ENDL
T E X A S

SAS-1.1 ISO FCD

T10 Review Draft Only

Not an official ISO/IEC document

This document is a preliminary draft of ISO/IEC SAS-1.1 (ISO/IEC 14776-151). This document is posted for review purposes only … prior to IR submission to JTC1 for ISO/IEC processing. This document is not an official ISO/IEC document.

The only difference between this revision and r0 appears in 5.3.3, and responds to the following comment from George Penokie:

The problem is that following text that occurs in section 5.3.3 General electrical characteristics:

" shielded twinaxial cable with 24 gauge solid wire up to 6 meters in length "

The problem is that the term << gauge >> has and English meaning that equates to AWG (which is in the acronym list) however it has a totally different meaning when used in the metric world and that's where the SAS standards live.

Normally this would have been caught during the ISO review because the wire gauge should have been stated as << 24 AWG >>.

The change that should be made to the ISO version would result in the following wording (which has been testing in may versions of SPI):

" shielded twinaxial cable with 0,205 mm^2 (24 AWG) solid wire up to 6 meters in length "

Also, SAS-2 has the same problem and I will be submitting a late letter ballot to get it fixed.

Final

ISO/IEC

Committee Draft

FCD 14776-151

Information technology - Small Computer System Interface (SCSI) - Part 151: Serial Attached SCSI - 1.1 (SAS-1.1)

PATENT STATEMENT

Contents

Tables

Figures

INFORMATION TECHNOLOGY –
SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 151: Serial attached SCSI-1.1 (SAS-1.1)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IECpublication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14776-151 was prepared by subcommittee 25: Interconnection of Information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14776 series, under the general title *Information technology – Small computer system interface (SCSI),* can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies and the voting results may be obtained from the address given on the second title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

Introduction

This International 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.
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 (Terms, definitions, symbols, abbreviations, keywords and conventions) defines terms and conventions used throughout this standard.

Clause 4 (General) describes architecture, names and identifiers, state machines, resets, I_T nexus loss and provides an expander device model.

Clause 5 (Physical layer) describes the physical layer. It describes passive interconnect components (connectors, cables and backplanes) and defines the transmitter and receiver electrical characteristics.

Clause 6 (Phy layer) describes the phy layer. It describes 8b10b encoding, bit order, out of band (OOB) signals, phy reset sequences, phy layer state machines, and spin-up.

Clause 7 (Link layer) describes the link layer. It describes primitives, clock skew management, idle physical links, CRC, scrambling, address frames, the identification sequence and its state machine, power management, SAS domain changes, connections, rate matching and SSP, STP and SMP connection rules and link layer state machines.

Clause 8 (Port layer) describes the port layer, which sits between one or more link layers and one or more transport layers. It includes port layer state machines.

Clause 9 (Transport layer) describes the transport layer. It includes SSP, STP and SMP frame definitions and transport layer state machines.

Clause 10 (Application layer) describes the application layer. It describes SCSI protocol services, mode parameters, log parameters and power conditions, ATA specifics and SMP functions.

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

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

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

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

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

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

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

Informative Annex H (ALIGN and/or NOTIFY insertion rate summary) describes the minimum ALIGN and/or NOTIFY insertion rates for clock skew management, rate matching and STP initiator phy throttling.

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

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

Informative Annex K (Messages between state machines) contains a list of messages between state machines.

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

Informative Annex M (SAS icon) defines the general SAS icon.

Information technology - Small Computer System Interface (SCSI) - Part 151: Serial Attached SCSI - 1.1 (SAS-1.1)

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 part of ISO/IEC 14776 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 International Standard to the other standards and related projects in the SCSI family of standards.
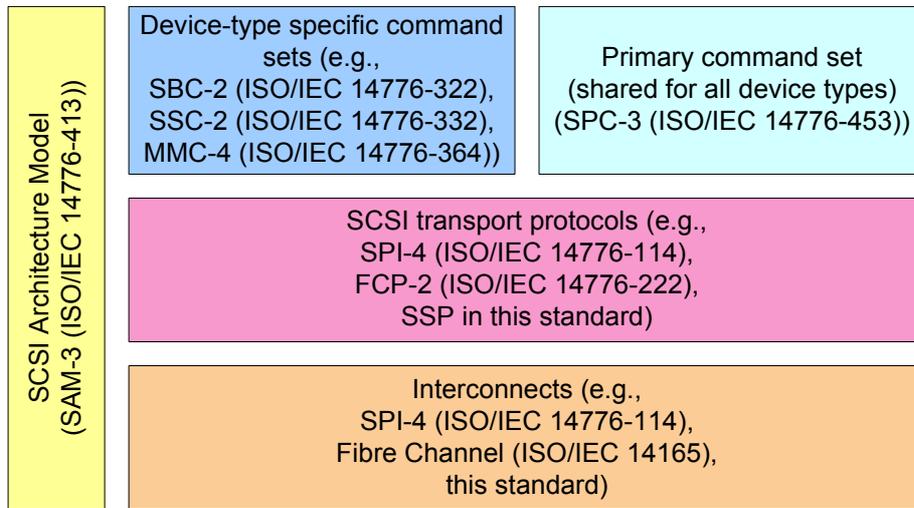
Figure 1 — SCSI document relationships

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

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

ATA register-delivered command set (ATA/ATAPI-7 Volume 1) (ISO/IEC 27439)	Device-type specific command sets (e.g., MMC-3 (ISO/IEC 14776-363))	Primary command set (shared for all device types) (SPC-3 (ISO/IEC 14776-453))
	ATAPI (Packet-delivered command set) (ATA/ATAPI-7 Volume 1 (ISO/IEC 27439))	

ATA/ATAPI logical register set
(ATA/ATAPI-7 Volume 1 (ISO/IEC 27439))

ATA transport protocols (e.g.,
STP in this standard,
ATA/ATAPI-7 Volume 2 (Parallel ATA)(ISO/IEC 27439),
ATA/ATAPI-7 Volume 3 (Serial ATA)(ISO/IEC 27439))

ATA physical interconnects (e.g.,
this standard,
ATA/ATAPI-7 Volume 2 (Parallel ATA)(ISO/IEC 27439),
ATA/ATAPI-7 Volume 3 (Serial ATA)(ISO/IEC 27439))

Figure 2 — ATA document relationships

Figure 1 and figure 2 show the general relationship of the documents to one another, and they 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 ISO and IEC publications

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC TR 14165-117, *Information technology – Fibre channel – Part 117: Methodologies for jitter and signal quality (MJSQ) (*ANSI INCITS TR-35-2004*).*

> NOTE 1 - When MJSQ is referenced from this standard, the FC Port terminology used within MJSQ should be substituted with SAS phy terminology.

ISO/IEC 14776-413, *Information technology – Small computer system interface (SCSI) – Part 413: Architecture model-3 (SAM-3)* (ANSI INCITS 402-2005)

ISO/IEC 14776-322, *Information technology – Small computer system interface (SCSI) – Part 322: Block commands-2 (SBC-2)* (ANSI INCITS 405-2005)

ISO/IEC 14776-372, *Information technology – Small computer system interface (SCSI) – Part 372: Enclosure services-2 (SES-2)* (ANSI INCITS 448-2008) (under consideration)

ISO/IEC 14776-453, *Information technology – Small computer system interface (SCSI) – Part 453: Primary commands-3 (SPC-3)* (ANSI INCITS 408-2005)

ISO/IEC 27439, *AT Attachment with Packet Interface-7 Volume 1 (ATA/ATAPI-7 V1)* (ANSI INCITS 397-2005)

ISO/IEC 27439, *AT Attachment with Packet Interface-7 Volume 3 (ATA/ATAPI-7 V3)(Serial ATA)* (ANSI INCITS 397-2005)

AT Attachement with Packet Interface-7 Amendment 1 (ANSI INCITS 397-2005/AMD1)

IEC/TS 62452, *Testing and performance requirements for high speed serial and parallel-serial copper links* (under consideration) [SFF-8410, HSS Copper Testing and Performance Requirements]

ISO/IEC 19501, *Unified Modeling Language Specification (UML) Version 1.4.2*

2.2 Other references

All references in this subclause were correct at the time of approval of this International Standard.

The provisions of the referenced specifications, as identified in this subclause, are valid within the context of this International Standard. The reference to a specification within this International Standard does not give it any further status within ISO/IEC; in particular, it does not give the referenced specification the status of an International Standard

SFF-8086, *Compact Multilane Series: Common Elements*

SFF-8087, *Compact Multilane Series: Unshielded*

SFF-8088, *Compact Multilane Series: Shielded*

SFF-8223, *2.5" Drive Form Factor with Serial Connector*

SFF-8323, *3.5" Drive Form Factor with Serial Connector*

SFF-8523, *5.25" Drive Form Factor with Serial Connector*

SFF-8416, *Measurement and Performance Requirements for HPEI Bulk Cable*

SFF-8460, *HSS Backplane Design Guidelines*

SFF-8470, *Shielded High Speed Multilane Copper Connector*

SFF-8482, *Unshielded Dual Port Serial Attachment Connector*

SFF-8484, *Multi-Lane Unshielded Serial Attachment Connectors*

SFF-8485, *Serial GPIO (SGPIO) Bus*

NOTE 2 - For more information on the current status of SFF document, contact the SFF Committee at 408-867-6630 (phone), or 408-867-2115 (fax). To obtain copies of these documents, contact the SFF Committee at 14426 Black Walnut Court, Saratoga, CA 95070 at 408-867-6630 (phone) or 408-741-1600 (fax) or see http://www.sffcommittee.org.

3 Terms, definitions, symbols, abbreviations, keywords and conventions

3.1 Terms and definitions

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

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

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

3.1.4
affiliation:
STP target port state of limiting acceptance of connection requests to those from a single STP initiator port. See 7.17.5.

3.1.5
aggregation:
When used in class diagrams, a form of association that defines a whole-part relationship between the whole (i.e., aggregate) class (see 3.1.22) and its parts. See 3.5.

3.1.6
application client:
An object that is the source of SCSI commands (see SAM-3), ATA commands (see ATA/ATAPI-7 V1), or management function requests.

3.1.7
association:
When used in class diagrams, a relationship between two or more classes (see 3.1.22) that specifies connections among their objects (see 3.1.126) (i.e., a relationship that specifies that objects of one class are connected to objects of another class). See 3.5.

3.1.8
attached SAS address:
The SAS address (see 3.1.165) of the attached phy (e.g., received by a physical phy in the incoming IDENTIFY address frame during the initialization sequence (see 4.1.2)), or the SAS address of the STP target port in an STP/SATA bridge (see 4.6.2).

3.1.9
attribute:
When used in class diagrams (see 3.1.23) or object diagrams (see 3.1.127), a named property of a class (see 3.1.22) that describes the range of values that its objects (see 3.1.126) may hold. See 3.5.

3.1.10
AT Attachment
ATA:
A standard for the internal attachment of storage devices to hosts. See ATA/ATAPI-7 V1.

3.1.11
ATA device:
A storage peripheral (analogous to a SCSI target device). See ATA/ATAPI-7 V1.

3.1.12
ATA domain:
I/O system consisting of an ATA host and one or more ATA devices that communicate with one another by means of a service delivery subsystem.

3.1.13
ATA host:
A host device that originates requests to be processed by an ATA device (analogous to a SCSI initiator device). See ATA/ATAPI-7 V1.

3.1.14
baud:
A unit of signaling speed, expressed as the maximum number of times per second the signal (see 3.1.200) may change the state of the physical link (see 3.1.139). The state change produces a transition (i.e., signal edge). Units of baud are symbols/second and in this standard, a symbol represents a single bit or single transition if the maximum transition rate (i.e., a 0101b pattern) is occurring.

3.1.15
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).

3.1.16
bit error ratio
BER:
The number of logical bits output from a receiver circuit that differ from the correct transmitted logical bits, divided by the number of transmitted logical bits. The BER is usually expressed as a coefficient and a power of 10 (e.g., 2 erroneous bits out of 100 000 bits transmitted is expressed as 2 out of 10^5 or 2×10^{-5}). See MJSQ.

3.1.17
broadcast primitive processor
BPP:
An object within an expander function that manages broadcast primitives. See 4.6.5.

3.1.18
burst time:
The part of an OOB signal (see 3.1.131) where the OOB burst (see 3.1.128) is transmitted. See 6.6.

3.1.19
byte:
A sequence of eight contiguous bits considered as a unit. A byte is encoded as a character (see 3.1.21) using 8b10b coding (see 6.2).

3.1.20
cable assembly:
Bulk cable plus a separable connector at each end plus any retention, backshell, or shielding features.

3.1.21
character:
A sequence of ten contiguous bits considered as a unit. A byte (see 3.1.19) is encoded as a character using 8b10b coding (see 6.2).

3.1.22
class:
A description of a set of objects (see 3.1.126) that share the same attributes (see 3.1.9), operations (see 3.1.132), relationships, and semantics. Classes may have attributes and may support operations.

3.1.23
class diagram:
A diagram that shows a collection of classes (see 3.1.22) and their contents and relationships. See 3.5.

3.1.24
clock data recovery
CDR:
The function provided by the receiver circuit responsible for producing a regular clock signal (i.e., the recovered clock) from the received signal and for aligning the recovered clock to the symbols (i.e., bits) being transmitted with the signal. The CDR uses the recovered clock to recover the bits. See MJSQ.

3.1.25
command descriptor block
CDB:
A structure used to communicate a command from a SCSI application client to a SCSI device server. See SAM-3.

3.1.26
compliance point:
An interoperability point where interoperability specifications are met. See 5.3.1.

3.1.27
compliant jitter tolerance pattern
CJTPAT:
A test pattern for jitter testing. See 5.3.5.4 and A.2.

3.1.28
configurable expander device:
An expander device that contains an expander route table that is configured with expander route entries. See 4.1.5.

3.1.29
confirmation:
Information passed from a lower layer state machine to a higher layer state machine, usually responding to a request (see 3.1.155) from that higher layer state machine, and sometimes relaying a response (see 3.1.157) from a peer higher layer state machine. See 3.6.

3.1.30
connection:
A temporary association between a SAS initiator port and a SAS target port. See 7.12.

3.1.31
connection rate:
The effective rate of dwords through the pathway between a SAS initiator phy and a SAS target phy, established through the connection request.

3.1.32
connector:
Electro-mechanical components consisting of a receptacle and a plug that provide a separable interface between two transmission segments.

3.1.33
constraint:
When used in class diagrams (see 3.1.23) and object diagrams (see 3.1.127), a mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations). See 3.5.

3.1.34
control byte:
A byte containing control information defined in table 54 (see 6.2).

3.1.35
control character
Kxx.y:
A character containing control information defined in table 54 (see 6.2).

3.1.36
cumulative distribution function
CDF:
The probability that jitter is less than a given value. See MJSQ.

3.1.37
cyclic redundancy check
CRC:
An error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum. See 7.5.

3.1.38
D.C. idle:
A differential signal level that is nominally 0 V(P-P), used during the idle time (see 3.1.92) of an OOB signal (see 3.1.131). See 5.3.6.

3.1.39
data byte:
A byte containing data information defined in table 53 (see 6.2).

3.1.40
data character
Dxx.y:
A character containing data information defined in table 53 (see 6.2).

3.1.41
data dword:
A dword containing four data bytes, or four data characters with correct disparity.

3.1.42
deadlock:
A condition in which two or more processes (e.g., connection requests) are waiting on the others to complete, resulting in none of the processes completing.

3.1.43
decibel
dB:
One-tenth of the common logarithm (i.e., $\log_{10}$) of the relative power ratio (i.e., dB = 10 $\log_{10}$ (P_1 / P_2)). Equivalent to one-twentieth of the common logarithm of the relative voltage ratio (i.e., dB = 20 $\log_{10}$ (V_1 / V_2)).

3.1.44
dependency:
When used in class diagrams (see 3.1.23), a relationship between two classes (see 3.1.22) where a change to one class (i.e., the independent class) may cause a change in the other class (i.e., the dependent class). See 3.5.

3.1.45
device server:
An object within a SAS target device that processes SCSI tasks (see SAM-3), ATA commands (see ATA/ATAPI-7 V1), or management functions.

3.1.46
direct current
D.C.:
The non-A.C. component of a signal. In this standard, all frequency components below 100 kHz.

3.1.47
direct routing attribute:
The attribute of an expander phy that indicates it may be used by the expander connection manager to route a connection request to an end device. See 4.6.7.1.

3.1.48
direct routing method:
The method the expander connection manager uses to establish a connection with an end device. See 4.6.7.1.

3.1.49
discover process:
The algorithm used by a management application client to configure the SAS domain. See 4.7.

3.1.50
disparity:
The difference between the number of ones and zeros in a character (see 6.2).

3.1.51
dispersion:
Signal pulse broadening and distortion from all causes.

3.1.52
domain:
A SAS domain, a SCSI domain, or an ATA domain.

3.1.53
dword:
A sequence of four contiguous bytes or four contiguous characters considered as a unit. The meaning depends on the context (e.g., when discussing the bits being transmitted over a physical link, dword represents four characters (i.e., 40 bits). When discussing the contents of a frame after 10b8b decoding (see 3.1.3), dword represents four bytes (i.e., 32 bits)).

3.1.54
dword synchronization:
Detection of an incoming stream of dwords from a physical link by a phy. See 6.9.

3.1.55
edge expander device:
An expander device that is part of a single edge expander device set (see 3.1.56). See 4.1.5.

3.1.56
edge expander device set:
A group of one or more edge expander devices (see 3.1.55) that may be attached to no more than one other edge expander device set or one fanout expander device (see 3.1.74). See 4.1.8.2.

3.1.57
enclosure:
The box, rack, or set of boxes providing the powering, cooling, mechanical protection, EMI protection, and external electronic interfaces for one or more SAS device(s) and/or expander device(s). The enclosure provides the outermost electromagnetic boundary and acts as an EMI barrier. An enclosure is not a class (see 3.1.22) in this standard.

3.1.58
enclosure in port:
A set of expander phys with subtractive routing attributes using the same external connector (see 5.2.3.3). See 4.6.2.

3.1.59
enclosure out port:
A set of expander phys with table routing attributes using the same external connector (see 5.2.3.3). See 4.6.2.

3.1.60
end device:
A SAS device that is not contained within an expander device.

3.1.61
event notification:
Information passed from the transport layer to the application layer notifying the application layer that a SCSI event has occurred. See SAM-3.

3.1.62
expander connection manager
ECM:
An object within an expander function that manages routing. See 4.6.3.

3.1.63
expander connection router
ECR:
The portion of an expander function that routes messages between expander phys. See 4.6.4.

3.1.64
expander device:
A device that is part of the service delivery subsystem and facilitates communication between SAS devices. See 4.1.5.

3.1.65
expander function:
An object within an expander device that contains an expander connection manager, expander connection router, and broadcast primitive processor. See 4.6.1.

3.1.66
expander phy:
A phy in an expander device that interfaces to a service delivery subsystem.

3.1.67
expander port:
An expander device object that interfaces to the service delivery subsystem and to SAS ports in other devices. See 4.6.2.

3.1.68
expander route entry:
A SAS address and an enable/disable bit in an expander route table (see 4.6.7.3).

3.1.69
expander route index:
A value used in combination with a phy identifier to select an expander route entry in an expander route table (see 4.6.7.3).

3.1.70
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 4.6.7.3.

3.1.71
external connector:
A bulkhead connector (see 3.1.32), whose purpose is to carry signals into and out of an enclosure (see 3.1.57), that exits the enclosure with only minor compromise to the shield effectiveness of the enclosure. See 5.2.3.3.1.1 and 5.2.3.3.1.2.

3.1.72
eye contour:
The locus of points in a signal level versus time eye diagram where the CDF of 10^{-12} in the actual signal population exists. Comparison of the measured eye contour to the jitter eye mask determines whether a jitter eye mask violation has occurred. See 5.3.5 and MJSQ.

3.1.73
fall time:
The time interval for the falling signal edge to transit between specified percentages of the signal amplitude. In this standard, the measurement points are the 80 % and 20 % voltage levels. Also see rise time (see 3.1.161).

3.1.74
fanout expander device:
An expander device that is capable of being attached to two or more edge expander device sets. See 4.1.5.

3.1.75
far-end crosstalk:
Crosstalk that is propagated in a disturbed channel in the same direction as the propagation of a signal in the disturbing channel. The terminals of the disturbed channel, at which the far-end crosstalk is present, and the energized terminals of the disturbing channel are usually remote from each other.

3.1.76
field:
A group of one or more contiguous bits.

3.1.77
frame:
A sequence of data dwords between a start of frame primitive (i.e., SOF, SOAF, or SATA_SOF) and an end of frame primitive (i.e., EOF, EOAF, or SATA_EOF).

3.1.78
frame information structure
FIS:
The SATA frame format. See ATA/ATAPI-7 V3.

3.1.79
generalization:
When used in class diagrams (see 3.1.23), a relationship among classes (see 3.1.22) where one class (i.e., the superclass) shares the attributes and operations of one or more other classes (i.e., the subclasses). See 3.5.

3.1.80
golden phase lock loop
golden PLL:
A function that conforms to the jitter timing reference frequency response requirements in MJSQ that extracts the jitter timing reference from the data stream under test to be used as the timing reference for the instrument used for measuring the jitter in the signal under test. See MJSQ.

3.1.81
hard reset:
A SAS device or expander device action in response to a reset event in which the device performs the operations described in 4.4.

3.1.82
hard reset sequence:
A sequence that causes a hard reset (see 4.4).

3.1.83
hardware maximum physical link rate:
The maximum physical link rate capability of a phy.

3.1.84
hardware minimum physical link rate:
The minimum physical link rate capability of a phy.

3.1.85
hash function:
A mathematical function that maps values from a larger set of values into a smaller set of values, reducing a long value into a shorter hashed value.

3.1.86
I_T nexus:
A nexus that exists between a SCSI initiator port and a SCSI target port. See SAM-3.

3.1.87
I_T nexus loss:
A condition where a SAS port determines that another SAS port is no longer available. See 4.5.

3.1.88
I_T_L nexus:
A nexus that exists between a SCSI initiator port, a SCSI target port, and a logical unit. This relationship extends the prior I_T nexus. See SAM-3.

3.1.89
I_T_L_Q nexus:
A nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a task. This relationship extends the prior I_T nexus or I_T_L nexus. See SAM-3.

3.1.90
identification sequence:
A sequence where phys exchange IDENTIFY address frames. See 7.9.

3.1.91
idle dword:
A vendor-specific data dword that is scrambled and is transmitted outside a frame. See 7.4.

3.1.92
idle time:
The part of an OOB signal (see 3.1.131) where D.C. idle (see 5.3.6) is being transmitted. See 6.6.

3.1.93
indication:
Information passed from a lower layer state machine to a higher layer state machine, usually relaying a request (see 3.1.155) from a peer higher layer state machine. See 3.6.

3.1.94
information unit
IU:
The portion of an SSP frame that carries command, task management function, data, response, or transfer ready information. See 9.2.2.

3.1.95
insertion loss
S_{21}:
The ratio, usually expressed in dB, of delivered power to incident power. See B.9.

3.1.96
insertion loss, differential
S_{DD21}:
Differential insertion loss measured at port 1. See B.9.

3.1.97
insertion loss, upstream differential
S_{DD12}:
Differential insertion loss measured at port 2. See B.9.

3.1.98
intersymbol interference
ISI:
Reduction in the distinction of a pulse caused by overlapping energy from neighboring pulses. Neighboring pulses are pulses that are close enough to have significant energy overlapping and does not imply or exclude adjacent pulses (i.e., many bit times may separate the pulses, especially in the case of reflections). ISI may result in DDJ and vertical eye closure. Several mechanisms produce ISI (e.g., dispersion, reflections, and circuits that lead to baseline wander). See MJSQ.

3.1.99
invalid character:
A character that is not a control character (see 3.1.35) or a data character (see 3.1.40).

3.1.100
invalid dword:
A dword that is not a data dword (see 3.1.41) or a primitive (see 3.1.144) (i.e., in the character context, a dword that contains an invalid character, a control character in other than the first character position, a control character other than K28.3 or K28.5 in the first character position, or one or more characters with a running disparity error).

3.1.101
jitter:
The collection of instantaneous deviations of signal edge times at a defined signal level of the signal from the reference times (e.g., as defined by the jitter timing reference) for those events. See MJSQ.

3.1.102
jitter, data dependent
DDJ:
Jitter that is added when the transmission pattern is changed from a clock-like to a non-clock-like pattern. See MJSQ.

3.1.103
jitter, deterministic
DJ:
Jitter with non-Gaussian distribution. See MJSQ.

3.1.104
jitter, random,
RJ:
Jitter that is characterized by a Gaussian distribution and is unbounded. See MJSQ.

3.1.105
jitter, sinusoidal
SJ:
Single frequency jitter applied during signal tolerance testing. See MJSQ.

3.1.106
jitter, total
TJ:
Jitter from all sources. See MJSQ.

3.1.107
jitter timing reference:
The signal used as the basis for calculating the jitter in the signal under test. See MJSQ.

3.1.108
jitter tolerance:
The ability of the receiver device to recover transmitted bits in an incoming data stream in the presence of specified jitter in the signal applied to the receiver device compliance point. See MJSQ.

3.1.109
jitter tolerance pattern
JTPAT:
A test pattern for jitter testing. See 5.3.5.4 and A.1.

3.1.110
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).

3.1.111
link reset:
Performing the link reset sequence (see 3.1.112).

3.1.112
link reset sequence:
For SATA, a phy reset sequence (see 3.1.138). For SAS, a phy reset sequence followed by an identification sequence (see 3.1.90), or a phy reset sequence followed by a hard reset sequence (see 3.1.82), another phy reset sequence, and an identification sequence. See 4.4.

3.1.113
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).

3.1.114
livelock:
A condition where two or more processes (e.g., connection requests) continually change their state in response to changes in other processes, resulting in none of the processes completing.

3.1.115
logical unit number
LUN:
An identifier for a logical unit. See SAM-3.

3.1.116
media:
Plural of medium (see 3.1.117).

3.1.117
medium:
The material on which data is stored (e.g., a magnetic disk).

3.1.118
message:
Information sent between state machines. See 3.6.

3.1.119
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).

3.1.120
multiplicity:
When used in class diagrams (see 3.1.23), an indication of the range of allowable instances of an object (see 3.1.126) or an attribute (see 3.1.9). See 3.5.

3.1.121
narrow link:
A physical link that attaches a narrow port to another narrow port. See 4.1.3.

3.1.122
narrow port:
A port that contains exactly one phy. See 4.1.3.

3.1.123
near-end crosstalk
NEXT:
Crosstalk that is propagated in a disturbed channel in the opposite direction as the propagation of a signal in the disturbing channel. The terminals of the disturbed channel, at which the near-end crosstalk is present, and the energized terminals of the disturbing channel are usually near each other.

3.1.124
negotiated physical link rate:
The current operational physical link rate established as a result of speed negotiation between two phys.

3.1.125
nexus:
A relationship between a SCSI initiator port and a SCSI target port that may extend to a logical unit and a task. See SAM-3.

3.1.126
object:
An entity with a well-defined boundary and identity that encapsulates state and behavior. All objects are instances of classes (see 3.1.22)(i.e., a concrete manifestation of a class is an object).

3.1.127
object diagram:
A diagram that encompasses objects and their relationships at a point in time. See 3.5.

3.1.128
OOB burst:
The transmission of signal transitions for a burst time (see 3.1.18). See 6.6.1.

3.1.129
OOB interval:
The time basis for burst times (see 3.1.18) and idle times (see 3.1.92) used to create OOB signals (see 3.1.131). See 6.6.1.

3.1.130
OOB sequence:
A sequence where two phys exchange OOB signals (see 3.1.131). See 6.7.2.1 and 6.7.4.1.

3.1.131
OOB signal:
Pattern of idle time (see 3.1.92) and burst time (see 3.1.18) used during the link reset sequence. See 6.6.

3.1.132
operation:
When used in class diagrams (see 3.1.23), a service that may be requested from any object (see 3.1.126) of the class (see 3.1.22) in order to effect behavior. Operations describe what a class is allowed to do and may be a request or a query. A request may change the state of the object but a query should not. See 3.5.

3.1.133
partial pathway:
The set of physical links participating in a connection request that have not yet conveyed a connection response. See 4.1.9.

3.1.134
pathway:
A set of physical links between a SAS initiator phy and a SAS target phy being used by a connection. See 4.1.9.

3.1.135
pathway blocked count:
The number of times the port has retried this connection request due to receiving OPEN_REJECT (PATHWAY BLOCKED).

3.1.136
phy:
A object in a device that is used to interface to other devices (e.g., an expander phy (see 3.1.66) or a SAS phy (see 3.1.171)). See 4.1.2.

3.1.137
phy identifier:
A identifier for a phy that is unique within the device containing it. See 4.2.7.

3.1.138
phy reset sequence:
An OOB sequence (see 3.1.130) followed by a speed negotiation sequence (see 3.1.212). See 4.4.

3.1.139
physical link:
Two differential signal pairs, one pair in each direction, that connect two physical phys. See 4.1.2.

3.1.140
physical phy:
A phy (see 3.1.136) that contains a transceiver (see 3.1.241) and electrically interfaces to a physical link to communicate with another physical phy. See 4.1.2.

3.1.141
port:
A SAS port or an expander port. Each port contains one or more phys. See 4.1.3.

3.1.142
potential pathway:
A set of physical links between a SAS initiator phy and a SAS target phy. See 4.1.9.

3.1.143
power on:
Power being applied.

3.1.144
primitive:
A dword containing a 7Ch or BCh control byte followed by three data bytes, or a K28.3 or K28.5 control character with correct disparity followed by three data characters with correct disparity. See 7.2.

3.1.145
primitive sequence:
A set of primitives treated as a single entity. See 7.2.4.

3.1.146
probe point:
Physical position in a test load where signal characteristics for compliance points are measured See 5.3.2.

3.1.147
programmed maximum physical link rate:
The maximum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function (see 10.4.3.10) or the Phy Control and Discover subpage (see 10.2.7.2.3)).

3.1.148
programmed minimum physical link rate:
The minimum operational physical link rate of a phy (e.g., as programmed via the SMP PHY CONTROL function (see 10.4.3.10) or the Phy Control and Discover subpage (see 10.2.7.2.3)).

3.1.149
rate:
Data transfer rate of a physical link (e.g., 1,5 Gbit/s or 3,0 Gbit/s).

3.1.150
rate change delay time
RCDT:
The time between rates during the speed negotiation sequence (see 6.7.4.2).

3.1.151
read data:
Data transferred to the application client's data-in buffer from the device server, as requested by the Send Data-In transport protocol service (see 10.2.1.6).

3.1.152
receiver circuit:
An electronic circuit that converts an analog serial input signal to a logic signal.

3.1.153
receiver device
Rx:
The device downstream from an IR or CR compliance point containing a portion of the physical link and a receiver circuit (see 3.1.152).

3.1.154
reflection coefficient
ρ**:**
The ratio of reflected voltage to incident voltage.

3.1.155
request:
Information passed from a higher layer state machine to a lower layer state machine, usually to initiate some action. See 3.6.

3.1.156
reset event:
An event that triggers a hard reset (see 4.4.2) in a SAS device.

3.1.157
response:
Information passed from a higher layer state machine to a lower layer state machine, usually in response to an indication (see 3.1.93). See 3.6.

3.1.158
return loss
S_{11}:
The ratio, usually expressed in dB, of incident power to reflected power. See B.9.

3.1.159
return loss, differential
S_{DD21}:
Differential return loss measured at port 1. See B.9.

3.1.160
return loss, upstream differential
S_{DD12}:
Differential return loss measured at port 2. See B.9.

3.1.161
rise time:
The time interval for the rising signal edge to transit between specified percentages of the signal amplitude. In this standard, the measurement points are the 20 % and 80 % voltage levels. Also see fall time (see 3.1.73).

3.1.162
role:
When used in class diagrams (see 3.1.23) and object diagrams (see 3.1.127), a label at the end of an association or aggregation that defines a relationship to the class on the other side of the association or aggregation. See 3.5.

3.1.163
run length:
Number of consecutive identical bits in the transmitted signal (e.g., the pattern 0011111010 includes the following run lengths: five 1s, one 0, one 1, and indeterminate run lengths of 0s at the start and end).

3.1.164
running disparity
RD:
A binary parameter with a negative (-) or positive (+) value indicating the cumulative encoded signal imbalance between the one and zero signal state of all characters since dword synchronization has been achieved. See 6.2.

3.1.165
SAS address:
A worldwide unique value assigned to a SAS initiator port, SAS target port, expander device, SAS initiator device, or SAS target device. See 4.2.2.

3.1.166
SAS device:
A SAS initiator device, SAS target device, or SAS target/initiator device.

3.1.167
SAS domain:
The I/O system defined by this standard that may serve as a SCSI domain. See 4.1.7.

3.1.168
SAS initiator device:
A device containing SSP, STP, and/or SMP initiator ports in a SAS domain. See 4.1.4.

3.1.169
SAS initiator phy:
A phy in a SAS initiator device.

3.1.170
SAS initiator port:
An SSP initiator port, STP initiator port, and/or SMP initiator port in a SAS domain.

3.1.171
SAS phy:
A phy in a SAS device that interfaces to a service delivery subsystem.

3.1.172
SAS port:
A SAS initiator port, SAS target port, or SAS target/initiator port.

3.1.173
SAS target device:
A device containing SSP, STP, and/or SMP target ports in a SAS domain. See 4.1.4.

3.1.174
SAS target phy:
A phy in a SAS target device.

3.1.175
SAS target port:
An SSP target port, STP target port, and/or SMP target port in a SAS domain.

3.1.176
SAS target/initiator device:
A device that has all the characteristics of a SAS target device and a SAS initiator device.

3.1.177
SAS target/initiator port:
A port that has all the characteristics of a SAS target port and a SAS initiator port in a SAS domain.

3.1.178
SATA device:
An ATA device that contains a SATA device port in an ATA domain (analogous to a SCSI target device).

3.1.179
SATA device port:
An ATA device object that interfaces to the service delivery subsystem with SATA (analogous to a SCSI target port).

3.1.180
SATA host:
An ATA host that contains a SATA host port in an ATA domain (analogous to a SCSI initiator device).

3.1.181
SATA host port:
An ATA host object that interfaces to the service delivery subsystem with SATA (analogous to a SCSI initiator port).

3.1.182
SATA phy:
A phy in a SATA device or SATA port selector that interfaces to a service delivery subsystem (analogous to a SAS phy).

3.1.183
SATA port selector:
A device that attaches to two SATA host ports (i.e., two ATA domains) and one SATA device port, and provides the means for one SATA host to access the device at any given time (see SATAII-PS).

3.1.184
scrambling:
Modifying data by XORing each bit with a pattern generated by a linear feedback shift register to minimize repetitive character patterns. See 7.6.

3.1.185
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.

3.1.186
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.

3.1.187
SCSI initiator device:
A SCSI device containing application clients and SCSI initiator ports that originates device service and task management requests to be processed by a SCSI target device and receives device service and task management responses from SCSI target devices. See SAM-3.

3.1.188
SCSI initiator port:
A SCSI initiator device object that acts as the connection between application clients and the service delivery subsystem through which indications and responses are routed. See SAM-3.

3.1.189
SCSI port:
A SCSI initiator port, SCSI target port, or SCSI target/initiator port. See SAM-3.

3.1.190
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.

3.1.191
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 confirmations are routed. See SAM-3.

3.1.192
SCSI target/initiator device:
A device that has all the characteristics of a SCSI target device and a SCSI initiator device. See SAM-3.

3.1.193
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.

3.1.194
Serial ATA
SATA:
The protocol defined by ATA/ATAPI-7 V3 (see 2.1).

3.1.195
Serial ATA Tunneled Protocol
STP:
The protocol defined in this standard used by STP initiator ports to communicate with STP target ports in a SAS domain. See 7.17 and 9.3.

3.1.196
Serial Attached SCSI
SAS:
The set of protocols and the interconnect defined by this standard.

3.1.197
Serial Management Protocol
SMP:
The protocol defined in this standard used by SMP initiator ports to communicate with SMP target ports in a SAS domain. See 7.18 and 9.4.

3.1.198
Serial SCSI Protocol
SSP:
The protocol defined in this standard used by SSP initiator ports to communicate with SSP target ports in a SAS domain. See 7.16 and 9.2.

3.1.199
service delivery subsystem:
The part of a SCSI I/O system that transmits information between a SCSI initiator port and a SCSI target port, or the part of an ATA I/O system that transmits information between an ATA host and an ATA device, or the part of a SAS I/O system that transmits information between a SAS initiator port and a SAS target port.

3.1.200
signal:
The entire voltage waveform during transmission.

3.1.201
signal amplitude:
A property of the overall signal (see 3.1.200) that describes the peak or peak-to-peak values of the signal level (see 3.1.202).

3.1.202
signal level:
The instantaneous intensity of a signal (see 3.1.200) measured in volts.

3.1.203
signal tolerance:
The ability of the receiver device to recover transmitted bits in an incoming data stream with maximum jitter and minimum amplitude. See MJSQ.

3.1.204
SMP initiator phy:
A SAS initiator phy in an SMP initiator port.

3.1.205
SMP initiator port:
A SAS initiator device object in a SAS domain that interfaces to the service delivery subsystem with SMP.

3.1.206
SMP phy:
A SAS phy in an SMP port.

3.1.207
SMP port:
An SMP initiator port, SMP target port, or SMP target/initiator port.

3.1.208
SMP target phy:
A SAS target phy in an SMP target port.

3.1.209
SMP target port:
A SAS target device object in a SAS domain that interfaces to the service delivery subsystem with SMP.

3.1.210
SMP target/initiator port:
A port that has all the characteristics of an SMP initiator port and an SMP target port.

3.1.211
speed negotiation lock time
SNLT:
The maximum time during a speed negotiation window for a transmitter to reply with ALIGN (1) (see 6.7.4.2).

3.1.212
speed negotiation sequence:
A sequence in which two phys negotiate the operational physical link rate. See 6.7.2.2 and 6.7.4.2.

3.1.213
speed negotiation transmit time
SNTT:
The time during which ALIGN (0) or ALIGN (1) is transmitted during the speed negotiation sequence (see 6.7.4.2).

3.1.214
spread spectrum clocking:
The technique of modulating the operating frequency of a transmitted signal to reduce the measured peak amplitude of radiated emissions.

3.1.215
SSP initiator phy:
A SAS initiator phy in an SSP initiator port.

3.1.216
SSP initiator port:
A SCSI initiator port in a SAS domain that implements SSP.

3.1.217
SSP phy:
A SAS phy in an SSP port.

3.1.218
SSP port:
An SSP initiator port, SSP target port, or SSP target/initiator port.

3.1.219
SSP target phy:
A SAS target phy in an SSP target port.

3.1.220
SSP target port:
A SCSI target port in a SAS domain that implements SSP.

3.1.221
SSP target/initiator port:
A port that has all the characteristics of an SSP initiator port and an SSP target port.

3.1.222
state machine variable:
A variable that exists within the context of a state machine that may log status determined in one state that is used in another state of the same state machine affecting subsequent state transitions or state machine outputs.

3.1.223
STP initiator phy:
A SAS initiator phy in an STP initiator port.

3.1.224
STP initiator port:
A SAS initiator device object in a SAS domain that interfaces to the service delivery subsystem with STP.

3.1.225
STP phy:
A SAS phy in an STP port.

3.1.226
STP port:
An STP initiator port, STP target port, or STP target/initiator port.

3.1.227
STP primitive:
A primitive used only inside STP connections and on SATA physical links. See 7.2.2.

3.1.228
STP target phy:
A SAS target phy in an STP target port.

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

3.1.230
STP target/initiator port:
A port that has all the characteristics of an STP initiator port and an STP target port.

3.1.231
STP/SATA bridge:
An expander device object containing an STP target port, a SATA host port, and the functions required to forward information between the STP target port and SATA host port to enable STP initiator ports in a SAS domain to communicate with SATA devices in an ATA domain.

3.1.232
subtractive routing attribute:
The attribute of an edge expander phy that indicates it may be used by the expander connection manager to route connection requests not resolved using the direct routing method or table routing method. See 4.6.7.1.

3.1.233
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 to an expander device. See 4.6.7.1.

3.1.234
symbol:
The smallest unit of data transmission on a physical link (i.e., a bit).

3.1.235
table routing attribute:
The attribute of an expander phy that indicates it may be used by the expander connection manager to route connection requests using an expander route table. See 4.6.7.1.

3.1.236
table routing method:
The method the expander connection manager uses to route connection requests to an expander device using an expander route table. See 4.6.7.1.

3.1.237
task:
An object within the logical unit representing the work associated with a command or group of linked commands.

3.1.238
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.

3.1.239
task manager:
An agent within the device server that processes task management functions. See SAM-3.

3.1.240
throttling:
Reducing the rate at which an STP initiator phy is sourcing dwords. See 7.17.2.

3.1.241
transceiver:
A physical entity that contains both a transmitter device (see 3.1.244) and a receiver device (see 3.1.153).

3.1.242
transmitter circuit:
An electronic circuit that converts a logic signal to an analog serial output signal.

3.1.243
transmitter compliance transfer function
TCTF:
The mathematical statement of the transfer function through which the transmitter shall be capable of producing acceptable signals as defined by a receive mask. See 5.3.6.1.

3.1.244
transmitter device
Tx:
The device upstream from an IT or CT compliance point containing a portion of the physical link and a transmitter circuit (see 3.1.242).

3.1.245
transport protocol service confirmation:
Information passed from the transport layer to the application layer (i.e., from the SSP initiator port to the SCSI application client) that notifies the application layer that a SCSI transport protocol service has completed.

3.1.246
transport protocol service indication:
Information passed from the transport layer to the application layer notifying the application layer (i.e., from the SSP target port to the SCSI device server) to begin a SCSI transport protocol service.

3.1.247
transport protocol service request:
Information passed from the SCSI application layer to the SSP transport layer (i.e., from the SCSI application client to the SCSI initiator port) to begin a SCSI transport protocol service.

3.1.248
transport protocol service response:
Information passed from the application layer to the transport layer (i.e., from the SCSI device server to the SSP target port) that completes the SCSI transport protocol service.

3.1.249
TxRx connection:
The complete simplex signal path between the transmitter circuit (see 3.1.242) and receiver circuit (see 3.1.152). See 5.3.3.

3.1.250
TxRx connection segment:
That portion of a TxRx connection (see 3.1.249) delimited by separable connectors or changes in the conductive material. See 5.3.3.

