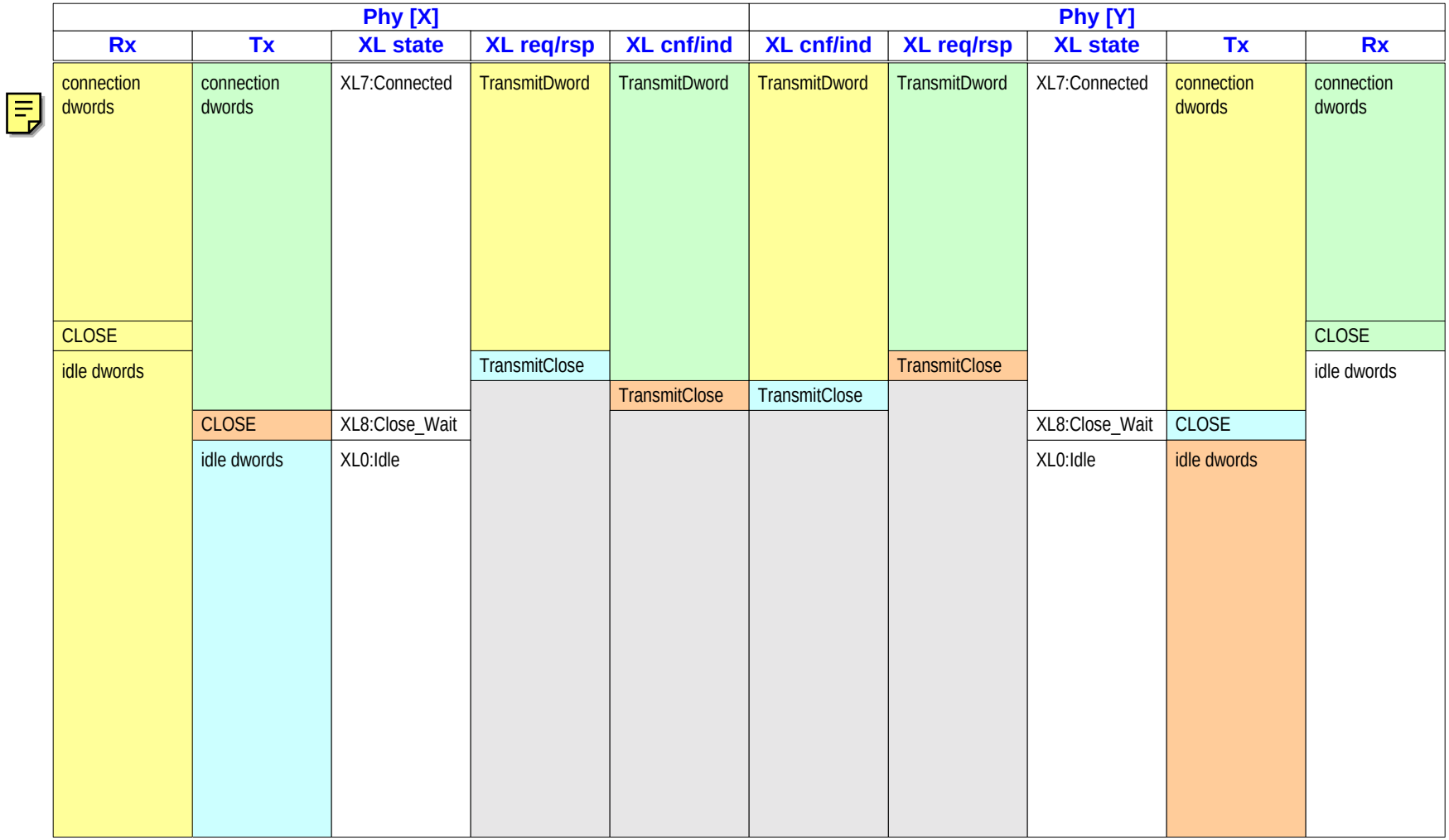


Figure G.9 — Connection close - simultaneous

G.10 Break handling during path arbitration

Figure G.10 shows an expander device responding to the reception of a BREAK during path arbitration.

Phy [X]					Phy [Y]				
Rx	**Tx**	**XL state**	**XL req/rsp**	**XL cnf/ind**	**XL cnf/ind**	**XL req/rsp**	**XL state**	**Tx**	**Rx**
idle dwords SOAF OPEN(A to B) EOAF	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
idle dwords	AIP(NORMAL)	XL1: Request_Path	RequestPath						
BREAK									
idle dwords	BREAK	XL9:Break							
	idle dwords	XL0:Idle							

Figure G.10 — Break handling during path arbitration

G.11 Break handling during connection

Figure G.11 shows an expander device responding to the reception of a BREAK during a connection.