3.1.251
unit interval
UI:
The normalized, dimensionless, nominal duration of a signal transmission bit (e.g., $666,\overline{6}$ ps at 1,5 Gbit/s and $333,\overline{3}$ ps at 3,0 Gbit/s). Unit interval is a measure of time that has been normalized such that 1 UI is equal to 1/baud seconds.

3.1.252
valid character:
A character that is a control character (see 3.1.35) or a data character (see 3.1.40).

3.1.253
valid dword:
A dword that is a data dword (see 3.1.41) or a primitive (see 3.1.144).

3.1.254
virtual phy:
A phy (see 3.1.136) that interfaces with a vendor-specific interface to another virtual phy inside the same device. See 4.1.2.

3.1.255
wide link:
A group of physical links that attaches a wide port to another wide port. See 4.1.3.

3.1.256
wide port:
A port that contains more than one phy. See 4.1.3.

3.1.257
write data:
Data transferred from the application client's data-out buffer to the device server, as requested by the Request Data-Out transport protocol service (see 10.2.1.8).

3.2 Symbols and abbreviations

Abbreviation	Meaning
AA	ATA application layer (see 10.3)
A.C.	alternating current

Abbreviation	Meaning
ACK	acknowledge primitive (see 7.2.6.1)
ACA	auto contingent allegiance (see SAM-3)
AIP	arbitration in progress primitive (see 7.2.5.1)
ASIC	application specific integrated circuit
ATA	AT attachment (see 3.1.10)
ATAPI	AT attachment packet interface
ATA/ATAPI-7	AT Attachment with Packet Interface - 7 standard (see 2.1)
AWG	American wire gauge
AWT	arbitration wait time
BCH	Bose, Chaudhuri and Hocquenghem code (see 4.2.3)
BER	bit error ratio (see 3.1.16)
BIST	built in self test
BPP	broadcast primitive processor (see 3.1.17)
CDB	command descriptor block (see 3.1.25)
CDF	cumulative distribution function (see 3.1.36)
CDR	clock data recovery (see 3.1.24)
CJTPAT	compliant jitter tolerance pattern (see 3.1.27)
CR	inter-enclosure (i.e., cabinet) receiver device compliance point (see 5.3.1)
CRC	cyclic redundancy check (see 3.1.37)
CT	inter-enclosure (i.e., cabinet) transmitter device compliance point (see 5.3.1)
dB	decibel (see 3.1.43)
D.C.	direct current (see 3.1.46)
Dxx.y	data character (see 3.1.40)
DDJ	data dependent jitter (see 3.1.102)
DJ	deterministic jitter (see 3.1.103)
e	2,718 28..., the base of the natural (i.e., hyperbolic) system of logarithms
ECM	expander connection manager (see 3.1.62)
ECR	expander connection router (see 3.1.63)
EMI	electromagnetic interference
EOAF	end of address frame primitive (see 7.2.5.6)
EOF	end of frame primitive (see 7.2.6.4)
ESD	electrostatic discharge
FIS	frame information structure (see 3.1.78)
G1	generation 1 physical link rate (1,5 Gbit/s)
G2	generation 2 physical link rate (3,0 Gbit/s)
G3	generation 3 physical link rate (defined in a future version of this standard)
Gbit	gigabits (10^9 bits)
Gbps	gigabits per second (10^9 bits per second)
Gen1i	SATA generation 1 physical link rate (1,5 Gbit/s)(see SATAII-PHY)
Gen1x	SATA generation 1 physical link rate (1,5 Gbit/s), extended length (see SATAII-PHY)
Gen2i	SATA generation 2 physical link rate (3,0 Gbit/s)(see SATAII-PHY)
Gen2x	SATA generation 2 physical link rate (3,0 Gbit/s), extended length (see SATAII-PHY)
GHz	gigahertz (10^9 cycles per second)
GPIO	general purpose input/output

Abbreviation	Meaning
Hz	hertz (cycles per second)
ID	identifier
IR	intra-enclosure (i.e., internal) receiver device compliance point (see 5.3.1)
ISI	intersymbol interference (see 3.1.98)
IT	intra-enclosure (i.e., internal) transmitter device compliance point (see 5.3.1)
IU	information unit (see 3.1.94)
JTPAT	jitter tolerance pattern (see 3.1.109)
kHz	kilohertz (10^3 cycles per second)
Kxx.y	control character (see 3.1.35)
LED	light-emitting diode
LSB	least significant bit (see 3.1.110)
LUN	logical unit number (see 3.1.115)
μA	microampere (10^{-6} amperes)
μs	microsecond (10^{-6} seconds)
MA	management application layer (see 10.4)
mA	milliampere (10^{-3} amperes)
Mbaud	megabaud (10^6 symbols per second)
MBps	megabytes per second (10^6 bytes per second)
MHz	megahertz (10^6 cycles per second)
MSB	most significant bit (see 3.1.119)
ms	millisecond (10^{-3} seconds)
MT	SMP transport layer state machines (see 9.4)
mV	millivolt (10^{-3} volts)
N/A	not applicable
NAA	name address authority
NAK	negative acknowledge primitive (see 7.2.6.5)
NEXT	near-end crosstalk (see 3.1.123)
nF	nanofarad (10^{-9} Farads)
ns	nanosecond (10^{-9} seconds)
NVRAM	non-volatile random-access memory
OOB	out-of-band
OOBI	out-of-band interval (see 3.1.129)
PCB	printed circuit board
PL	port layer state machines (see 8.2)
PLL	phase lock loop
P-P	peak-to-peak
ppm	parts per million (10^{-6})
ps	picosecond (10^{-12} seconds)
ρ	reflection coefficient (rho)(see 3.1.154)
RCDT	rate change delay time (see 3.1.150)
RD	running disparity (see 3.1.164)
RJ	random jitter (see 3.1.104)
rms	root mean square
RRDY	receiver ready primitive (see 7.2.6.6)

Abbreviation	Meaning
Rx	receiver device (see 3.1.153)
SA	SCSI application layer (see 10.2)
SAM-3	SCSI Architecture Model - 3 standard (see 2.1)
SAS	Serial Attached SCSI (see 3.1.196)
SATA	Serial ATA (see 3.1.194)
SBC-2	SCSI Block Commands - 2 standard (see 2.1)
SCSI	Small Computer System Interface family of standards
S_{ij}	S-parameter for port j to port i (see B.9)
S_{CCij}	S-parameter for common-mode to common-mode port j to port i (see B.9)
S_{CDij}	S-parameter for differential to common-mode port j to port i (see B.9)
S_{DCij}	S-parameter for common-mode to differential port j to port i (see B.9)
S_{DDij}	S-parameter for differential to differential port j to port i (see B.9)
SGPIO	Serial General Purpose Input/Output
SJ	sinusoidal jitter (see 3.1.105)
SL	link layer for SAS phys state machines (see 7.14)
SL_IR	link layer identification and hard reset state machines (see 7.9.5)
SMA	subminiature version A connector
SMP	Serial Management Protocol (see 3.1.197), or link layer for SMP phys state machines (see 7.18.4)
SNLT	speed negotiation lock time (see 3.1.211)
SNTT	speed negotiation transmit time (see 3.1.213)
SOAF	start of address frame primitive (see 7.2.5.12)
SOF	start of frame primitive (see 7.2.6.7)
SP	phy layer state machine (see 6.8)
SP_DWS	phy layer dword synchronization state machine (see 6.9)
SPC-3	SCSI Primary Commands - 3 standard (see 2.1)
SSP	Serial SCSI Protocol (see 3.1.198), or link layer for SSP phys state machines (see 7.16.8)
ST	SSP transport layer state machines (see 9.2)
STP	Serial ATA Tunneled Protocol (see 3.1.195), or link layer for STP phys state machines (see 7.17.9)
s	second (unit of time)
TCTF	transmitter compliance transfer function (see 3.1.243)
TDNA	time domain network analyzer (i.e., TDR/TDT plus analysis software that performs a VNA-style output)
TDR	time domain reflectometer
TDT	time domain transmission
TJ	total jitter (see 3.1.106)
TT	STP transport layer state machines (see 9.3)
Tx	transmitter device (see 3.1.244)
UI	unit interval (see 3.1.251)
V	volt
VNA	vector network analyzer
VPD	vital product data (see 10.2.11)
XL	link layer for expander phys state machine (see 7.15)

Abbreviation	Meaning
XOR	exclusive logical OR
^	exclusive logical OR
×	multiplication
/	division
±	plus or minus
>	greater than
≥	greater than or equal to
<	less than
≤	less than or equal to
|v|	the absolute value of (i.e., magnitude) of v

3.3 Keywords

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

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

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

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

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

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

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

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

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

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

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

3.4 Editorial conventions

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

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

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

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

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

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

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

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

This standard uses the ISO convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a space and a comma is used as the decimal point). Table 1 shows some examples of decimal numbers using the ISO and American numbering conventions.

Table 1 — ISO and American numbering conventions

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

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

Lists sequenced by lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the colors named:
- a)   red, specificity one of the following colors:
  - A)   crimson; or
  - B)   amber;
- b)   blue; or
- c)   green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 - The following list shows the order in which a page is meant to be read:
1)   top;
2)   middle; and
3)   bottom.

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

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

Notes do not constitute any requirements for implementers.

3.5 Class diagram and object diagram conventions

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

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

Notation for a class with no attributes or operations:

Class Name

Notation for a class with attributes and no operations:

Class Name
Attribute 1
Attribute 2
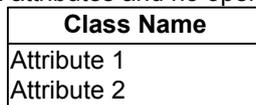
Notation for a class with operations and no attributes:

Class Name
Operation 1()
Operation 2()

Notation for a class with attributes and operations:

Class Name
Attribute 1
Attribute 2
Operation 1()
Operation 2()
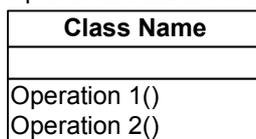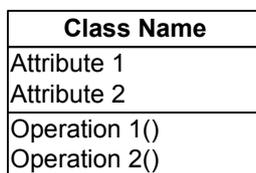
Notation for a class with attributes showing multiplicity and operations:

Class Name
Attribute 1[1..*]
Attribute 2[1]
Operation 1()
Operation 2()
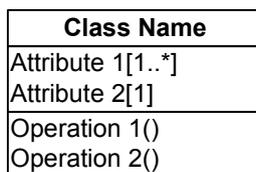
Figure 3 — Classes in class diagrams

Table 2 defines the notation used for multiplicity in class diagrams.

Table 2 — Multiplicity notation in class diagrams

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

Figure 4 defines the notation used for association relationships between classes.
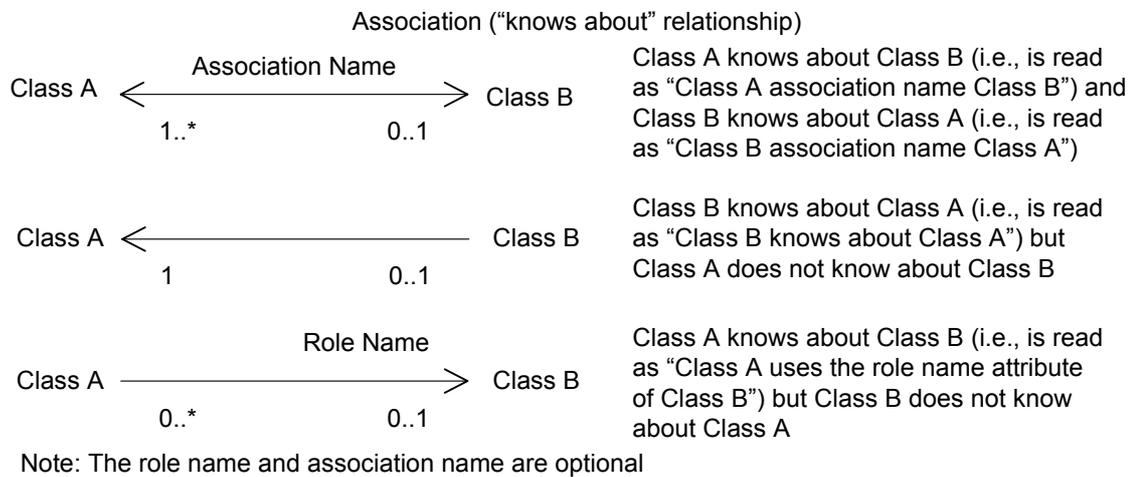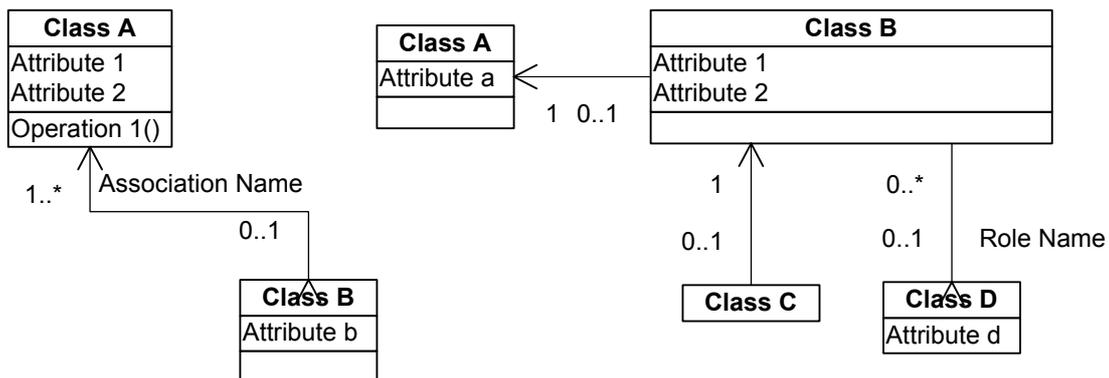
Figure 4 — Association relationships in class diagrams

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

Aggregation ("is a part of" or "contains" relationship)
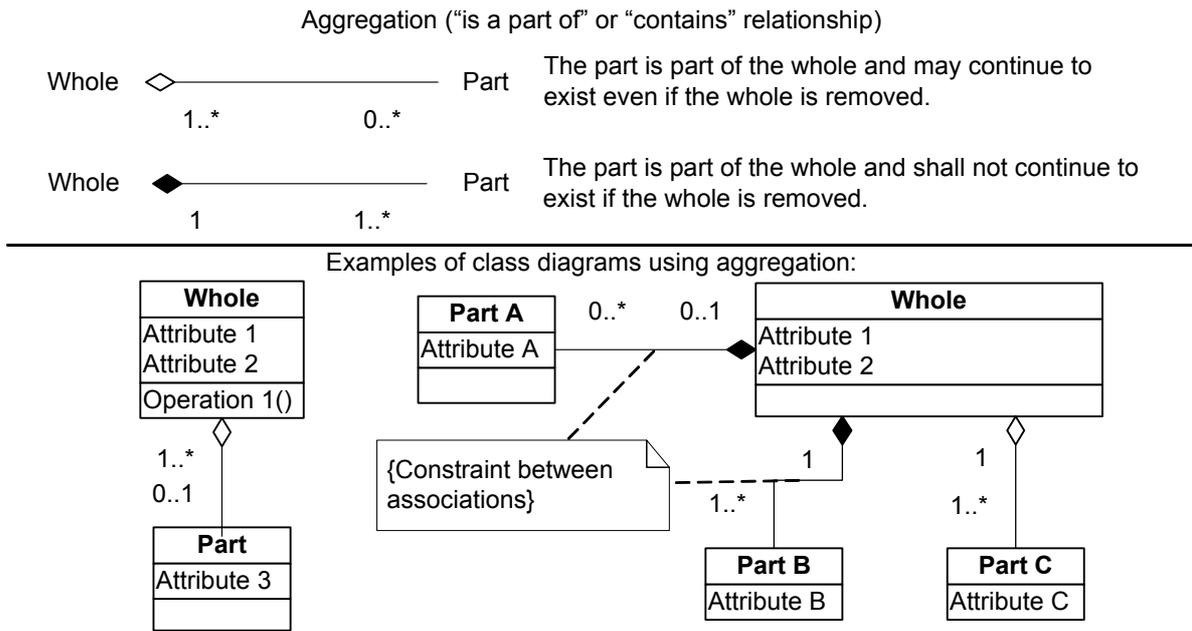
Figure 5 — Aggregation relationships in class diagrams

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

Generalization ("is a kind of" relationship)
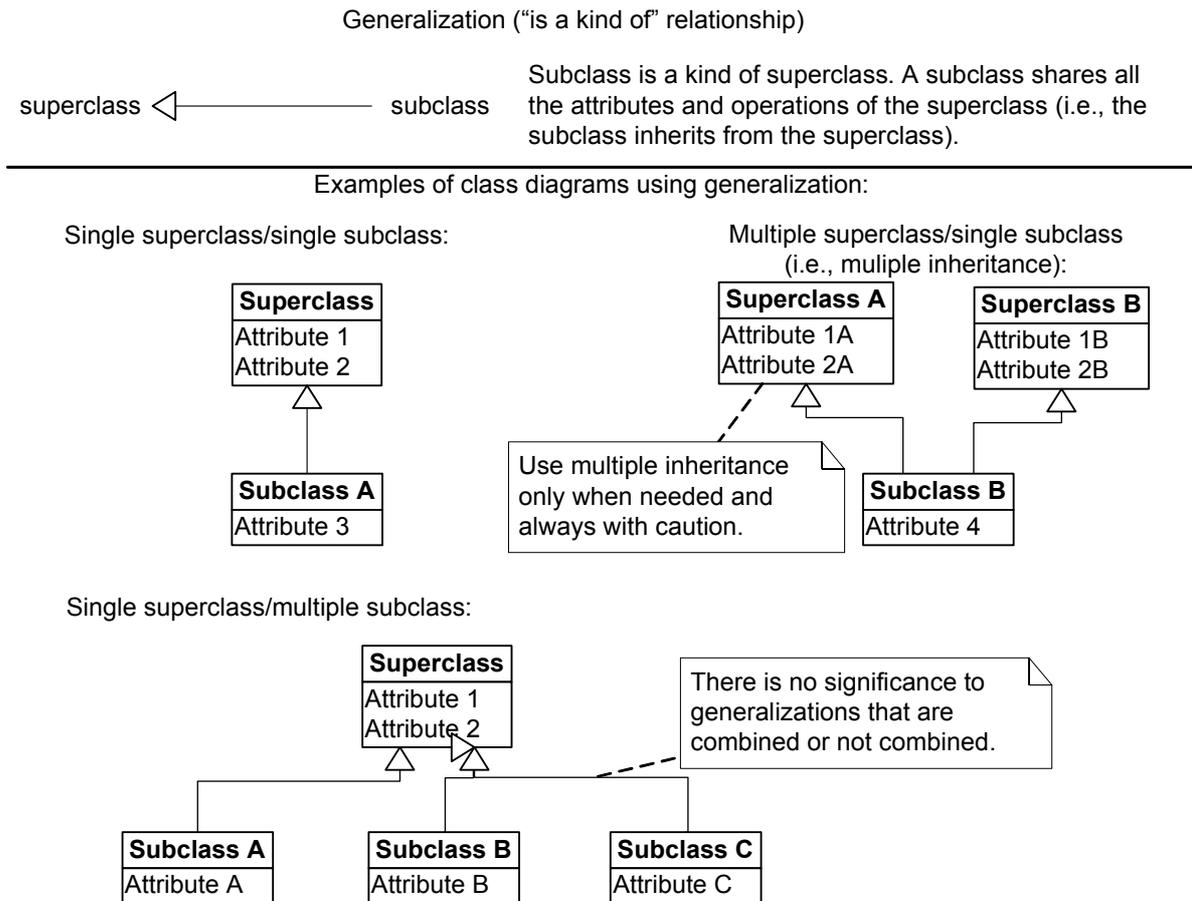
Figure 6 — Generalization relationships in class diagrams

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

Dependency ("depends on" relationship)

Class A   `------------------>`   Class B   Class A depends on class B. A change in class B may cause a change in class A.

Example of class diagram using dependency:

| Dependent | `------------------>` | Independent |
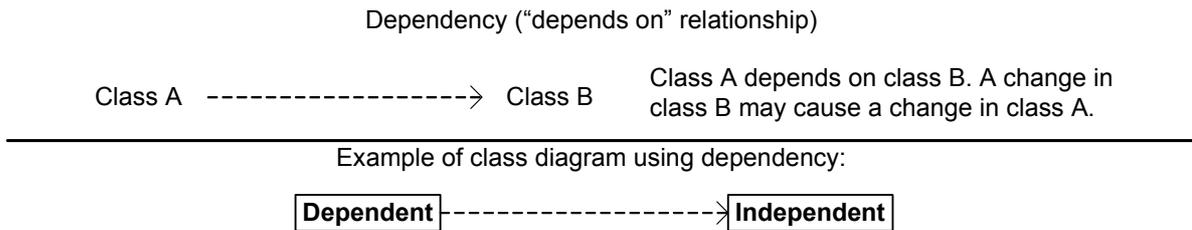
Figure 7 — Dependency relationships in class diagrams

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

Notation for a named object with no attributes:

| <u>label : Class Name</u> |

Notation for a named object with attributes:

| <u>label : Class Name</u> |
| Attribute 1 = x |
| Attribute 2 = y |
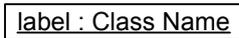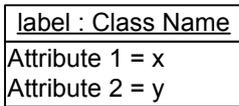
Notation for an anonymous object with no attributes:

| <u> : Class Name</u> |

Notation for an anonymous object with attributes:

| <u> : Class Name</u> |
| Attribute 1 = x |
| Attribute 2 = y |
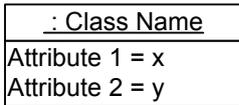
Figure 8 — Objects in object diagrams

3.6 State machine conventions

3.6.1 State machine conventions overview

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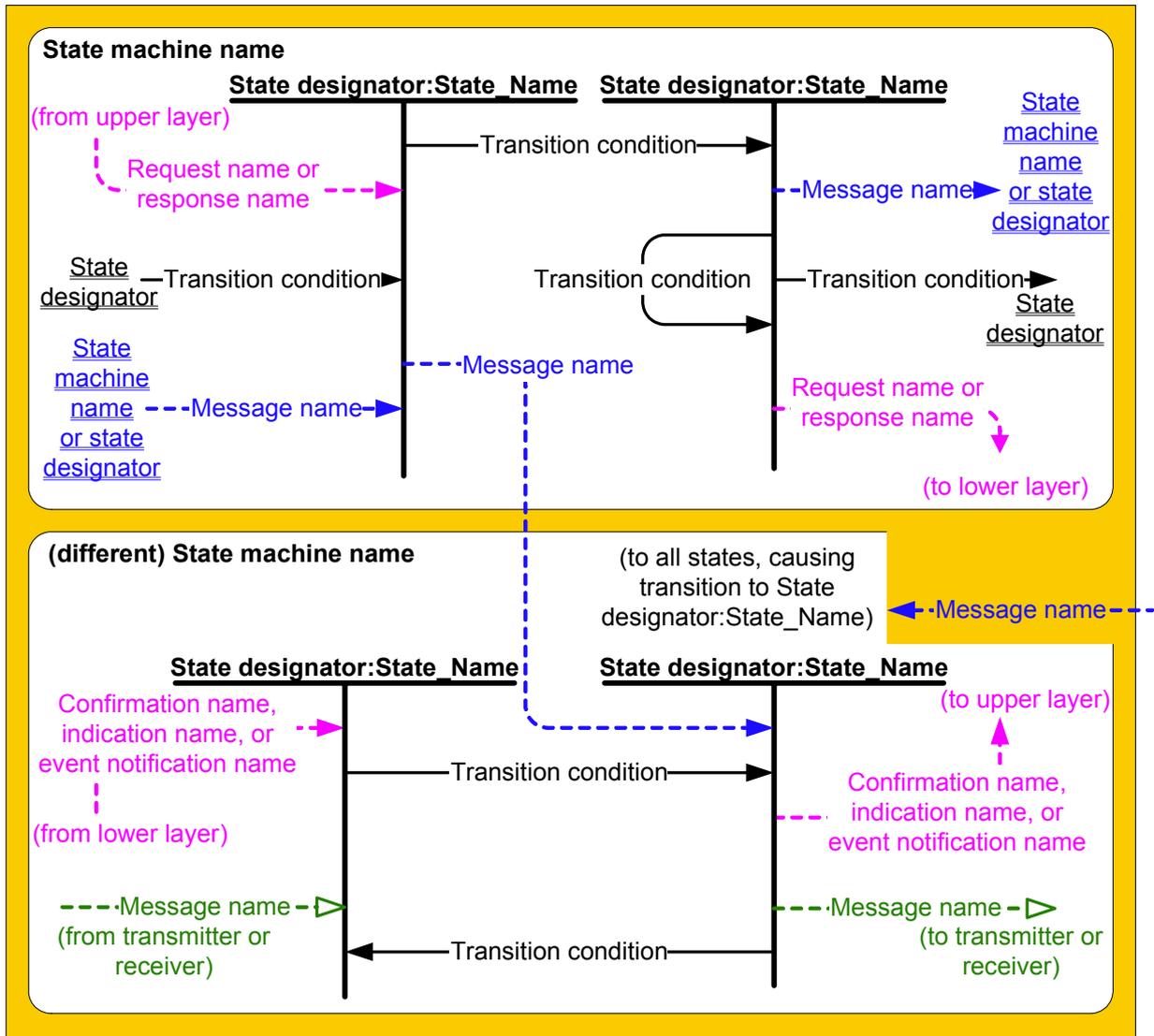
Figure 9 — State machine conventions

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

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

3.6.2 Transitions

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

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

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

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

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

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

    a) a dashed line to or from a state machine name label with double underlines and/or state name label with double underlines, if the destination is in a different figure from the source;

    b) a dashed line to or from a state in another state machine in the same figure; or

    c) a dashed line from a state machine name label with double underlines to a "(to all states)" label, if the destination is every state in the state machine.

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

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

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

3.6.4 State machine counters, timers, and variables

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

    a) their scope is the state machine itself;

    b) they are created and deleted with the state machines with which they are associated;

    c) their initialization and modification is specified in the state descriptions and the transition descriptions; and

    d) their current values may be used to determine the behavior of a state and select the transition out of a state.

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

3.6.5 State machine arguments

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

    a) the state machine that sends the argument owns that argument's value;

    b) the state machine that receives the argument shall not modify those argument's values;

    c) the state machine that sends the argument may resend those arguments with different values;

    d) the scope of a state machine argument is the state machine itself; and

    e) state machine argument usage is described in the state descriptions and the transition descriptions.

3.7 Bit and byte ordering

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

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

    NOTE 3 - SATA numbers bits within fields the same as this standard, but uses little-endian byte ordering.

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

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

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

A data dword consists of 32 bits. Table 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

3.8 Notation for procedures and functions

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

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

where:

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

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

4 General

4.1 Architecture

4.1.1 Architecture overview

A SAS domain (see 4.1.7) contains one or more SAS devices and a service delivery subsystem. A SAS domain may be a SCSI domain (see SAM-3).

A SAS device (see 4.1.4) contains one or more SAS ports (see 4.1.3). A SAS device may be a SCSI device (see SAM-3).

A SAS port (see 4.1.3) contains one or more phys (see 4.1.2). A SAS port may be a SCSI port (see SAM-3).

The service delivery subsystem (see 4.1.6) in a SAS domain may contain expander devices (see 4.1.5).

Expander devices contain expander ports (see 4.1.3) and one SMP port.

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

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

Figure 10 shows the class diagram for a SAS domain, showing the relationships between SAS domain, SCSI domain, service delivery subsystem, expander device, expander port, SAS device, SCSI device, SAS port, SCSI port and phy classes. Not all attributes are shown.

Figure 10 — SAS domain class diagram

4.1.2 Physical links and phys

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

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

Phys are contained in ports (see 4.1.3). Phys interface to the service delivery subsystem (see 4.1.6).

Figure 11 shows two phys attached with a physical link.
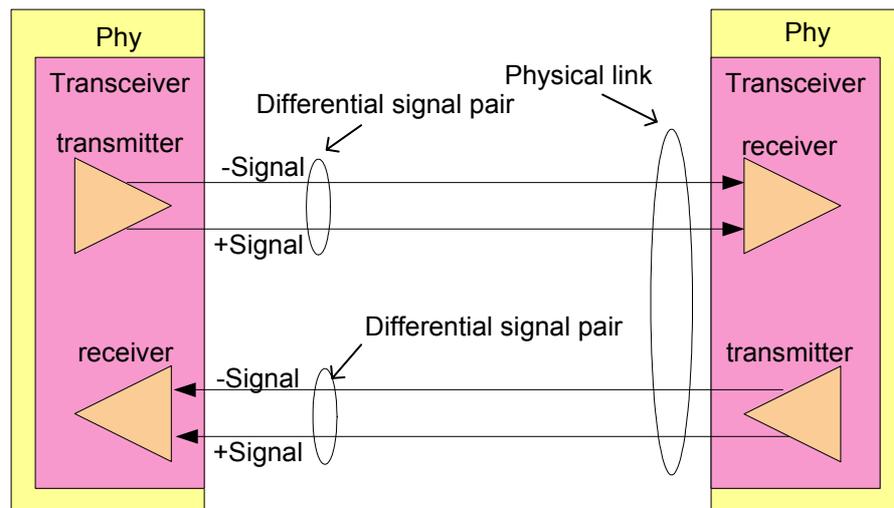
Figure 11 — Physical links and phys

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

A device (i.e., a SAS device (see 4.1.4) or expander device (see 4.1.5)) contains one or more phys.

Each phy has:

a) a SAS address (see 4.2.2), inherited from the SAS port (see 4.1.3) or expander device;
b) a phy identifier (see 4.2.7) which is unique within the device;
c) optionally, support for being an SSP initiator phy;
d) optionally, support for being an STP initiator phy;
e) optionally, support for being an SMP initiator phy;
f) optionally, support for being an SSP target phy;
g) optionally, support for being an STP target phy; and
h) optionally, support for being an SMP target phy.

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

a) transmits an IDENTIFY address frame including the device type (i.e., end device, edge expander device, or fanout expander device) of the device containing the phy, the SAS address of the SAS port or expander device containing the phy, phy identifier, SSP initiator phy capability, STP initiator phy capability, SMP initiator phy capability, SSP target phy capability, STP target phy capability and SMP target phy capability.
b) receives an IDENTIFY address frame containing the same set of information from the attached phy, including the attached device type, attached SAS address, attached phy identifier, attached SSP initiator phy capability, attached STP initiator phy capability, attached SMP initiator phy capability, attached SSP target phy capability, attached STP target phy capability, and attached SMP target phy capability.

The transceiver follows the electrical specifications defined in 5.3. Phys transmit and receive bits at physical link rates defined in 5.3. The physical link rates supported by a phy are specified or indicated by the NEGOTIATED PHYSICAL LINK RATE field, HARDWARE MINIMUM PHYSICAL LINK RATE field, the HARDWARE MAXIMUM PHYSICAL LINK RATE field, the PROGRAMMED MINIMUM PHYSICAL LINK RATE field, and the PROGRAMMED MAXIMUM PHYSICAL LINK RATE field in the SMP DISCOVER function (see 10.4.3.5), SMP PHY CONTROL function (see 10.4.3.10) and Phy Control and Discover subpage (see 10.2.7.2.3). The bits are part of dwords (see 6.2.1), each of which has been encoded using 8b10b coding into four 10-bit characters (see 6.2).

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

a) phy;
b) SAS phy;
c) expander phy;
d) SAS initiator phy;
e) SAS target phy;
f) SSP phy;
g) STP phy; and

h) SMP phy.

SATA phys are also referenced in this standard but are defined by SATA (see ATA/ATAPI-7 V3).
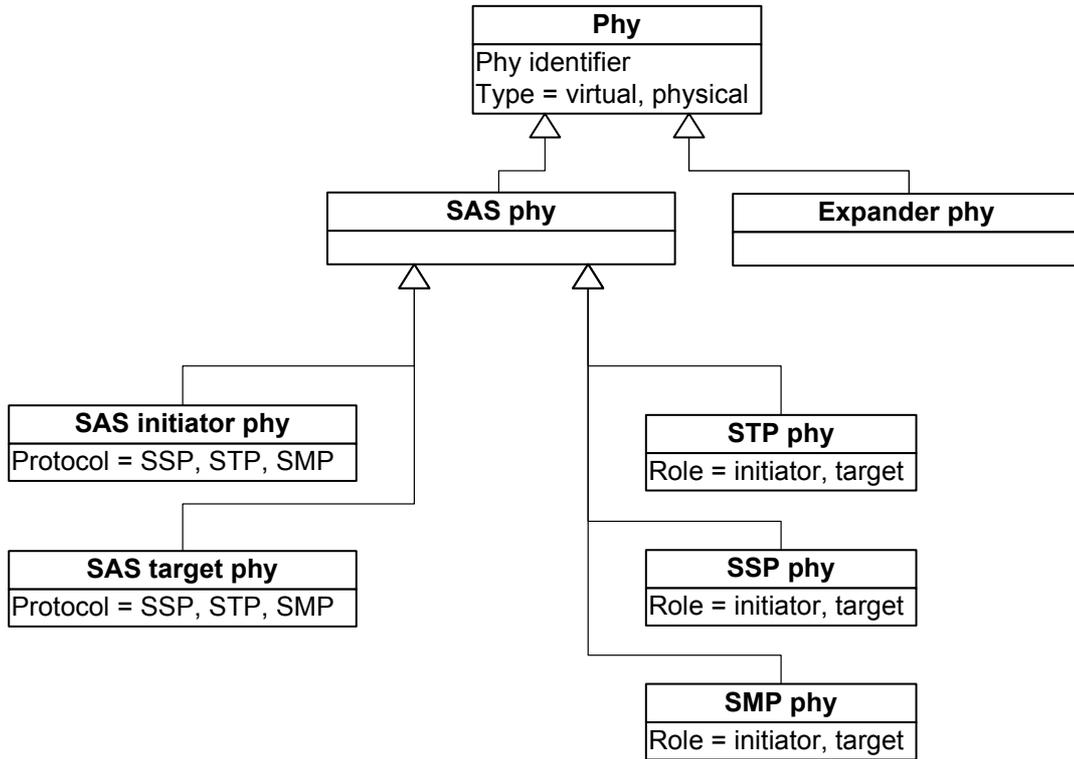
Figure 12 — Phy class diagram

Figure 13 shows the objects instantiated from the phy classes, including:

a) from the SAS phy class:
    A) SSP initiator phy;
    B) SSP target phy;
    C) virtual SSP initiator phy;
    D) virtual SSP target phy;
    E) STP initiator phy;
    F) STP target phy;
    G) virtual STP initiator phy;
    H) virtual STP target phy;
    I) SMP initiator phy;
    J) SMP target phy;
    K) virtual SMP initiator phy; and
    L) virtual SMP target phy;

    and

b) from the expander phy class:
    A) expander phy; and
    B) virtual expander phy.

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

Valid objects for the expander phy class:

Expander phy : Expander phy
Phy identifier Type = physical

Virtual expander phy : Expander phy
Phy identifier Type = Virtual

Valid objects for the SAS phy class:

SSP initiator phy : SAS phy
Phy identifier Type = physical Role = initiator Protocol = SSP

SSP target phy : SAS phy
Phy identifier Type = physical Role = target Protocol = SSP

STP initiator phy : SAS phy
Phy identifier Type = physical Role = initiator Protocol = STP

STP target phy : SAS phy
Phy identifier Type = physical Role = target Protocol = STP

SMP initiator phy : SAS phy
Phy identifier Type = physical Role = initiator Protocol = SMP

SMP target phy : SAS phy
Phy identifier Type = physical Role = target Protocol = SMP

Virtual SSP initiator phy : SAS phy
Phy identifier Type = virtual Role = initiator Protocol = SSP

Virtual SSP target phy : SAS phy
Phy identifier Type = virtual Role = target Protocol = SSP

Virtual STP initiator phy : SAS phy
Phy identifier Type = virtual Role = initiator Protocol = STP

Virtual STP target phy : SAS phy
Phy identifier Type = virtual Role = target Protocol = STP

Virtual SMP initiator phy : SAS phy
Phy identifier Type = virtual Role = initiator Protocol = SMP

Virtual SMP target phy : SAS phy
Phy identifier Type = virtual Role = target Protocol = SMP

Figure 13 — Phy object diagram

4.1.3 Ports (narrow ports and wide ports)

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

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

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

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

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

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

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

Figure 14 shows examples of narrow ports and wide ports, with a representation of the SAS address transmitted during the identification sequence. Although several phys on the left transmit SAS addresses of B, only phys attached to the same SAS addresses become part of the same ports. The set of phys with SAS address B attached to the set of phys with SAS address Y become one port, while the set of phys with SAS address B attached to the set of phys with SAS address Z become another port.
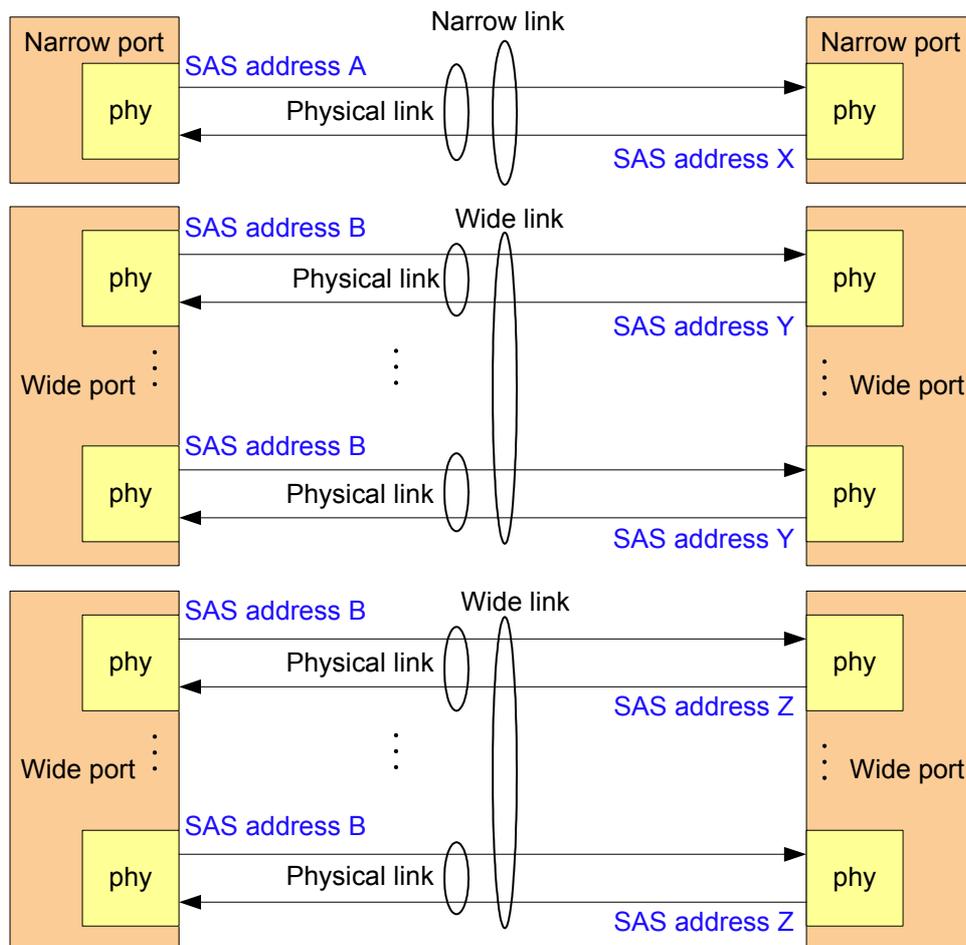
Each horizontal line represents a differential signal pair

Figure 14 — Ports (narrow ports and wide ports)

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

Figure 15 defines the port classes, showing the relationships between the following classes:

    a)   port;
    b)   expander port;
    c)   SAS port;
    d)   SAS initiator port;
    e)   SAS target port;
    f)   SSP port;
    g)   STP port; and
    h)   SMP port.
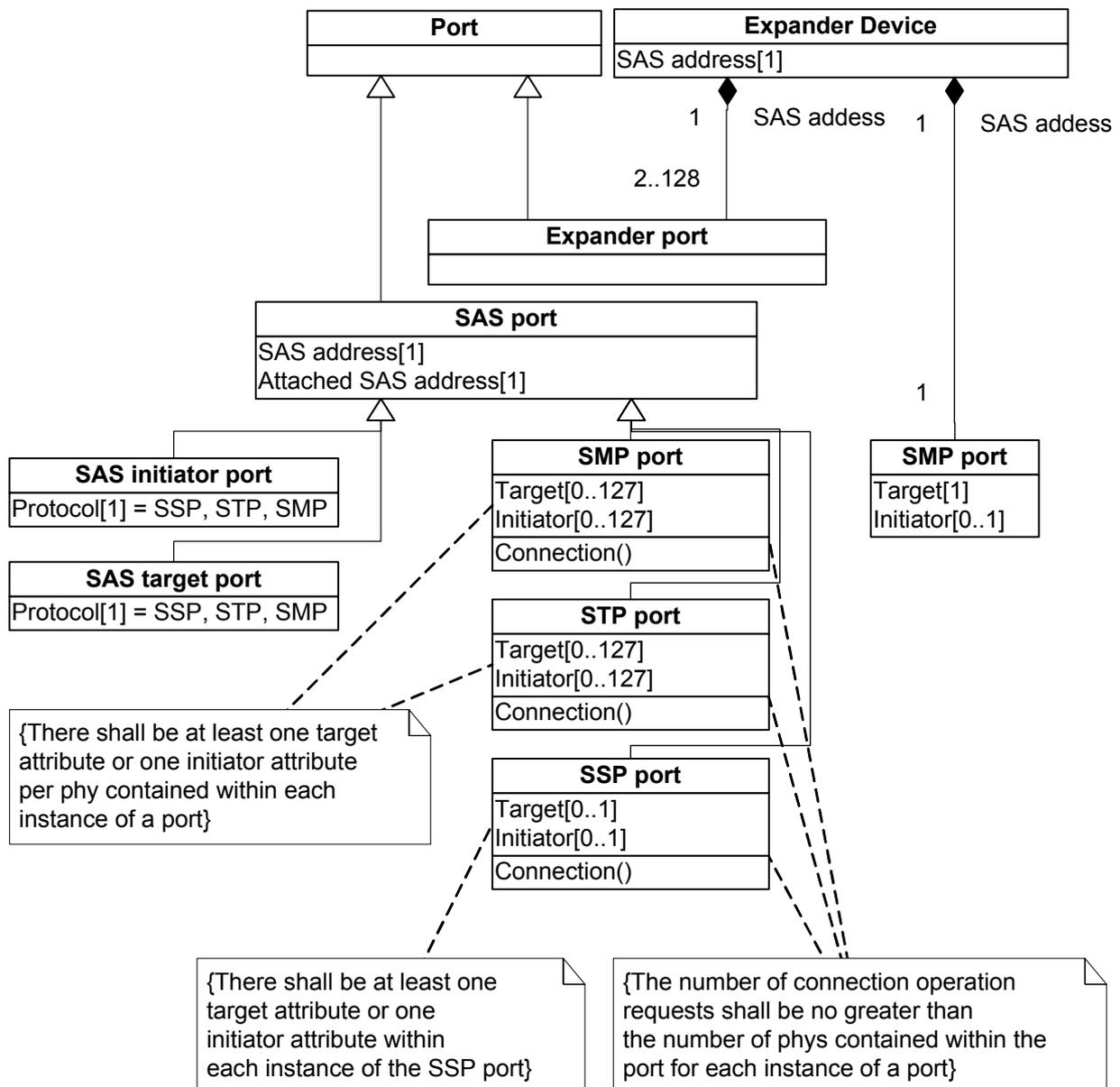
Figure 15 — Port class diagram

Figure 16 shows the objects instantiated from the port classes:

    a)   SAS target port class (i.e., SSP target port, STP target port, SMP target port);
    b)   SAS initiator port class (i.e., SSP initiator port, STP initiator port, SMP initiator port);
    c)   STP port class (i.e., STP initiator port, STP target port, STP port);

d) SMP port class (i.e., SMP initiator port, SMP target port, SMP port);
e) SSP port class (i.e., SSP initiator port, SSP target port, SSP port);
f) expander device SMP port class (i.e., SMP target port, SMP port); and
g) expander port class (i.e., expander port).

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

Valid objects for the
SAS initiator port class

SSP initiator port : SAS initiator port
SAS address
Attached SAS address
Protocol = SSP

STP initiator port : SAS initiator port
SAS address
Attached SAS address
Protocol = STP

SMP initiator port : SAS initiator port
SAS address
Attached SAS address
Protocol = SMP

Valid objects for the
SAS target port class

SSP target port : SAS target port
SAS address
Attached SAS address
Protocol = SSP

STP target port : SAS target port
SAS address
Attached SAS address
Protocol = STP

SMP target port : SAS target port
SAS address
Attached SAS address
Protocol = SMP

Valid objects for the
expander device
SMP port class

SMP target port : SMP port
Target

SMP port : SMP port
Target
Initiator

Valid object for the
expander port class

Expander port : Expander port
SAS address
Attached SAS address

Valid objects
for the SSP port class

SSP target port : SSP port
SAS address
Attached SAS address
Target

SSP initiator port : SSP port
SAS address
Attached SAS address
Initiator

SSP port : SSP port
SAS address
Attached SAS address
Initiator
Target

Examples of valid objects
for the STP port class

STP target port : STP port
SAS address
Attached SAS address
Target01
Target02

STP initiator port : STP port
SAS address
Attached SAS address
Initiator

STP port : STP port
SAS address
Attached SAS address
Target01
Target02
Initiator

Examples of valid objects
for the SMP port class

SMP target port : SMP port
SAS address
Attached SAS address
Target

SMP initiator port : SMP port
SAS address
Attached SAS address
Initiator01
Initiator02

SMP port : SMP port
SAS address
Attached SAS address
Target01
Target02
Initiator01
Initiator02

Figure 16 — Port object diagram

4.1.4 SAS devices

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

Figure 17 shows examples of SAS devices with different port and phy configurations.

Each horizontal line represents a differential signal pair
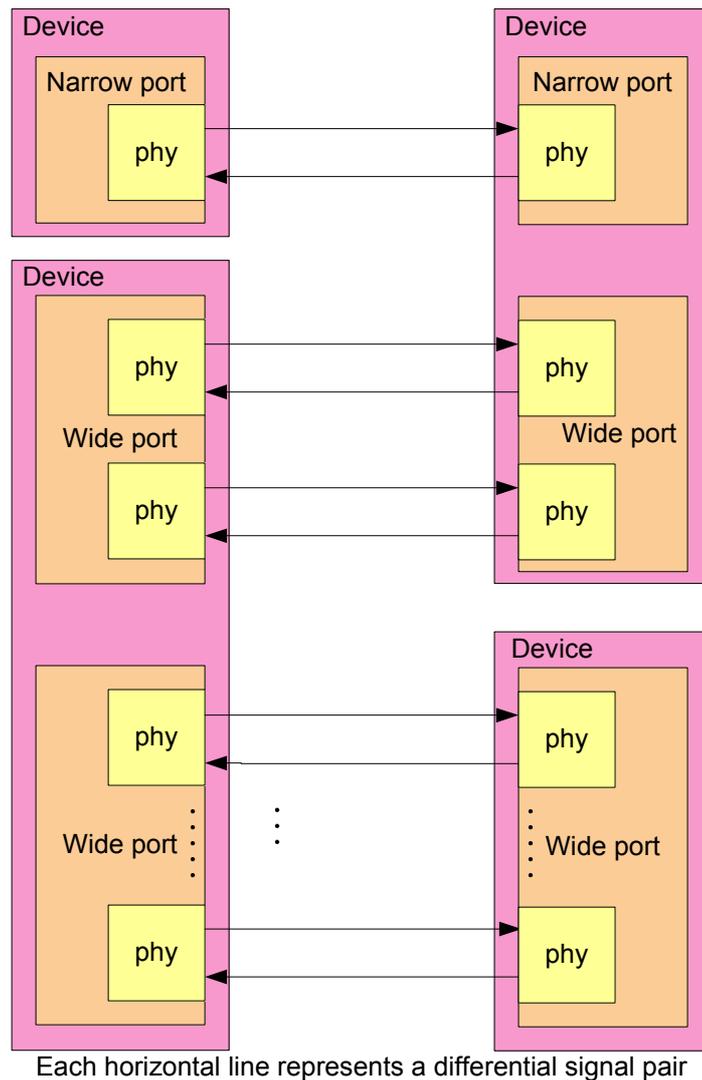
Figure 17 — SAS devices

An end device is a SAS device that is not contained in an expander device (see 4.1.5).

4.1.5 Expander devices (edge expander devices and fanout expander devices)

Expander devices are part of the service delivery subsystem and facilitate communication between multiple SAS devices. Expander devices contain two or more external expander ports. Each expander device contains at least one SMP target port for management and may contain SAS devices (e.g., an expander device may include an SSP target port for access to a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-3 and SES-2)).

Figure 18 shows an expander device.
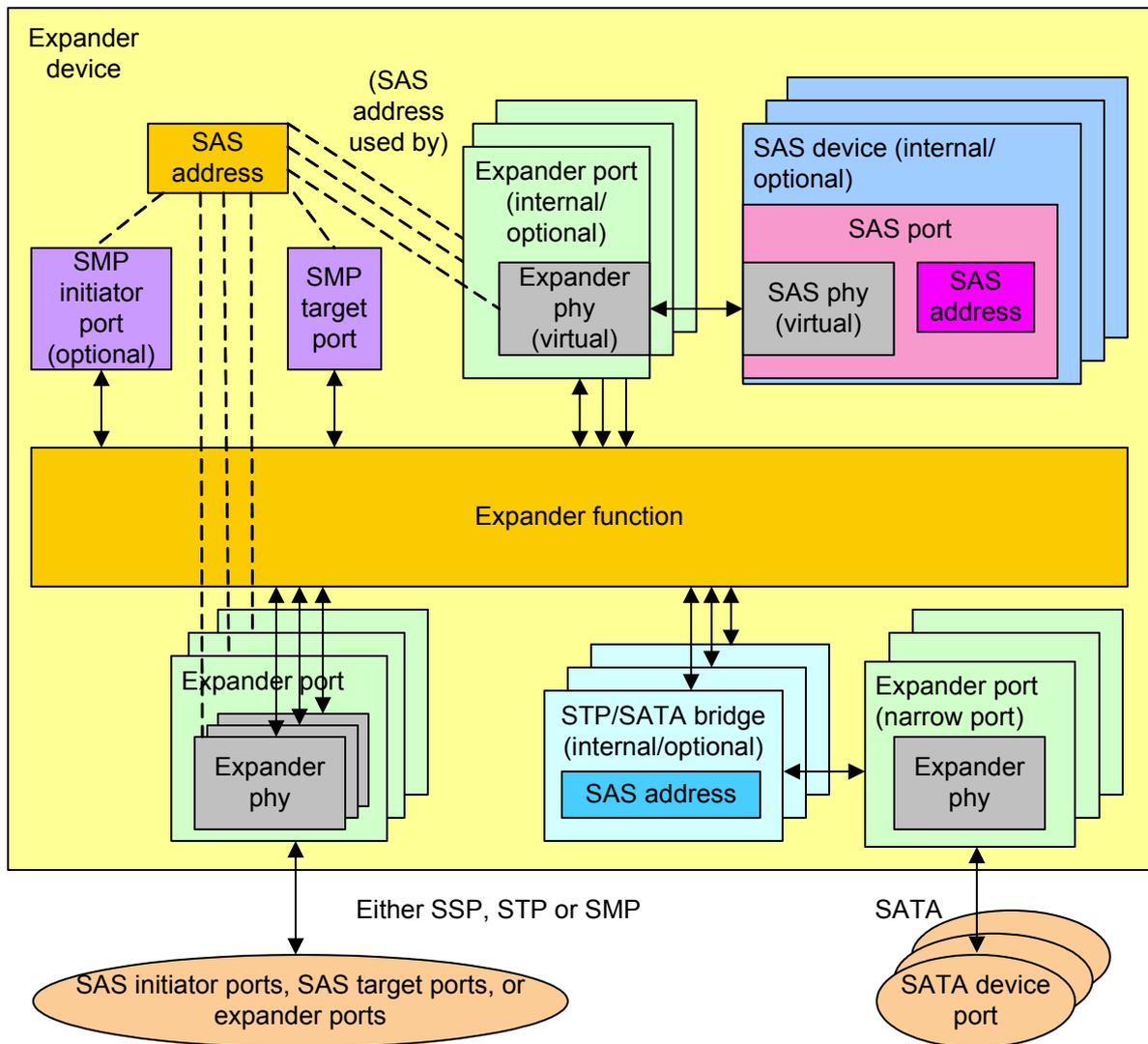
Figure 18 — Expander device

There are two types of expander devices:

    a) edge expander devices; and
    b) fanout expander devices.

An edge expander device is part of an edge expander device set (see 4.1.8.2). A fanout expander device is an expander device capable of being attached to two or more edge expander device sets.

See 4.6 for a detailed model of an expander device.

Figure 19 shows the expander device classes, showing the relationship between the following classes:

    a) expander device;
    b) edge expander device;
    c) edge expander device set; and
    d) fanout expander device.
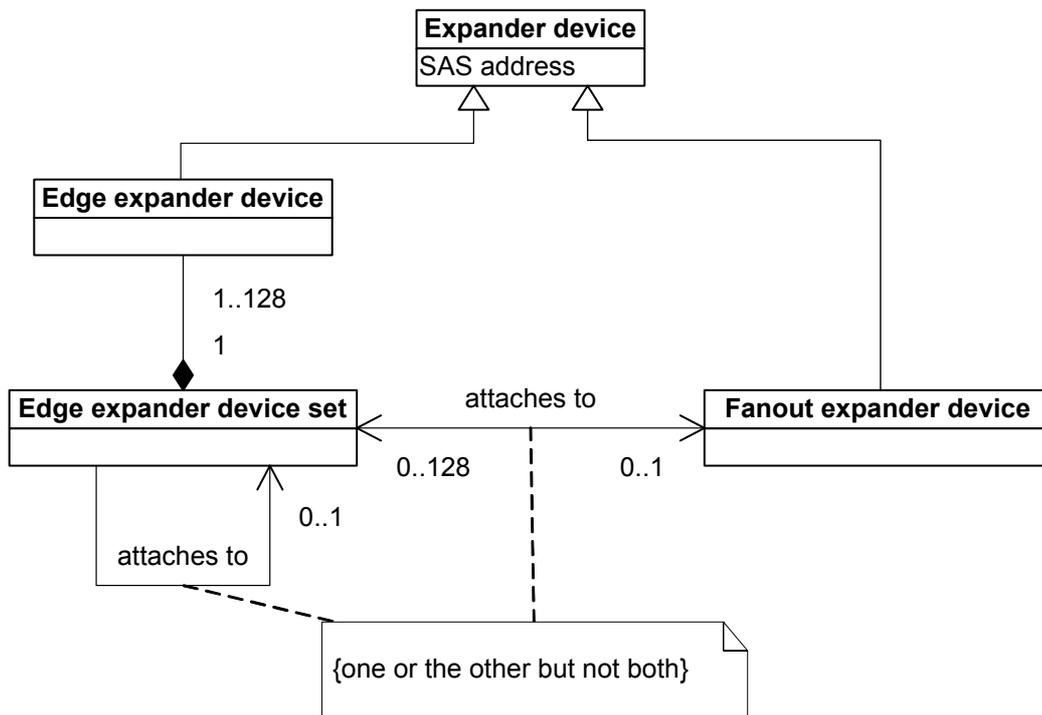
Figure 19 — Expander device class diagram

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

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

Edge expander devices may contain phys with the subtractive routing attribute. A fanout expander device shall not contain any phys with the subtractive routing attribute. Edge expander devices and fanout expander devices may also contain phys with direct routing and table routing attributes.

Expander devices with expander phys with the table routing attribute contain an expander route table. The expander route table may be configurable. An expander device with a configurable route table depends on a management application client within the SAS domain to use the discover process (see 4.7) to configure the expander route table. An expander device with expander phys with the table routing attribute that does not have a configurable route table shall be self-configuring and shall contain a management application client and SMP initiator port to perform the discover process to configure its own expander route table.

4.1.6 Service delivery subsystem

The service delivery subsystem is either:

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

See 4.1.8 for rules on constructing service delivery subsystems from multiple expander devices.

4.1.7 Domains

Figure 20 shows examples of SAS domains and ATA domains.
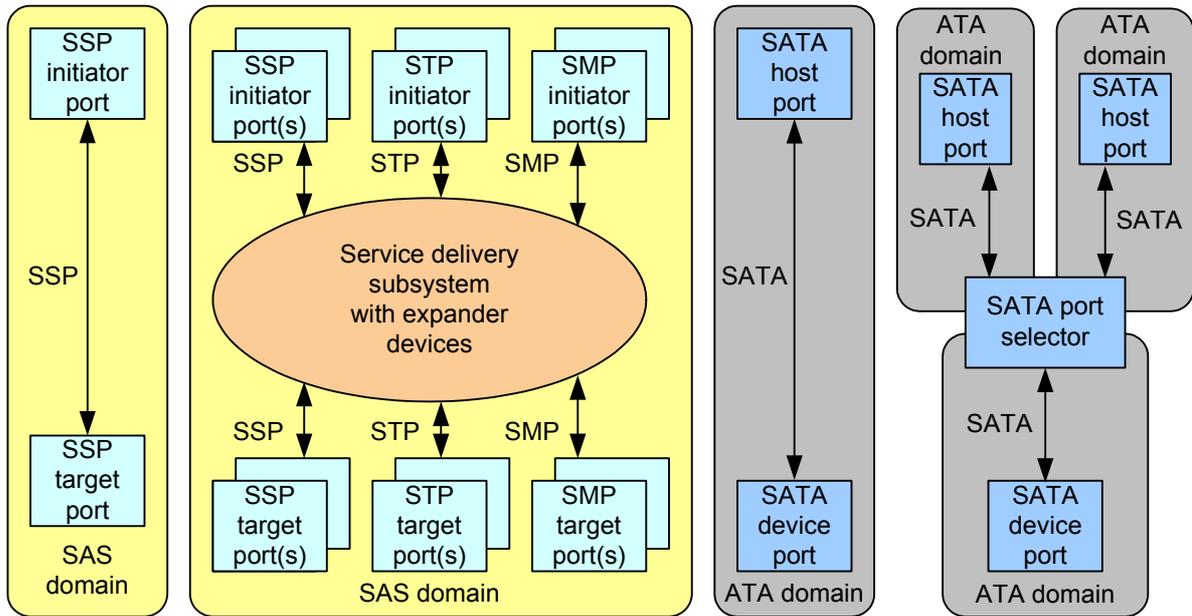
Note: When expander devices are present, SAS target ports may be
located in SAS devices contained in expander devices.

Figure 20 — Domains

Figure 21 shows a SAS domain bridging to one or more ATA domains.
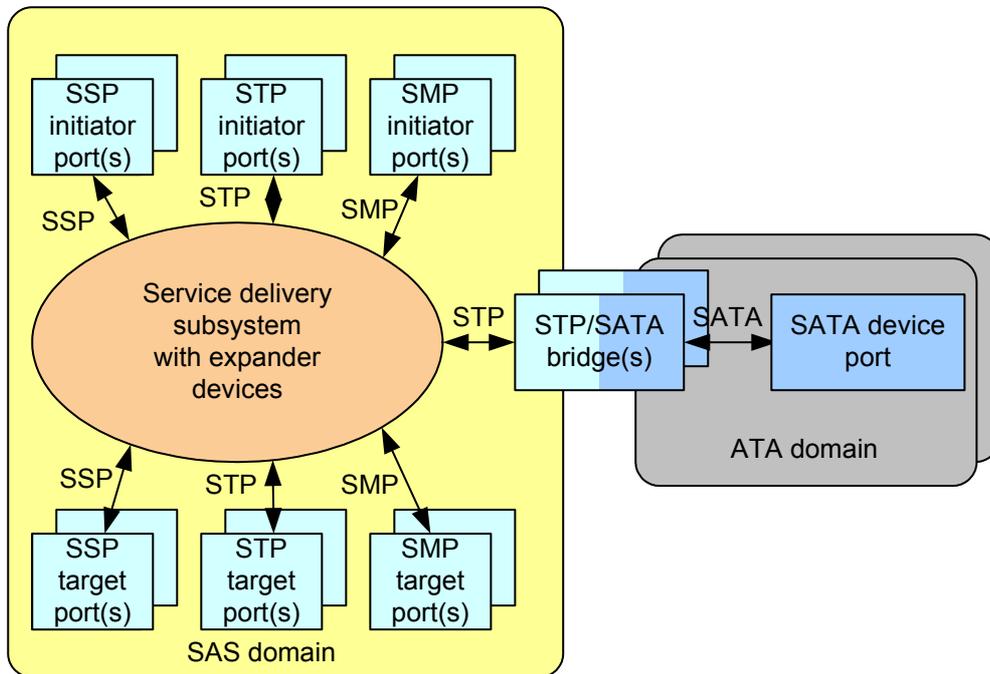
Figure 21 — SAS domain bridging to ATA domains

Figure 22 shows two SAS domains bridging to one or more ATA domains containing SATA devices with SATA port selectors.
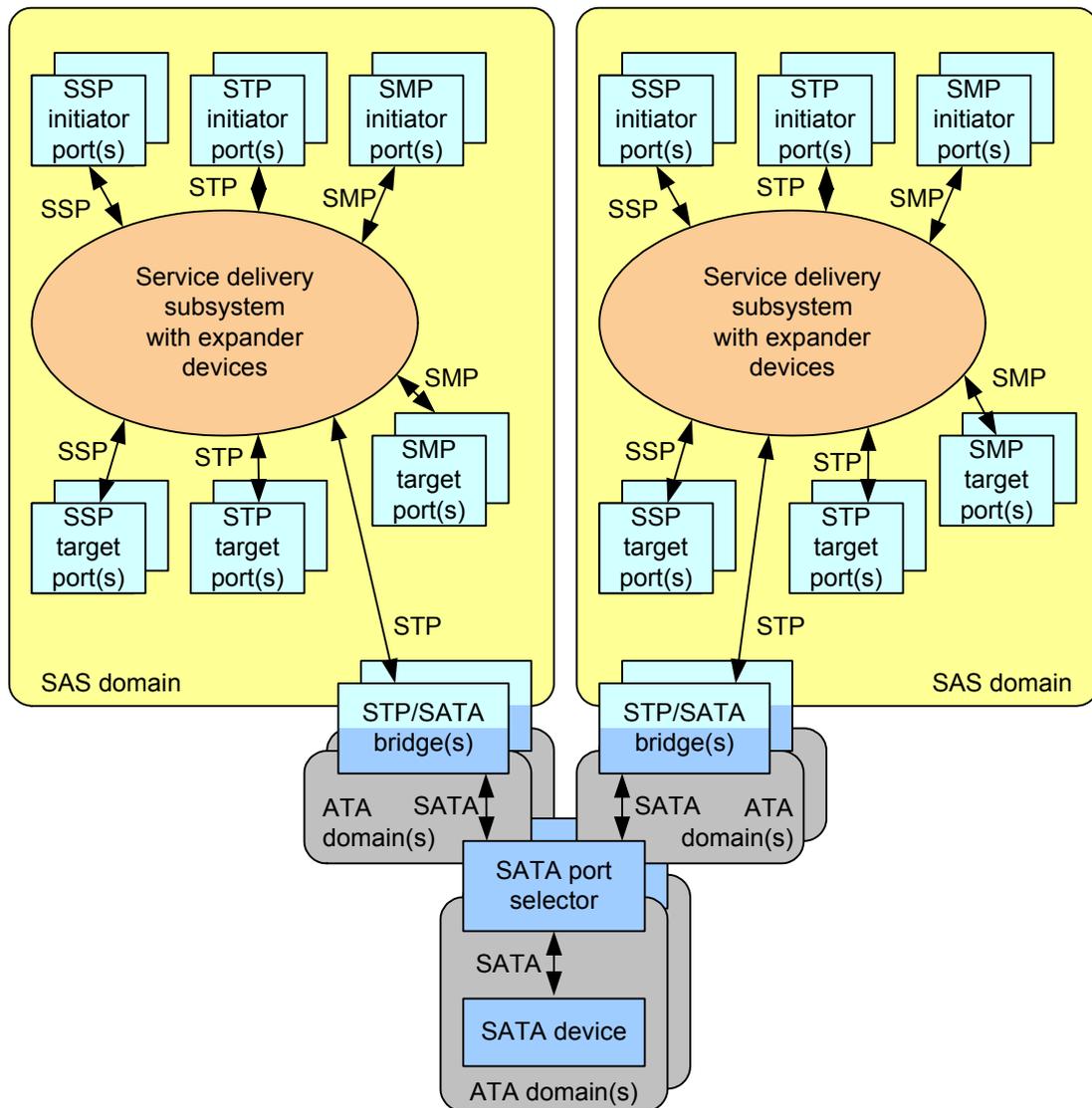
Figure 22 — SAS domains bridging to ATA domains with SATA port selectors

Figure 23 shows SAS initiator devices and SAS target devices with SAS ports in the same SAS domains and in different SAS domains. When a SAS device has ports in different SAS domains, the ports may have the same SAS address (see 4.2); when its ports are in the same SAS domain, they shall have different SAS addresses.
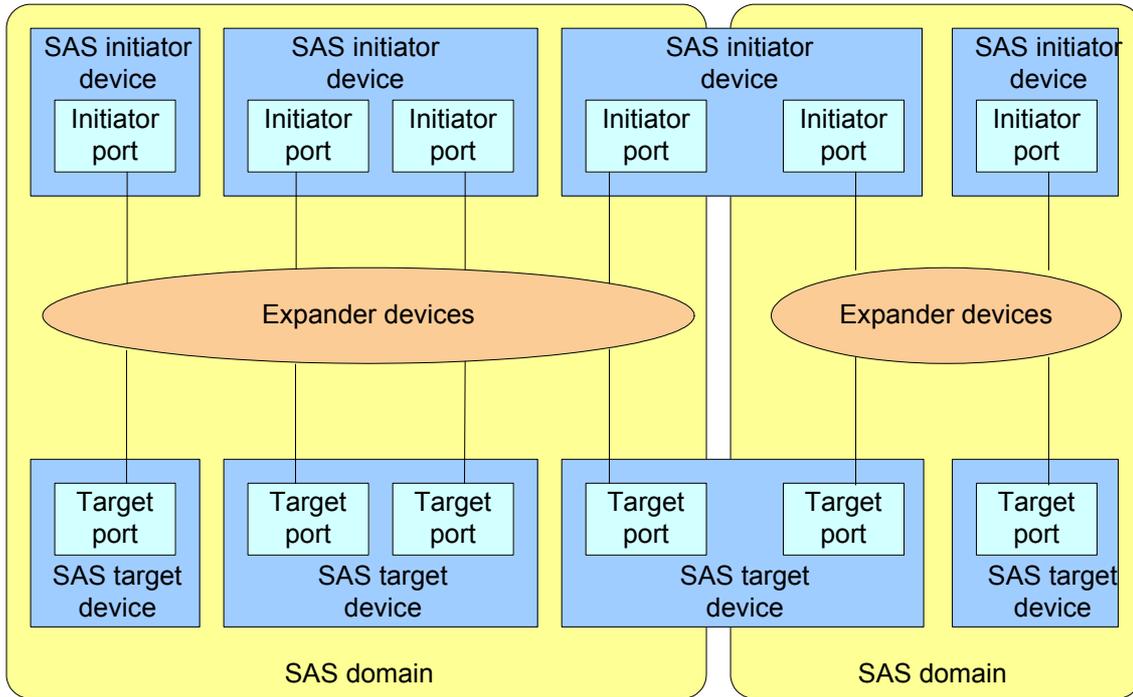
Figure 23 — Devices spanning SAS domains

4.1.8 Expander device topologies

4.1.8.1 Expander device topology overview

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

4.1.8.2 Edge expander device set

One or more edge expander devices may be grouped into an edge expander device set. The number of edge expander devices and the routing attributes (see 4.6.7.1) of the expander phys in the edge expander devices within an edge expander device set shall be established when the edge expander device set is constructed. The method used to construct edge expander sets is outside the scope of this standard. The routing method used by an expander phy depends on the type of device to which it is attached (see 4.6.7.1).

An edge expander device set is bounded by expander phys with the direct routing attribute that are either

    a) attached to end devices or
    b) not attached to any device,

and by expander phys with the subtractive routing attribute that are

    a) attached to expander phys of a fanout device,
    b) attached to expander phys with subtractive routing attributes on another edge expander device set,
    c) attached to an end device or
    d) not attached to any device.

Phys with table routing attributes within an edge expander device set are used to provide attachments between edge expander devices in the edge expander device set.

The number of SAS addresses used by an edge expander device set shall not exceed 128. This includes SAS addresses for

    a) the edge expander devices within the edge expander device set,
    b) SAS devices contained in the expander devices,
    c) STP target ports in STP/SATA bridges contained in the expander devices and
    d) the end devices potentially attached to the edge expander device set.

To avoid an overflow of an edge expander route index during the discover process (see 4.7), an edge expander device set shall be constructed such that the number of edge expander route indexes available for each edge expander phy is greater than or equal to the number of SAS addresses addressable through that edge expander phy.

Figure 24 shows an edge expander device set.
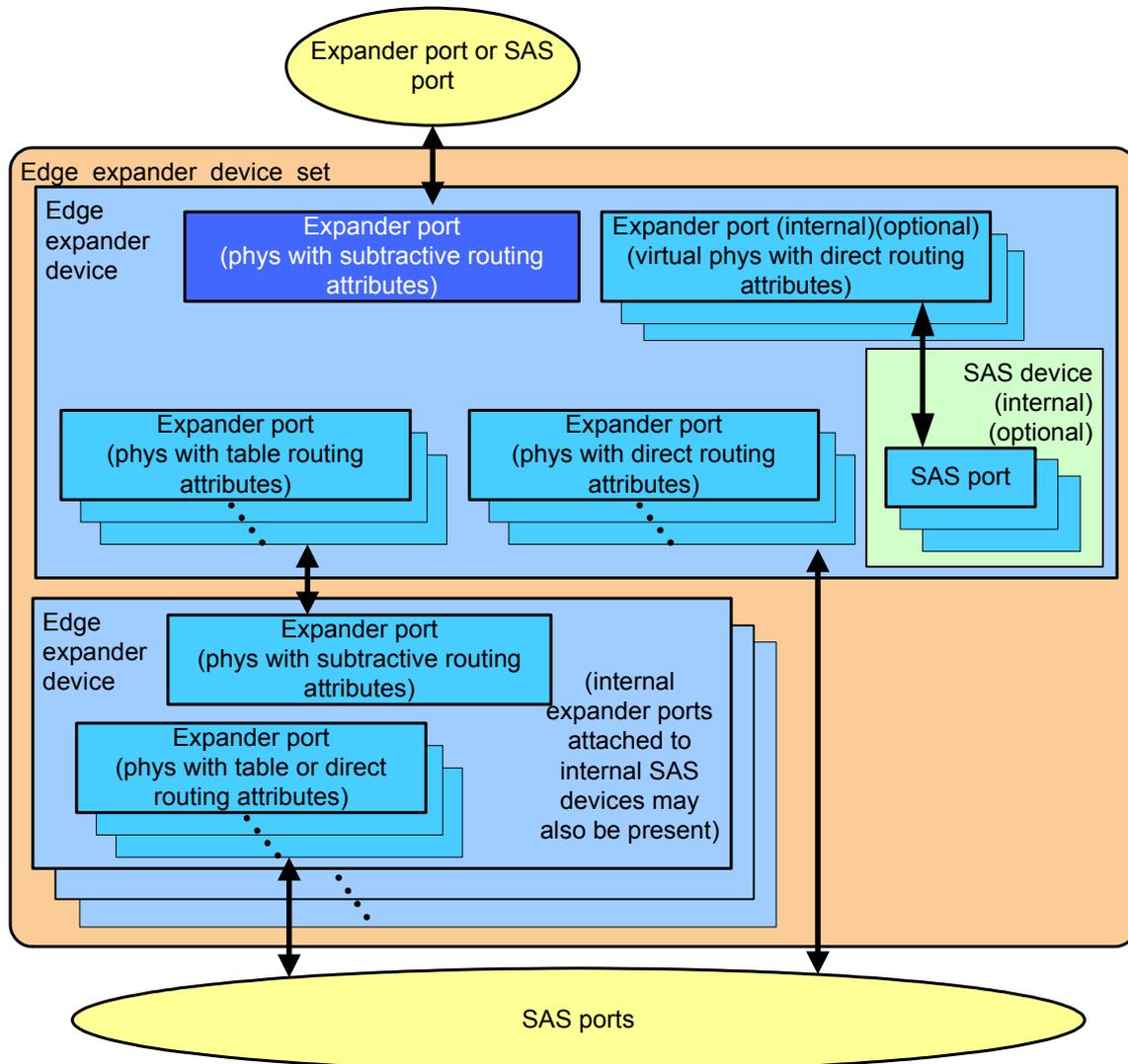
Figure 24 — Edge expander device set

4.1.8.3 Expander device topologies

An 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

    a) there are only two edge expander device sets in the SAS domain,
    b) the edge expander device sets are attached using expander phys with subtractive routing attributes and
    c) there are no fanout expander devices in the SAS domain.

No more than one fanout expander device shall be included in a SAS domain. The fanout expander device may be attached to up to 128 edge expander device sets or SAS ports.

Figure 25 shows a SAS domain with the maximum number of expander device sets.
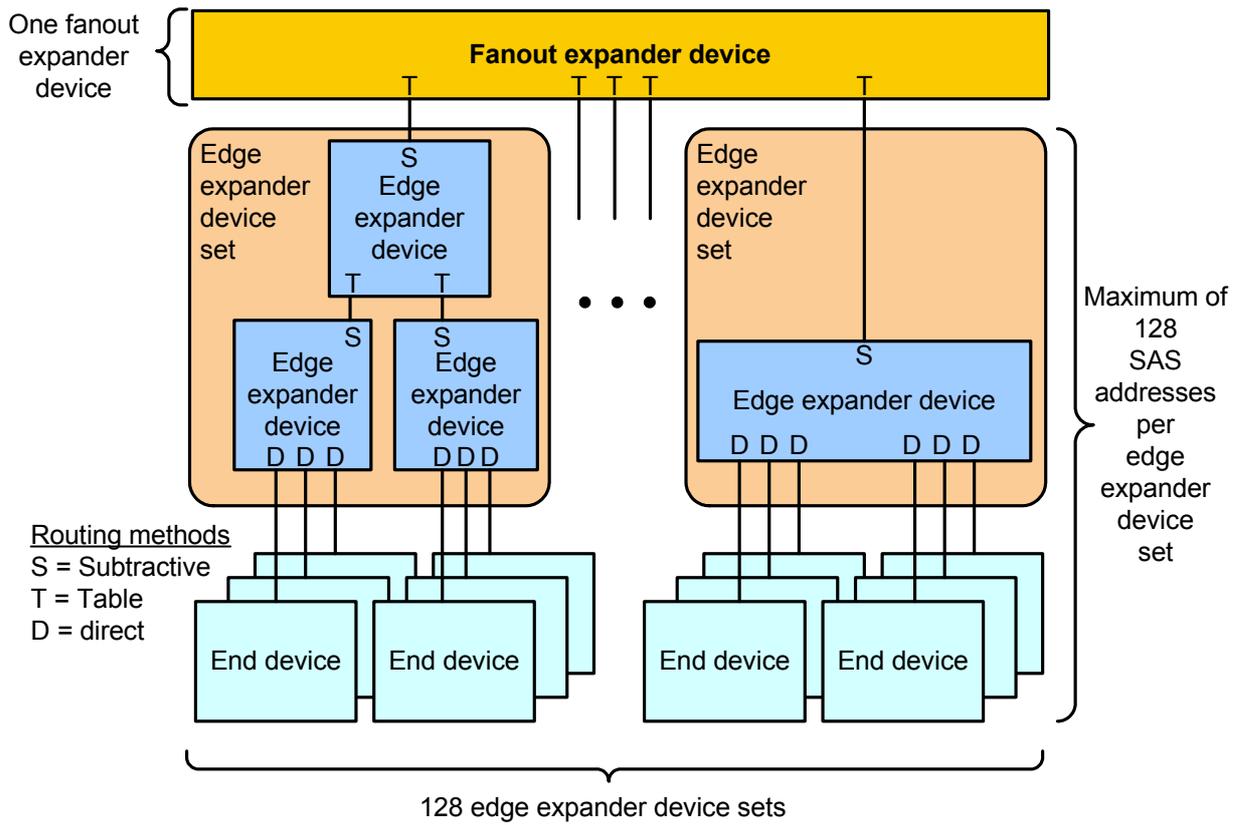
Figure 25 — Maximum expander device set topology

End devices may be attached directly to the fanout expander device, as shown in figure 26.
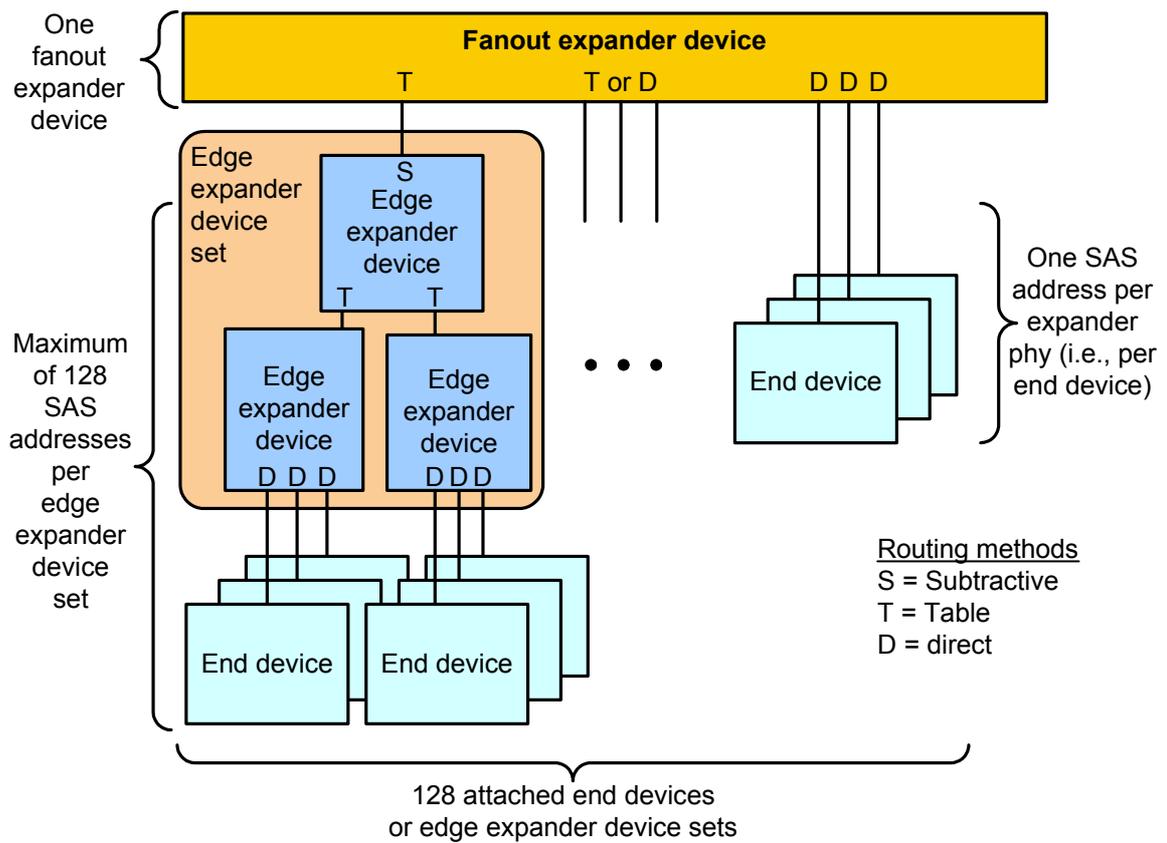
Figure 26 — Fanout expander device topology

SAS initiator devices and SAS 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 27.

Figure 27 — Edge expander device set to edge expander device set topology

4.1.9 Pathways

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

A pathway is a set of physical links between a SAS initiator phy and a SAS target phy being used by a connection (see 4.1.10).

Figure 28 shows examples of potential pathways.
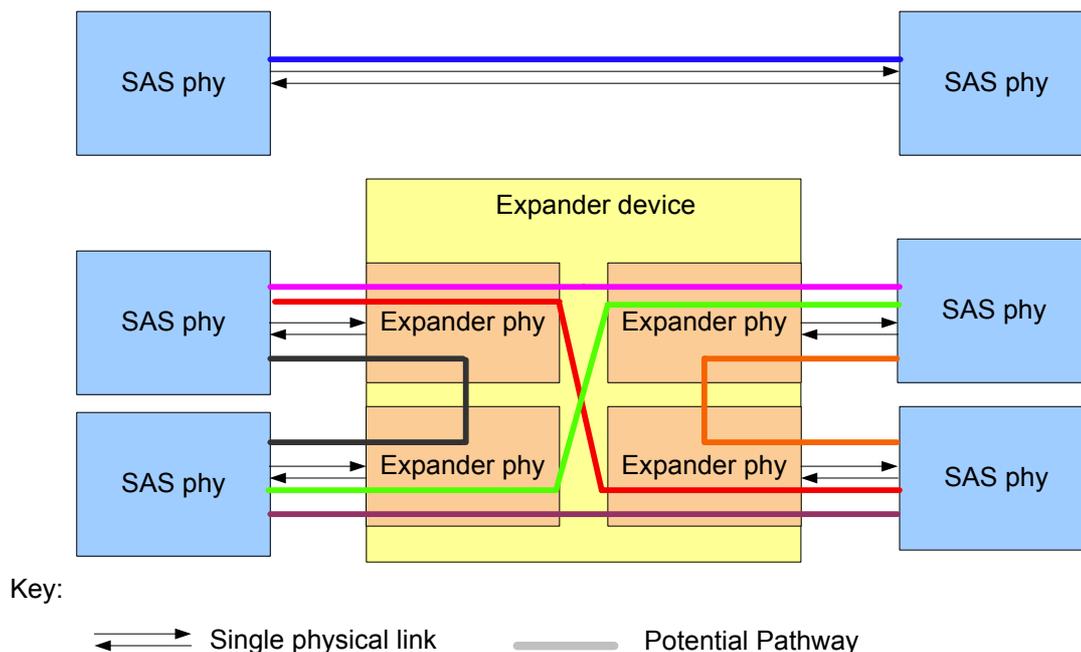
Figure 28 — Potential pathways

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

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

4.1.10 Connections

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

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

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

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

No more than one connection is 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.