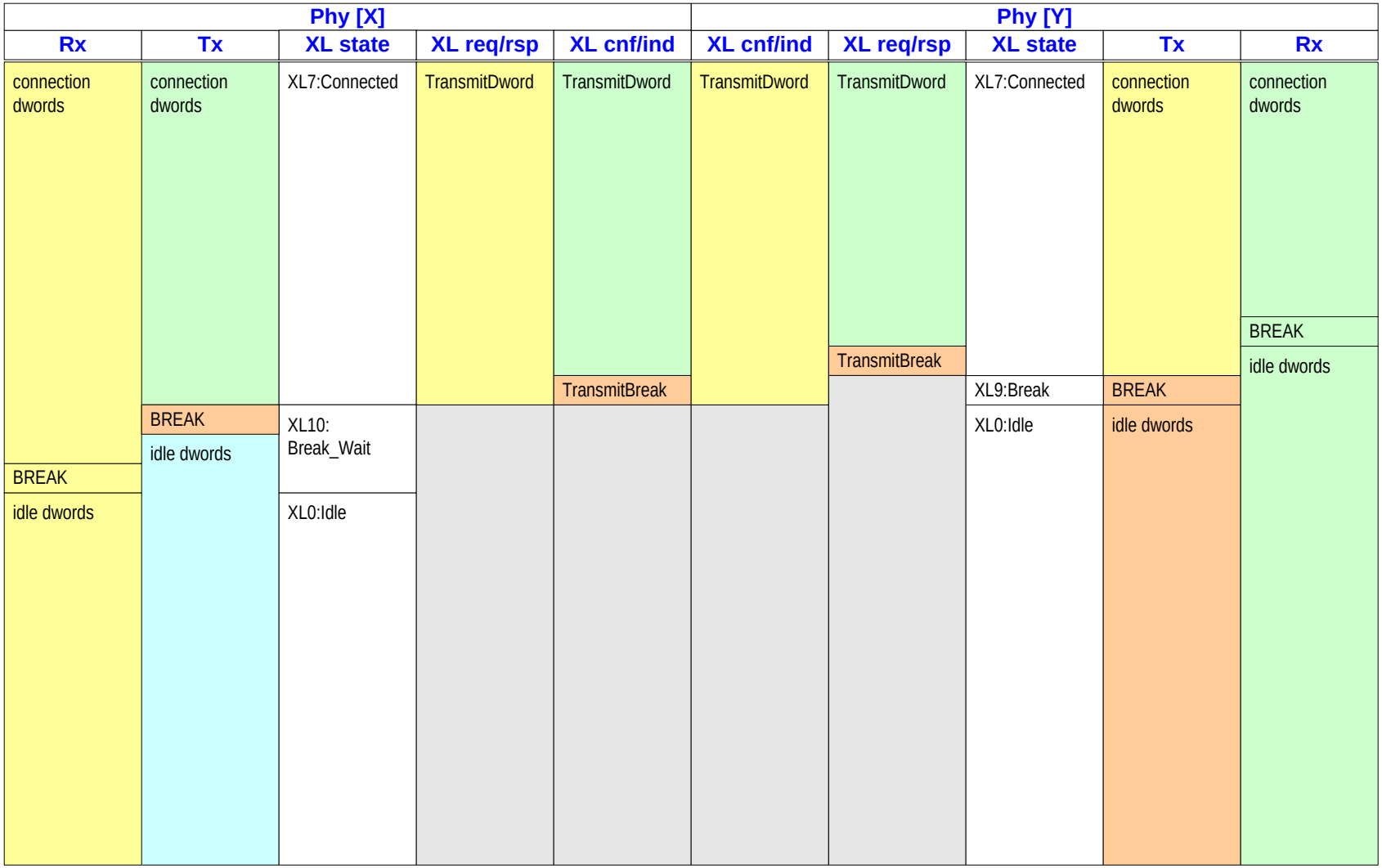
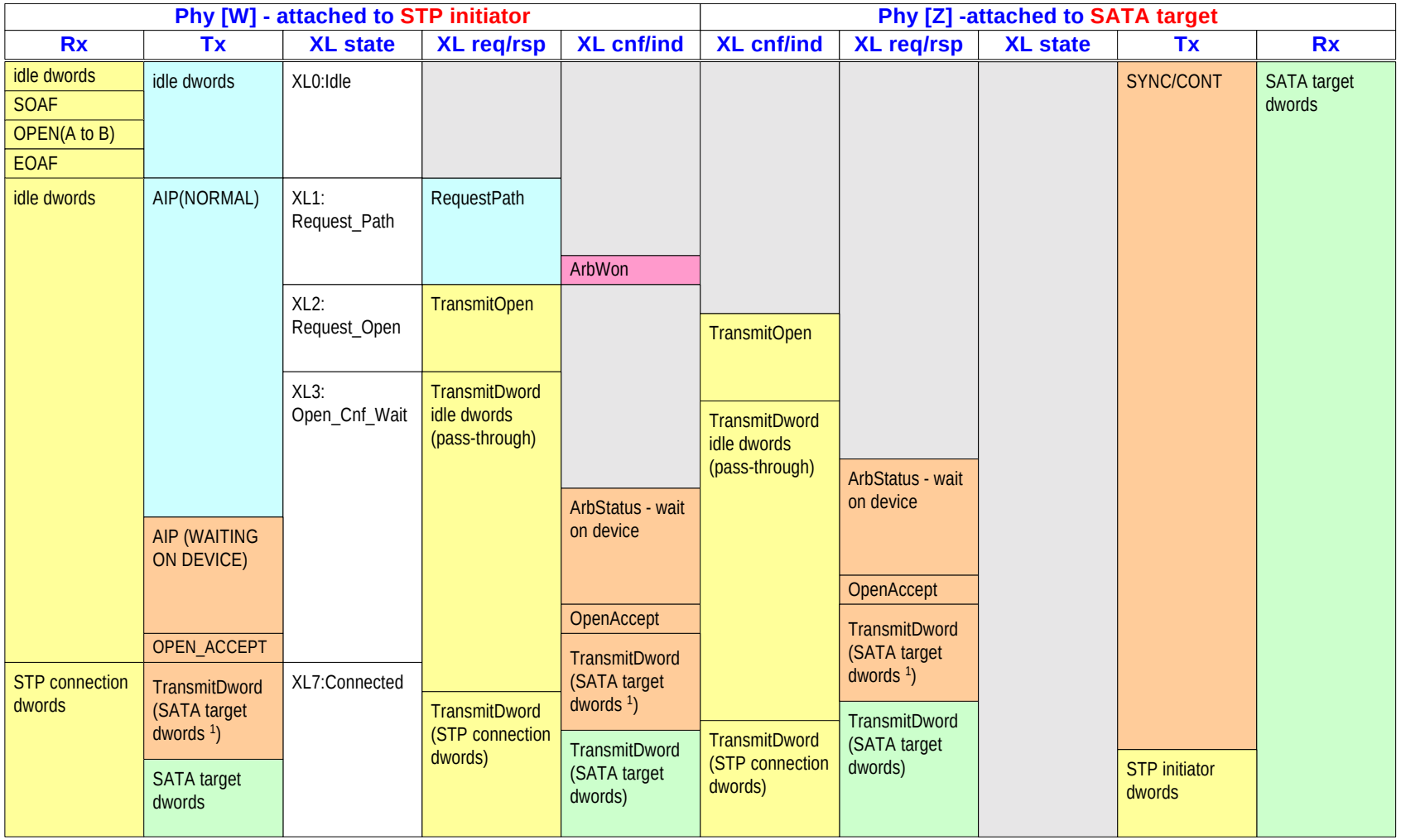


Figure G.11 — Break handling during connection

G.12 STP connection - originated by STP initiator port

Figure G.12 shows an STP initiator originating a connection to a SATA target.