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

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

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

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

Figure 29 shows examples of connections between wide and narrow ports. All the connections shown may occur simultaneously. Additionally

  a) the connections labeled A and B are an example of one SAS initiator port with connections to multiple SAS target ports,
  b) the connections labeled A and C are an example of one SAS target port with connections to multiple SAS initiator ports,
  c) the connections labeled E and F are an example of multiple connections between one SAS initiator port and one SAS target port and
  d) the connections labeled C, D, E, and F are an example of one SAS initiator port with connections to multiple SAS target ports with one of those SAS target ports having multiple connections with that SAS initiator port.
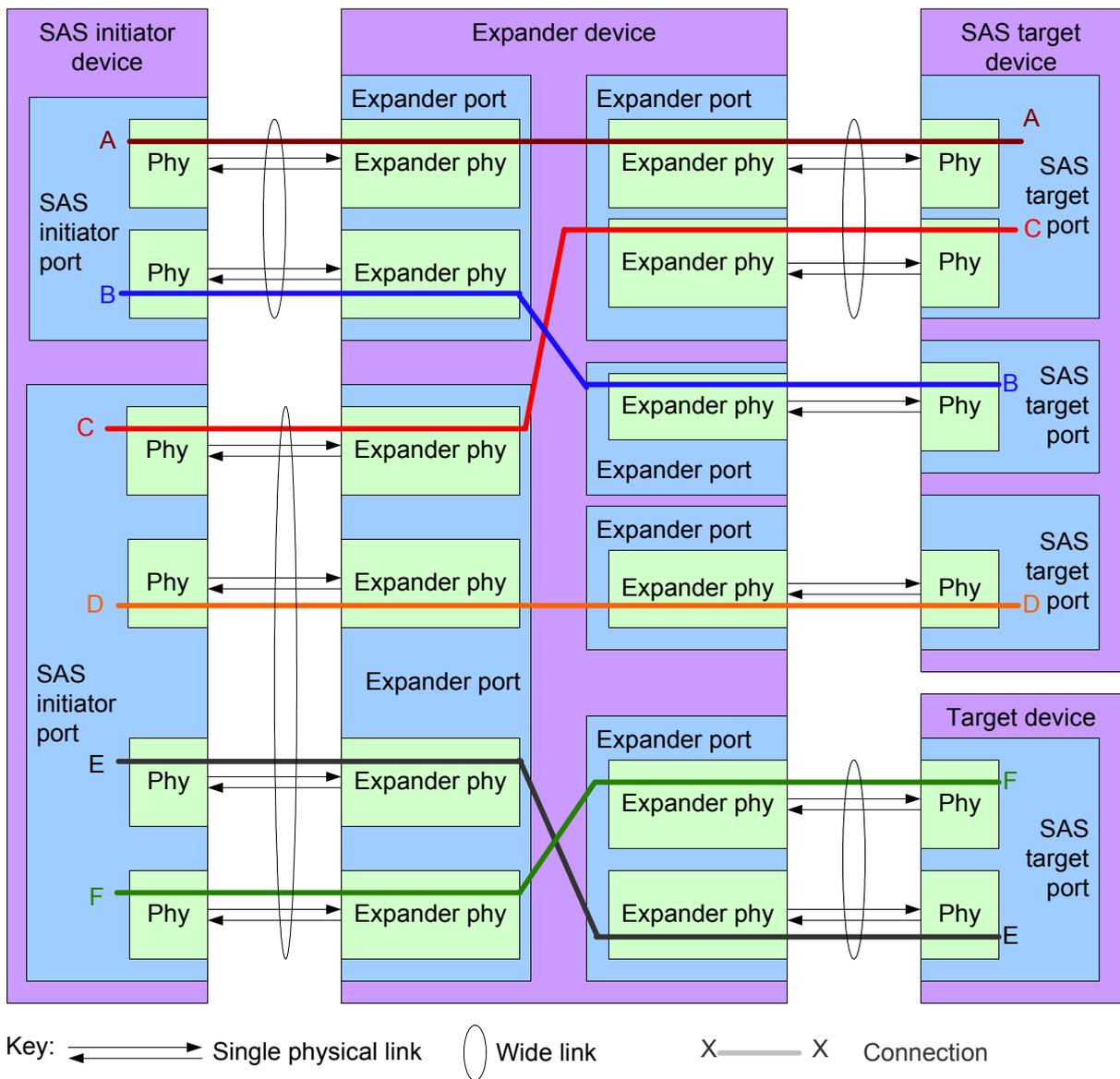
Note: The expander device has a unique SAS address. Each SAS initiator port and SAS target port has a unique SAS address. Connections E and F represent a wide SAS initiator port with two simultaneous connections to a wide SAS target port.

Figure 29 — Multiple connections on wide ports

4.2 Names and identifiers

4.2.1 Names and identifiers overview

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

Table 5 describes the definitions of names and identifiers for SAS.

Table 5 — Names and identifiers

Attribute	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 attributes are implemented in SSP.

Table 6 — SAM-3 attribute mapping

SAM-3 attribute	SSP implementation
Initiator port identifier	SAS address of SSP initiator port
Initiator port name	Not defined
Target port identifier	SAS address of SSP target port
Target port name	Not defined
SCSI device name	SAS address of SCSI device

4.2.2 SAS address

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

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

NOTE 4 - Information about IEEE company identifiers may be obtained from the IEEE Registration Authority web site at http://standards.ieee.org/regauth/oui.

The VENDOR-SPECIFIC IDENTIFIER field contains a 36-bit numeric value that is assigned by the organization associated with the company identifier in the IEEE COMPANY ID field. The VENDOR-SPECIFIC IDENTIFIER field shall be assigned so the SAS address is 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 to provide an additional level of verification of proper frame routing.

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

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

Annex E contains information on SAS address hashing.

4.2.4 Device names

Each expander device, SAS initiator device, SAS target device, and SAS target/initiator device shall include a SAS address (see 4.2.2) as its device name. A SAS address used as a device name shall not be used as any other name or identifier (e.g., a device name, port name, port identifier or logical unit name (see SAM-3)).

Expander devices report their device names in the IDENTIFY address frame (see 7.8.2).

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

NOTE 5 - There is no way to retrieve SAS initiator device names defined in this standard.

4.2.5 Port names

Port names are not defined in SAS.

NOTE 6 - The SAS addresses used by SAS ports in different SAS domains may be the same (e.g., when a set of phys transmit the same SAS address in the identification sequence but receive different SAS addresses, indicating they are attached to two separate SAS domains) so the SAS address serves as a port identifier (see 4.2.6) rather than a port name.

4.2.6 Port identifiers

Each SAS initiator port, SAS target port (e.g., including STP target ports in STP/SATA bridges), and SAS target/initiator port shall include a SAS address (see 4.2.2) as its port identifier. A SAS address used as a port identifier shall not be used as any other name or identifier (e.g., a device name, port name or logical unit name (see SAM-3)) but may be used as a port identifier in one or more other SAS domains (see 4.1.3).

SAS ports in end devices report their port identifiers in the IDENTIFY address frame (see 7.8.2). Expander devices containing SAS ports (e.g., SAS ports attached to virtual phys or STP target ports in STP/SATA bridges) report the port identifiers of those SAS ports in the SMP DISCOVER function (see 10.4.3.5).

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

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

4.2.7 Phy identifiers

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

Phy identifiers shall be greater than or equal to 00h and less than 80h and should be numbered starting with 00h. In an expander device or in a SAS device containing an SMP target port, phy identifiers shall be less than the value of the NUMBER OF PHYS field in the SMP REPORT GENERAL function (see 10.4.3.3). In a SAS device containing an SSP target port, phy identifiers shall be less than the value of the NUMBER OF PHYS field in the Protocol-Specific Port mode page for SAS SSP - Phy Control And Discover subpage (see 10.2.7.2.3).

4.3 State machines

4.3.1 State machine overview

Figure 30 shows the state machines for SAS devices, their relationships to each other and to the SAS device, SAS port, and SAS phy classes.
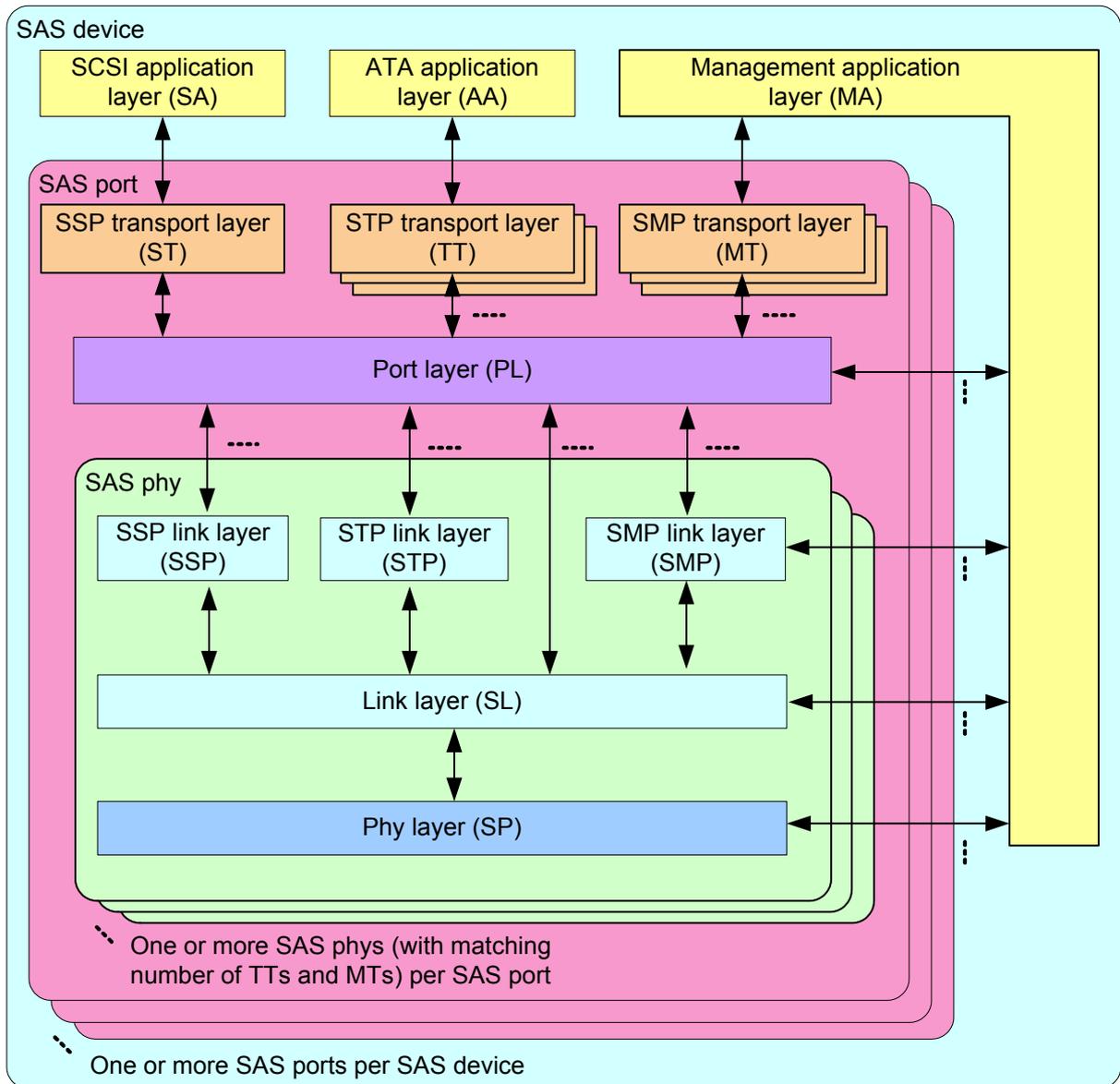
Figure 30 — State machines for SAS devices

Figure 31 shows the state machines for expander devices, their relationships to each other and to the expander device, expander port and expander phy classes. Expander function state machines are not defined in this standard, but the interface to the expander function is defined in 4.6.6.
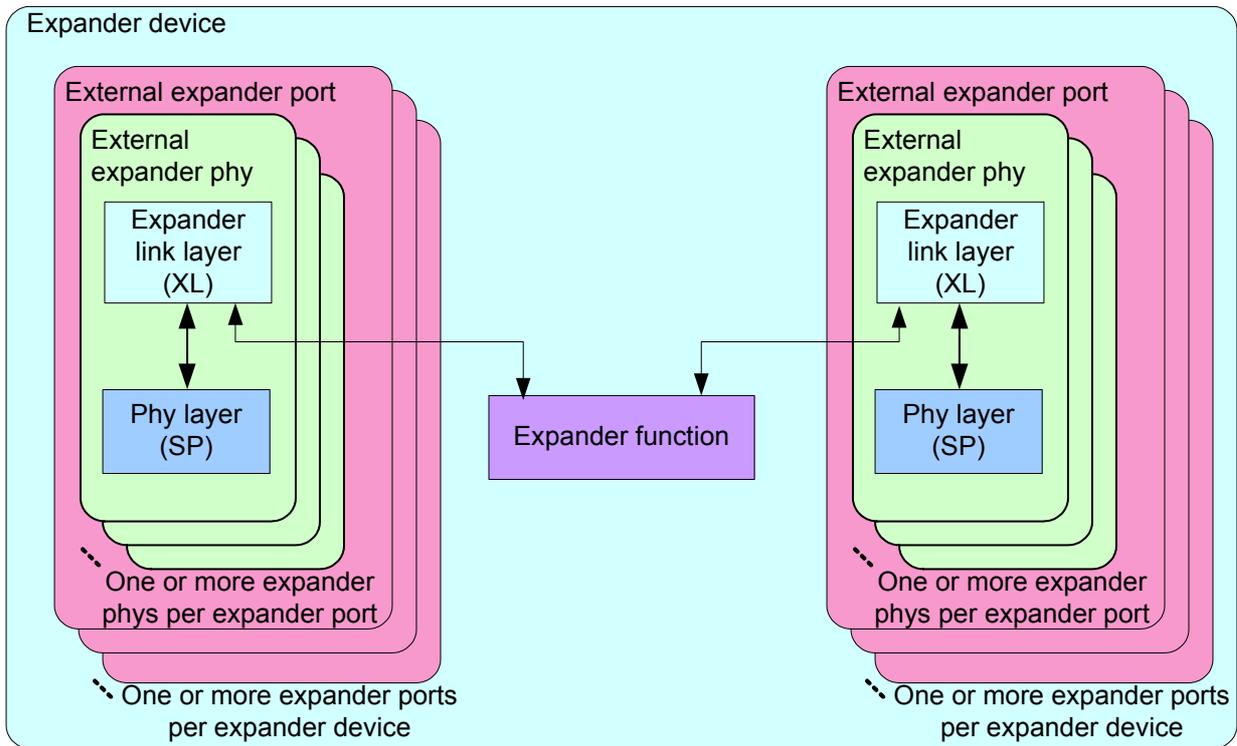
Figure 31 — State machines for expander devices

Annex K contains a list of messages between state machines.

4.3.2 Transmit data path

Figure 32 shows the transmit data path in a SAS phy.
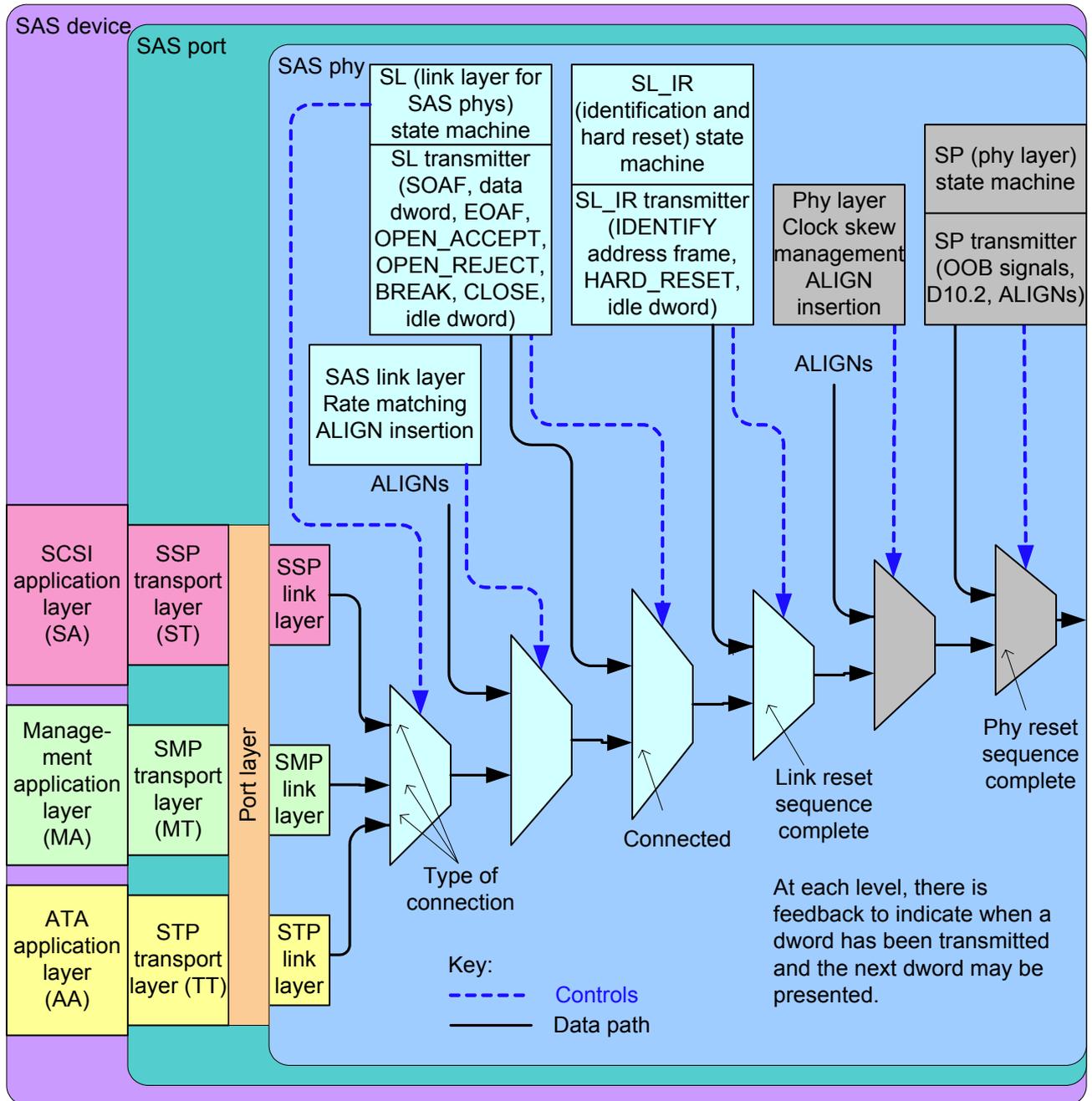
Figure 32 — Transmit data path in a SAS phy

Figure 33 shows the transmit data path for the SSP link layer and the communication to the port layer, SSP transport layer and SCSI application layer. Only the SSP link layer (i.e., not the port, transport, or application layer) transmits dwords.
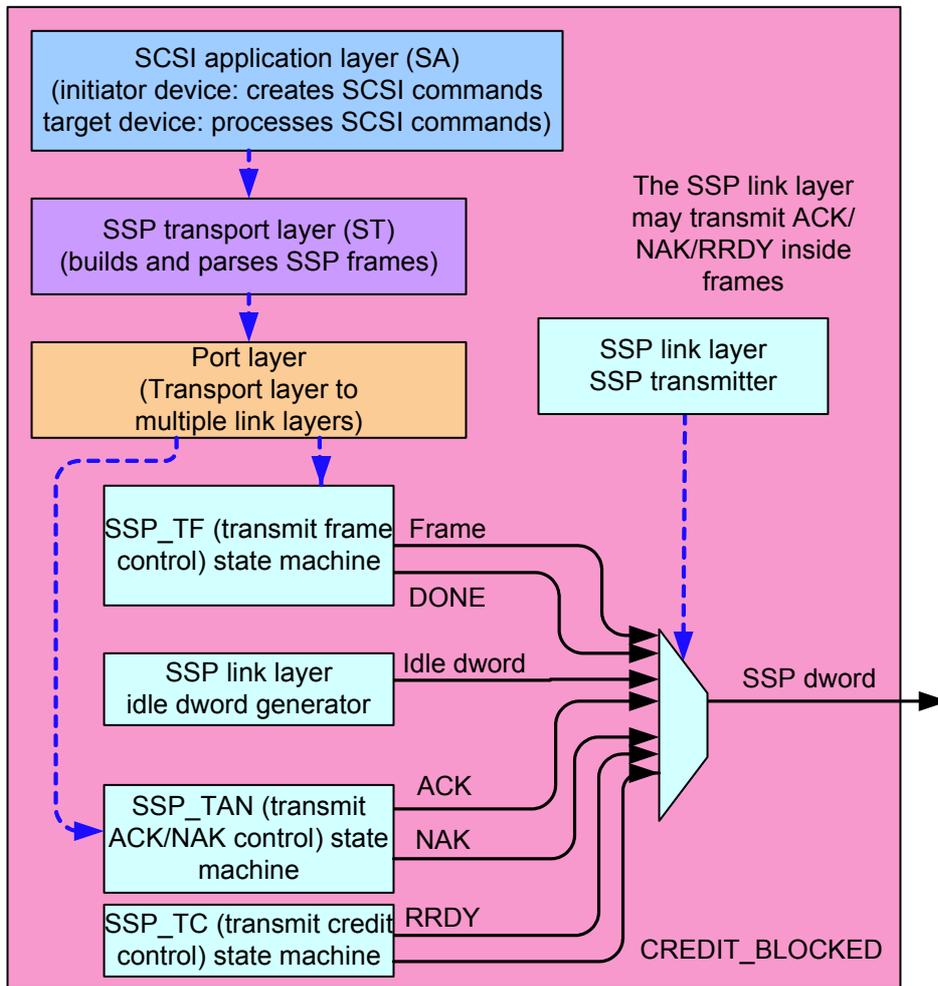
Figure 33 — SSP link, port, SSP transport and SCSI application layer state machines

Figure 34 shows the transmit data path for the SMP link layer and the communication to the port layer, SMP transport layer and management application layer. Only the SMP link layer (i.e., not the port, transport or application layer) transmits dwords.
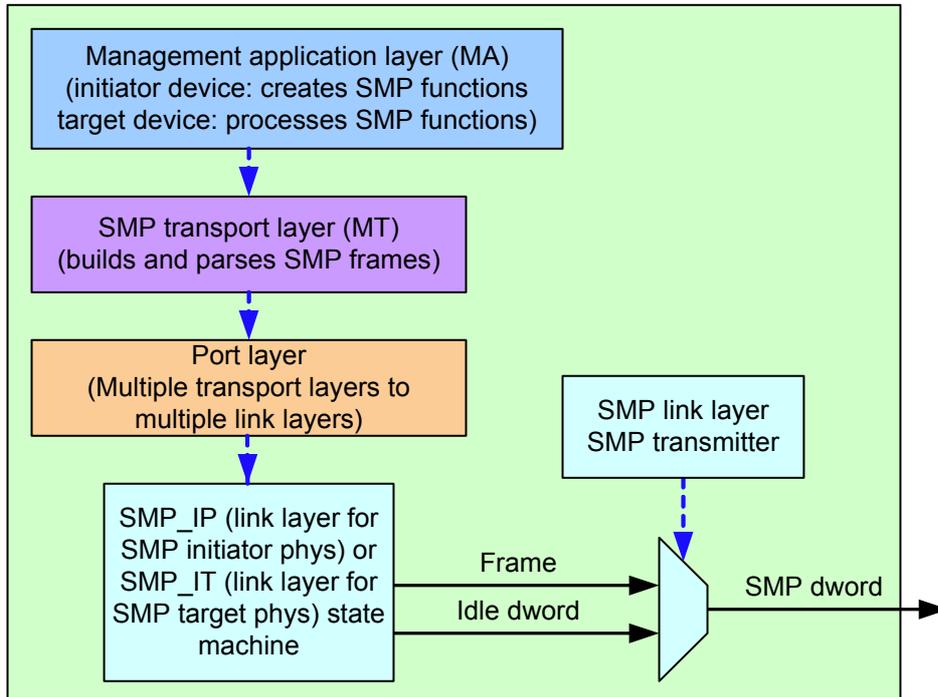
Figure 34 — SMP link, port, SMP transport and management application layer state machines

Figure 35 shows the transmit data path for the STP link layer and communication to the port layer, STP transport layer and ATA application layer. Only the STP link layer (i.e., not the port, transport or application layer) transmits dwords.
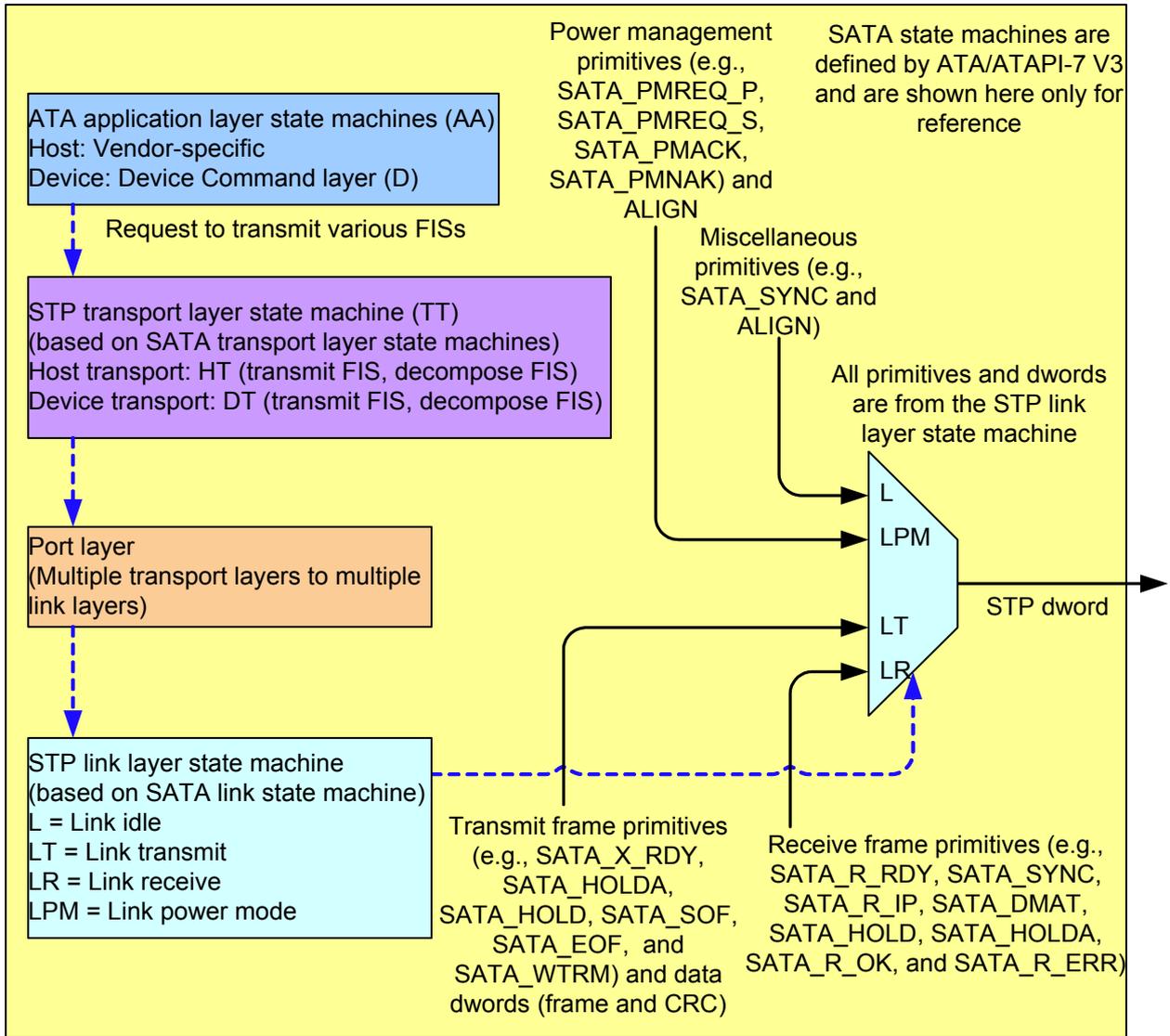
Figure 35 — STP link, port, STP transport and ATA application layer state machines

Figure 36 shows the transmit data path in an expander phy.
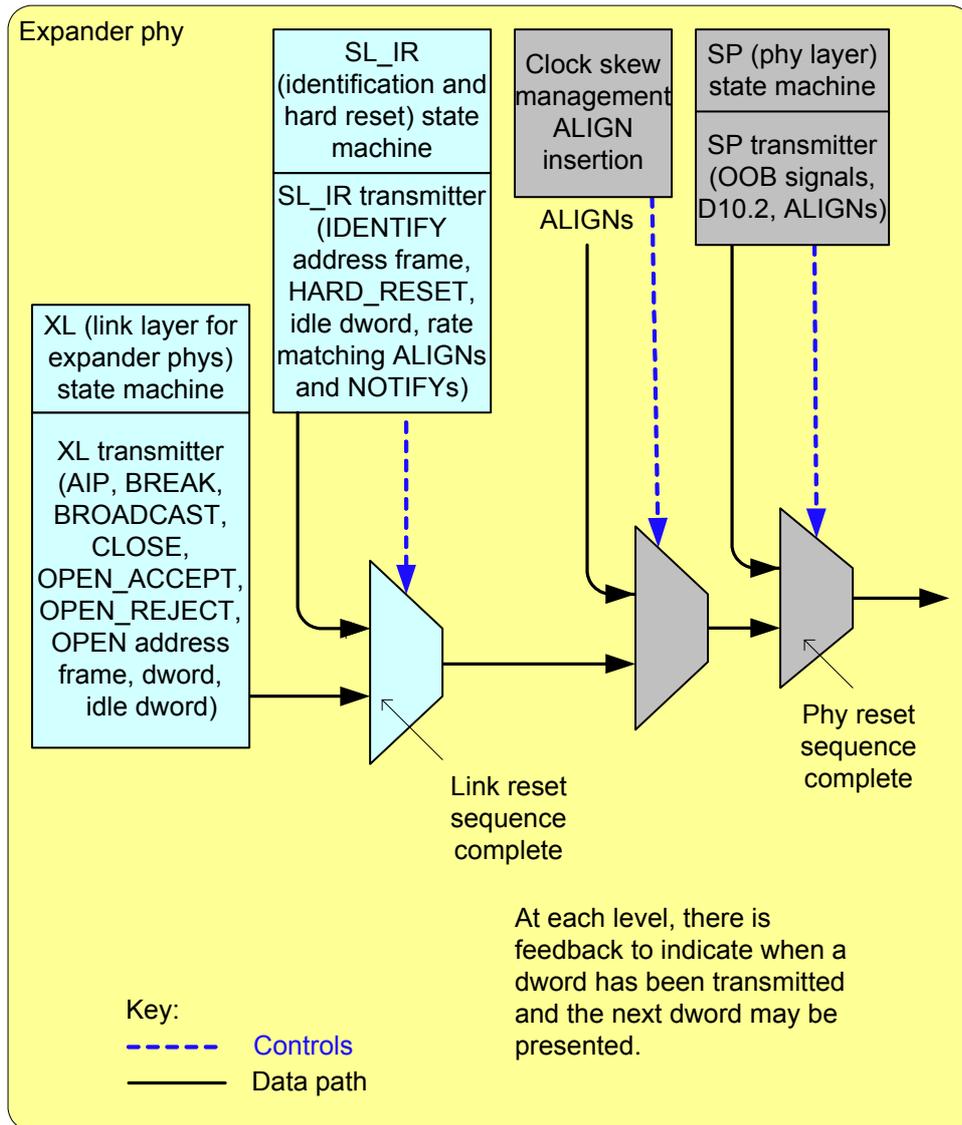
Figure 36 — Transmit data path and state machines in an expander phy

4.3.3 Receive data path

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

Figure 37 shows the receive data path in a SAS phy.
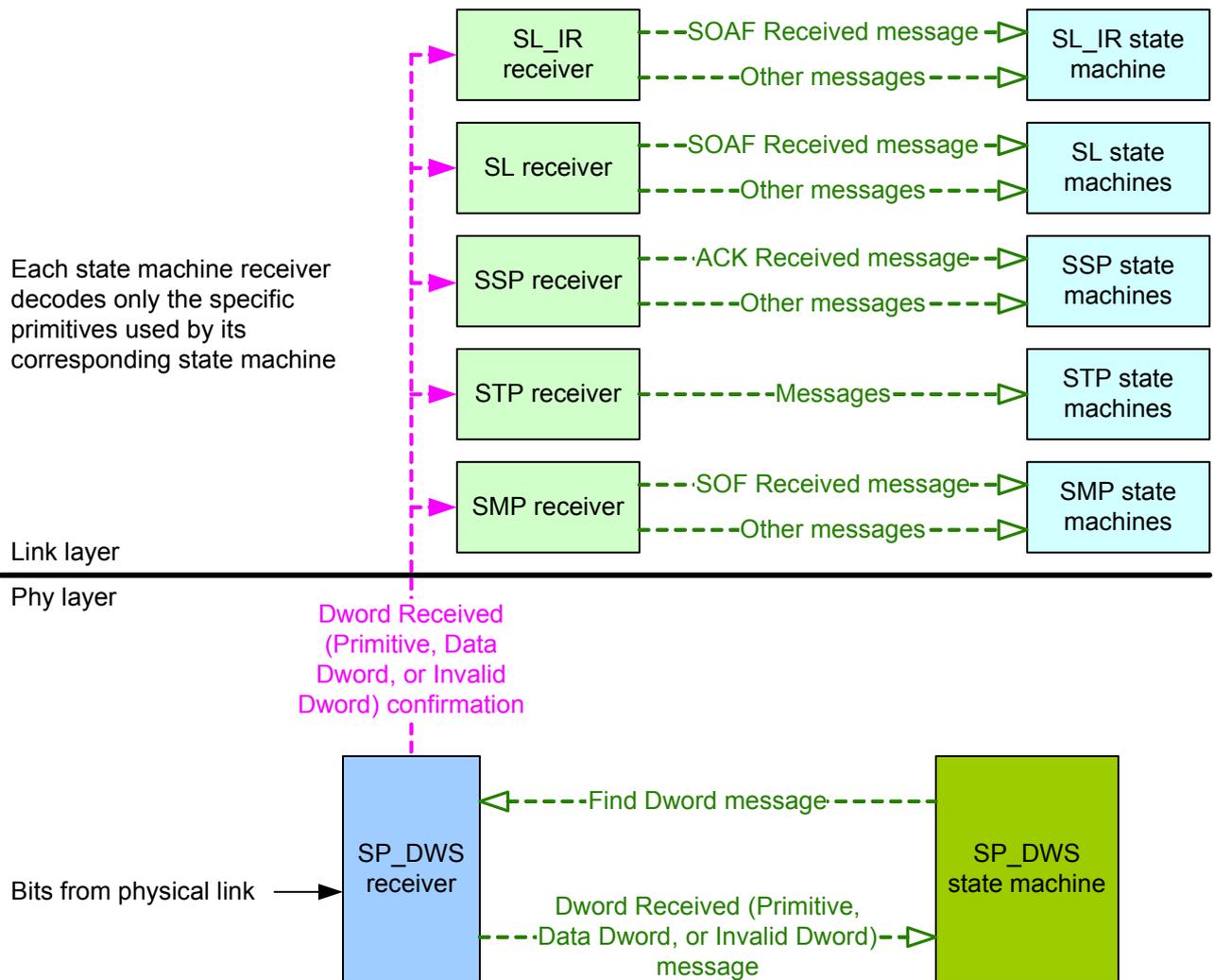
Figure 37 — Receive data path in a SAS phy

Figure 38 shows the receive data path in an expander phy.
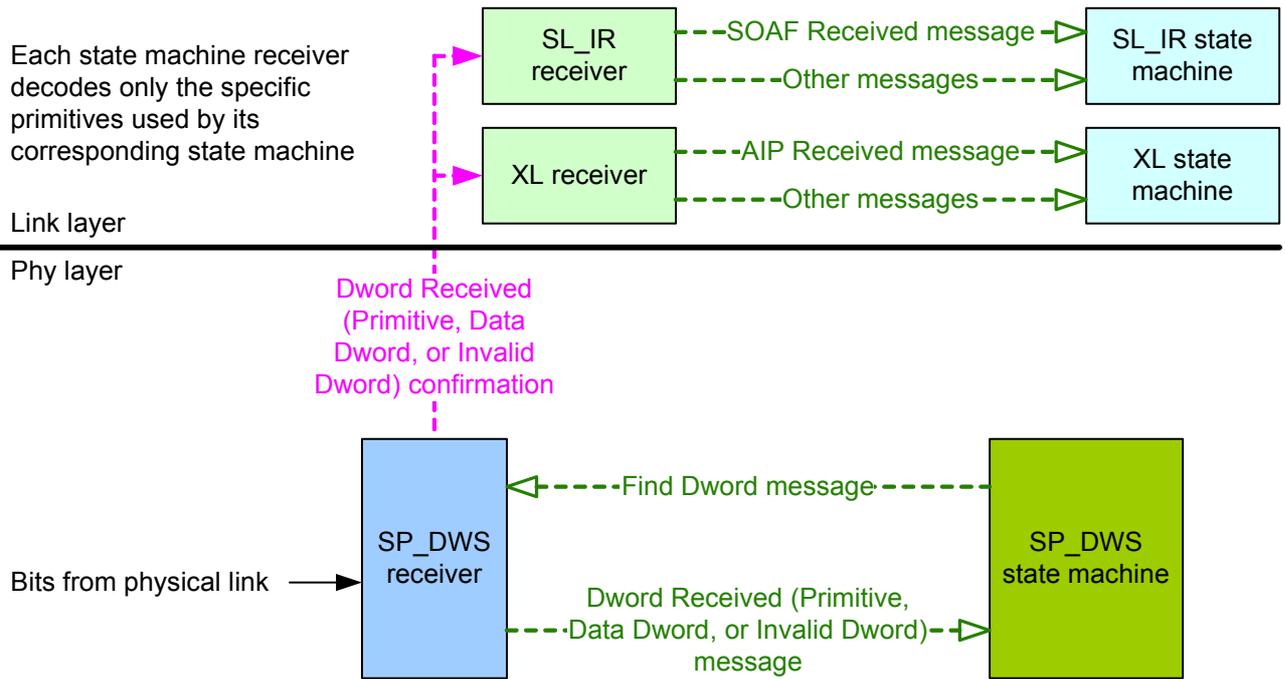
Figure 38 — Receive data path in an expander phy

4.3.4 State machines and SAS device, SAS port and SAS phy classes

Figure 39 shows which state machines are contained within the SAS device, SAS port and SAS phy classes.
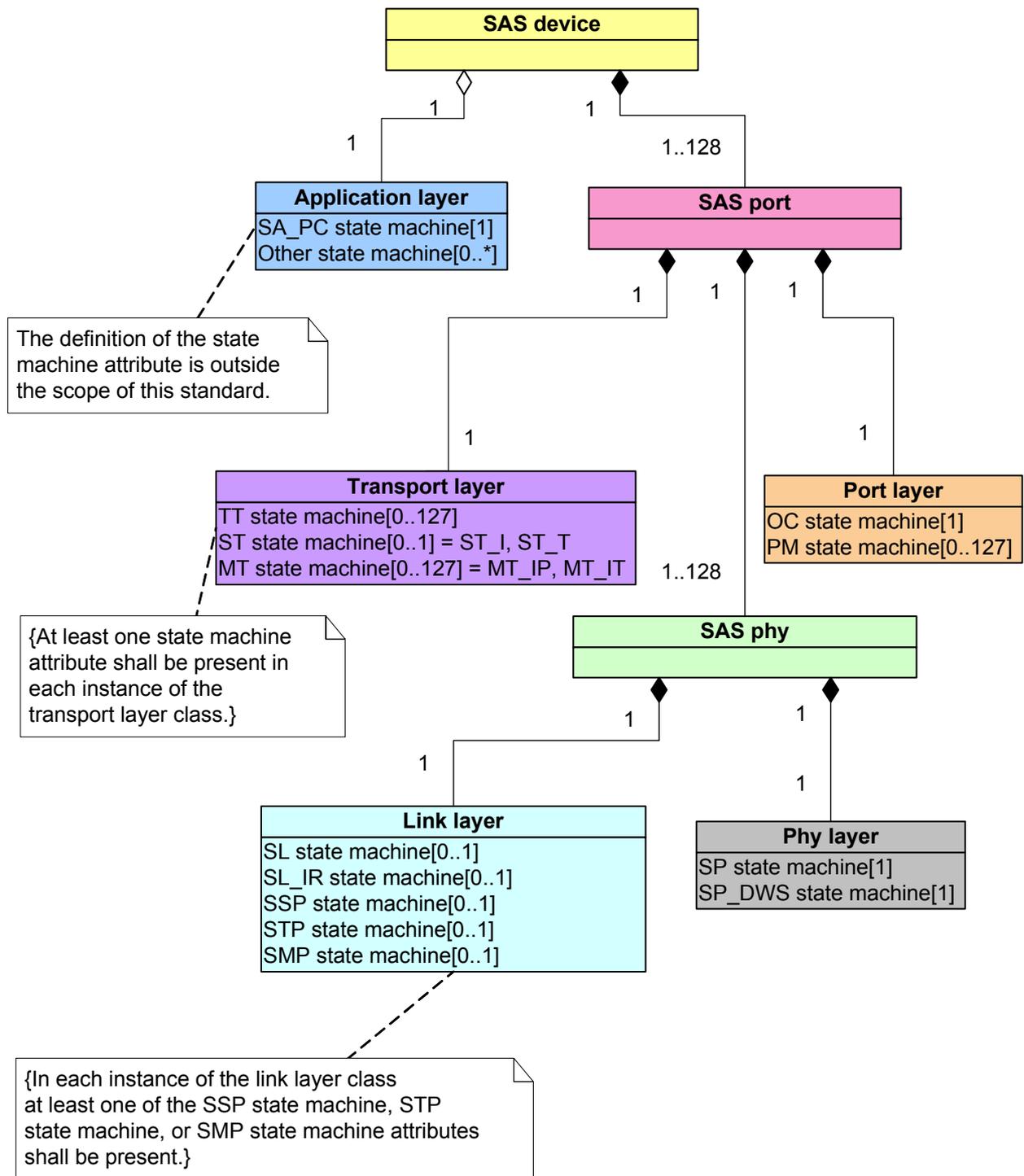
Figure 39 — State machines and SAS device, SAS port and SAS phy classes

Figure 40 shows which state machines are contained within the expander device, expander port and expander phy classes.
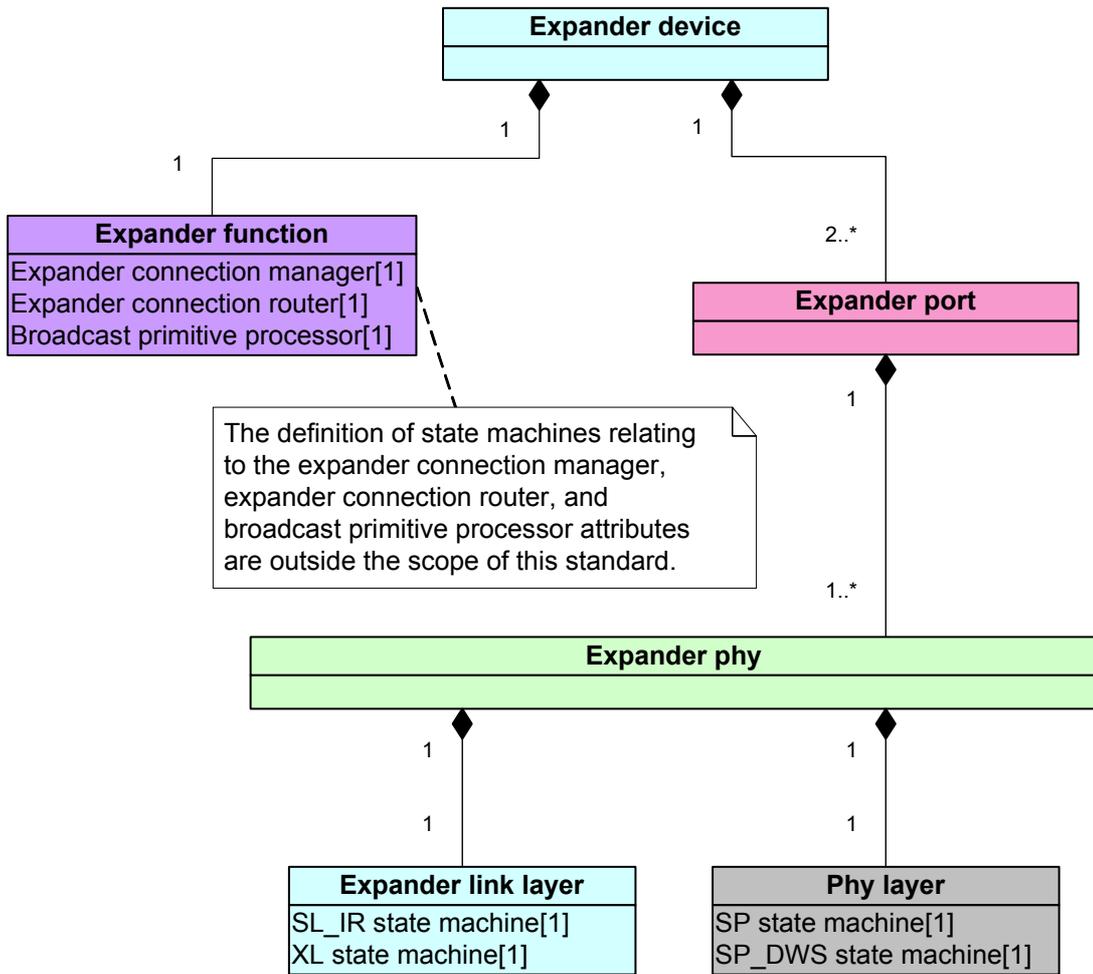
Figure 40 — State machine and expander device, expander port, and expander phy classes

4.4 Resets

4.4.1 Reset overview

Figure 41 illustrates the reset terminology used in this standard as follows:

    a) link reset sequence;
    b) phy reset sequence (see 6.7);
    c) SATA OOB sequence (see 6.7.2.1);
    d) SATA speed negotiation sequence (see 6.7.2.2);
    e) SAS OOB sequence (see 6.7.4.1);
    f) SAS speed negotiation sequence (see 6.7.4.2);
    g) hard reset sequence (see 7.9); and
    h) identification sequence (see 7.9).
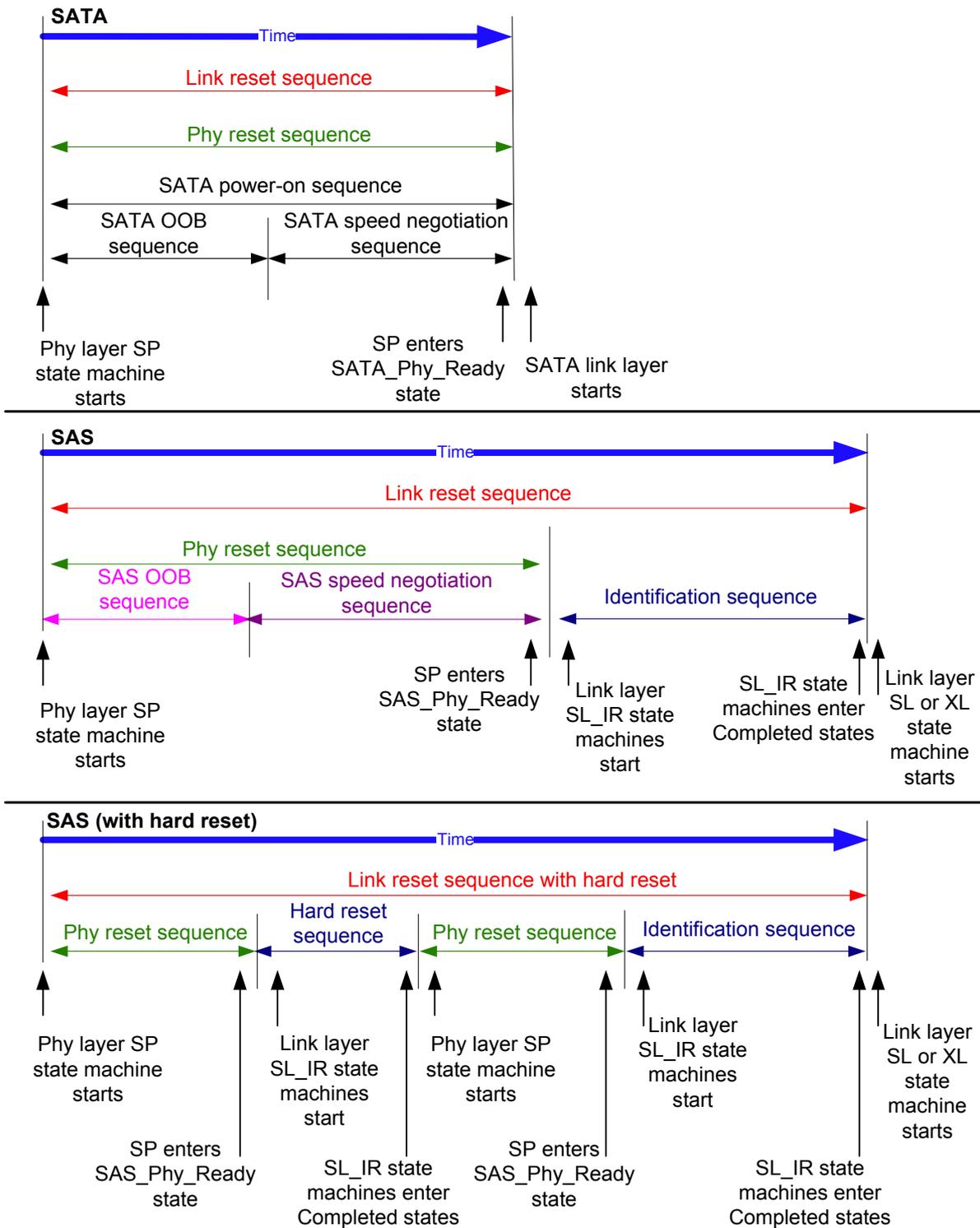
Figure 41 — Reset terminology

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

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

4.4.2 Hard reset

4.4.2.1 Hard reset overview

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

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

If a SAS device is contained in an expander device, its SSP ports, STP ports and/or SATA ports shall initiate a hard reset when an SMP PHY CONTROL function with a phy operation of HARD RESET and phy identifier specifying a virtual expander phy attached to such a SAS port is processed (see 10.4.3.10).

4.4.2.2 Additional hard reset processing by SAS ports

If the port processing the hard reset is an SSP port, the hard reset causes a Transport Reset event notification to the SCSI application layer (see 10.2.5) and the SCSI device shall perform the actions defined for hard reset in SAM-3. After processing the hard reset, each logical unit to which the SSP target port has access shall establish a unit attention condition for all SSP initiator ports with the additional sense code set to SCSI BUS RESET OCCURRED (see SAM-3 and SPC-3).

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

If the port processing the hard reset is an STP port that is not in an STP/SATA bridge, the STP target device shall perform the actions defined for power-on or hardware reset in ATA/ATAPI-7 V1.

4.4.2.3 Additional hard reset processing by expander ports

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

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

4.5 I_T nexus loss

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

   a)   the connection is established;
   b)   for SSP target ports, the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port mode page (see 10.2.7.2) expires; or
   c)   for STP ports, SMP ports or SSP initiator ports, a vendor-specific I_T nexus loss time expires.

An SSP initiator port should retry the connection request for the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port mode page (see 10.2.7.2) for the SSP target port to which it is trying to establish a connection.

If the I_T nexus loss time expires in an SSP port, then the port shall send a Nexus Loss event notification to the SCSI application layer (see 10.2.5) and the SCSI device shall perform the actions defined for I_T nexus loss in SAM-3.

I_T nexus loss is handled by the port layer state machines (see 8.2.2.3).

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)   broadcast primitive processor (BPP);
  b) two or more physical expander phys. For the maximum number of phys, see 4.1.5;
  c) an expander port available per phy; and
  d) an SMP target port.

An expander device may contain SAS devices with SSP ports, STP ports, and/or SMP ports.

Figure 42 shows a model of an expander device showing the state machines in each expander port. The internal SMP port is not shown.
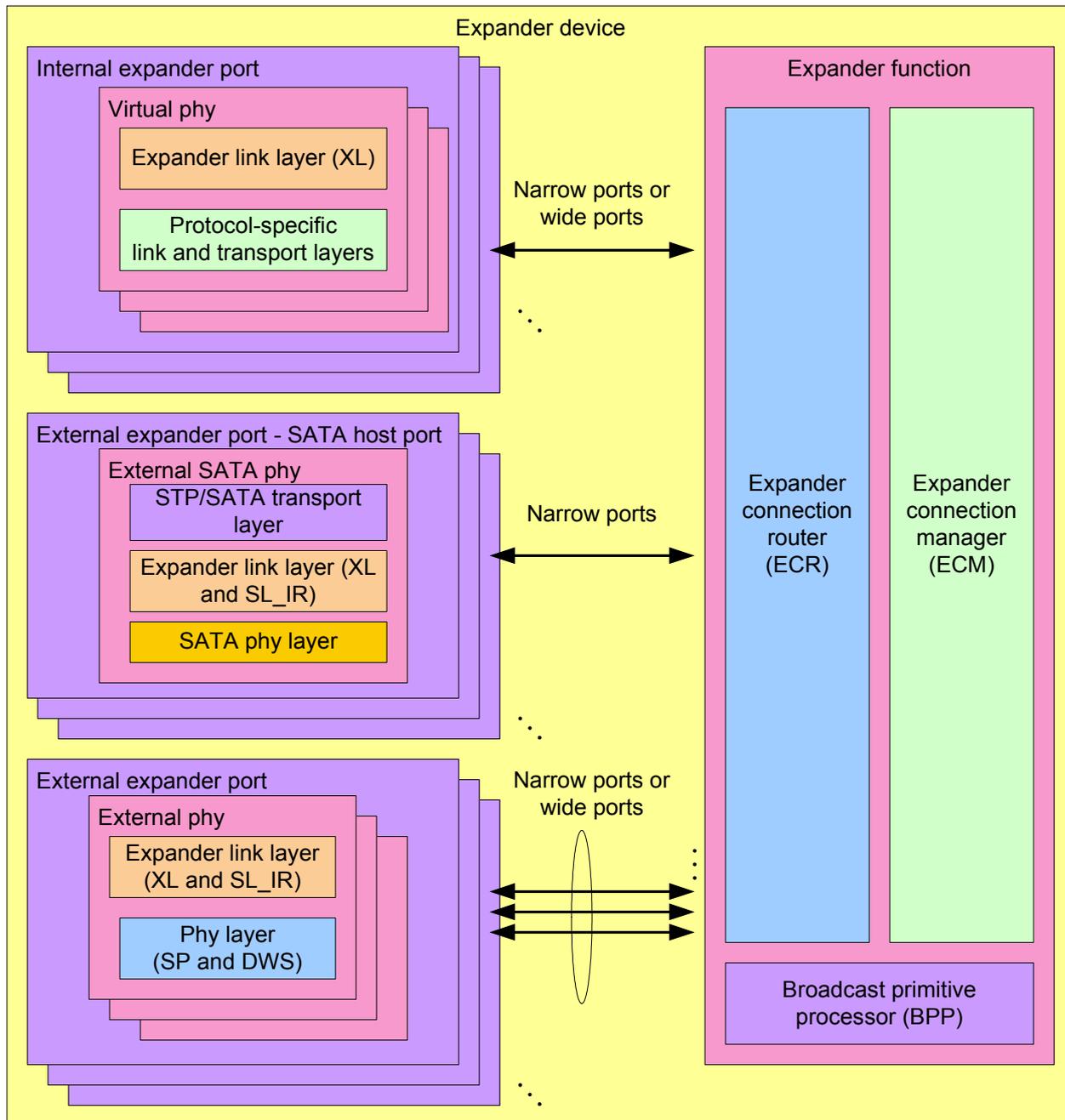
Figure 42 — Expander device model

4.6.2 Expander ports

An external expander port contains one or more physical phys (see 4.1.2). Since each phy in the expander device has the same SAS address, expander ports are created based on the attached SAS addresses (see 4.1.3).

Each phy in an expander port shall have the same routing attribute (see 4.6.7.1) and the DISCOVER function (see 10.4.3.5) shall return the same value in the ROUTING ATTRIBUTE field for each phy in an expander port.

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

A set of phys with table routing attributes using the same external connector (see 5.2.3.3) is called an enclosure out port. A set of phys with subtractive routing attributes using the same external connector is called an enclosure in port.

Each expander phy contains an expander link layer with an XL state machine (see 7.15) and an SL_IR state machine (see 7.9.5). The XL state machine in each expander phy within an expander port processes connection requests independently of the XL state machines in other expander phys.

An internal expander port contains a virtual phy with an expander link layer and a protocol-specific transport layer (e.g., to provide access as an SSP target port to a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-3 and SES-2)).

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

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

Each STP/SATA bridge shall have a unique SAS address. This SAS address is reported in the ATTACHED SAS ADDRESS field in the DISCOVER function (see 10.4.3.5) for the expander phy containing the STP/SATA bridge (i.e., the expander phy attached to the SATA device or SATA port selector).

4.6.3 Expander connection manager (ECM)

The ECM performs the following functions:

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

4.6.4 Expander connection router (ECR)

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

4.6.5 Broadcast primitive processor (BPP)

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

In a self-configuring expander device (see 4.1.5), the BPP requests transmission of a BROADCAST (CHANGE) when it completes configuration (see 10.4.3.3).

4.6.6 Expander device interfaces

4.6.6.1 Expander device interface overview

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 consists of requests, confirmations, indications, and responses. This interaction is called the expander device interface.

Figure 43 describes the interfaces present within an expander device.
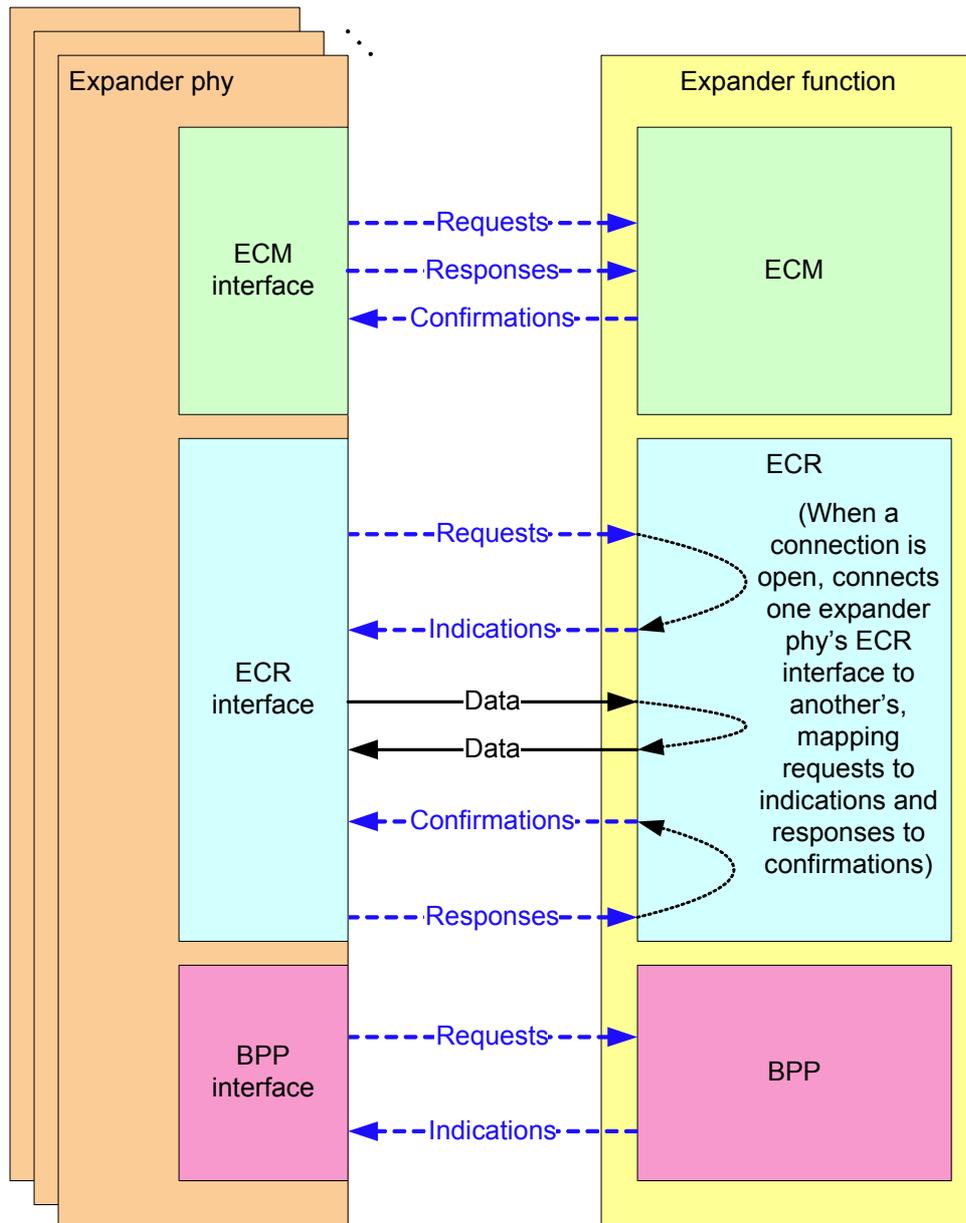
Figure 43 — Expander device interfaces

4.6.6.2 Expander device interfaces detail

Figure 44 shows the interface requests, confirmations, indications, and responses used by an expander device to manage connections.
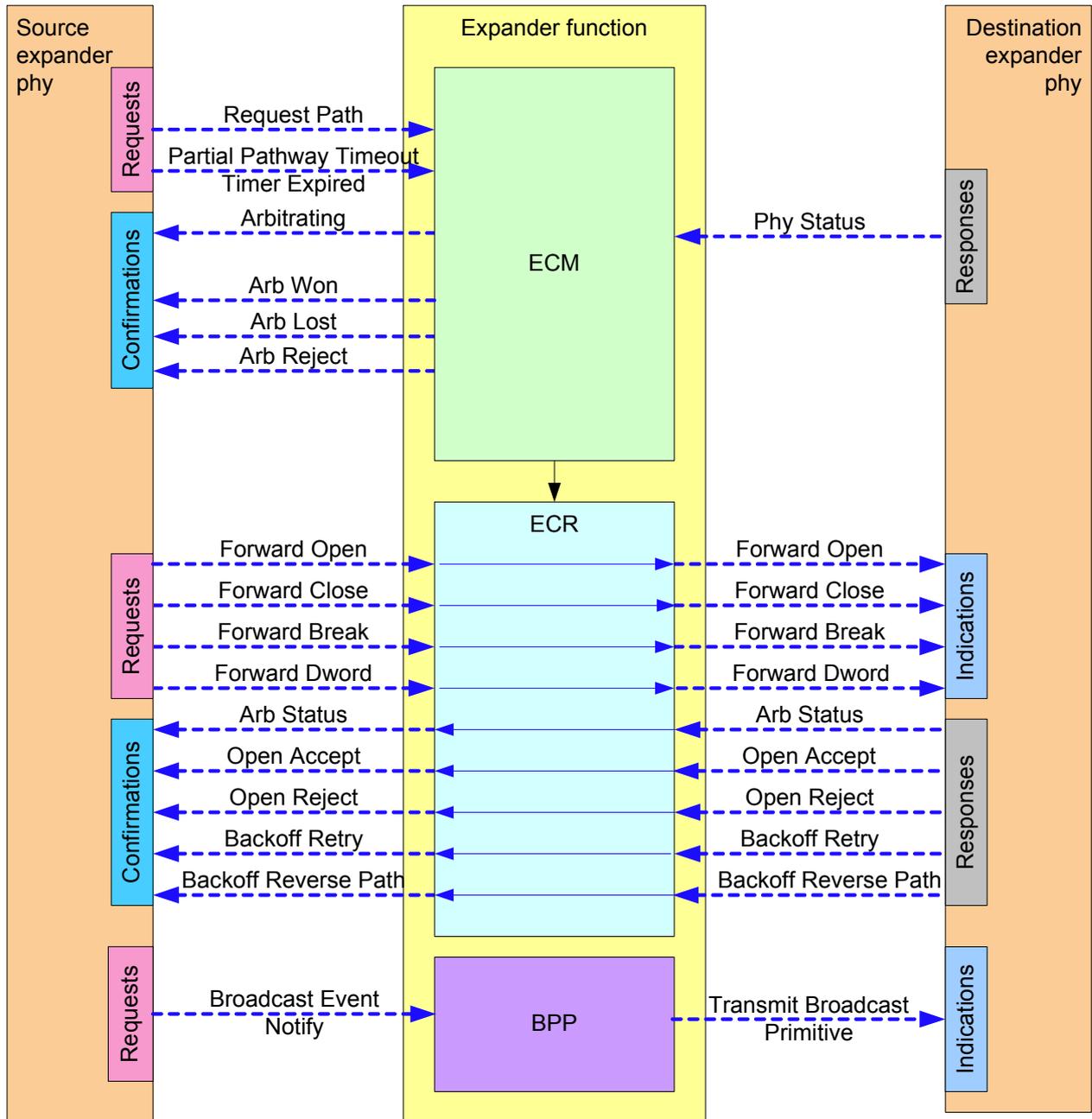
Figure 44 — Expander device interface detail

4.6.6.3 ECM interface

Table 9 describes the requests from an expander phy to the ECM.

Table 9 — Expander phy to ECM requests

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

Table 10 describes the responses from an expander phy to the ECM.

Table 10 — Expander phy to ECM responses

Message	Description
Phy Status (Partial Pathway)	Response meaning that an expander phy: a) is being used for an unblocked partial pathway (i.e., the expander phy is in the XL3:Open_Confirm_Wait state or XL6:Open_Response_Wait state and the last AIP transmitted or received was not AIP (WAITING ON PARTIAL)); or b) has sent a Request Path request to the ECM and is receiving Arbitrating (Waiting On Partial) from the ECM.
Phy Status (Blocked Partial Pathway)	Response meaning that an expander phy: a) is being used for a blocked partial pathway (i.e., the expander phy is in the XL3:Open_Confirm_Wait state or XL6:Open_Response_Wait state and the last AIP transmitted or received was AIP (WAITING ON PARTIAL)); or b) has sent a Request Path request to the ECM and is receiving Arbitrating (Blocked On Partial) from the ECM.
Phy Status (Connection)	Response meaning that an expander phy: a) is being used for a connection (i.e., the expander phy is in the XL7:Connected or XL8:Close_Wait state); or b) has sent a Request Path request to the ECM and is receiving Arbitrating (Waiting On Connection) from the ECM.

Table 11 describes the confirmations from the ECM to an expander phy. These confirmations are sent in confirmation of a Request Path request.

Table 11 — ECM to expander phy confirmations (Sheet 1 of 2)

Message	Description
Arbitrating (Normal)	Confirmation that the ECM has received the Request Path request.
Arbitrating (Waiting On Partial)	Confirmation that the ECM has determined that: a) there is a destination port capable of routing to the requested destination SAS address; b) at least one phy within the destination port supports the requested connection rate; c) each of the phys within the destination port is returning a Phy Status (Partial Pathway) or Phy Status (Blocked Partial Pathway) response; and d) at least one of the phys within the destination port is returning a Phy Status (Partial Pathway) response.
Arbitrating (Blocked On Partial)	Confirmation that the ECM has determined that: a) there is a destination port capable of routing to the requested destination SAS address; b) at least one phy within the destination port supports the requested connection rate; and c) each of the phys within the destination port is returning a Phy Status (Blocked Partial Pathway) response.

Table 11 — ECM to expander phy confirmations (Sheet 2 of 2)

Message	Description
Arbitrating (Waiting On Connection)	Confirmation that the ECM has determined that the connection request is blocked due to one of the following reasons: a) the connection request is blocked by an active connection; or b) there are insufficient routing resources within the expander to complete the connection request. A connection request shall be considered blocked by an active connection when: a) there is a destination port capable of routing to the requested destination SAS address; b) at least one phy within the destination port supports the requested connection rate; c) each of the phys within the destination port is returning a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response; and d) at least one of the phys within the destination port is returning a Phy Status (Connection) response.
Arb Won	Confirmation that an expander phy has won path arbitration.
Arb Lost	Confirmation that an expander phy has lost path arbitration.
Arb Reject (No Destination)	Confirmation that: a) there is no operational expander phy capable of routing to the requested destination SAS address; or b) the requested destination SAS address maps back to the requesting port (see 7.12.4.3 and 7.12.4.4).
Arb Reject (Bad Destination)	Confirmation that: a) the requested destination SAS address maps back to the requesting port; b) the requesting port is using the direct routing method or the table routing method; and c) the ECM has not chosen to return Arb Reject (No Destination) (see 7.12.4.3 and 7.12.4.4).
Arb Reject (Bad Connection Rate)	Confirmation that the ECM has determined that there is a destination port capable of routing to the requested destination SAS address but no phys within the destination port are configured to support the requested connection rate.
Arb Reject (Pathway Blocked)	Confirmation that the ECM has determined that the requesting expander phy shall back off according to SAS pathway recovery rules.

4.6.6.4 ECR interface

Table 12 describes the requests from an expander phy to the ECR and the corresponding indications from the ECR to another expander phy.

Table 12 — Expander phy to ECR to expander phy requests and indications

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

Table 13 describes the responses from an expander phy to the ECR and the corresponding confirmations from the ECR to another expander phy. These responses are sent in response to a Forward Open indication.

Table 13 — Expander phy to ECR to expander phy responses and confirmations

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

4.6.6.5 BPP interface

Table 14 describes the requests from an expander phy to the BPP. See 7.11 for more information on broadcasts.

Table 14 — Expander phy to BPP requests

Message	Description
Broadcast Event Notify (Phy Not Ready)	Request to transmit a BROADCAST (CHANGE) on all other ports because an expander phy's SP state machine transitioned from the SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready state to the SP0:OOB_COMINIT state (see 6.8).
Broadcast Event Notify (SATA Spinup Hold)	Request to transmit a BROADCAST (CHANGE) on all other ports because the SATA spinup hold state has been reached (see 6.8 and 6.10).
Broadcast Event Notify (Identification Sequence Complete)	Request to transmit a BROADCAST (CHANGE) on all other ports because an expander phy has completed the identification sequence (see 7.9).
Broadcast Event Notify (SATA Port Selector Change)	Request to transmit a BROADCAST (CHANGE) on all other ports because a SATA port selector appeared or disappeared.
Broadcast Event Notify (CHANGE Received)	Request to transmit a BROADCAST (CHANGE) on all other ports because a BROADCAST (CHANGE) was received.
Broadcast Event Notify (RESERVED CHANGE Received)	Request to transmit a BROADCAST (RESERVED CHANGE) on all other ports because a BROADCAST (RESERVED CHANGE) was received.
Broadcast Event Notify (SES Received)	Request to transmit a BROADCAST (SES) on all other ports because a BROADCAST (SES) was received.
Broadcast Event Notify (RESERVED 1 Received)	Request to transmit a BROADCAST (RESERVED 1) on all other ports because a BROADCAST (RESERVED 1) was received.

Table 15 describes the indications from the BPP to an expander phy.

Table 15 — BPP to expander phy indications

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

4.6.7 Expander device routing

4.6.7.1 Routing attributes and routing methods

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

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

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

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

Table 16 describes the routing methods that the ECM uses based on the routing attributes of a phy.

Table 16 — Routing attributes and routing methods

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

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

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

An edge expander device shall 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 performing the discover process (see 4.7) shall report an error in a vendor-specific manner.

4.6.7.2 Connection request routing

The ECM shall determine how to route a connection request from a source expander phy to a destination expander phy in a different expander port if the destination expander phy is enabled and operating at a valid physical link rate (e.g., the DISCOVER function reports a NEGOTIATED PHYSICAL LINK RATE field set to G1 (i.e., 8h) or G2 (i.e., 9h)) using the following precedence:

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

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

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

4.6.7.3 Expander route table

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 10.4.3.3). Each expander route entry shall be disabled after power on.

A management application client may reference a specific expander route entry within an expander route table with the SMP REPORT ROUTE INFORMATION function (see 10.4.3.8) and the SMP CONFIGURE ROUTE INFORMATION function (see 10.4.3.9).

Figure 45 shows a representation of an expander route table.
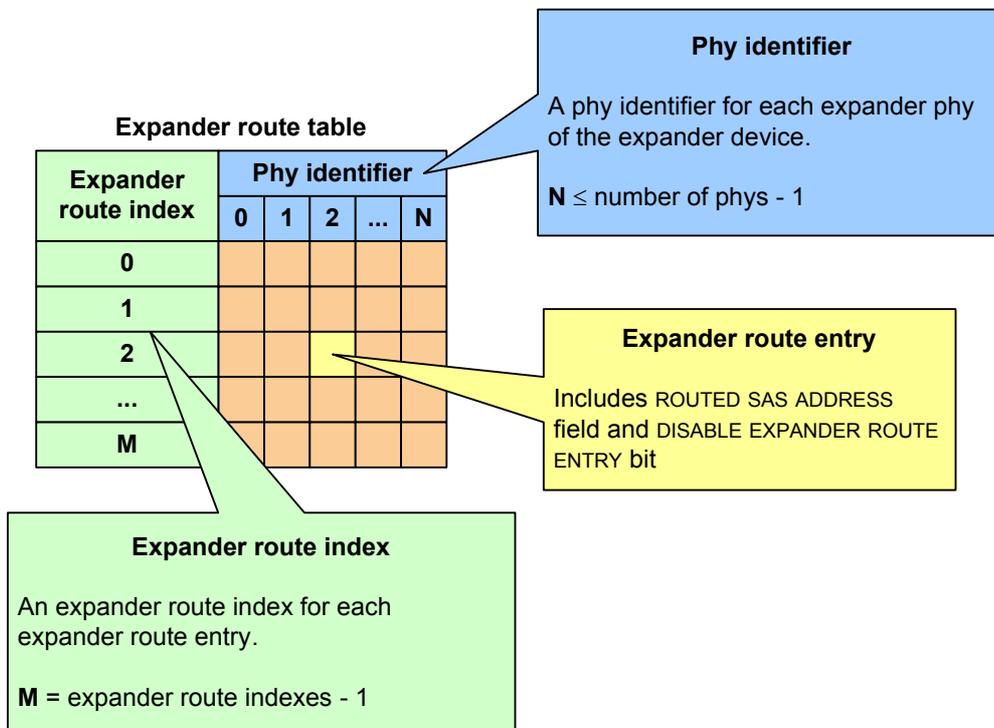
Figure 45 — Expander route table example

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.

4.7 Discover process

4.7.1 Discover process overview

Management application clients direct an SMP initiator port to request SMP functions from an SMP target port. Management application clients are located in every SAS initiator device and every self-configuring expander device. The discover process is the process of:

a) discovering all the SAS devices and expander devices in the SAS domain (i.e., determining their device types, SAS addresses, and supported protocols); and
b) configuring routing tables in the expander devices as needed.

A management application client performing the discover process shall perform a level-order (i.e., breadth-first) traversal of the SAS domain. The order of traversal shall be to discover:

1) the device(s) to which the device containing the management application client is attached;
2) if an attached device is an expander device, every device attached to that expander device; and
3) for each expander device found, every device attached to that expander device.

This order is repeated until all expander devices have been traversed. If the management application client is inside an expander device, then the process shall be repeated on each expander port.

Figure 46 shows an example of level-order traversal.
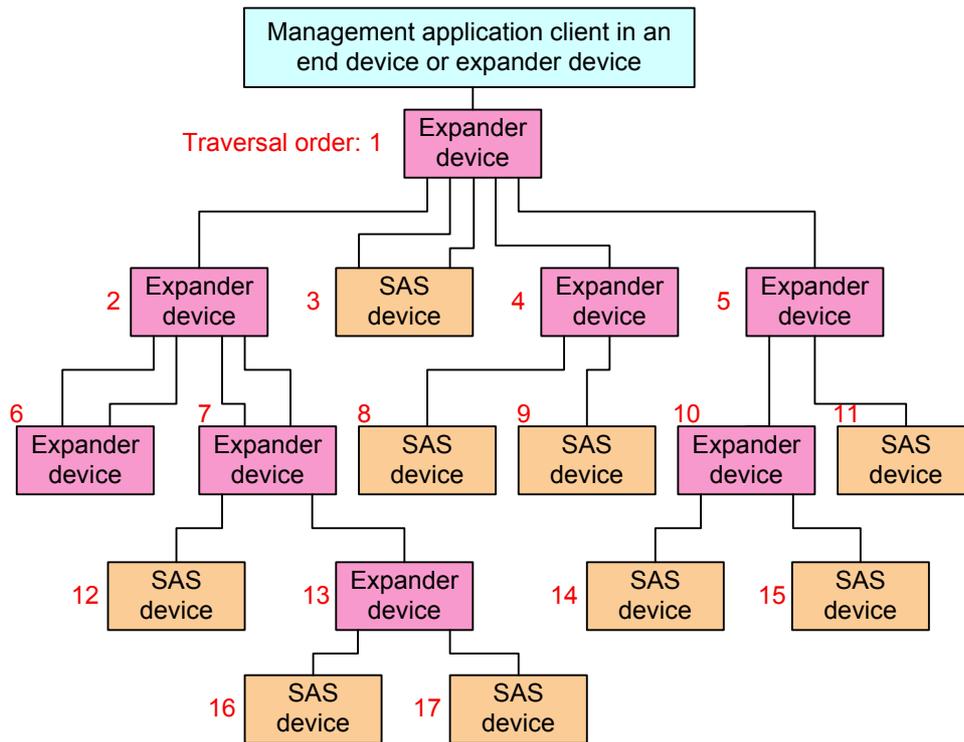
Note: Assume that the phy with the lowest phy identifier in each expander
device is on the top right, and the remaining phys have increasing phy
identifiers assigned in a counter-clockwise direction

Figure 46 — Level-order traversal example

The management application client determines whether an expander device or SAS device is attached at each point in the traversal. For the first device (i.e., the device directly attached), this is determined from the DEVICE TYPE field in the IDENTIFY address frame information received by the phy that the management application client is using. For other devices (i.e., devices not directly attached), this is determined from ATTACHED DEVICE TYPE field the SMP DISCOVER function response.

If an expander device is attached, the management application client shall use the SMP REPORT GENERAL function (see 10.4.3.3) to determine how many phys are in the expander device and then use the SMP DISCOVER function (see 10.4.3.5) to determine what is attached to each expander phy (e.g., the device type, SAS address, and supported protocol(s)).

If the expander device's CONFIGURABLE ROUTE TABLE bit is set to one in the SMP REPORT GENERAL function response, the management application client shall configure its expander route table as described in 4.7.3 and 4.7.4.

If a SAS device is attached, the discover process is not required to obtain any more information about the SAS device. Additional discovery software may access that SAS device, however:

a) if the SAS device supports an SMP target port, the management application client may use SMP functions (e.g., REPORT GENERAL and REPORT MANUFACTURER INFORMATION) to determine additional information about the SAS target device;

b) if the SAS device supports an SSP target port, a SCSI application client may transmit SCSI commands (e.g., INQUIRY and REPORT LUNS) to determine additional information about the SCSI target device; and

c) if the end device supports an STP target port, an ATA application client may transmit ATA commands (e.g., IDENTIFY DEVICE and IDENTIFY PACKET DEVICE) to determine additional information about the ATA device.

The result of the discover process is that the management application client has the necessary information (e.g., the device type, SAS address, and supported protocol(s)) to communicate with each SAS device and expander

device in the SAS domain and each configurable expander device is configured with the necessary expander route entries to allow routing of connection requests through the SAS domain.

Annex L contains an example implementation of how a management application client may perform the discover process.

The discover process may be aborted prior to completion and restarted if there is an indication that it may be based on incorrect information (e.g., arrival of a BROADCAST (CHANGE)).

4.7.2 Allowed topologies

The management application client shall allow the following attachments between expander phys:

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

If the management application client detects any other combination of expander phy attachment (e.g., expander phy with table routing attached to expander phy with table routing, or an expander phy with direct routing attached to an expander phy), it shall report an error in a vendor-specific manner.

If the management application client detects an overflow of the edge expander route index, it shall report an error in a vendor-specific manner.

If the discover process optimization (see 4.7.3) is disabled and the management application client detects an expander route entry that references the SAS address of the expander device itself (i.e., self-reference), it shall disable the expander route entry by setting the DISABLE EXPANDER ROUTE ENTRY bit to one in the SMP CONFIGURE ROUTE INFORMATION function (see 10.4.3.9). The management application client shall disable each expander route entry in the route table by setting the DISABLE EXPANDER ROUTE ENTRY bit to one in the SMP CONFIGURE ROUTE INFORMATION function (see 10.4.3.9) for each expander phy that has its attached device type set to 000b (i.e., no device attached).

If the management application client detects a port that the discover process has not already configured with a SAS address it has already found attached to another expander device, it should use the SMP PHY CONTROL function (see 10.4.3.10) to disable all the expander phys attached to that SAS address except for phys in the expander device with the lowest SAS address. Figure 47 shows some illegal topologies.

Problem: Multiple paths to SAS address B in the domain.
From A to B:
 A to Y to B
 A to Y to Z to B
From C to B:
 C to Z to B
 C to Z to Y to B
SAS device B considers itself in two domains.

Remedy: disable phy(s) in expander device Z attached to SAS address B.

Problem: Multiple paths in the domain, including:
From A to C:
 A to Y to Z to C
 A to Y to X to Z to C
From A to B:
 A to Y to X to B
 A to Y to Z to X to B

Also, introduces a loop where BROADCASTs cycle forever:
 Y to X to Z (to Y…)
 Y to Z to X (to Y…)

Remedy: disable phy(s) in expander device Z attached to expander device X.

Figure 47 — Examples of invalid topologies

4.7.3 Discover process optimization

The management application client shall support a discover process optimization that reduces the number of entries required in an expander route table. The method used to enable and disable the discover process optimization is vendor specific.

If the discover process optimization is enabled, then the management application client shall exclude discovered SAS addresses from the expander device route table when any of the following conditions are met:

a) in the discovered phy SMP DISCOVER response:
   A) the FUNCTION RESULT field is set to a non-zero value (i.e., not SMP FUNCTION ACCEPTED);
b) in the discovered phy SMP DISCOVER response:
   A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
   B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table); and
   C) the ATTACHED DEVICE TYPE field is set to zero (i.e., no device attached);
c) in the discovered phy SMP DISCOVER response:
   A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
   B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);
   C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device, edge expander device, or fanout expander device); and
   D) the ATTACHED SAS ADDRESS field contains the SAS address of the expander device being configured (i.e. self-referencing address);
d) in the discovered phy SMP DISCOVER response:
   A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);
   B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);

    C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device, edge expander device, or fanout expander device); and

    D) the ATTACHED SAS ADDRESS field contains the SAS address of a device directly attached to the expander device being configured;

   or

  e) in the discovered phy SMP DISCOVER response:

    A) the FUNCTION RESULT field is set to zero (i.e., SMP FUNCTION ACCEPTED);

    B) the ROUTING ATTRIBUTE field is set to 1h (i.e., subtractive) or 2h (i.e., table);

    C) the ATTACHED DEVICE TYPE field is set to a non-zero value (e.g., end device, edge expander device, or fanout expander device); and

    D) the ATTACHED SAS ADDRESS field contains a SAS address that already exists in the expander device route table.

If the discovered SAS address being included in the expander device route table is for a device that is currently not attached (i.e. SMP DISCOVER response contains an ATTACHED DEVICE TYPE field set to zero (i.e., no device attached) and the ROUTE ATTRIBUTE field set to 0h (i.e., direct)), then the entry shall be inserted with the ROUTED SAS ADDRESS field set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit set to one (see 10.4.3.9).

If the discover process optimization is disabled, then all SAS addresses shall be qualified for insertion in the expander device route table.

If the management application client supports the discover process optimization, then the management application client should provide a vendor-specific method for initiating a check of the resulting expander route tables. The check should be performed under the following situations:

  a) when an OPEN_REJECT (NO DESTINATION) is received for a connection request to a SAS address that is expected to be accepted;

  b) when a discover process has been completed;

  c) when another SMP initiator port is discovered in the SAS domain; or

  d) when a self-configuring expander device is discovered in the SAS domain.

If the management application client detects an inconsistency in the expander route tables when the discover process optimization is enabled (e.g., detects entries that appear to have been filled in by a discover process with optimization disabled), then the management application client shall report an error in a vendor-specific manner and shall disable the discover process optimization. The management application client should then re-initiate a discover process with the discover process optimization disabled.

4.7.4 Expander route index order

The expander route table shall be configured for each expander phy that has a table routing attribute.

If the phy is not attached to an edge expander device, every expander route entry for that phy shall be disabled (i.e., the ROUTED SAS ADDRESS field shall be set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit shall be set to one).

If the phy is attached to an edge expander device, the expander route table shall be configured for that phy as follows. For purposes of configuring the expander route table for that phy, the edge expander devices attached to the expander phy are assigned levels:

  1) the expander device in which the expander route table is being configured is level 0;

  2) the attached edge expander device is considered level 1;

  3) devices attached to the level 1 edge expander device, except for the level 0 expander device, are considered level 2;

  4) devices attached to level 2 edge expander devices, except for level 1 edge expander devices, are considered level 3; and

  5) for each n greater than 3, devices attached to level n-1 edge expander devices, except for level n-2 edge expander devices, are considered level n.

The expander route table for each expander phy shall be configured starting from expander route index 0 by level (i.e., all level 1 entries first, then all level 2 entries, then all level 3 entries, etc.) up to the value of the EXPANDER ROUTE INDEXES field reported by the SMP REPORT GENERAL function (see 10.4.3.3).

Assuming the level 1 edge expander device has expander phys attached to N phys with qualified SAS addresses (see 4.7.3), the first N entries shall be used for those SAS addresses in expander phy order (i.e., the addresses attached to lower expander phy numbers first).

For each of the level 2 devices that:

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

the next M entries shall be used for the level 2 edge expander device's qualified SAS addresses in expander phy order (i.e., lower phy numbers first).