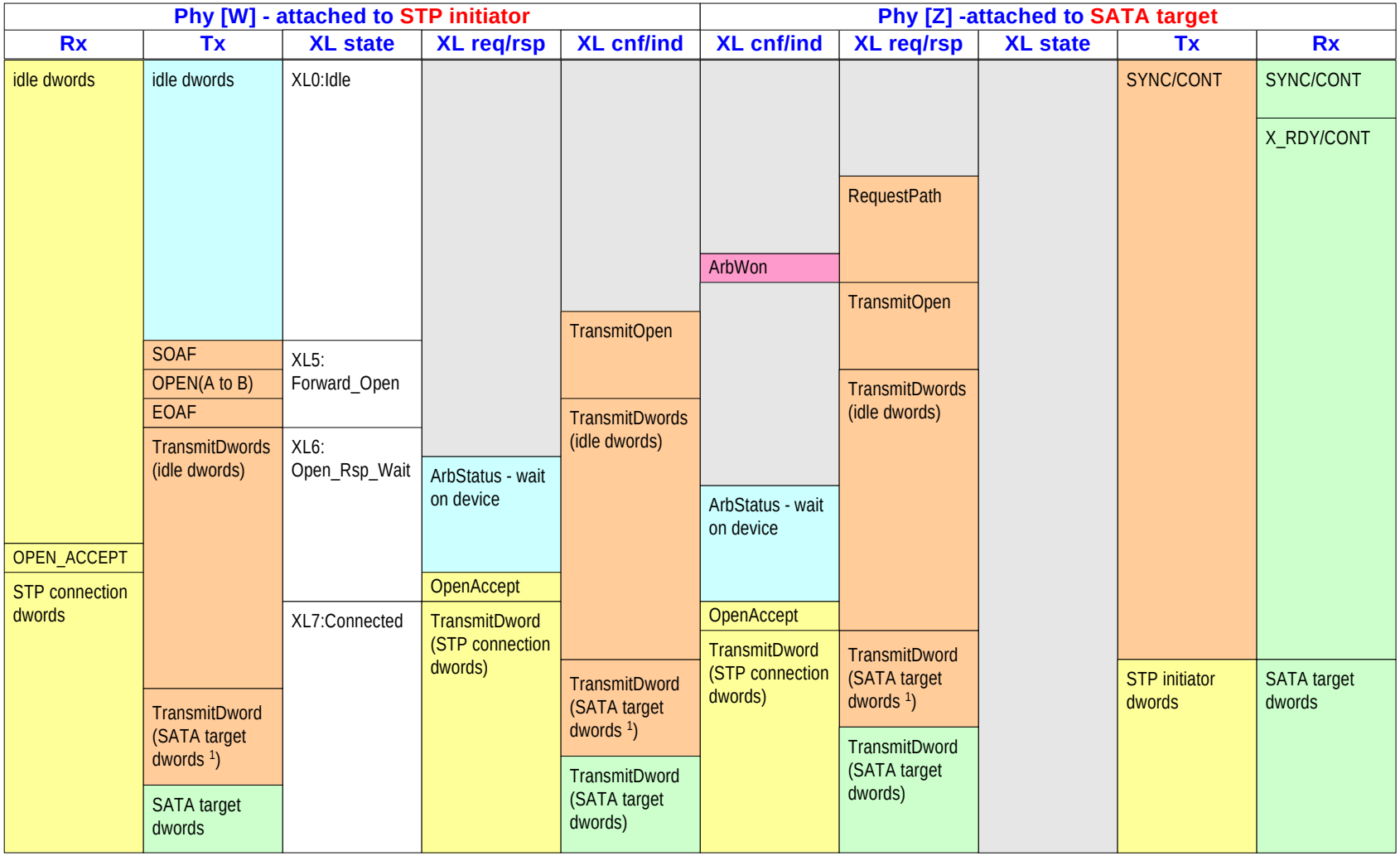


[1] Expander device duplicates the dword stream which is being received from the SATA target port before forwarding dwords - this ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

Figure G.12 — STP connection - originated by STP initiator port

G.13 STP connection - originated by expander device

Figure G.13 shows an expander originating a connection on behalf of a SATA target which is requesting to transmit a frame.



[1] Expander device duplicates the dword stream which is being received from the SATA target port before forwarding dwords - this ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

Figure G.13 — STP connection - originated by expander device

G.14 STP connection close - originated by STP initiator port

Figure G.14 shows an STP initiator closing a connection to a SATA target.

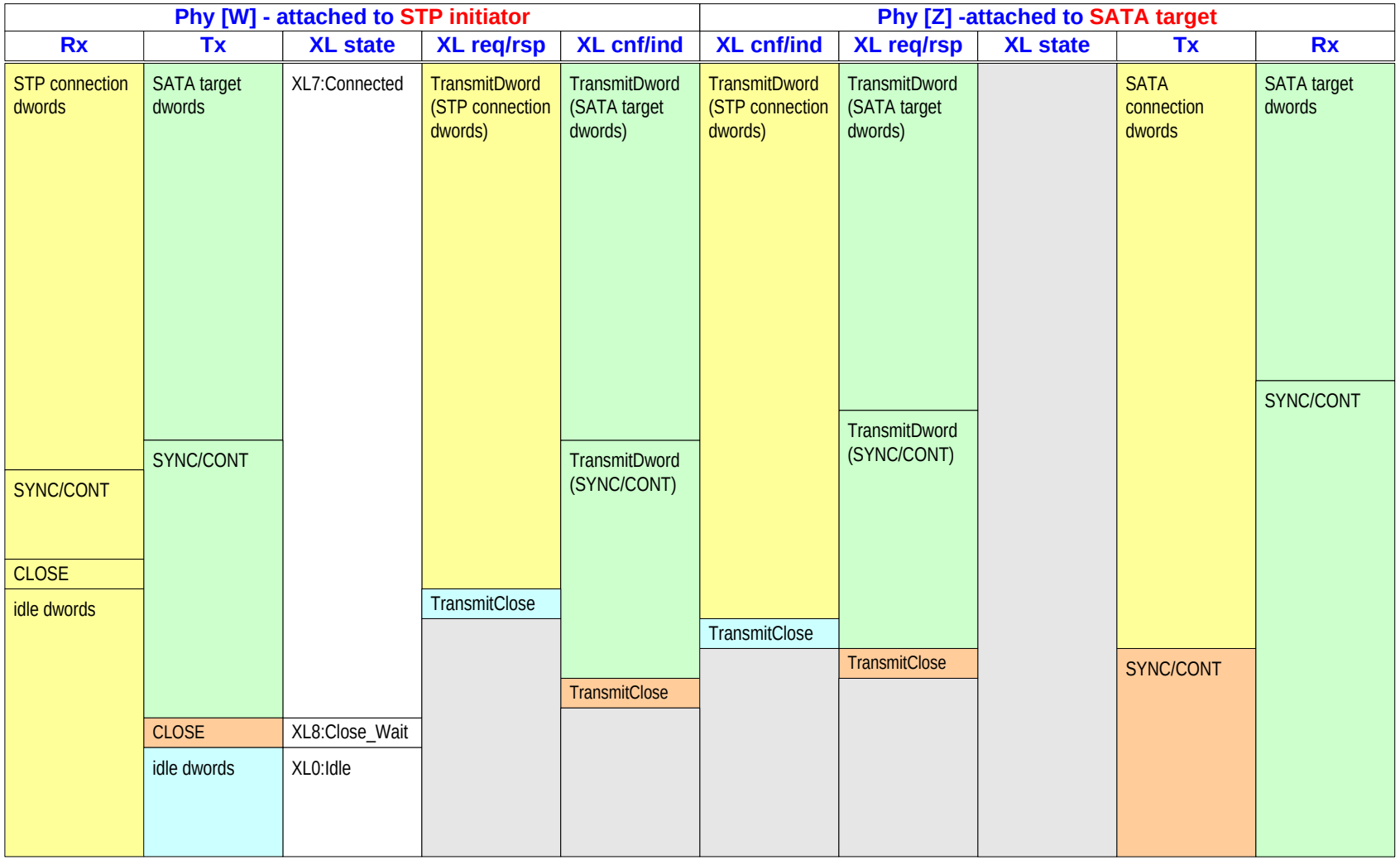


Figure G.14 — STP connection close - originated by STP initiator port

G.15 STP connection close - originated by expander device

Figure G.15 shows an expander closing an STP connection.