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

After the expander route table has been configured with entries for all levels of edge expander devices, all remaining expander route table entries, if any, shall be disabled (i.e., the ROUTED SAS ADDRESS field shall be set to 00000000 00000000h and the DISABLE EXPANDER ROUTE ENTRY bit shall be set to one). The management application client is not required to disable entries if the topology of expander devices has not changed.

Figure 48 shows a portion of an edge expander device set, where phy A in edge expander device R is being configured.
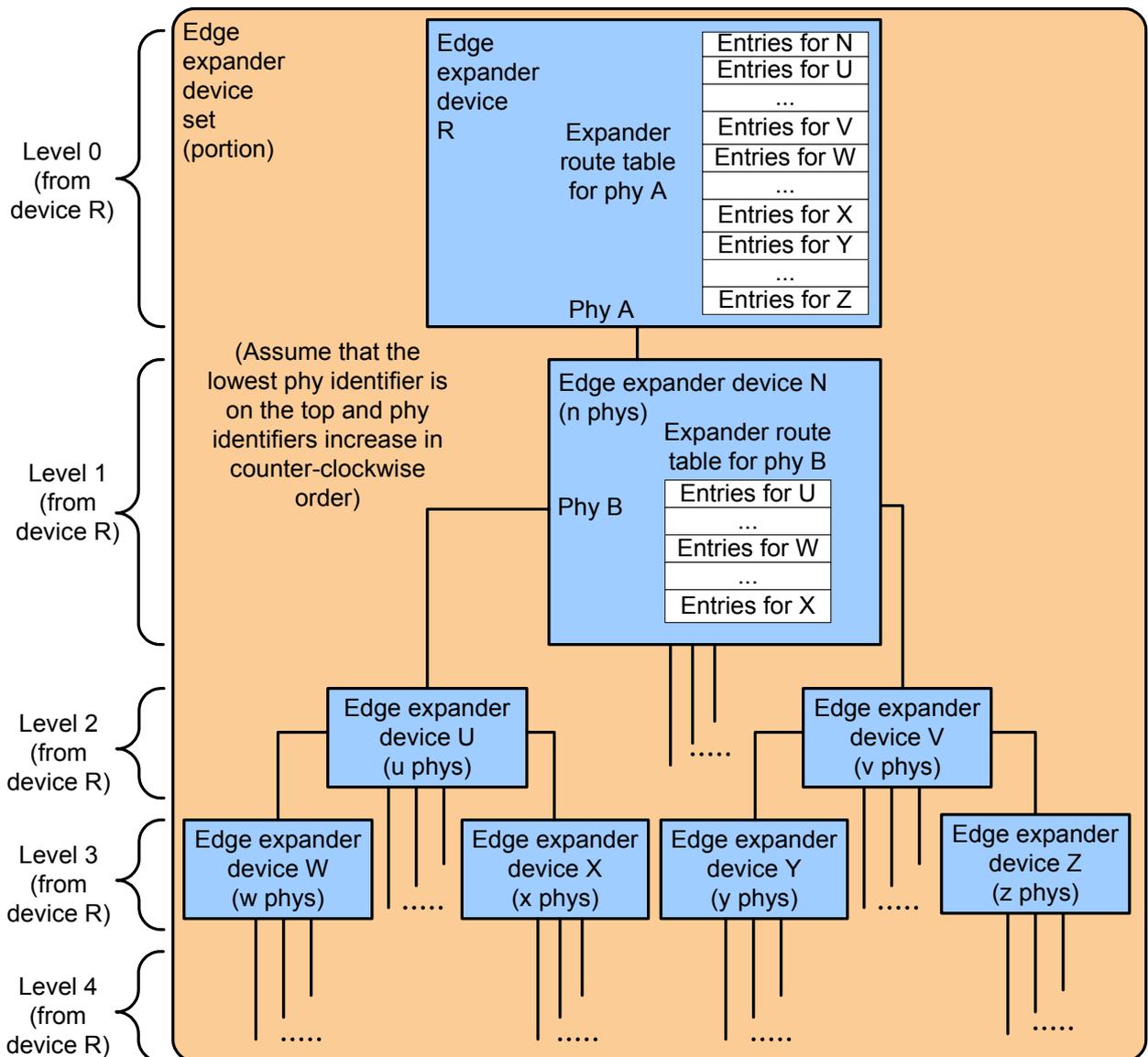
Figure 48 — Expander route index levels example

Figure 49 shows a fanout expander device and an edge expander device set, where phy A in the fanout expander device is being configured.
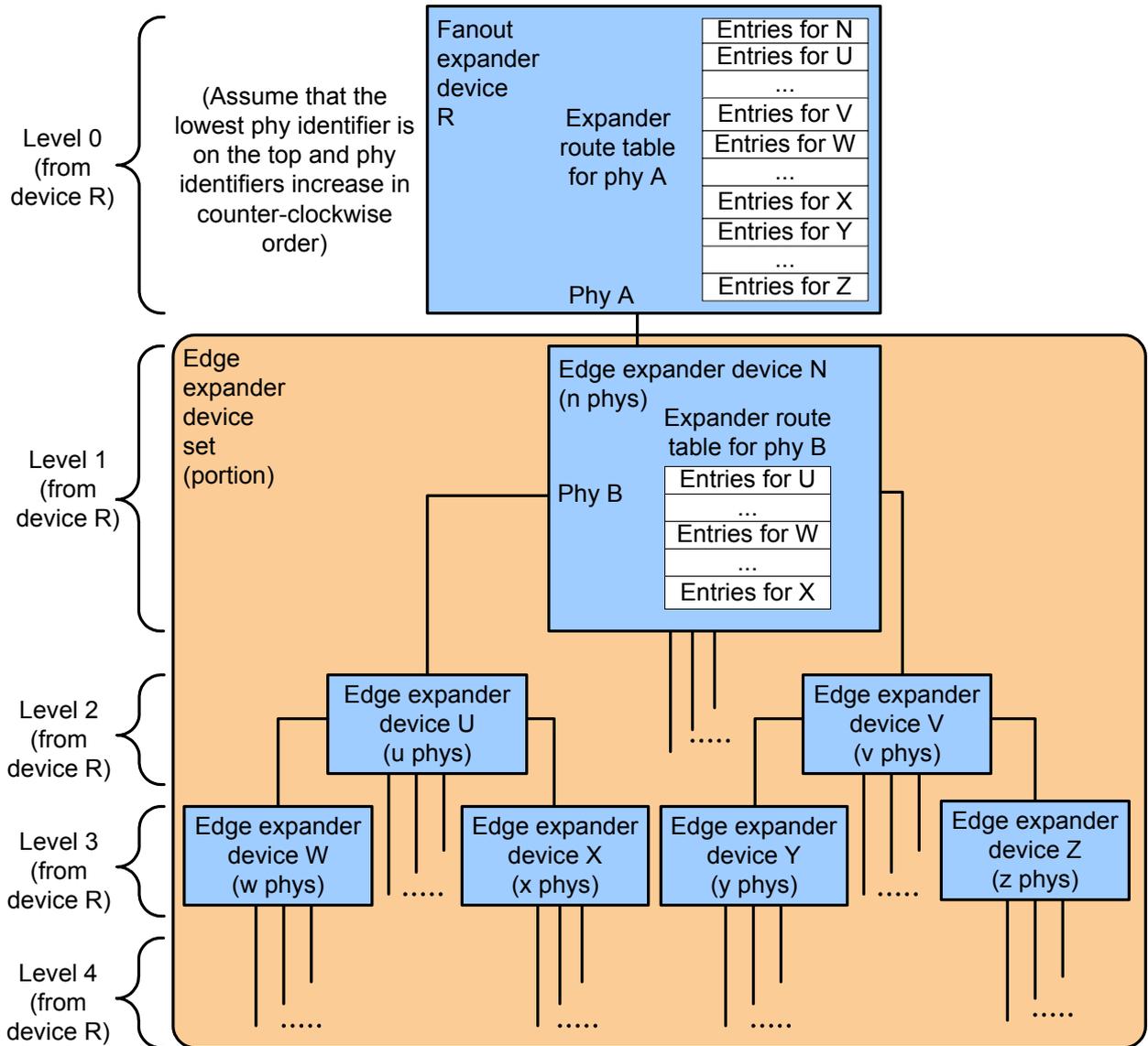
Figure 49 — Expander route index levels example with fanout expander device

Table 17 shows how the expander route table is configured for edge expander device R phy A in figure 48 and the fanout expander device R phy A in figure 49.

Table 17 — Expander route table levels for edge expander device R or fanout expander device R

Expander route index	Expander route entry contents
Level 1 (from device R) entries	
0 .. (≤ n entries)	Qualified SAS addresses attached to edge expander device N
Level 2 (from device R) entries	
(≤ u entries)	Qualified SAS addresses attached to edge expander device U
...	...additional qualified SAS addresses for expander devices at level 2...
(≤ v entries)	Qualified SAS addresses attached to edge expander device V
Level 3 (from device R) entries	
(≤ w entries)	Qualified SAS addresses attached to edge expander device W
...	...additional qualified SAS addresses for expander devices at level 3...
(≤ x entries)	Qualified SAS addresses attached to edge expander device X
(≤ y entries)	Qualified SAS addresses attached to edge expander device Y
...	...additional qualified SAS addresses for expander devices at level 3...
(≤ z entries)	Qualified SAS addresses attached to edge expander device Z
Entries for additional levels	
...	...
Disabled entries	
...	...

Table 18 shows how the expander route table is configured for edge expander device N phy B in figure 48 and figure 49.

Table 18 — Expander route table levels for edge expander device N

Expander route index	Expander route entry contents
Level 1 (from device N) entries	
(≤ u entries)	Qualified SAS addresses attached to edge expander device U
Level 2 (from device N) entries	
(≤ w entries)	Qualified SAS addresses attached to edge expander device W
...	...additional qualified SAS addresses for expander devices at level 2...
(≤ x entries)	Qualified SAS addresses attached to edge expander device X
Entries for additional levels	
...	...
Disabled entries	
...	...

Figure 50 shows an example topology with a fanout expander device.
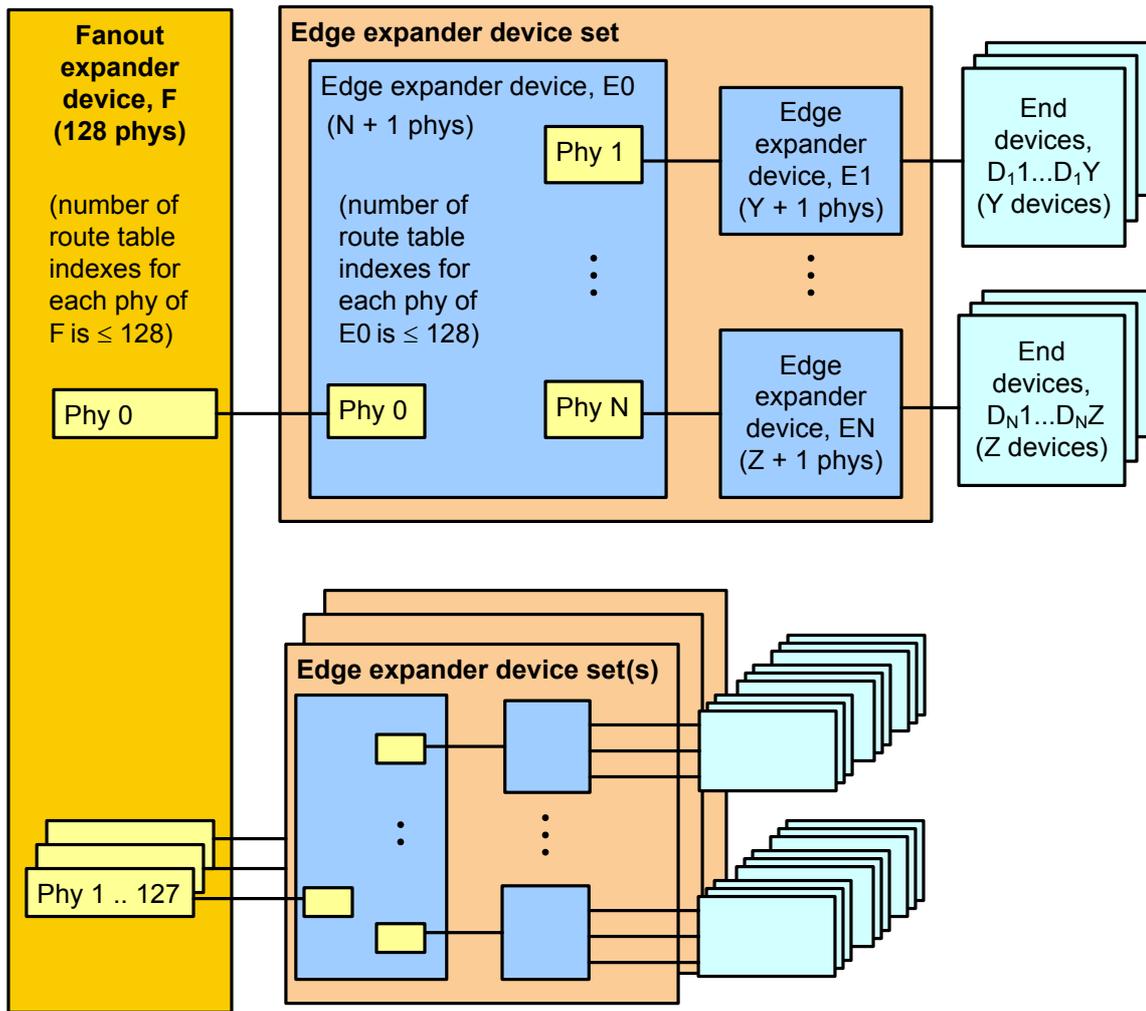
Figure 50 — Expander route index order example

Table 19 shows the expander route index order for the edge expander E0 phy 1 in figure 50.

Table 19 — Expander route entries for edge expander E0 phy 1

Expander route index	Expander route entry contents
Level 1 entries	
0	SAS address (e.g., $D_1 1$) of the device attached to phy 1 of edge expander device E1
1	SAS address (e.g., $D_1 2$) of the device attached to phy 2 of edge expander device E1
...	...
Y - 1	SAS address (e.g., $D_1 Y$) of the device attached to phy Y of edge expander device E1
Level 2 and beyond: no entries	
Disabled entries	

Table 20 shows the expander route index order for the fanout expander F phy 0 in figure 50.

Table 20 — Expander route entries for fanout expander device F phy 0

Expander route index	Expander route entry contents
Level 1 entries	
0	SAS address (e.g., E1) of the device attached to phy 1 of edge expander device E0
...	...additional qualified SAS addresses for edge expander device E0...
N - 1	SAS address (e.g., EN) of the device attached to phy N of edge expander device E0
Level 2 entries	
N	SAS address (e.g., $D_1 1$) of the device attached to phy 1 of edge expander device E1
...	...additional qualified SAS addresses for edge expander device E1...
	SAS address (e.g., $D_1 Y$) of the device attached to phy Y of edge expander device E1
...	...additional qualified SAS addresses for edge expander devices E2 through EN-1...
	SAS address (e.g., $D_N 1$) of the device attached to phy 1 of edge expander device EN
...	...additional qualified SAS addresses for edge expander device EN...
	SAS address (e.g., $D_N Z$) of the device attached to phy Z of edge expander device EN
Level 3 and beyond: no entries since all devices attached to E1 through EN, except for E0, are end devices	
Disabled entries	

4.8 Phy test functions

The optional Protocol-Specific diagnostic page for SAS (see 10.2.9.1) provide methods for an application client to enable and disable a phy test function (e.g., transmission of the CJTPAT) for a selected phy in a SAS target device with an SSP target port. The optional SMP PHY TEST FUNCTION function (see 10.4.3.11) provides similar methods for expander devices and SAS target devices with SMP target ports.

The application client sends a SEND DIAGNOSTIC command with the Protocol-Specific diagnostic page or an SMP PHY TEST FUNCTION function specifying the phy in the SAS target device that is to perform the phy test function and the phy test function to be performed. If the phy test function requires a specific phy test pattern and/or phy test pattern physical link rate, then it also specifies the phy test pattern and phy test pattern physical link rate.

The SEND DIAGNOSTIC command may be sent through any SSP target port to any logical unit in the SAS target device that contains the phy that is to perform the phy test function.

For the SEND DIAGNOSTIC command, the phy shall begin the specified phy test function after the SSP target port receives an ACK for the RESPONSE frame transmitted in response to the SEND DIAGNOSTIC command that requested the phy test function. For the SMP PHY TEST FUNCTION function, the phy shall begin the specified phy test function after the SMP target port transmits the SMP response frame.

Once a SAS phy has begun performing a phy test function, it shall ignore its receiver. To stop a SAS phy from performing a phy test function, an application client sends a SEND DIAGNOSTIC command or an SMP PHY TEST FUNCTION function to a SAS phy in the SAS target device that is not performing a phy test function requesting a phy test function of 00h (i.e., STOP). If no such phy is available, the phy test function only stops on power loss.

5 Physical layer

5.1 Physical layer overview

The physical layer defines:

a) passive interconnect (e.g., connectors and cable assemblies); and
b) transmitter and receiver device electrical characteristics.

Within this standard, references to connector gender use the terms plug and receptacle as equivalent to the terms free and fixed, respectively, that may be used in the references that define the connectors. Fixed and free terminology has no relationship to the application of the connector.

5.2 Passive interconnect

5.2.1 SATA connectors and cable assemblies

Figure 51 shows a schematic representation of the connectors and cables defined by SATA (see ATA/ATAPI-7 V3). A SATA host is analogous to a SAS initiator device and a SATA device is analogous to a SAS target device.
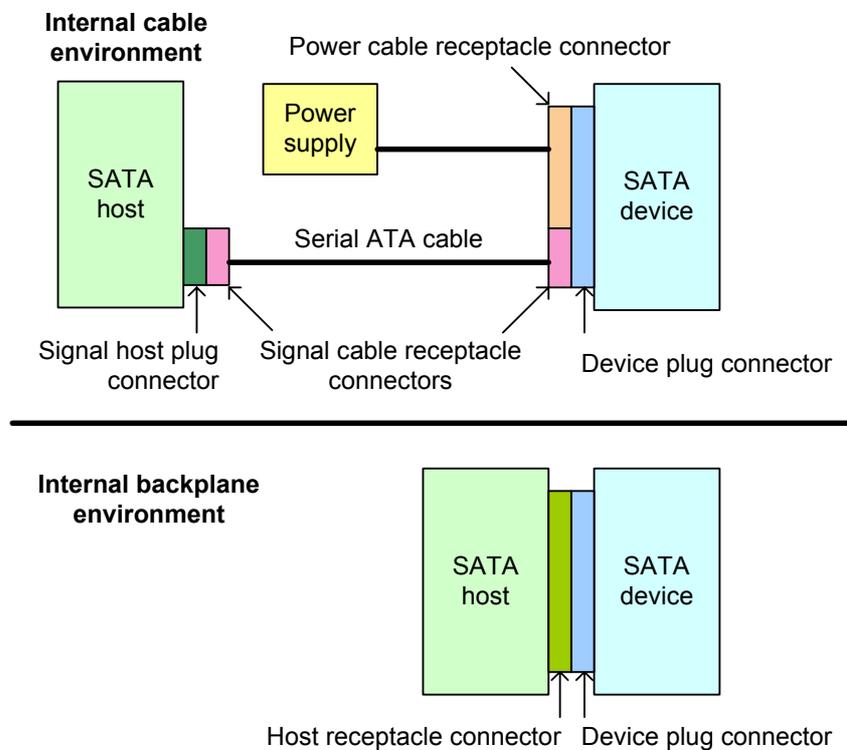
Figure 51 — SATA connectors and cables

5.2.2 SAS connectors and cables

This standard defines SAS Drive cable, SAS Drive backplane, SAS internal cable, and SAS external cable environments.

Figure 52 shows a schematic representation of the SAS Drive cable environments.

Single-port SAS Drive cable environment

Dual-port SAS Drive cable environment
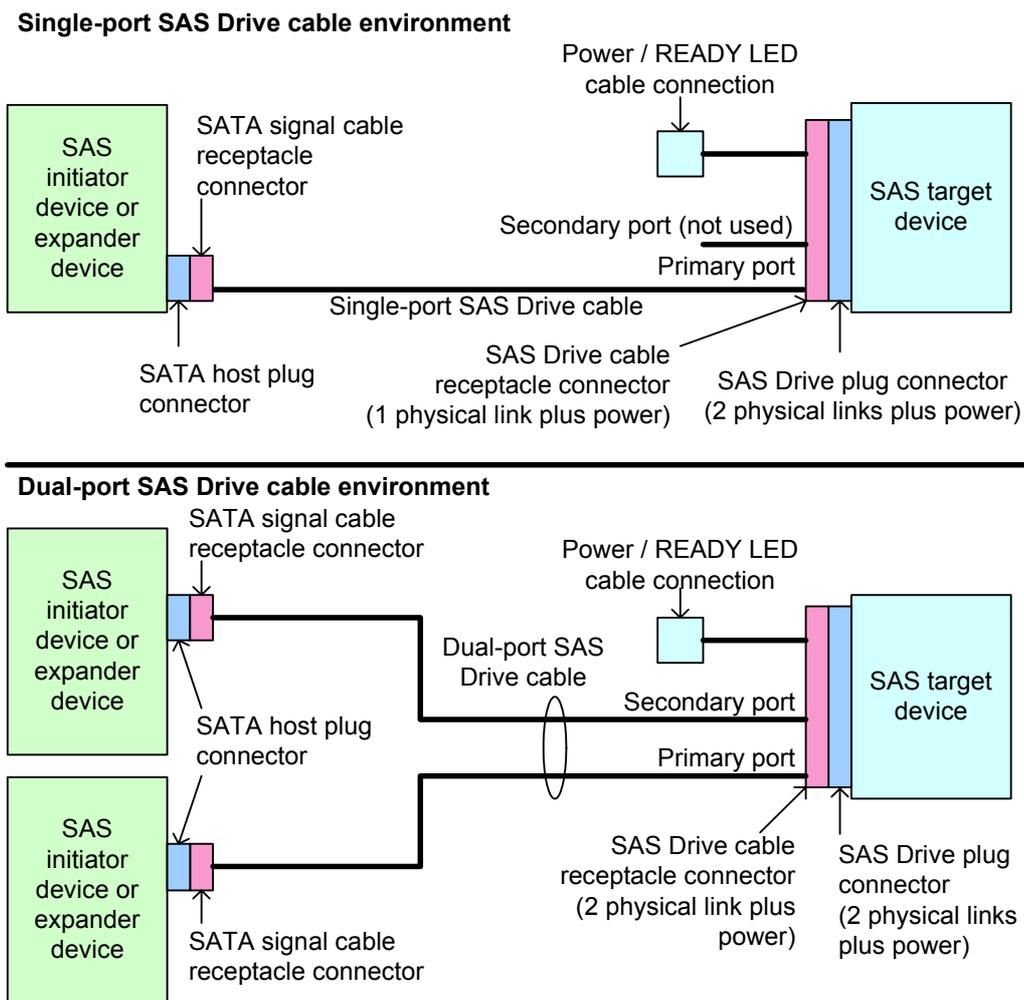
Figure 52 — SAS Drive cable environments

Figure 53 shows a schematic representation of the SAS Drive backplane environment.
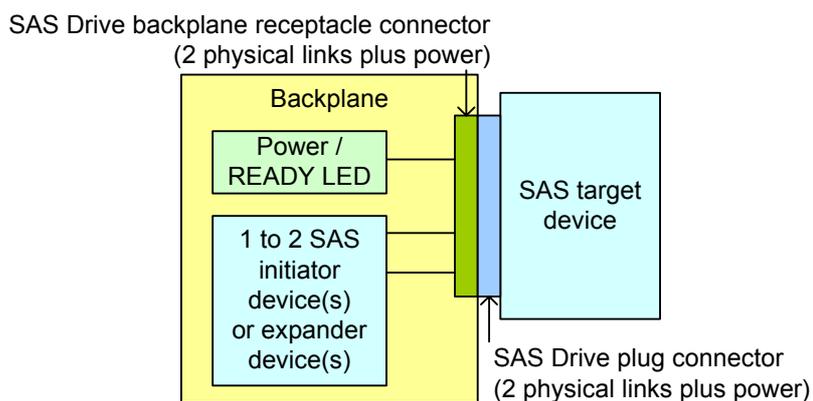
Figure 53 — SAS Drive backplane environment

Figure 54 shows a schematic representation of the SAS external cable environment.
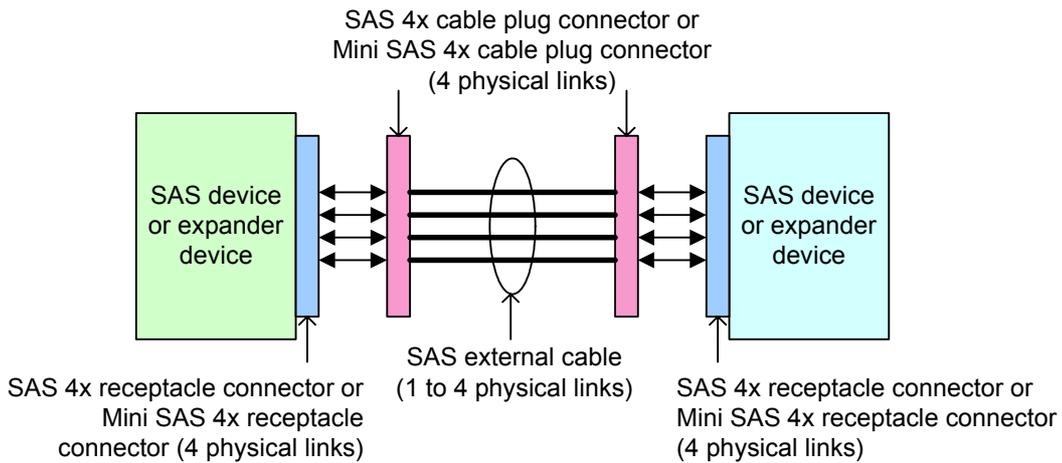
Figure 54 — SAS external cable environment

Figure 55 shows a schematic representation of the SAS internal cable environment attaching a controller to a backplane using a SAS internal symmetric cable.
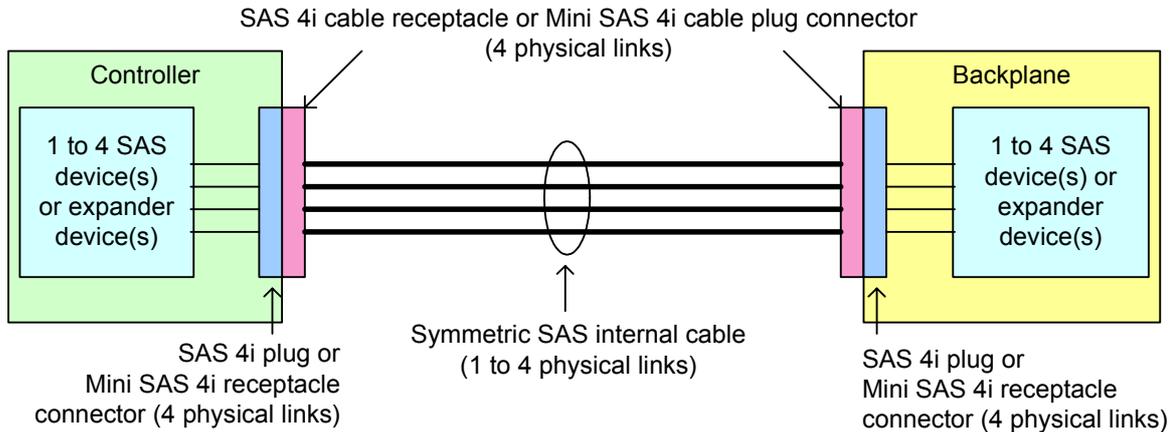
Figure 55 — SAS internal symmetric cable environment - controller to backplane

A SAS internal symmetric cable provides one to four physical links, and may be used as any combination of wide links and narrow links (see 4.1.3) using those physical links.

Figure 56 shows a schematic representation of the SAS internal cable environment attaching a controller to a controller using a SAS internal symmetric cable. Two controllers may also be attached together with a SAS internal symmetric cable. If SAS 4i connectors are used, all four physical links are used (see 5.2.4.1.2).
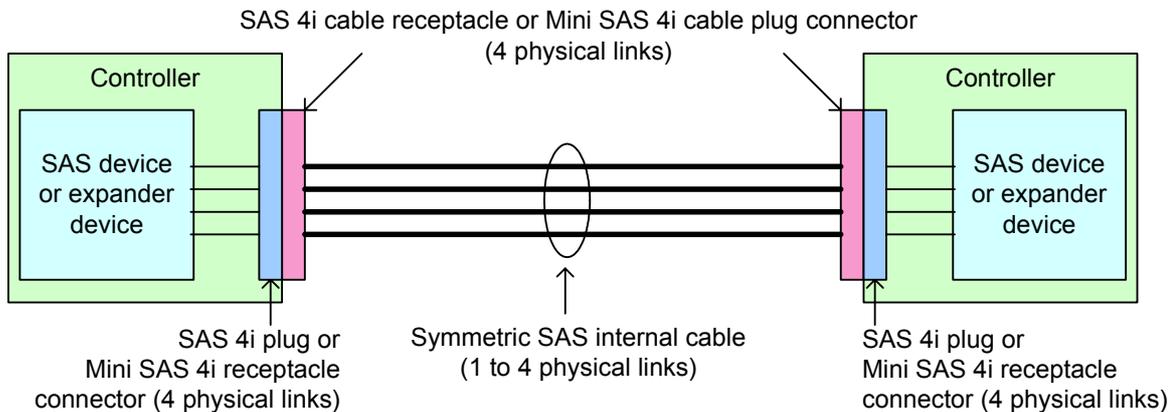
Figure 56 — SAS internal symmetric cable environment - controller to controller

Figure 57 shows a schematic representation of the SAS internal cable environment using a SAS controller-based fanout cable.

SAS Drive cable receptacle connectors
(each only using primary physical link)

Power / READY LED cable connection

Secondary physical link (not used)

SAS device or SATA device

SAS 4i cable receptacle or Mini SAS 4i cable plug connector (4 physical links)

Controller

1 to 4 SAS device(s) or expander device(s)

Power / READY LED cable connection

Secondary physical link (not used)

SAS device or SATA device

SAS 4i plug or Mini SAS 4i receptacle connector (4 physical links)

Power / READY LED cable connection

Secondary physical link (not used)

SAS device or SATA device

SAS internal controller-based fanout cable (4 physical links)

Power / READY LED cable connection

Secondary physical link (not used)

SAS device or SATA device

SAS Drive plug connectors
(each only using primary physical link)
or SATA device plug connectors

Figure 57 — SAS internal controller-based fanout cable environment

Figure 58 shows a schematic representation of the SAS internal cable environment using a SAS backplane-based fanout cable.

SATA signal cable receptacle connectors

SAS 4i cable receptacle or Mini SAS 4i cable plug connector (4 physical links)

Backplane

1 to 4 SAS device(s) or expander device(s)

1 to 4 SAS device(s) or expander device(s)

SAS internal backplane-based fanout cable (4 physical links)

SAS 4i plug or Mini SAS 4i receptacle connector (4 physical links)

SATA host plug connectors
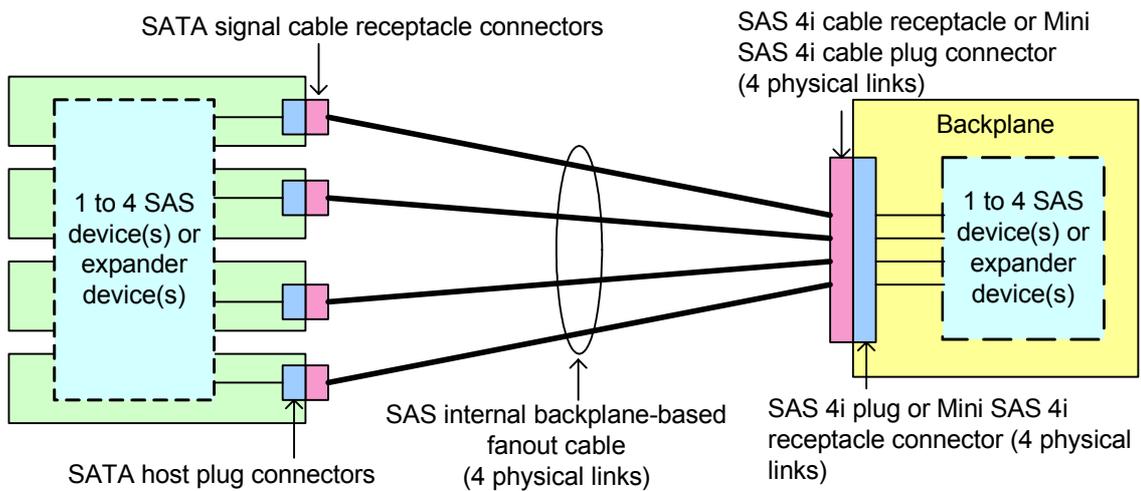
Figure 58 — SAS internal backplane-based fanout cable environment

5.2.3 Connectors

5.2.3.1 Connectors overview

Table 21 summarizes the connectors defined in this standard.

Table 21 — Connectors

Type of connector	Physical links	Reference	Attaches to		
			Type of connector	Physical links	Reference
SATA internal connectors used by SAS					
SATA signal cable receptacle	1	ATA/ATAPI-7 V3	SATA host plug	1	ATA/ATAPI-7 V3
SATA host plug	1	ATA/ATAPI-7 V3	SATA signal cable receptacle	1	ATA/ATAPI-7 V3
SATA device plug	1	ATA/ATAPI-7 V3	SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2
			SAS Drive backplane receptacle	2	5.2.3.2.1.3
SAS internal connectors - SAS Drive connectors					
SAS Drive plug	2	5.2.3.2.1.1	SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2
			SAS Drive backplane receptacle	2	5.2.3.2.1.3
SAS Drive cable receptacle	1 or 2	5.2.3.2.1.2	SAS Drive plug	2	5.2.3.2.1.1
			SATA device plug	1	ATA/ATAPI-7 V3
SAS Drive backplane receptacle	2	5.2.3.2.1.3	SAS Drive plug	2	5.2.3.2.1.1
			SATA device plug	1	ATA/ATAPI-7 V3
SAS internal connectors - other					
SAS 4i cable receptacle	4	5.2.3.2.2.1	SAS 4i plug	4	5.2.3.2.2.2
SAS 4i plug	4	5.2.3.2.2.2	SAS 4i cable receptacle	4	5.2.3.2.2.1
Mini SAS 4i cable plug	4	5.2.3.2.3.1	Mini SAS 4i receptacle	4	5.2.3.2.3.2
Mini SAS 4i receptacle	4	5.2.3.2.3.2	Mini SAS 4i cable plug	4	5.2.3.2.3.1
SAS external connectors					
SAS 4x cable plug	4	5.2.3.3.1.1	SAS 4x receptacle	4	5.2.3.3.1.2
SAS 4x receptacle	4	5.2.3.3.1.2	SAS 4x cable plug	4	5.2.3.3.1.1
Mini SAS 4x cable plug	4	5.2.3.3.2.1	Mini SAS 4x receptacle	4	5.2.3.3.2.2
Mini SAS 4x receptacle	4	5.2.3.3.2.2	Mini SAS 4x plug	4	5.2.3.3.2.1

The general SAS icon (see Annex M) should be placed on or near each SAS connector.

5.2.3.2 SAS internal connectors

5.2.3.2.1 SAS Drive connectors

5.2.3.2.1.1 SAS Drive plug connector

The SAS Drive plug connector is the Device Free (Plug) connector defined in SFF-8482.

See SFF-8223, SFF-8323, and SFF-8523 for the SAS Drive plug connector locations on common form factors.

Figure 59 shows the SAS Drive plug connector.
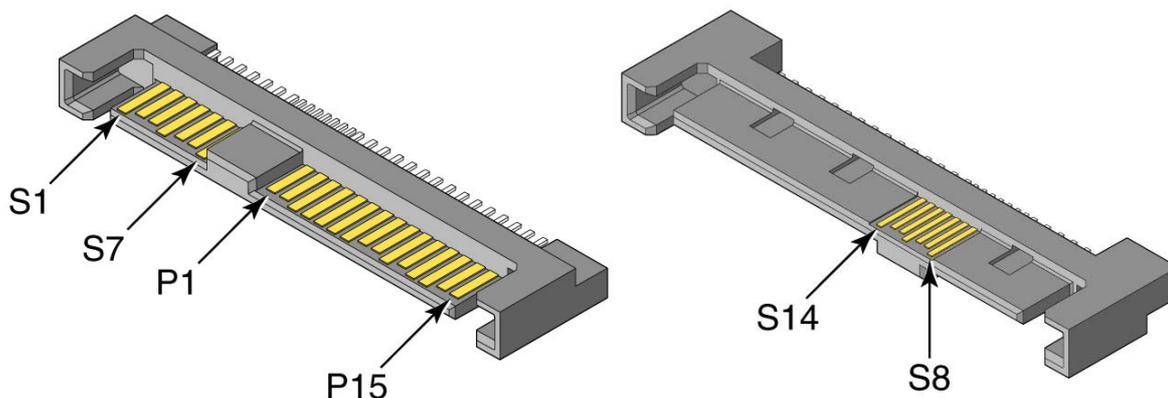
Figure 59 — SAS Drive plug connector

Table 22 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive plug connector.

5.2.3.2.1.2 SAS Drive cable receptacle connector

The SAS Drive cable receptacle connector is the Internal Cable Fixed (Receptacle) connector defined in SFF-8482.

The single-port version attaches to:

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

Figure 60 shows the single-port version of the SAS Drive cable receptacle connector.
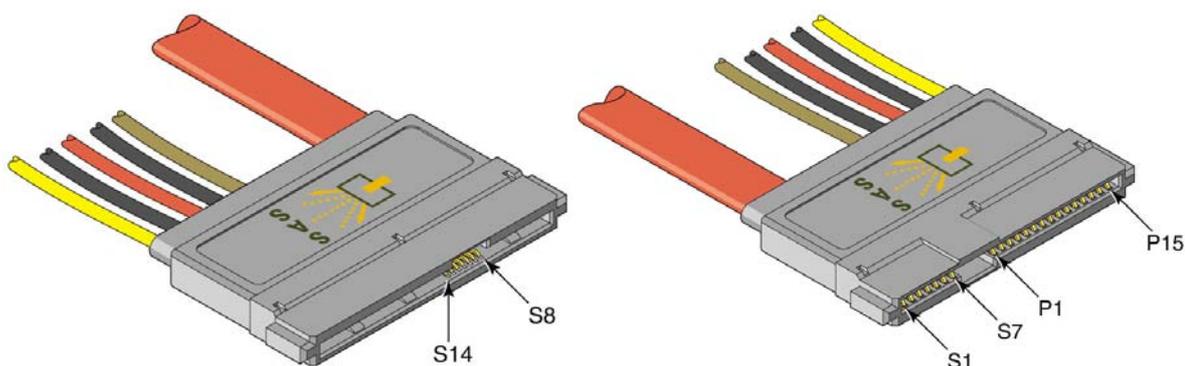
Figure 60 — Single-port SAS Drive cable receptacle connector

The dual-port version attaches to:

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

Figure 61 shows the dual-port version of the SAS Drive cable receptacle connector.
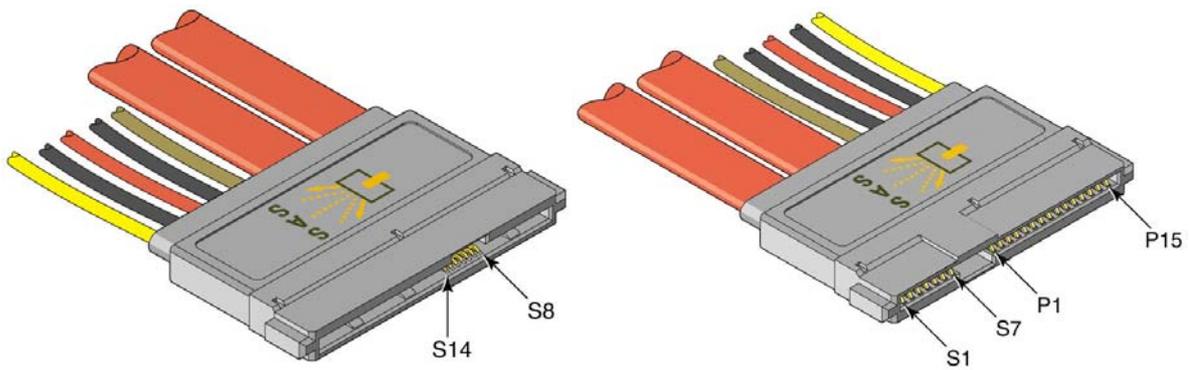
Figure 61 — Dual-port SAS Drive cable receptacle connector

Table 22 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive cable receptacle connector. The secondary physical link (i.e., pins S8 through S14) is not supported by the single-port internal cable receptacle.

5.2.3.2.1.3 SAS Drive backplane receptacle connector

The SAS Drive backplane receptacle connector is the Backplane Fixed (Receptacle) connector defined in SFF-8482.

The SAS Drive backplane receptacle connector attaches to:

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

Figure 62 shows the SAS Drive backplane receptacle connector.
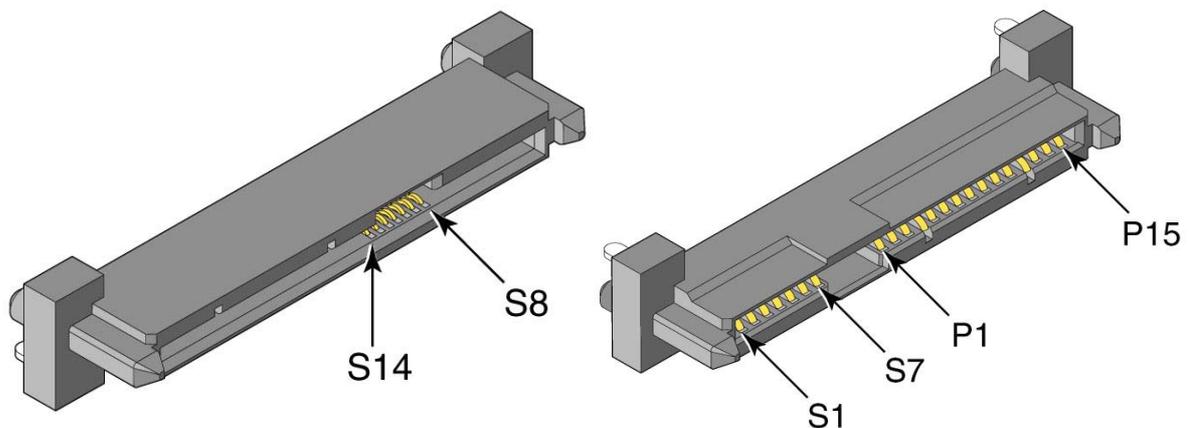
Figure 62 — SAS Drive backplane receptacle connector

Table 22 (see 5.2.3.2.1.4) defines the pin assignments for the SAS Drive backplane receptacle connector.