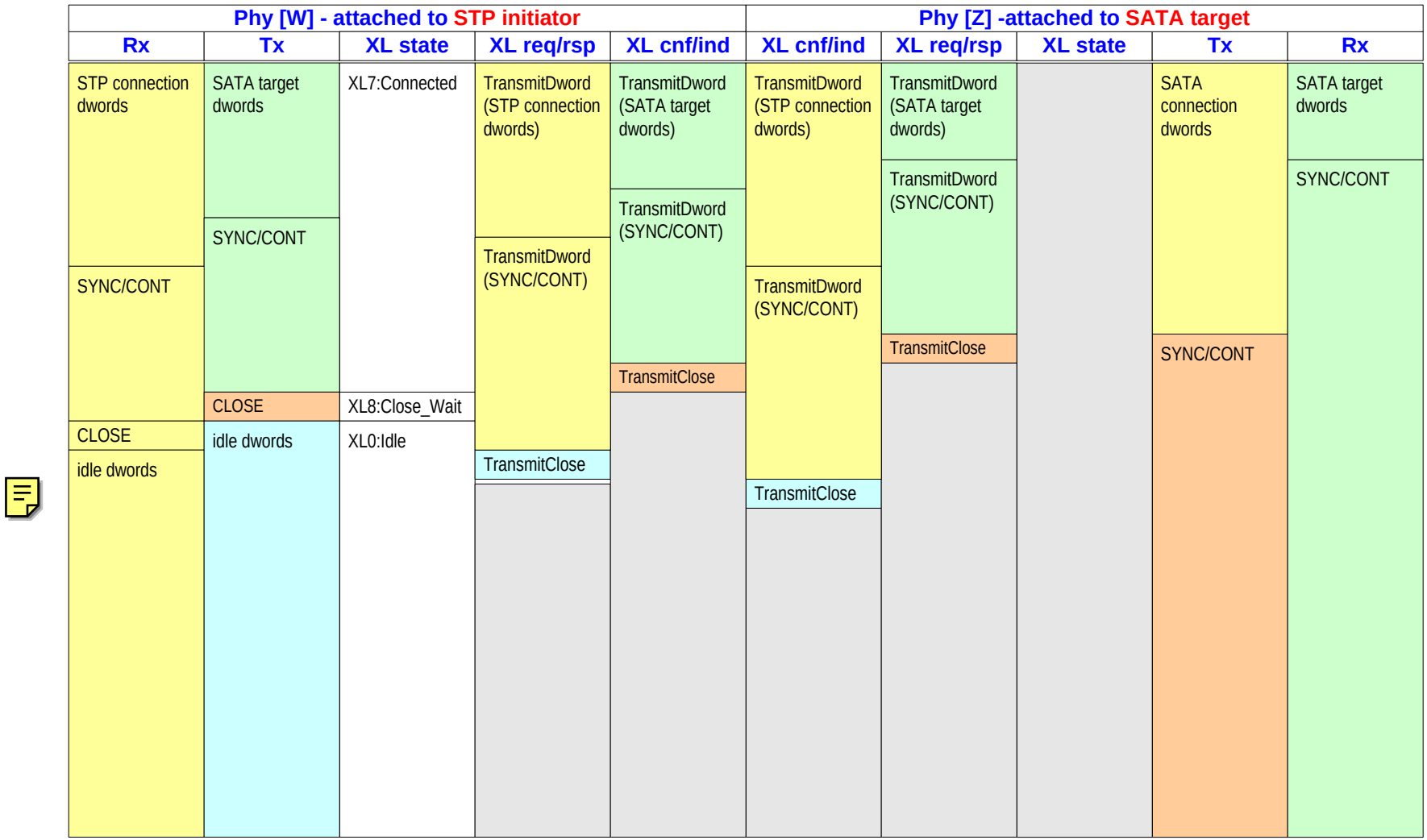


Figure G.15 — STP connection close - originated by expander device

G.16 Pathway blocked and recover example

Figure G.16 shows a topology used to illustrate pathway recovery.

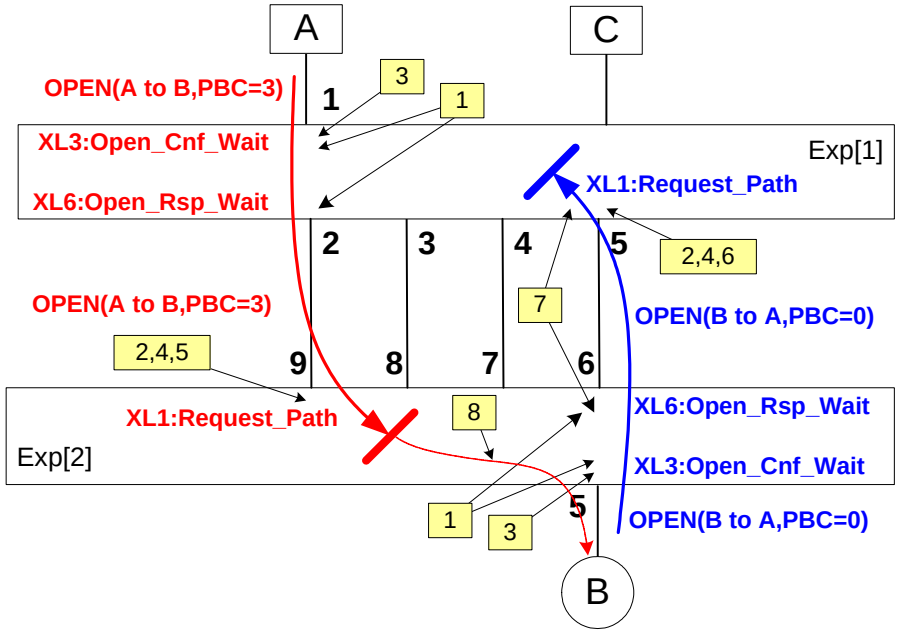


Figure G.16 — Partial pathway recovery

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] send Phy Status (Partial Pathway) to Connection Manager to indicate that they contain partial pathways;
2) Exp[1].Phy[5] and Exp[2].Phy[9] receive Arbitrating (Waiting On Partial) from Connection Manager which cause them to transmit AIP (WAITING ON PARTIAL);
3) AIP(WAITING ON PARTIAL) received by Exp[1].Phy[2] and Exp[2].Phy[6] then forwarded to Exp[1].Phy[1] and Exp[2].Phy[5] as Arb Status (Waiting On Partial). Exp[1].Phy[1] and Exp[2].Phy[5] send Phy Status (Blocked On Partial) to Connection Manager as confirmation that they are blocked waiting on a partial pathway in another expander;
4) Exp[1].Phy[5] and Exp[2].Phy[9] receive Arbitrating (Blocked On Partial) from Connection Manager while all destination phys contain a Phy Status of Blocked On Partial which cause them to run their Partial Pathway Timeout timers;
5) PPT expires in Exp[2].Phy[9] – this causes a request to the Connection Manager to resolve pathway blockage. Pathway recovery priority for this phy is not lower than all phys within the destination port which are also blocked. Connection Manager does not provide Arb Reject (Pathway Blocked) to Exp[2].Phy[9] so this phy waits for pathway resolution to occur elsewhere in the topology;
6) PPT expires in Exp[1].Phy[5] – this causes a request to the Connection Manager to resolve pathway blockage. Pathway recovery priority for this phy is lower than all phys within the destination port which are also blocked. Connection Manager provides Arb Reject (Pathway Blocked) to Exp[1].Phy[5] which instructs this phy to reject the connection request using OPEN_REJECT (PATHWAY BLOCKED);
7) OPEN_REJECT (PATHWAY BLOCKED) tears down partial pathway all the way to the originating end device (Device B); and
8) Partial pathway established between Exp[2].Phy[9] and Exp[2].Phy[5] and OPEN(A to B) is delivered to device B.