5.2.3.2.1.4 SAS Drive connector pin assignments

Table 22 defines the SAS target device pin assignments for the SAS Drive plug connector (see 5.2.3.2.1.1), the SAS Drive cable receptacle connector (see 5.2.3.2.1.2), and the SAS Drive backplane receptacle connector (see 5.2.3.2.1.3). The TP+, TP-, RP+, and RP- signals are used by the primary physical link. The TS+, TS-, RS+, and RS- signals are used by the secondary physical link, if any.

SAS Drive plug connector pin assignments, except for the addition of the secondary physical link when present, are in the same locations as they are in a SATA device plug connector (see ATA/ATAPI-7 V3).

Table 22 — SAS Drive connector pin assignments

Segment	Pin	Backplane receptacle	SAS Drive plug and SAS Drive cable receptacle
Primary signal segment	S1	SIGNAL GROUND	
	S2	TP+	RP+
	S3	TP-	RP-
	S4	SIGNAL GROUND	
	S5	RP-	TP-
	S6	RP+	TP+
	S7	SIGNAL GROUND	
Secondary signal segment [a]	S8	SIGNAL GROUND	
	S9	TS+	RS+
	S10	TS-	RS-
	S11	SIGNAL GROUND	
	S12	RS-	TS-
	S13	RS+	TS+
	S14	SIGNAL GROUND	
Power segment [b]	P1	V_{33} [c]	
	P2	V_{33} [c]	
	P3	V_{33}, precharge [c]	
	P4	GROUND	
	P5	GROUND	
	P6	GROUND	
	P7	V_5, precharge [c]	
	P8	V_5 [c]	
	P9	V_5 [c]	
	P10	GROUND	
	P11	READY LED [d]	
	P12	GROUND	
	P13	V_{12}, precharge [c]	
	P14	V_{12} [c]	
	P15	V_{12} [c]	

[a] S8 through S14 are no-connects on single-port implementations.

[b] Backplane receptacle connectors and SAS Drive cable receptacle connectors provide V_{33}, V_5, and V_{12}. SAS Device plug connectors receive V_{33}, V_5, and V_{12}.

[c] Behind a SAS Drive plug connector, the precharge pin and each corresponding voltage pin shall be connected together on the SAS target device (e.g., the V_5, precharge pin P7 is connected to the two V_5 pins P8 and P9).

[d] Electrical characteristics for READY LED are defined in 5.4 and signal behavior is defined in 10.4.1. SATA devices use P11 for activity indication and staggered spin-up disable and have different electrical characteristics (see SATAII-EXT).

5.2.3.2.2 SAS 4i connectors

5.2.3.2.2.1 SAS 4i cable receptacle connector

The SAS 4i cable receptacle connector is the 4 Lane Cable Receptacle (fixed) with Backshell connector defined in SFF-8484.

Figure 63 shows the SAS 4i cable receptacle connector.
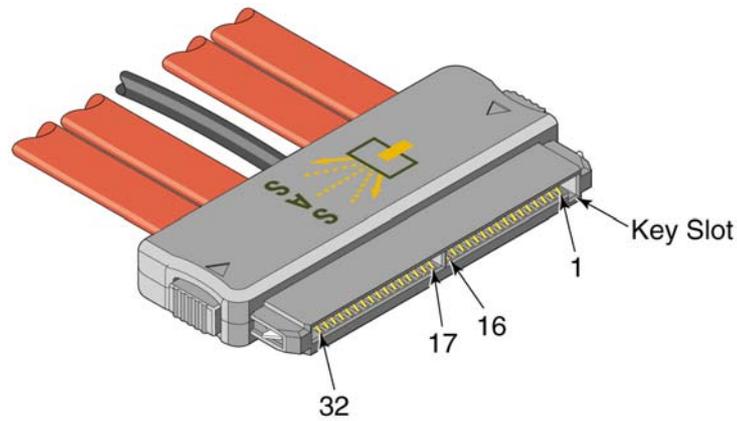
Figure 63 — SAS 4i cable receptacle connector

Table 23 and table 24 (see 5.2.3.2.2.3) define the pin assignments for the SAS 4i cable receptacle connector.

5.2.3.2.2.2 SAS 4i plug connector

The SAS 4i plug connector is the 4 Lane Vertical Plug (free) or 4 Lane R/A Plug (free) connector defined in SFF-8484.

Figure 64 shows the SAS 4i plug connector.
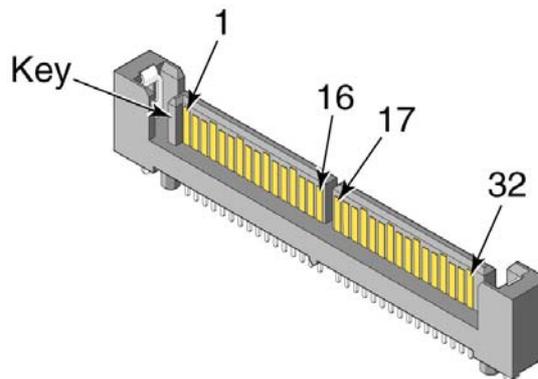
Figure 64 — SAS 4i plug connector

Table 23 and table 24 (see 5.2.3.2.2.3) define the pin assignments for the SAS 4i plug connector.

5.2.3.2.2.3 SAS 4i connector pin assignments

Table 23 defines the pin assignments for SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) and SAS 4i plug connectors (see 5.2.3.2.2.2) for controller applications using one, two, three, or four of the physical links.

Table 23 — Controller SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]			
	One	**Two**	**Three**	**Four**
Rx 0+	2	2	2	2
Rx 0-	3	3	3	3
Tx 0-	5	5	5	5
Tx 0+	6	6	6	6
Rx 1+	N/C	8	8	8
Rx 1-	N/C	9	9	9
Tx 1-	N/C	11	11	11
Tx 1+	N/C	12	12	12
Sideband 0	14	14	14	14
Sideband 1	15	15	15	15
Sideband 2	16	16	16	16
Sideband 3	17	17	17	17
Sideband 4	18	18	18	18
Sideband 5	19	19	19	19
Rx 2+	N/C	N/C	21	21
Rx 2-	N/C	N/C	22	22
Tx 2-	N/C	N/C	24	24
Tx 2+	N/C	N/C	25	25
Rx 3+	N/C	N/C	N/C	27
Rx 3-	N/C	N/C	N/C	28
Tx 3-	N/C	N/C	N/C	30
Tx 3+	N/C	N/C	N/C	31
SIGNAL GROUND	1, 4, 7, 10, 13, 20, 23, 26, 29, 32			
[a] N/C = not connected				

The use of the sideband signals by a controller is vendor-specific. One implementation of the sideband signals by a controller is an SGPIO initiator interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

Table 24 defines the pin assignments for SAS 4i plug connectors (see 5.2.3.2.2.1) and SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) for backplane applications using one, two, three, or four of the physical links.

Table 24 — Backplane SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]			
	One	Two	Three	Four
Rx 3+	N/C	N/C	N/C	2
Rx 3-	N/C	N/C	N/C	3
Tx 3-	N/C	N/C	N/C	5
Tx 3+	N/C	N/C	N/C	6
Rx 2+	N/C	N/C	8	8
Rx 2-	N/C	N/C	9	9
Tx 2-	N/C	N/C	11	11
Tx 2+	N/C	N/C	12	12
Sideband 5	14	14	14	14
Sideband 4	15	15	15	15
Sideband 3	16	16	16	16
Sideband 2	17	17	17	17
Sideband 1	18	18	18	18
Sideband 0	19	19	19	19
Rx 1+	N/C	21	21	21
Rx 1-	N/C	22	22	22
Tx 1-	N/C	24	24	24
Tx 1+	N/C	25	25	25
Rx 0+	27	27	27	27
Rx 0-	28	28	28	28
Tx 0-	30	30	30	30
Tx 0+	31	31	31	31
SIGNAL GROUND	1, 4, 7, 10, 13, 20, 23, 26, 29, 32			
[a] N/C = not connected				

The use of the sideband signals by a backplane is vendor-specific. One implementation of the sideband signals by a backplane is an SGPIO target interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

5.2.3.2.3 Mini SAS 4i connectors

5.2.3.2.3.1 Mini SAS 4i cable plug connector

The Mini SAS 4i cable plug connector is the free (plug) 36-circuit Unshielded Compact Multilane connector defined in SFF-8087 and SFF-8086.

Figure 65 shows the Mini SAS 4i cable plug connector.
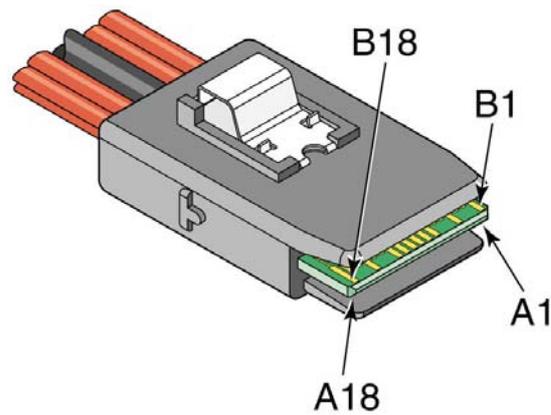
Figure 65 — Mini SAS 4i cable plug connector

Table 25 and table 26 (see 5.2.3.2.3.3) define the pin assignments for the Mini SAS 4i cable plug connector.

5.2.3.2.3.2 Mini SAS 4i receptacle connector

The Mini SAS 4i receptacle connector is the fixed (receptacle) 36-circuit Unshielded Compact Multilane connector defined in SFF-8087 and SFF-8086.

Figure 66 shows the Mini SAS 4i receptacle connector.
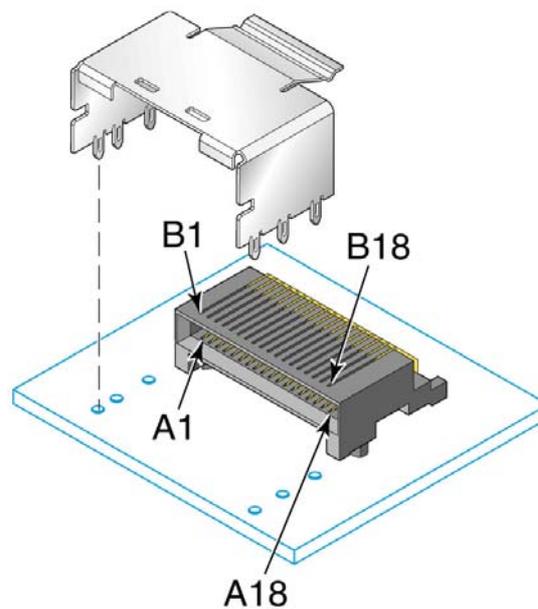
Figure 66 — Mini SAS 4i receptacle connector

Table 25 and table 26 (see 5.2.3.2.3.3) define the pin assignments for the Mini SAS 4i receptacle connector.

5.2.3.2.3.3 Mini SAS 4i connector pin assignments

Table 25 defines the pin assignments for Mini SAS 4i plug connectors (see 5.2.3.2.3.1) and Mini SAS 4i cable receptacle connectors (see 5.2.3.2.3.2) for controller applications using one, two, three, or four of the physical links.

Table 25 — Controller Mini SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]				Mating length
	One	Two	Three	Four	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Sideband 7	A8	A8	A8	A8	First
Sideband 3	A9	A9	A9	A9	
Sideband 4	A10	A10	A10	A10	
Sideband 5	A11	A11	A11	A11	
Rx 2+	N/C	N/C	A13	A13	Third
Rx 2-	N/C	N/C	A14	A14	
Rx 3+	N/C	N/C	N/C	A16	
Rx 3-	N/C	N/C	N/C	A17	
Tx 0+	B2	B2	B2	B2	Third
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Sideband 0	B8	B8	B8	B8	First
Sideband 1	B9	B9	B9	B9	
Sideband 2	B10	B10	B10	B10	
Sideband 6	B11	B11	B11	B11	
Tx 2+	N/C	N/C	B13	B13	Third
Tx 2-	N/C	N/C	B14	B14	
Tx 3+	N/C	N/C	N/C	B16	
Tx 3-	N/C	N/C	N/C	B17	
SIGNAL GROUND	A1, A4, A7, A12, A15, A18, B1, B4, B7, B12, B15, B18				First
[a] N/C = not connected					

The use of the sideband signals by a controller is vendor-specific. One implementation of the sideband signals by a controller is an SGPIO initiator interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

Table 26 defines the pin assignments for Mini SAS 4i plug connectors (see 5.2.3.2.3.1) and Mini SAS 4i cable receptacle connectors (see 5.2.3.2.3.2) for backplane applications using one, two, three, or four of the physical links.

Table 26 — Backplane Mini SAS 4i connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]				Mating length
	One	**Two**	**Three**	**Four**	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Sideband 0	A8	A8	A8	A8	First
Sideband 1	A9	A9	A9	A9	
Sideband 2	A10	A10	A10	A10	
Sideband 6	A11	A11	A11	A11	
Rx 2+	N/C	N/C	A13	A13	Third
Rx 2-	N/C	N/C	A14	A14	
Rx 3+	N/C	N/C	N/C	A16	
Rx 3-	N/C	N/C	N/C	A17	
Tx 0+	B2	B2	B2	B2	Third
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Sideband 7	B8	B8	B8	B8	First
Sideband 3	B9	B9	B9	B9	
Sideband 4	B10	B10	B10	B10	
Sideband 5	B11	B11	B11	B11	
Tx 2+	N/C	N/C	B13	B13	Third
Tx 2-	N/C	N/C	B14	B14	
Tx 3+	N/C	N/C	N/C	B16	
Tx 3-	N/C	N/C	N/C	B17	
SIGNAL GROUND	A1, A4, A7, A12, A15, A18, B1, B4, B7, B12, B15, B18				First
[a] N/C = not connected					

The use of the sideband signals by a backplane is vendor-specific. One implementation of the sideband signals by a backplane is an SGPIO target interface (see SFF-8485). Other implementations shall be compatible with the signal levels defined in SFF-8485.

5.2.3.3 SAS external connectors

5.2.3.3.1 SAS 4x connectors

5.2.3.3.1.1 SAS 4x cable plug connector

The SAS 4x cable plug connector is the 4X free (plug) connector with jack screws defined in SFF-8470. The SAS 4x cable plug connector shall not include keys and may include key slots. Key slots for the SAS 4x cable plug connector are not defined by this standard.

Figure 67 shows the SAS 4x cable plug connector.
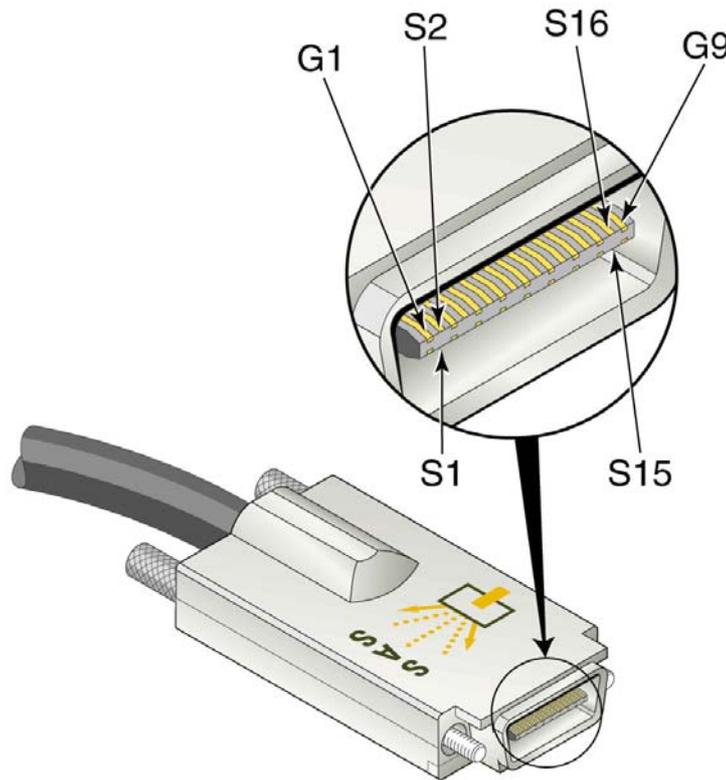
Figure 67 — SAS 4x cable plug connector

Table 29 (see 5.2.3.3.1.3) defines the pin assignments for the SAS 4x cable plug connector.

Table 27 defines the icons that should be placed on or near SAS 4x cable plug connectors.

Table 27 — SAS 4x cable plug connector icons

Use	Icon
End of a SAS external cable that attaches to an end device or an enclosure out port	Diamond
End of a SAS external cable that attaches to an end device or an enclosure in port	Circle

5.2.3.3.1.2 SAS 4x receptacle connector

The SAS 4x receptacle connector is the 4X fixed (receptacle) connector with jack screws defined in SFF-8470. The SAS 4x receptacle connector shall not include keys and may include key slots. Key slots for the SAS 4x receptacle connector are not defined by this standard.

A SAS 4x receptacle connector may be used by one or more SAS devices (e.g., one SAS device using physical links 0 and 3, another using physical link 1, and a third using physical link 2).

A SAS 4x receptacle connector shall be used by no more than one expander device at a time, and all physical links shall be used by the same expander port (i.e., all the expander phys shall have the same routing attribute (e.g., subtractive or table) (see 4.6.2)).

Figure 68 shows the SAS 4x receptacle connector.
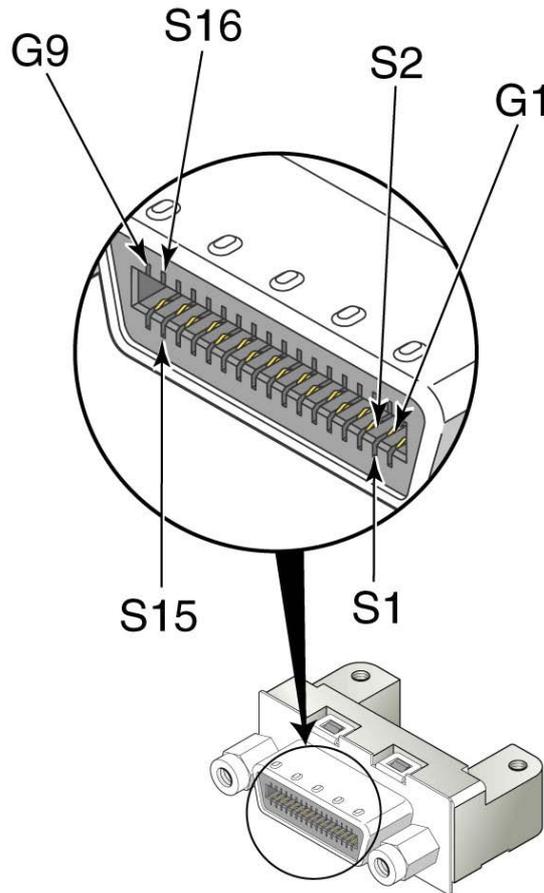
Figure 68 — SAS 4x receptacle connector

Table 29 (see 5.2.3.3.1.3) defines the pin assignments for the SAS 4x receptacle connector.

Table 28 defines the icons that should be placed near SAS 4x receptacle connectors.

Table 28 — SAS 4x receptacle connector icons

Use	Icons
Enclosure out port (see 4.6.2)	Diamond
End device	Diamond and circle
Enclosure in port (see 4.6.2)	Circle

5.2.3.3.1.3 SAS 4x connector pin assignments

Table 29 defines the pin assignments for SAS 4x cable plug connectors (see 5.2.3.3.1.1) and SAS 4x receptacle connectors (see 5.2.3.3.1.2) for applications using one, two, three, or four of the physical links.

Table 29 — SAS 4x connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]			
	One	**Two**	**Three**	**Four**
Rx 0+	S1	S1	S1	S1
Rx 0-	S2	S2	S2	S2
Rx 1+	N/C	S3	S3	S3
Rx 1-	N/C	S4	S4	S4
Rx 2+	N/C	N/C	S5	S5
Rx 2-	N/C	N/C	S6	S6
Rx 3+	N/C	N/C	N/C	S7
Rx 3-	N/C	N/C	N/C	S8
Tx 3-	N/C	N/C	N/C	S9
Tx 3+	N/C	N/C	N/C	S10
Tx 2-	N/C	N/C	S11	S11
Tx 2+	N/C	N/C	S12	S12
Tx 1-	N/C	S13	S13	S13
Tx 1+	N/C	S14	S14	S14
Tx 0-	S15	S15	S15	S15
Tx 0+	S16	S16	S16	S16
SIGNAL GROUND	G1 - G9			
CHASSIS GROUND	Housing			
[a] N/C = not connected				

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the connector when used in a cable assembly.

5.2.3.3.2 Mini SAS 4x connectors

5.2.3.3.2.1 Mini SAS 4x cable plug connector

The Mini SAS 4x cable plug connector is the free (plug) 26-circuit Shielded Compact Multilane connector defined in SFF-8088 and SFF-8086.

Figure 69 shows the Mini SAS 4x cable plug connector.
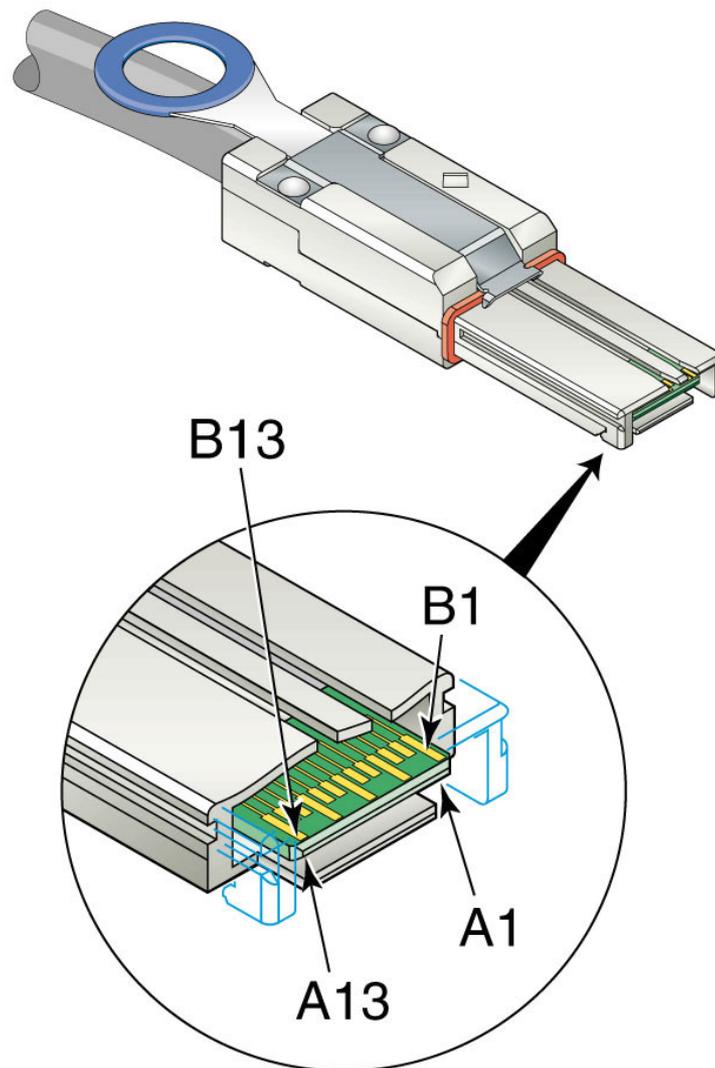
Figure 69 — Mini SAS 4x cable plug connector

Table 32 (see 5.2.3.3.2.3) defines the pin assignments for the Mini SAS 4x cable plug connector.

Mini SAS 4x cable plug connectors shall include key slots to allow attachment to Mini SAS 4x receptacle connectors (see 5.2.3.3.2.2) with matching keys.

Table 30 defines the icons that shall be placed on or near Mini SAS 4x cable plug connectors and the key slot positions (see SFF-8088) that shall be used by Mini SAS 4x cable plug connectors.

Table 30 — Mini SAS 4x cable plug connector icons and key slot positions

Use	Icon	Key slot positions	Reference
End of a SAS external cable that attaches to an end device or an enclosure out port	Diamond	2, 4	Figure 70
End of a SAS external cable that attaches to an end device or an enclosure in port	Circle	4, 6	Figure 71

Figure 70 shows the key slots on the Mini SAS 4x cable plug connector that attaches to either an end device (see figure 73 in 5.2.3.3.2.2) or an enclosure out port (see figure 74 in 5.2.3.3.2.2).
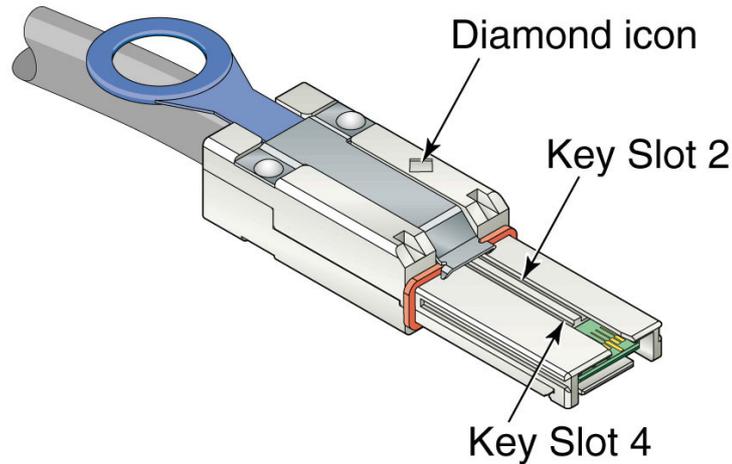
Figure 70 — Mini SAS 4x cable plug connector that attaches to an enclosure out port

Figure 71 shows the key slots on the Mini SAS 4x cable plug connector that attaches to either an end device (see figure 73 in 5.2.3.3.2.2) or an enclosure in port (see figure 75 in 5.2.3.3.2.2).
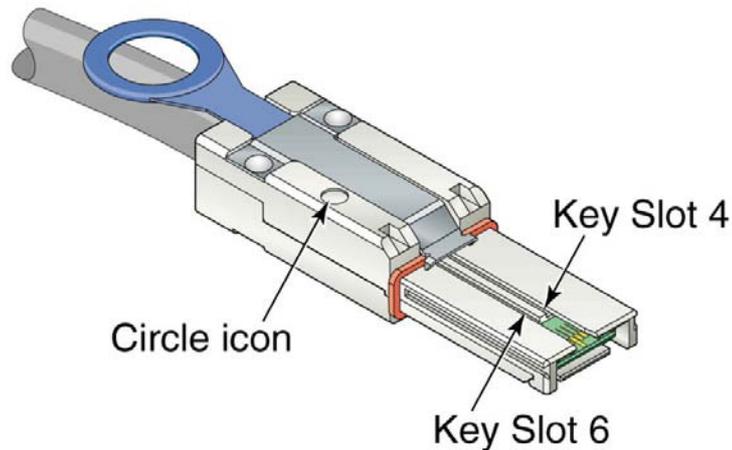
Figure 71 — Mini SAS 4x cable plug connector that attaches to an enclosure in port

5.2.3.3.2.2 Mini SAS 4x receptacle connector

The Mini SAS 4x receptacle connector is the fixed (receptacle) 26-circuit Shielded Compact Multilane connector defined in SFF-8088 and SFF-8086.

A SAS 4x receptacle connector may be used by one or more SAS devices (e.g., one SAS device using physical links 0 and 3, another using physical link 1, and a third using physical link 2).

A SAS 4x receptacle connector shall be used by no more than one expander device at a time, and all physical links shall be used by the same expander port (i.e., all the expander phys shall have the same routing attribute (e.g., subtractive or table) (see 4.6.2)).

Figure 72 shows the Mini SAS 4x receptacle connector.
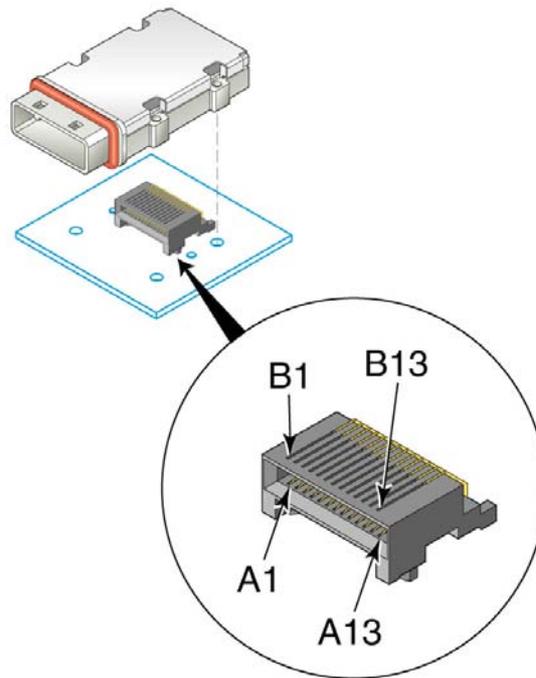
Figure 72 — Mini SAS 4x receptacle connector

Table 32 (see 5.2.3.3.2.3) defines the pin assignments for the Mini SAS 4x receptacle connector.

Mini SAS 4x receptacle connectors shall include keys to prevent attachment to Mini SAS 4x cable plug connectors (see 5.2.3.3.2.1) without matching key slots.

Table 31 defines the icons that shall be placed on or near Mini SAS 4x receptacle connectors and the key positions (see SFF-8088) that shall be used by Mini SAS 4x receptacle connectors.

Table 31 — Mini SAS 4x receptacle connector icons and key positions

Use	Icons	Key position	Reference
End device	Diamond and circle	4	Figure 73
Enclosure out port (see 4.6.2)	Diamond	2	Figure 74
Enclosure in port (see 4.6.2)	Circle	6	Figure 75

Figure 73 shows the key on a Mini SAS 4x receptacle connector used by end devices. This connector may be attached to the Mini SAS 4x cable plug shown in figure 70 or the Mini SAS 4x cable plug shown in figure 71 (see 5.2.3.3.2.2).
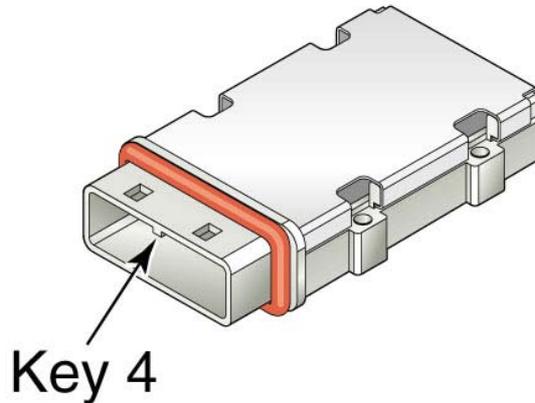
Figure 73 — Mini SAS 4x receptacle connector - end device

Figure 74 shows the key on a Mini SAS 4x receptacle connector used by an enclosure out port. This connector may be attached to the Mini SAS 4x cable plug shown in figure 70 (see 5.2.3.3.2.2).
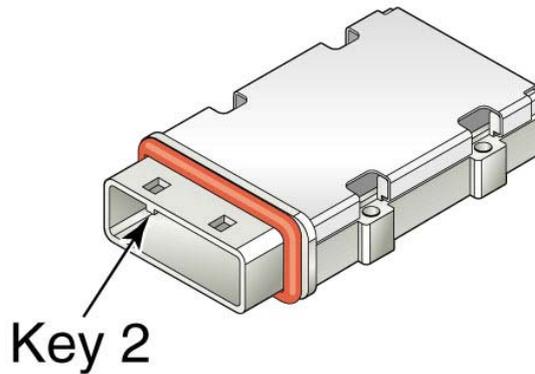
Figure 74 — Mini SAS 4x receptacle connector - enclosure out port

Figure 75 shows the key on a Mini SAS 4x receptacle connector used by an enclosure in port. This connector may be attached to the Mini SAS 4x cable plug shown in figure 71 (see 5.2.3.3.2.2).
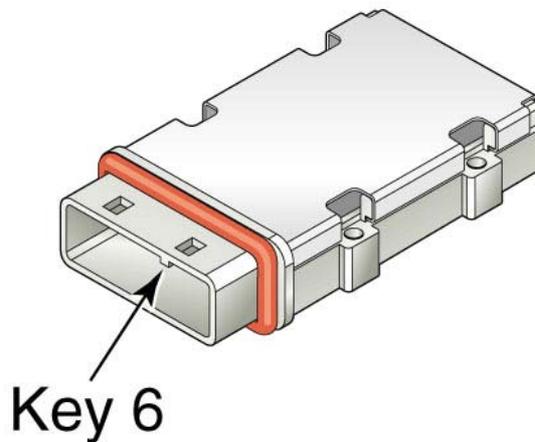
Figure 75 — Mini SAS 4x receptacle connector - enclosure in port

5.2.3.3.2.3 Mini SAS 4x connector pin assignments

Table 32 defines the pin assignments for Mini SAS 4x cable plug connectors (see 5.2.3.3.2.1) and Mini SAS 4x receptacle connectors (see 5.2.3.3.2.2) for applications using one, two, three, or four of the physical links.

Table 32 — Mini SAS 4x connector pin assignments and physical link usage

Signal	Pin usage based on number of physical links supported by the cable assembly [a]				Mating length
	One	**Two**	**Three**	**Four**	
Rx 0+	A2	A2	A2	A2	Third
Rx 0-	A3	A3	A3	A3	
Rx 1+	N/C	A5	A5	A5	
Rx 1-	N/C	A6	A6	A6	
Rx 2+	N/C	N/C	A8	A8	
Rx 2-	N/C	N/C	A9	A9	
Rx 3+	N/C	N/C	N/C	A11	
Rx 3-	N/C	N/C	N/C	A12	
Tx 0+	B2	B2	B2	B2	
Tx 0-	B3	B3	B3	B3	
Tx 1+	N/C	B5	B5	B5	
Tx 1-	N/C	B6	B6	B6	
Tx 2+	N/C	N/C	B8	B8	
Tx 2-	N/C	N/C	B9	B9	
Tx 3+	N/C	N/C	N/C	B11	
Tx 3-	N/C	N/C	N/C	B12	
SIGNAL GROUND	A1, A4, A7, A10, A13 B1, B4, B7, B10, B13				First
CHASSIS GROUND	Housing				N/A
[a] N/C = not connected					

SIGNAL GROUND shall not be connected to CHASSIS GROUND in the connector when used in a cable assembly.

5.2.4 Cable assemblies

5.2.4.1 SAS internal cable assemblies

5.2.4.1.1 SAS Drive cable assemblies

There are two types of SAS Drive cable assemblies:

   a)   Single-port SAS Drive cable assembly; and
   b)   Dual-port SAS Drive cable assembly.

Both SAS Drive cable assemblies shall use:

   a)   a SAS Drive cable receptacle connector (see 5.2.3.2.1.2) on the SAS target device end; and
   b)   a SATA signal cable receptacle connector (see ATA/ATAPI-7 V3) on the SAS initiator device or expander device end.

The power and READY LED signal connection is vendor specific.

A SAS initiator device shall use a SATA host plug connector (see ATA/ATAPI-7 V3) for connection to a SAS Drive cable assembly. The signal assignment for the SAS initiator device or expander device with this connector shall be the same as that defined for a SATA host (see ATA/ATAPI-7 V3).

Figure 76 shows the Single-port SAS Drive cable assembly.
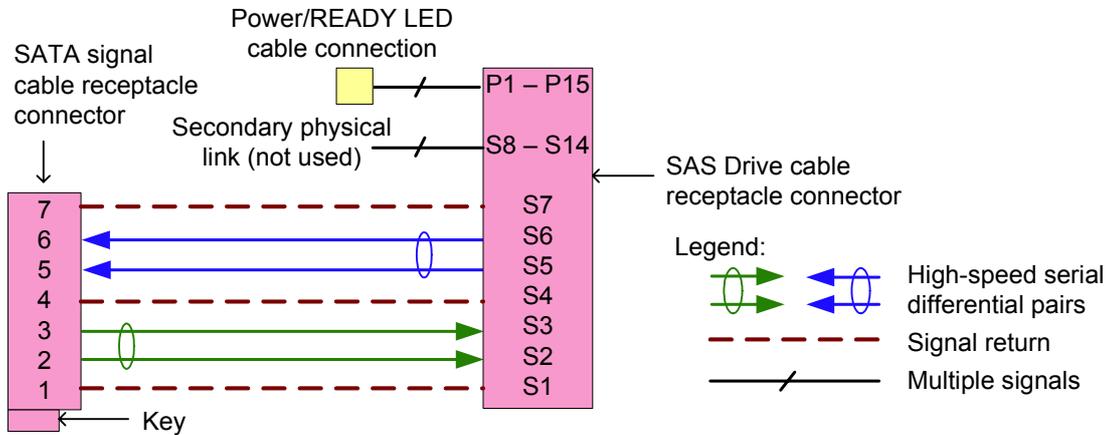
Figure 76 — Single-port SAS Drive cable assembly

Figure 77 shows the Dual-port SAS Drive cable assembly.
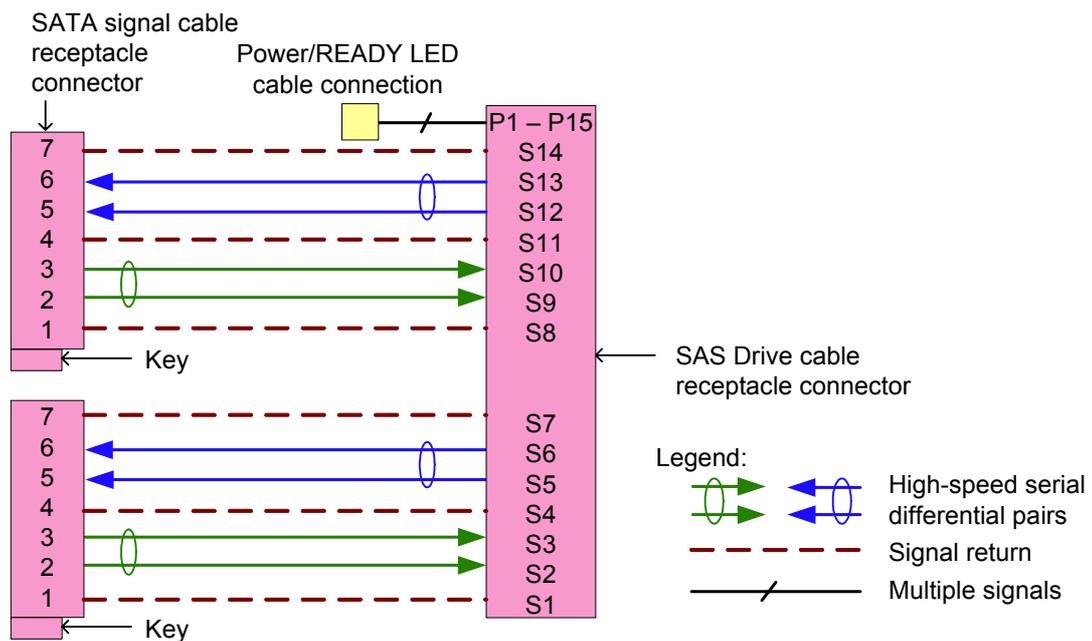
Figure 77 — Dual-port SAS Drive cable assembly

5.2.4.1.2 SAS internal symmetric cable assemblies

5.2.4.1.2.1 SAS internal symmetric cable assemblies overview

There are several types of SAS internal symmetric cable assemblies:

   a) SAS 4i cable receptacle connectors (see 5.2.3.2.2.1) on each end (see 5.2.4.1.2.2);
   b) Mini SAS 4i cable plug connectors (see 5.2.3.2.3.2) on each end (see 5.2.4.1.2.3); and
   c) a SAS 4i cable receptacle connector on one end and a Mini SAS 4i cable plug connector on the other end (see 5.2.4.1.2.4).

In a SAS internal symmetric cable assembly, the Tx signals on one end shall be connected to Rx signals on the other end (e.g., a Tx + of one connector shall connect to an Rx + of the other connector. SAS internal symmetric

cable assemblies should be labeled to indicate how many physical links are included (e.g., 1X, 2X, 3X, and 4X on each connector's housing)

With the SAS 4i cable plug connector, the physical link number of the signal depends on the application (e.g., controller-to-controller applications and controller-to-backplane applications differ).

Although the SAS 4i cable receptacle connector and Mini SAS 4i cable plug connector always support four physical links:

a) a SAS internal symmetric cable assembly using SAS 4i cable receptacle connectors may support one, two, three, or four physical links when used for controller-to-backplane applications;

b) a SAS internal symmetric cable assembly using SAS 4i cable receptacle connectors shall support four physical links when used for controller-to-controller applications; and

c) a SAS internal symmetric cable assembly using Mini SAS 4i cable plug connectors may support one, two, three, or four physical links for either controller-to-backplane or controller-to-controller applications.