Annex H
(informative)

Primitive encoding

H.1 Overview

This annex describes a set of the K28.5-based primitive encodings whose 40-bit values (after 8b10b encoding with either starting disparity) have a Hamming distance (the number of bits different in two patterns) of at least 8. All the primitive encodings in 7.1 were selected from this list. Unassigned encodings may be used by future versions of this standard.

Table H.1 — Primitives with Hamming distance of 8 (part 1 of 3)

1st	2nd	3rd	4th	Assignment
K28.5	D01.3	D01.3	D01.3	ALIGN (3)
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	
K28.5	D02.0	D24.0	D07.3	BREAK
K28.5	D02.0	D29.7	D16.7	
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 (RESERVED 1)
K28.5	D04.7	D02.0	D01.4	BROADCAST (CHANGE)
K28.5	D04.7	D04.7	D04.7	
K28.5	D04.7	D07.3	D29.7	BROADCAST (RESERVED 0)
K28.5	D04.7	D16.7	D02.0	
K28.5	D04.7	D24.0	D31.4	BROADCAST (RESERVED CHANGE)
K28.5	D04.7	D27.4	D07.3	
K28.5	D04.7	D29.7	D30.0	
K28.5	D04.7	D31.4	D27.4	
K28.5	D07.0	D07.0	D07.0	ALIGN (1)

Table H.1 — Primitives with Hamming distance of 8 (part 2 of 3)

1st	2nd	3rd	4th	Assignment
K28.5	D07.3	D01.4	D31.4	
K28.5	D07.3	D02.0	D04.7	
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)

Table H.1 — Primitives with Hamming distance of 8 (part 3 of 3)

1st	2nd	3rd	4th	Assignment
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)
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 (RESERVED 0)
K28.5	D31.3	D27.3	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 (RESERVED ABANDON 0)
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)

Annex I
(informative)

Discover process example implementation

I.1 Overview

This annex includes a C program implementing the discover process. Table I.1 describes the source files.

Table I.1 — C program files

Filename	Description
SASDiscoverSimulation.h	header file
SASDiscoverSimulation.cpp	C source file

I.2 Header file

The following is the C header file for the discover process.

```
// SASDiscoverSimulation.h
// assume the maximum number of phys in an expander device is 16
#define MAXIMUM_EXPANDER_PHYS          16
#define MAXIMUM_EXPANDER_INDEXES       64
#define MAXIMUM_INITIATORS             8

#define SMP_REQUEST_FRAME              0x40
#define SMP_RESPONSE_FRAME             0x41

#define REPORT_GENERAL                 0x00
#define DISCOVER                       0x10
#define CONFIGURE_ROUTE_INFORMATION    0x80

#define SMP_REQUEST_ACCEPTED           0x00
#define SMP_UNKNOWN_FUNCTION           0x01
#define SMP_REQUEST_FAILED             0x02

// DeviceTypes
enum DeviceTypes
{
   END = 0,
   EDGE,
   FANOUT
};

// RoutingAttribute
enum RoutingAttribute
{
   DIRECT = 0,
   SUBTRACTIVE,
   TABLE
};

// RouteFlag
enum DisableRouteEntry
{
   ENABLED = 0,
```

```
   DISABLED
};

// PhyLinkRate(s)
enum PhysicalLinkRate
{
   RATE_UNKNOWN = 0,
   PHY_DOES_NOT_EXIST,
   PHY_DISABLED,
   PHY_FAILED,
   SPINUP_HOLD_OOB,
   GBPS_1_5,
   GBPS_3_0
};

// provide the simple type definitions
typedef unsigned char byte;
typedef unsigned word;
typedef unsigned long dword;
typedef unsigned _int64 quadword;

// defines for the protocol bits
#define SATA   0x01
#define SMP    0x02
#define STP    0x04
#define SSP    0x08

// 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 OOB, for this
// code it contains the identity information for the attached device
// and the initiator application client
struct Identify
{
   // byte 0
   byte AddressFrame:4;       // for an identify frame the value is 0
   byte DeviceType:3;
   byte IgnoredByte0Bit7:1;

   // byte 1
   byte IgnoredByte1;

   // byte 2
   union
   {
      struct
      {
         byte ReservedByte2Bit0:1;
         byte SMPInitiator:1;
         byte STPInitiator:1;
         byte SSPInitiator:1;
         byte ReservedByte2Bit4_7:4;
      };
      byte InitiatorBits;
   };
```

```
      // byte 3
      union
      {
         struct
         {
            byte IgnoredByte3Bit0:1;
            byte SMPTarget:1;
            byte STPTarget:1;
            byte SSPTarget:1;
            byte ReservedByte3Bit4_7:4;
         };
         byte TargetBits;
      };

      // byte 4-11
      byte IgnoredByte4_11[8];

      // byte 12-19
      quadword SASAddress;

      // byte 20-21
      byte IgnoredByte20_21[2];

      // byte 22-27
      byte ReservedByte22_27[8];

      // byte 28-31
      dword CRC[4];
   };

   // request specific bytes for SMP Report General function
   struct SMPRequestReportGeneral
   {
      // byte 4-7
      byte CRC[4];
   };

   // request specific bytes for SMP Discover function
   struct SMPRequestDiscover
   {
      // 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
      byte CRC[4];
```

```
};

// the ConfigureRouteInformation structure is used to provide the
// expander route entry for the expander route table, it is intended
// to be referenced by SMP RequestConfigureRouteInformation
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];

   // byte 40-43
   dword CRC[4];
};

// request specific bytes for SMP ConfigureRouteInformation function
struct SMPRequestConfigureRouteInformation
{
   // byte 4-7
   byte ReservedByte4_5[2];

   // byte 6-7
   word ExpanderRouteIndex;

   // byte 8
   byte ReservedByte8;

   // byte 9
   byte PhyIdentifier;

   // byte 10
   byte IgnoredByte10;

   // byte 11
   byte ReservedByte11;

   // byte 12-15
   struct ConfigureRouteInformation Configure;
};

// generic structure referencing an SMP Request, must be initialized
// before being used
```

```
struct SMPRequest
{
   // byte 0
   byte SMPFrameType;                  // always 40h for SMP Frame request

   // byte 1
   byte Function;                      // 00h ReportGeneral
                                       // 10h Discover
                                       // 80h ConfigureRouteInformation

   // byte 2-3
   byte ReservedByte2_3[2];

   // bytes 4-n
   union
   {
      struct SMPRequestDiscover Discover;
      struct SMPRequestReportGeneral ReportGeneral;
      struct SMPRequestConfigureRouteInformation ConfigureRouteInformation;

// remaining structures would be necessary for a complete implementation
// for this example however they are comments only
//
//      struct SMPRequestSATACapabilities SATACapabilities;
//      struct SMPRequestReportManfacturerInformation
ManufacturerInformation;
//      struct SMPRequestReportRouteInformation RouteInformation;
//      struct SMPRequestReportPhyErrorLog PhyErrorLog;
//      struct SMPRequestReportPhySATA PhySATA;
//      struct SMPRequestPhyControl PhyControl;
//      struct SMPRequestPhyMarginControl PhyMarginControl;

   } Request;
};

// request specific bytes for SMP Report General response, intended to be
// referenced by SMPResponse
struct SMPResponseReportGeneral
{
   // byte 4-5
   byte ReservedByte4_5;

   // byte 6-7
   word ExpanderRouteIndexes;

   // byte 8
   byte ReservedByte8;

   // byte 9
   byte NumberOfPhys;

   // byte 10
   byte ConfigurableRouteTable:1;
   byte ReservedByte10Bit1_7:7;

   // byte 11
   byte ReservedByte11;
```

```
      // byte 12-15
      byte CRC[4];
   };


   // 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 RoutingAttribute:4;
      byte AttachedDeviceType:3;
      byte RouteEntryDisabled:1;

      // byte 13
      byte PhysicalLinkRate:4;
      byte ReservedByte13Bit4_7:4;

      // byte 14
      union
      {
         struct
         {
            byte ReservedByte2Bit0:1;
            byte AttachedSMPInitiator:1;
            byte AttachedSTPInitiator:1;
            byte AttachedSSPInitiator:1;
            byte ReservedByte14Bit4_7:4;
         };
         byte InitiatorBits;
      };

      // byte 15
      union
      {
         struct
         {
            byte AttachedSATATarget:1;
            byte AttachedSMPTarget:1;
            byte AttachedSTPTarget:1;
            byte AttachedSSPTarget:1;
            byte ReservedByte15Bit4_7:4;
         };
         byte TargetBits;
      };

      // byte 16-23
      quadword AttachedSASAddress;

      // byte 24-31
      quadword SASAddress;

      // byte 32
      byte HardwareMinimumPhysicalLinkRate:4;
      byte ProgrammedMinimumPhysicalLinkRate:4;
```

```
      // byte 33
      byte HardwareMaximumPhysicalLinkRate:4;
      byte ProgrammedMaximumPhysicalLinkRate:4;

      // byte 34-35
      byte VendorSpecific[2];

      // byte 36-39
      byte ReservedByte36_39[4];

      // byte 40-43
      dword CRC[4];
   };

   // 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 10-11
      byte ReservedByte10_11;

      // byte 12-43
      struct Discover Results;
   };

   // response specific bytes for SMP Configure Route Information,
   // intended to be referenced by SMPResponse
   struct SMPResponseConfigureRouteInformation
   {
      // byte 4-7
      byte CRC[4];
   };

   // 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 SMPResponseDiscover Discover;
         struct SMPResponseReportGeneral ReportGeneral;
         struct SMPResponseConfigureRouteInformation ConfigureRouteInformation;

   // remaining structures would be necessary for a complete implementation
   // for this example however they are comments only
   //
   //      struct SMPRequestSATACapabilities SATACapabilities;
   //      struct SMPRequestReportManfacturerInformation
   ManufacturerInformation;
   //      struct SMPRequestReportRouteInformation RouteInformation;
   //      struct SMPRequestReportPhyErrorLog PhyErrorLog;
   //      struct SMPRequestReportPhySATA PhySATA;
   //      struct SMPRequestPhyControl PhyControl;
   //      struct SMPRequestPhyMarginControl PhyMarginControl;

      } Response;
   }; // SMPResponse

   // this structure is how this simulation obtains it's knowledge about the
   // initiator port that is doing the discover, it is not defined as part of
   // the standard...
   // the simulation assumes that whatever is marked as the initiator in the
   // ini file will perform the discover process on all the phys of the device
   struct ApplicationClientKnowledge
   {
      quadword SASAddress;
      byte NumberOfPhys;
      byte InitiatorBits;
      byte TargetBits;
   };

   // 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 explictly
   struct TopologyTable
   {
      struct TopologyTable *Next;

      // information from REPORT_GENERAL
      struct SMPResponseReportGeneral Device;

      // information from DISCOVER
      struct SMPResponseDiscover Phy[MAXIMUM_EXPANDER_PHYS];

      // route index for the configuration chain functions
      word RouteIndex[MAXIMUM_EXPANDER_PHYS];

      //
      // 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...
```

```
    //
}; // TopologyTable
```

I.3 Source file

The following is the C header file for the discover process.

```
// SASDiscoverSimulation.cpp
// this is a simple simulation and code implementation of
// the initiator based expander discovery and configuration process

// there is no attempt to handle phy errors, arbitration issues, etc.
// production level implementation would have to handle errors appropriately

// basic assumptions
//
// 1. change primitives will initate a rediscovery/configuration sequence
// 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 intiator and the end devices is increased
// 4. configuration of a single expander occurs before proceeding to
subsequent
//    expanders attached
// 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 “SASDiscoverSimulation.h”

#define ADDRESS_CHAIN_LEVELS  16
#define MAXIMUM_ROUTE_ENTRIES (64*64)

// loaded by the application client, in this simulation it is provided
// in a text file, SASDomainExample.ini
struct ApplicationClientKnowledge Iam[MAXIMUM_INITIATORS] = { 0 };

// obtained following OOB from the attached phy, in this simulation
// it is provided in a text file
struct Identify Attached[MAXIMUM_INITIATORS] = { 0 };

// buffers used to request and return SMP data
struct SMPRequest SMPRequestFrame = { 0 };
struct SMPResponse SMPResponseFrame = { 0 };

// resulting discover information will end up in this table
struct TopologyTable *SASDomain[MAXIMUM_INITIATORS] = { 0 };

// 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 Function,
                        ...);

// this is the configuration chain it contains the list of expander between
// the programming initiator and the discover extent, each expander in the
// chain must be configured when a new SASaddress is encounter from the top
// of the chain to the bottom
static struct ConfigurationChain
{
   quadword SASAddress;
   struct TopologyTable *Expander;
   byte PhyIdentifier;
} TableChain[ADDRESS_CHAIN_LEVELS] = { 0 };

static int ChainIndex = 0;
static int ChainEntry = 0;

// this is the upstream chain and contains the list of expanders that have
// already been traversed upstream, this reduces the number of times we
// go through the configuration
static quadword UpstreamChain[ADDRESS_CHAIN_LEVELS] = { 0 };

// this is the route entry chain. It contains the list of route entries that
// have already been configured, this reduces duplication of configuration
// requests
static struct RouteEntryChain
{
   quadword SASAddress;
   word RouteIndex;
   byte PhyIdentifier;
} ConfiguredChain[MAXIMUM_ROUTE_ENTRIES] = { 0 };

// this function gets the report general and discover information for
// a specific expander
struct TopologyTable *DiscoverExpander(byte PhyIdentifier,
                                       quadword SourceSASAddress,
                                       quadword DestinationSASAddress,
                                       struct TopologyTable *Expander)
{
   struct TopologyTable *expander;
   byte phyCount;
   int error = 1;

   // 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)
   {
      // get the report general information for the expander
      SMPRequest(PhyIdentifier,
                 SourceSASAddress,
                 DestinationSASAddress,
```

```
                &SMPRequestFrame,
                &SMPResponseFrame,
                REPORT_GENERAL);

            // don't worry about too much in the 'else' case for this example,
            // production code must handle
            if(SMPResponseFrame.FunctionResult == SMP_REQUEST_ACCEPTED)
            {
               // copy the result into the topology table
               memcpy((void *)&(expander->Device),
                      (void *)&SMPResponseFrame.Response.ReportGeneral,
                      sizeof(struct SMPResponseReportGeneral));

               if(expander->Device.NumberOfPhys <= MAXIMUM_EXPANDER_PHYS)
               {
                  // 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,
                                DISCOVER,
                                phyCount);

                     // don't worry about the 'else' case for this example,
                     // production code must handle
                     if(SMPResponseFrame.FunctionResult == SMP_REQUEST_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
                     {
                        // something happened so just bailout on this expander
                        error = 1;

                        // release the memory we allocated for this...
                        free(expander);
                        break;
                     }
                  }
               }
            }

            // if we did not have an error, then link in the information from this
            // expander into the topology table
            if(!error &&
```

```
            Expander)
         {
            Expander->Next = expander;
         }
      }

      // the expander pointer is the error return, a null indicates something
      // bad happened...
      return(expander);
   } // DiscoverExpander

   // this routine will add a SASAddress to the configuration chain
   void PushSASAddress(quadword SASAddress,
                       struct TopologyTable *Expander,
                       byte PhyIdentifier)
   {
      TableChain[ChainIndex].SASAddress = SASAddress;
      TableChain[ChainIndex].Expander = Expander;
      TableChain[ChainIndex].PhyIdentifier = PhyIdentifier;

      if(ChainIndex < ADDRESS_CHAIN_LEVELS)
      {
         ChainIndex++;
      }
   }

   // this routine will remove a SASAddress from the configuration chain
   void PopSASAddress(void)
   {
      if(ChainIndex)
      {
         ChainIndex--;
      }

      TableChain[ChainIndex].SASAddress = 0;
      TableChain[ChainIndex].Expander = 0;
      TableChain[ChainIndex].PhyIdentifier = 0;
   }

   // this routine will reset the ChainEntry and return the first entry on
   // the Chain
   quadword FirstSASAddress(void)
   {
      ChainEntry = 0;

      return(TableChain[ChainEntry].SASAddress);
   }

   // this routine will return the next entry on the Chain
   quadword NextSASAddress(void)
   {
      if(ChainEntry < ChainIndex)
      {
         ChainEntry++;
      }

      return(TableChain[ChainEntry].SASAddress);
```

```
}

// this routine will get the route index from the expander structure
word IndexForSASAddress(void)
{
   struct TopologyTable *expander = TableChain[ChainEntry].Expander;
   word index = 0;

   index = expander->RouteIndex[TableChain[ChainEntry].PhyIdentifier]++;

   return(index);
}

// this routine will get the phy identifier from the configuration chain
byte PhyForSASAddress(void)
{
   return(TableChain[ChainEntry].PhyIdentifier);
}

// this routine keeps track of which SASAddresses have already been
// traversed upstream and avoid duplicate configuration passes
byte AlreadyTraversed(quadword SASAddress)
{
   byte chain = 0;
   byte itsHere = 0;

   while(UpstreamChain[chain])
   {
      if(UpstreamChain[chain] == SASAddress)
      {
         itsHere = 1;
         break;
      }

      if(chain < ADDRESS_CHAIN_LEVELS)
      {
         chain++;
      }
   }

   UpstreamChain[chain] = SASAddress;

   return(itsHere);
} // AlreadyTraversed

// this routine keeps track of which expanders have been configured and
// avoids duplicate configuration requests
byte AlreadyConfigured(quadword SASAddress,
                       word RouteIndex,
                       byte PhyIdentifier)
{
   word chain = 0;
   byte itsHere = 0;

   while(ConfiguredChain[chain].SASAddress)
   {
      if((ConfiguredChain[chain].SASAddress == SASAddress) &&
```

```
            (ConfiguredChain[chain].RouteIndex == RouteIndex) &&
            (ConfiguredChain[chain].PhyIdentifier == PhyIdentifier))
         {
            itsHere = 1;
            break;
         }

         if(chain < MAXIMUM_ROUTE_ENTRIES)
         {
            chain++;
         }
      }

      ConfiguredChain[chain].SASAddress = SASAddress;
      ConfiguredChain[chain].RouteIndex = RouteIndex;
      ConfiguredChain[chain].PhyIdentifier = PhyIdentifier;

      return(itsHere);
   } // AlreadyConfigured

   // this function is recursive and discovers then configures as necessary
   // the expanders it finds that are “downstream”
   struct TopologyTable *DiscoverAndConfigure(byte PhyIdentifier,
                                              quadword SourceSASAddress,
                                              struct TopologyTable *Start,
                                              byte Upstream)
   {
      struct TopologyTable *expander = Start;
      struct TopologyTable *nextExpander;
      struct TopologyTable *attachedDeviceSet = 0;

      struct SMPResponseDiscover *discover;
      struct SMPResponseDiscover *configure;

      quadword sasAddress;
      byte phyCount;
      word indexOffset;
      byte phyOffset;

      // if the upstream flag is set, then we’re moving upstream looking for
      // the boundary of the device set
      if(Upstream)
      {
         // walk through all the phys of the expander
         for(phyCount = 0;
             (phyCount < expander->Device.NumberOfPhys);
             phyCount++)
         {
            // this is just a pointer helper
            discover = &(expander->Phy[phyCount]);

            // look for phys with edge or fanout devices attached...
            if(((discover->Results.RoutingAttribute == SUBTRACTIVE) &&
                (discover->Results.AttachedDeviceType == EDGE)) ||
               (discover->Results.AttachedDeviceType == FANOUT))
            {
               if(!AlreadyTraversed(discover->Results.AttachedSASAddress))
```

```
            {
               nextExpander
               = DiscoverExpander(PhyIdentifier,
                                  SourceSASAddress,
                                  discover->Results.AttachedSASAddress,
                                  expander);

               // if we successfully got the information from the next
expander
               // then link it in the topology table...
               if(nextExpander)
               {
                  // if the attached device has a subtractive phy, then
                  // stop going upstream, we have two expander device sets
                  // connected without a fanout expander, save the address
                  // of the attachedDeviceSet, and do the configuration of
                  // it as a “separate domain”...
                  if(nextExpander->Phy[phyCount].Results.RoutingAttribute
                     == SUBTRACTIVE)
                  {
                     attachedDeviceSet = nextExpander;
                  }
                  // the attached device does not have a subtractive phy, so
                  // continue to move upstream until we hit the edge of the
                  // device set...
                  else
                  {
                     // go upstream to the next expander
                     attachedDeviceSet = DiscoverAndConfigure(PhyIdentifier,
                                                        SourceSASAddress,
                                                           nextExpander,
                                                           1);

                     expander->Next = nextExpander;
                  }
               }
            }
         }
      }
   }

   // this is a maximal decent traversal with a configuration stage
   // at each transition to a new level, if a configuration is required
   // by the expander

   // walk through all the phys of the expander looking for table routes
   // only, we’ll do configuration as we find them...
   for(phyCount = 0;
       (phyCount < expander->Device.NumberOfPhys);
       phyCount++)
   {
      // this is just a pointer helper
      discover = &(expander->Phy[phyCount]);

      // look for phys with edge or fanout devices attached...
      if((discover->Results.RoutingAttribute == TABLE) &&
         ((discover->Results.AttachedDeviceType == EDGE) ||
```

```
        (discover->Results.AttachedDeviceType == FANOUT)))
    {
       // add the expander to the configuration chain
       PushSASAddress(discover->Results.SASAddress,
                      expander,
                      phyCount);

       nextExpander = DiscoverExpander(PhyIdentifier,
                                       SourceSASAddress,
                                     discover->Results.AttachedSASAddress,
                                       expander);

       // if we successfully got the information from the next expander
       // then link it in the topology table...
       if(nextExpander)
       {
          byte nextPhyCount = 0;

          // loop through the phys of the nextExpander, gathering
          // the SASAddresses for this expander
          for(nextPhyCount = 0;
              (nextPhyCount < nextExpander->Device.NumberOfPhys);
              nextPhyCount++)
          {
             // configure the expanders between the initiator and
             // the destination
             for(sasAddress = FirstSASAddress();
                 sasAddress;
                 sasAddress = NextSASAddress())
             {
                word routeIndex = IndexForSASAddress();
                byte phyIdentifier = PhyForSASAddress();

                // this avoids duplicate configure cycles as we walk the
                // topology
                if(!AlreadyConfigured(sasAddress,
                                      routeIndex,
                                      phyIdentifier))
                {
                   struct SMPResponseDiscover *nextDiscover;

                   // this is just a helper pointer to reduce line length
                   nextDiscover = &(nextExpander->Phy[nextPhyCount]);

                   // configure the indexes for the nextExpander in the
                   // AttachedExpander
                   SMPRequest(PhyIdentifier,
                              SourceSASAddress,
                              sasAddress,
                              &SMPRequestFrame,
                              &SMPResponseFrame,
                              CONFIGURE_ROUTE_INFORMATION,
                              routeIndex,
                              phyIdentifier,
                              0,
                              nextDiscover->Results.AttachedSASAddress);
                }
```

```
                }
            }

            // descend to the next expander
            DiscoverAndConfigure(PhyIdentifier,
                                 SourceSASAddress,
                                 nextExpander,
                                 0);

            // remove the last expander from the configuration chain
            PopSASAddress();

            expander->Next = nextExpander;
         }
      }
   }

   return(attachedDeviceSet);
} // DiscoverAndConfigure

// this routine will append the leaf to the tree domain
void ConcatenateDomains(struct TopologyTable *Tree,
                        struct TopologyTable *Leaf)
{
   while(Tree)
   {
      if(Tree->Next == 0)
      {
         Tree->Next = Leaf;
         break;
      }

      Tree = Tree->Next;
   }
} // ConcatenateDomains

// this routine clear the state variables that control execution, this allows
// recursion to work, and allows the different initiator passes to work...
void InitializeForRecursion(void)
{
   word chain = 0;

   for(chain = 0;
       chain < MAXIMUM_ROUTE_ENTRIES;
       chain++)
   {
      if(chain < ADDRESS_CHAIN_LEVELS)
      {
         TableChain[chain].SASAddress = 0;
         TableChain[chain].Expander = 0;
         TableChain[chain].PhyIdentifier = 0;

         UpstreamChain[chain] = 0;
      }

      ConfiguredChain[chain].SASAddress = 0;
      ConfiguredChain[chain].RouteIndex = 0;
```

```
         ConfiguredChain[chain].PhyIdentifier = 0;
      }

      ChainIndex = 0;
      ChainEntry = 0;
   } // InitializeForRecursion

   // the application client for the initiator device would make a call to
   // this function to begin the discover process...
   // to simplify the setup for the simulation, the DiscoverProcess will get
   // the Initiator number to allow multiple initiators...
   void DiscoverProcess(byte Initiator)
   {
      // clear the state variables
      InitializeForRecursion();

      // check to see if an expander is attached

      // this example ignores the other phys on initiators with multiple phys,
      // the results would be the same, just the determination of wide ports
      // and independent SASDomains would need to be handled...
      if((Attached[Initiator].DeviceType == EDGE) ||
         (Attached[Initiator].DeviceType == FANOUT))
      {
         // expander is attached, so begin to walk the topology, building
         // the necessary tables for each expander
         SASDomain[Initiator] = DiscoverExpander(0,
                                                 Iam[Initiator].SASAddress,
                                                Attached[Initiator].SASAddress,
                                                 0);
      }

      // make sure we got the information from the attached expander before
   going
      // on...
      if(SASDomain[Initiator])
      {
         struct TopologyTable *attachedDeviceSet;

         // 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 will naturally move upstream
         // due to the subtractive addressing method
         // if during the discover and configuration 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
         attachedDeviceSet = DiscoverAndConfigure(0,
                                                  Iam[Initiator].SASAddress,
                                                  SASDomain[Initiator],
                                                  1);

         // if two device sets are connected, then the attached device set
         // has to be discovered and configured separately, but there is no
         // need to do the upstream check...
         if(attachedDeviceSet)
```

```
          {
             DiscoverAndConfigure(0,
                                  Iam[Initiator].SASAddress,
                                  attachedDeviceSet,
                                  0);

             // put the domains together
             ConcatenateDomains(SASDomain[Initiator],
                                attachedDeviceSet);
          }
       }
    } // DiscoverProcess
```

Annex J
(informative)

SAS logo

The SAS logo is shown in figure J.1. This logo should be included on all connectors to SAS phys.

Figure J.1 — SAS logo