5.2.4.1.2.2 SAS internal symmetric cable assembly - SAS 4i

Figure 78 shows the SAS internal symmetric cable assembly with SAS 4i cable receptacle connectors at each end.
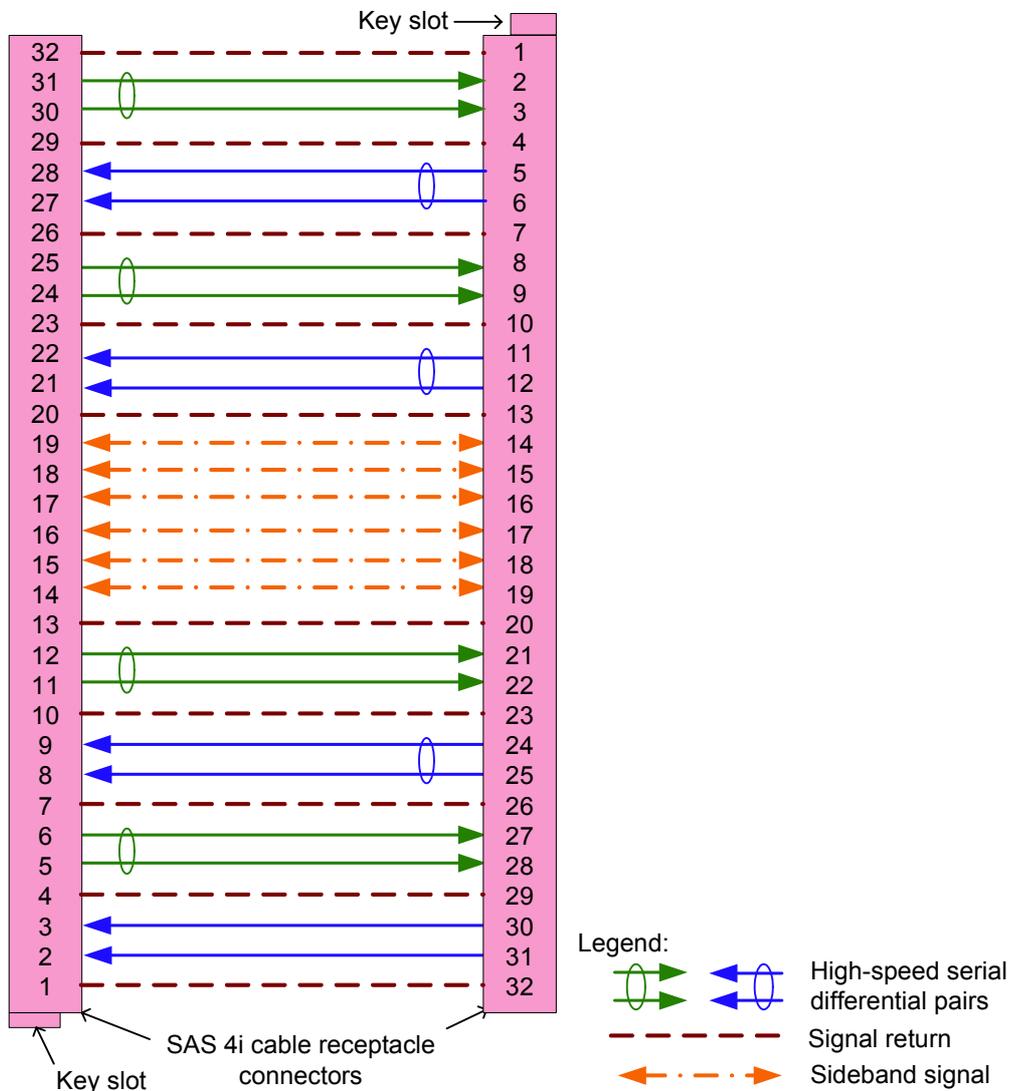
Figure 78 — SAS internal symmetric cable assembly - SAS 4i

In addition to the signal return connections shown in figure 78, the cable assembly may connect one or more of the signal returns together.

For controller-to-backplane applications, the cable assembly may support one to four physical links. SIDEBAND signals on the controller are attached to the corresponding SIDEBAND signals on the backplane (e.g., SIDEBAND0 of the controller is attached to SIDEBAND0 of the backplane).

For controller-to-controller applications, the cable assembly shall support all four physical links and the controllers should use all four physical links, because one controller's physical links 0 and 1 are attached the other controller's physical links 3 and 2, respectively. If both controllers use one or two physical links starting with physical links 0, communication is not possible. If both controllers use physical links 0, 1, and 2, then only communication over physical links 1 and 2 is possible. SIDEBAND signals on one controller are not attached to their corresponding SIDEBAND signals on the other controller (e.g., SIDEBAND0 of one controller is attached to SIDEBAND5 of the other controller).

5.2.4.1.2.3 SAS internal symmetric cable assembly - Mini SAS 4i

Figure 79 shows the SAS internal cable assembly with Mini SAS 4i cable plug connectors at each end.
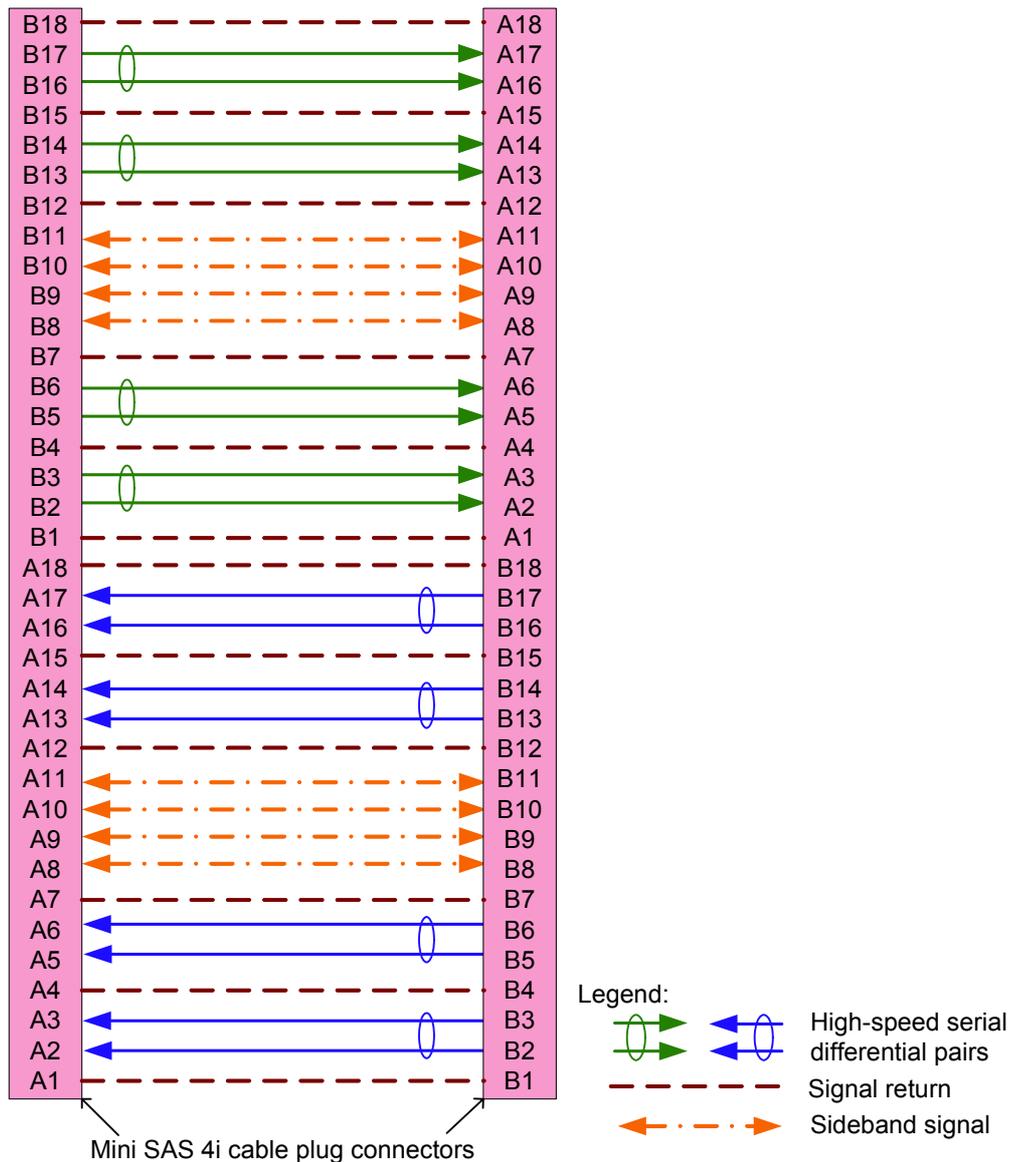
Figure 79 — SAS internal symmetric cable assembly - Mini SAS 4i

In addition to the signal return connections shown in figure 79, the cable assembly may connect one or more of the signal returns together.

The cable assembly may support one to four physical links.

For controller-to-backplane applications, SIDEBAND signals on the controller are attached to the corresponding SIDEBAND signals on the backplane (e.g., SIDEBAND0 of the controller is attached to SIDEBAND0 of the backplane).

For controller-to-controller applications, SIDEBAND signals on one controller are not attached to their corresponding SIDEBAND signals on the other controller (e.g., SIDEBAND0 of one controller is attached to SIDEBAND6 of the other controller).

5.2.4.1.2.4 SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i

Figure 80 shows the SAS internal symmetric cable assembly with a SAS 4i cable receptacle connector at one end and a Mini SAS 4i cable plug connector at the other end.
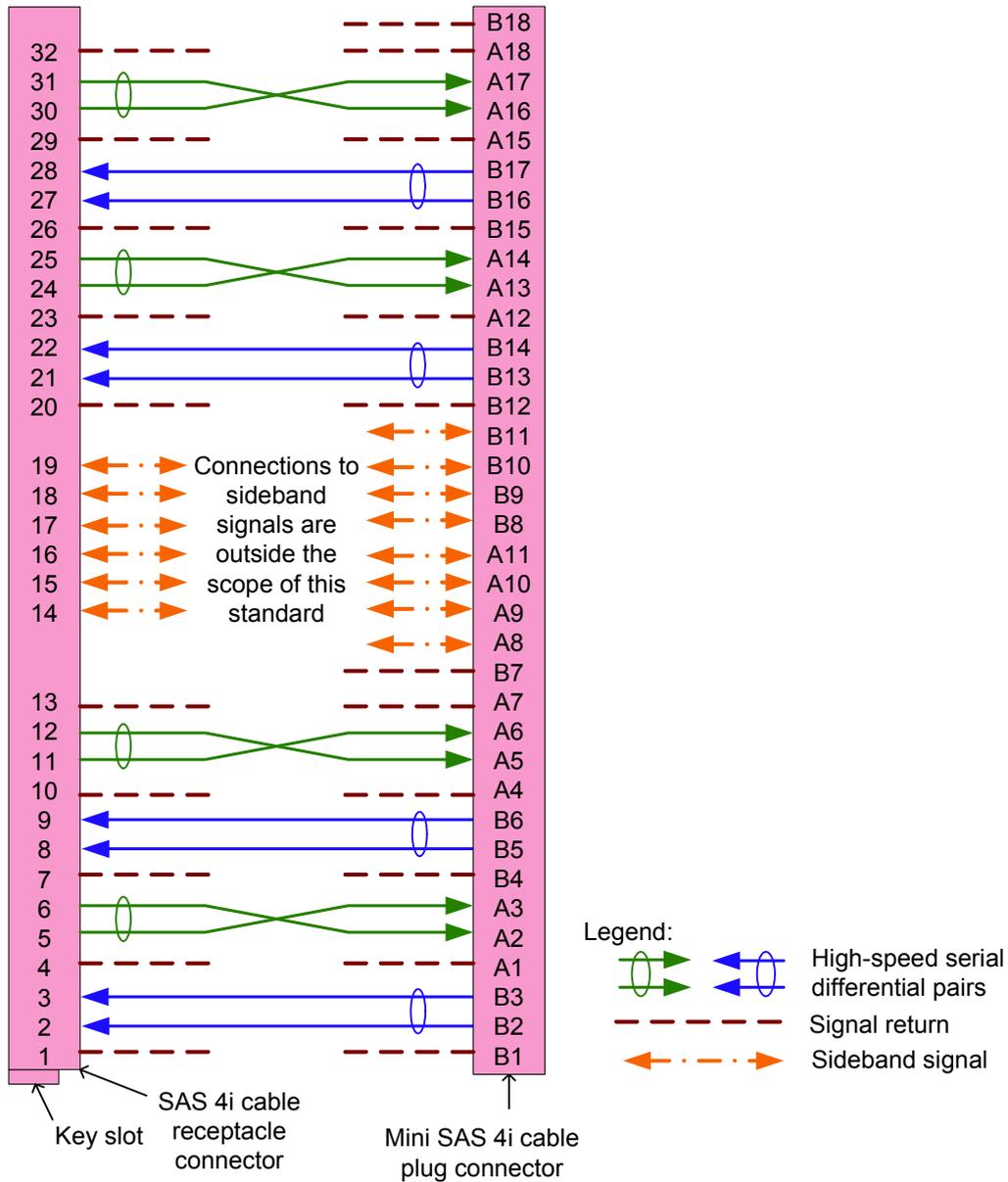
Figure 80 — SAS internal symmetric cable assembly - SAS 4i to Mini SAS 4i

NOTE 7 - The cable assembly needs different SIDEBAND signal routing based on whether the controller or backplane is using the SAS 4i connector.

The cable assembly shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

For controller-to-backplane applications with the SAS 4i cable receptacle connector on the controller end, the cable assembly may support one to four physical links.

For controller-to-controller applications, the cable assembly may support one to four physical links.

For controller-to-backplane applications with the Mini SAS 4i cable receptacle connector on the controller end, the cable assembly shall support all four physical links and the controller should use all four physical links, because the controller's physical links 0, 1, 2, and 3 are attached to the backplane's physical links 3, 2, 1, and 0, respectively. If both controllers use one or two physical links starting with physical links 0, communication is not possible. If both controllers use physical links 0, 1, and 2, then only communication over physical links 1 and 2 is possible.

5.2.4.1.3 SAS internal fanout cable assemblies

5.2.4.1.3.1 SAS internal fanout cable assemblies overview

There are several types of SAS internal fanout cable assemblies:

   a) SAS internal controller-based fanout cable assemblies (see 5.2.4.1.3.2) with:
     A) a SAS 4i cable receptacle connector on one end (i.e., the controller end) and four SAS Drive cable receptacle connectors on the other end (i.e., the backplane end); and
     B) a Mini SAS 4i cable plug connector on one end (i.e., the controller end) and four SAS Drive cable receptacle connectors on the other end (i.e., the backplane end);

     and

   b) SAS internal backplane-based fanout cable assemblies (see 5.2.4.1.3.3):
     A) four SATA signal cable receptacle connectors on one end (i.e., the controller end) and a SAS 4i cable receptacle connector on the other end (i.e., the backplane end); and
     B) four SATA signal cable receptacle connectors on one end (i.e., the controller end) and a Mini SAS 4i cable plug connector on the other end (i.e., the backplane end).

In a SAS internal fanout symmetric cable assembly, the Tx signals on one end shall be connected to Rx signals on the other end (e.g., a Tx + of one connector shall connect to an Rx + of the other connector.

5.2.4.1.3.2 SAS internal controller-based fanout cable assemblies

Figure 81 shows the SAS internal controller-based fanout cable assembly with a SAS 4i cable receptacle connector at the controller end.
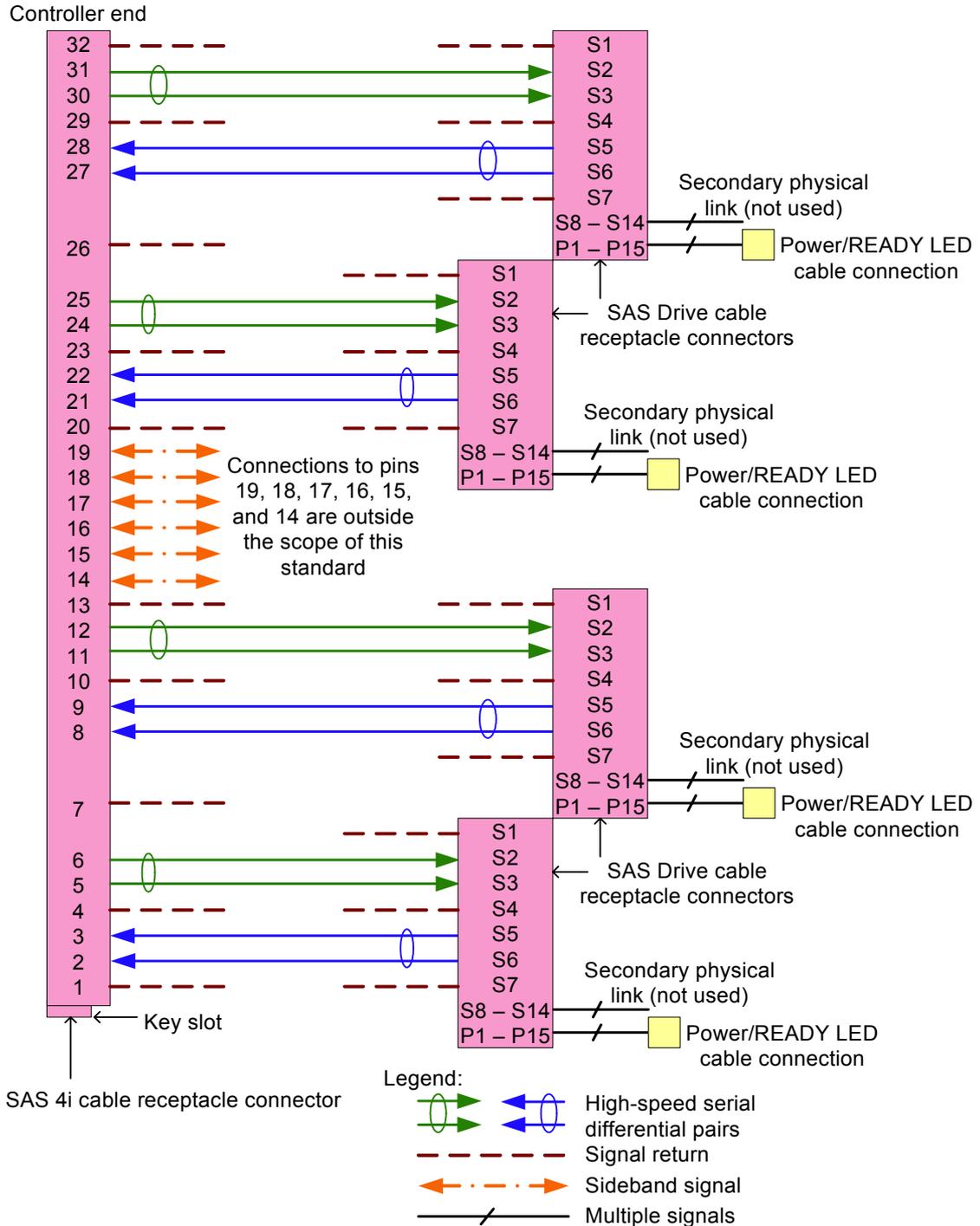
Figure 81 — SAS internal controller-based fanout cable assembly - SAS 4i

The cable assembly shown in figure 81 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

Figure 82 shows the SAS internal controller-based fanout cable assembly with a Mini SAS 4i cable plug connector at the controller end.
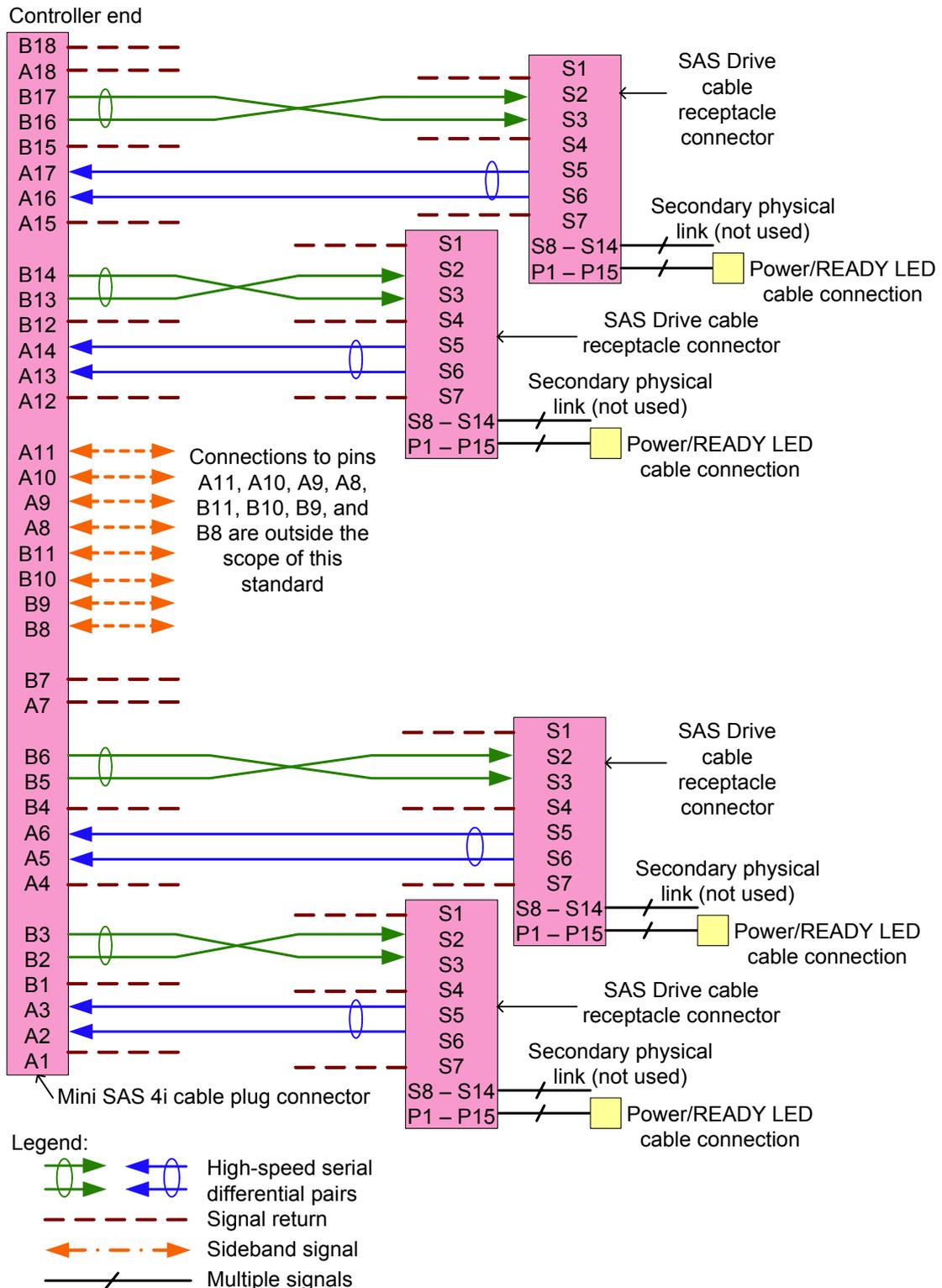
Figure 82 — SAS internal controller-based fanout cable assembly - Mini SAS 4i

The cable assembly shown in figure 82 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.4.1.3.3 SAS internal backplane-based fanout cable assemblies

Figure 83 shows the SAS internal backplane-based fanout cable assembly with the SAS 4i cable receptacle connector.
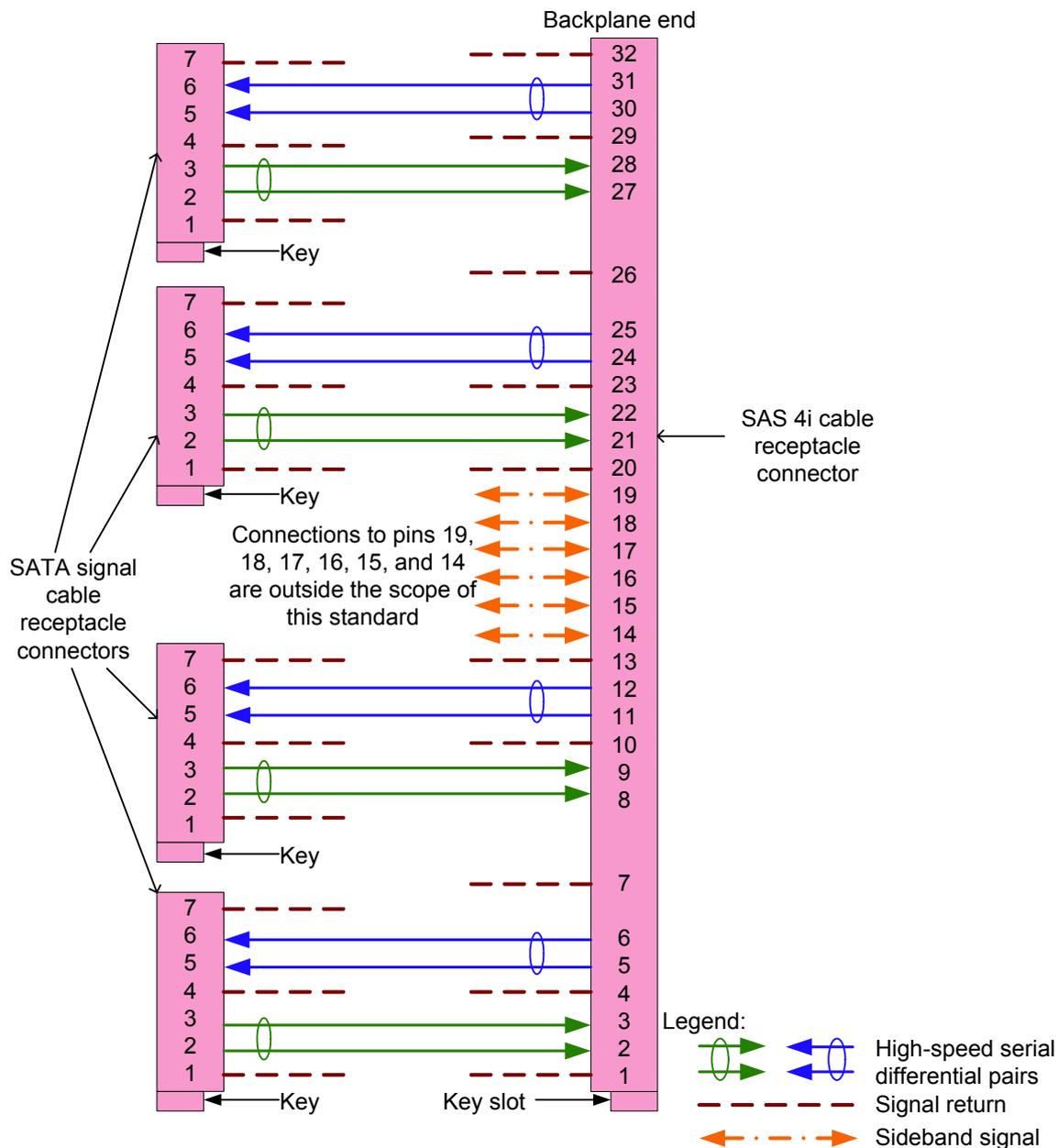
Figure 83 — SAS internal backplane-based fanout cable assembly - SAS 4i

The cable assembly shown in figure 83 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

Figure 84 shows the SAS internal backplane-based fanout cable assembly with the Mini SAS 4i cable receptacle connector.

Figure 84 — SAS internal backplane-based fanout cable assembly - Mini SAS 4i

The cable assembly shown in figure 84 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.4.2 SAS external cable assemblies

5.2.4.2.1 SAS external cable assemblies overview

There are several types of SAS external cable assemblies:

   a)   SAS 4x cable plug connector (see 5.2.3.3.1.1) at each end (see 5.2.4.2.2);
   b)   Mini SAS 4x cable plug connector (see 5.2.3.3.2.1) at each end (see 5.2.4.2.3); and
   c)   SAS 4x cable plug connector at one end and Mini SAS 4x cable plug connector at the other end (see 5.2.4.2.4).

SAS external cable assemblies do not include power or the READY LED signal.

Although the connector always supports four physical links, a SAS external cable assembly may support one, two, three, or four physical links. SAS external cable assemblies should be labeled to indicate how many physical links are included (e.g., 1X, 2X, 3X, and 4X on each connector's housing)

The Tx signals on one end shall be connected to the corresponding Rx signals of the other end (e.g., Tx 0+ of one connector shall be connected to Rx 0+ of the other connector).

Signal returns shall not be connected to CHASSIS GROUND in the cable assembly.

In addition to the general SAS icon, additional icons are defined for external connectors to guide users into making compatible attachments (i.e., not attaching expander device table routing phys to expander device table routing phys, which is not allowed by this standard). Connectors that have one or more matching icons are intended to be attached. Connectors that do not have a matching icon should not be attached together.

One end of the SAS external cable assembly shall support being attached to an end device or an enclosure out port. The other end of the SAS external cable assembly shall support being attached to an end device or an enclosure in port. If a SAS 4x cable plug connector is used, it should include icons as defined in 5.2.3.3.1.1. If a Mini SAS 4x cable plug connector is used, it shall include icons and key slots as defined in 5.2.3.3.2.1.

5.2.4.2.2 SAS external cable assembly - SAS 4x

Figure 85 shows the SAS external cable assembly with SAS 4x cable plug connectors at each end.
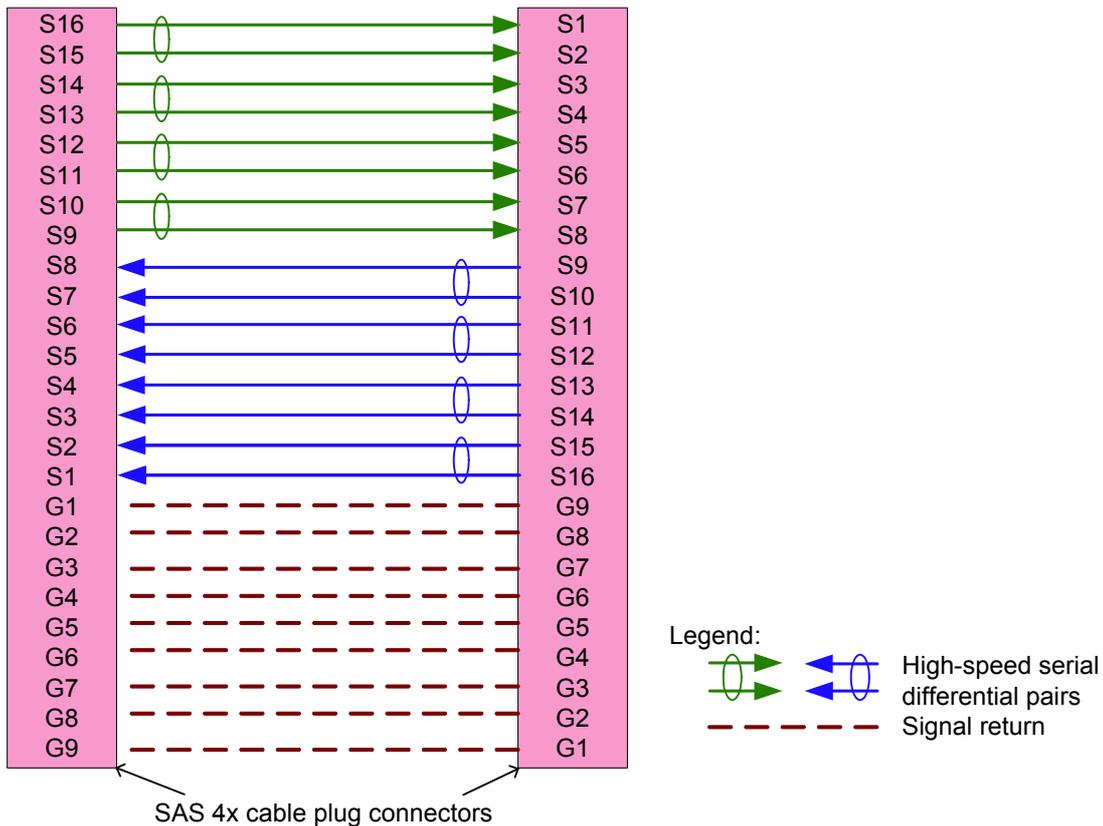
Figure 85 — SAS external cable assembly - SAS 4x

In addition to the signal return connections shown in figure 85, the cable assembly may connect one or more of the signal returns together.

5.2.4.2.3 SAS external cable assembly - Mini SAS 4x

Figure 86 shows the SAS external cable assembly with Mini SAS 4x cable plug connectors at each end.

Mini SAS 4x cable plug connectors

Legend:

High-speed serial
differential pairs
Signal return
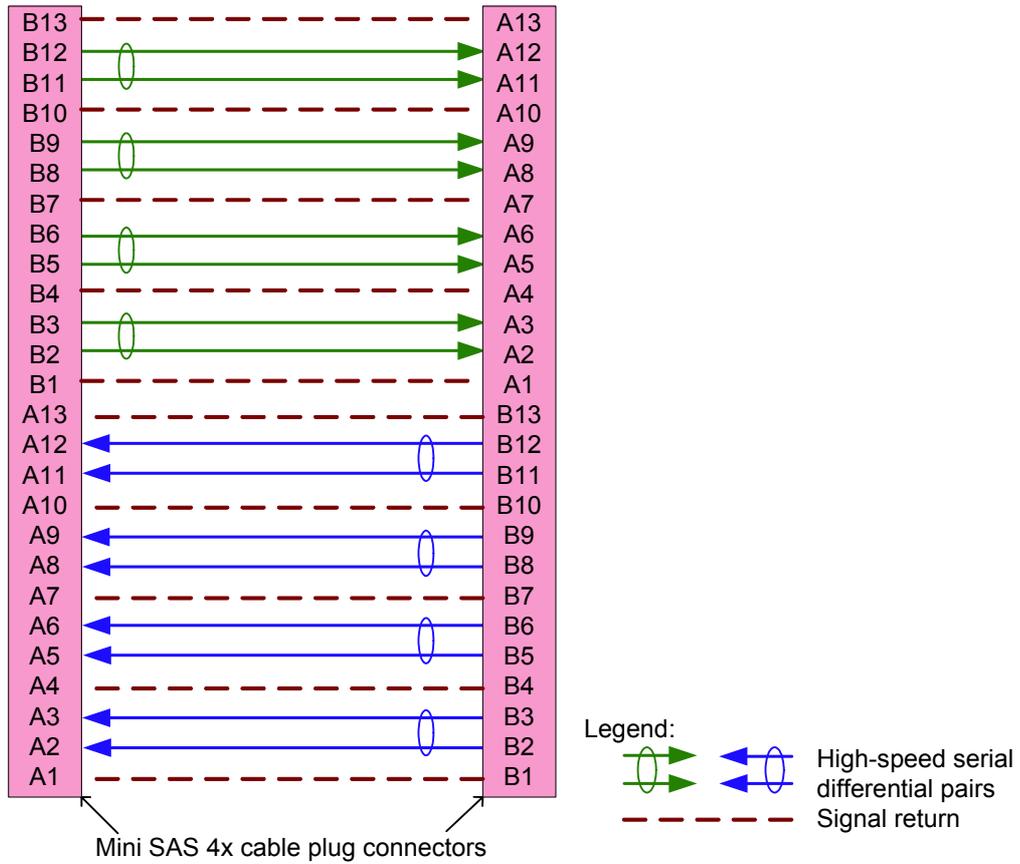
Figure 86 — SAS external cable assembly - Mini SAS 4x

In addition to the signal return connections shown in figure 86, the cable assembly may connect one or more of the signal returns together.

Figure 87 shows the icons and key slots in the SAS external cable assembly with Mini SAS 4x cable plug connectors at each end.
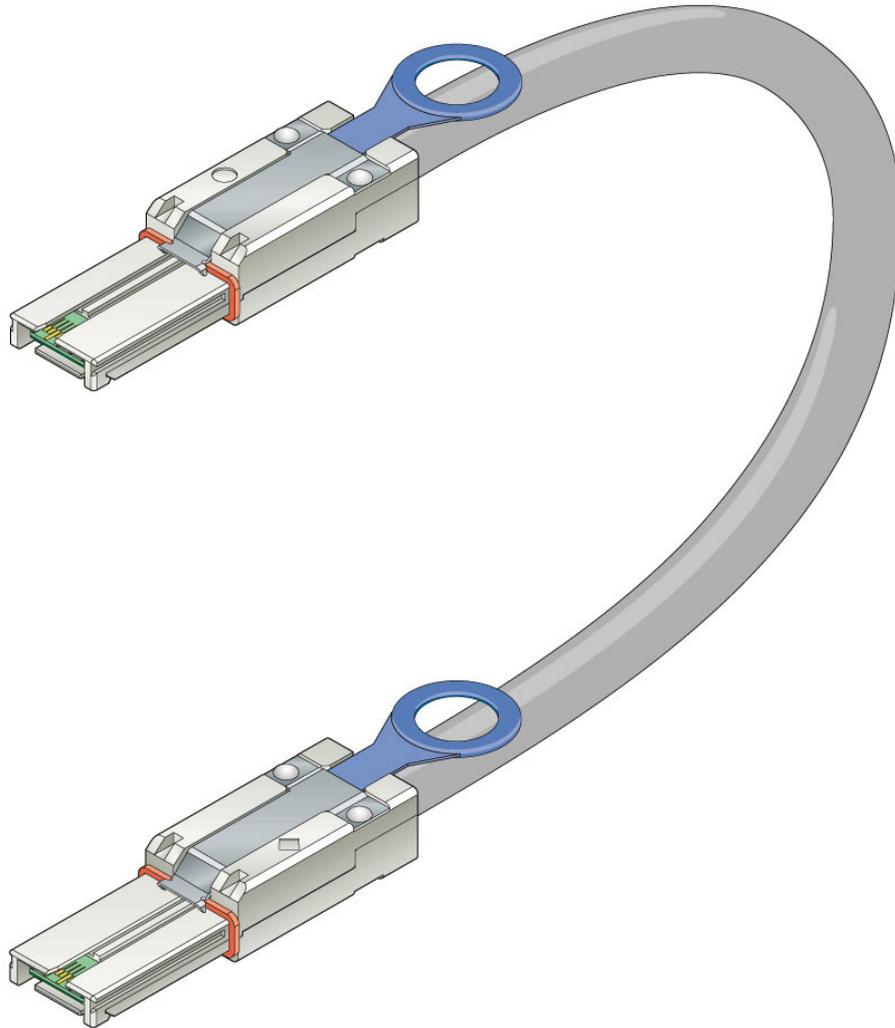
Figure 87 — SAS external cable assembly with Mini SAS 4x cable plug connectors

Although the topology is supported by this standard, a SAS external cable assembly with Mini SAS 4x connectors on each end that attaches an enclosure in port to another enclosure in port is not defined by this standard.

5.2.4.2.4 SAS external cable assembly - SAS 4x to Mini SAS 4x

Figure 88 shows the SAS external cable assembly with a SAS 4x cable plug connector at one end and a Mini SAS 4x cable plug connector at the other end.
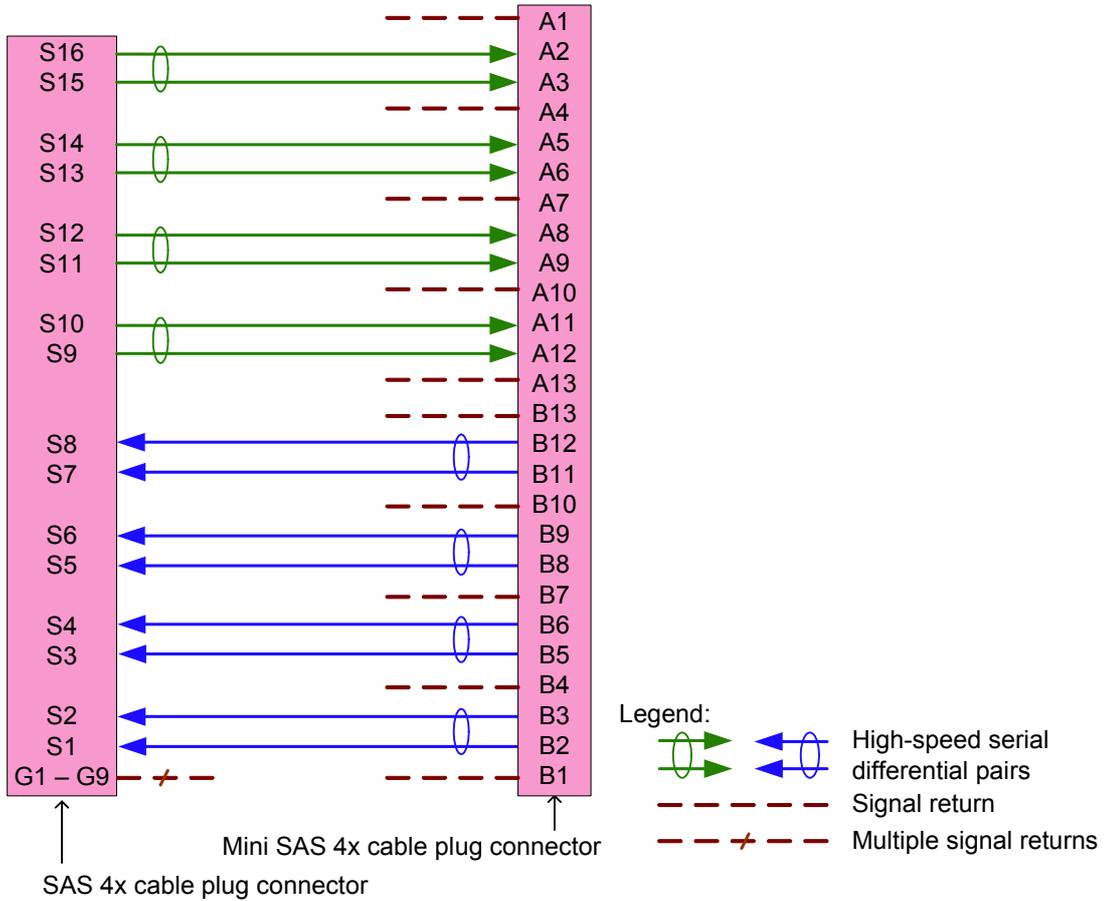
Figure 88 — SAS external cable assembly - SAS 4x to Mini SAS 4x

The cable assembly shown in figure 88 shall connect each signal return on one end to at least one signal return on the other end. The cable assembly may connect one or more of the signal returns together.

5.2.5 Backplanes

SAS backplane designs should follow the recommendations in SFF-8460.

5.2.6 Cable assembly and backplane specifications

Table 33 defines requirements for internal cable assemblies using SAS Drive connectors and backplanes.

Table 33 — Requirements for internal cable assemblies using SAS Drive connectors and backplanes

Requirement [a, b]	Units	1,5 Gbit/s	3,0 Gbit/s
Bulk cable or backplane: [c, d]			
Differential impedance	ohm	100 ± 10	
Maximum differential impedance imbalance [e]	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors: [d, f]			
Differential impedance	ohm	100 ± 15	
Maximum differential impedance imbalance [e]	ohm	5	

[a] All measurements are made through mated connector pairs.
[b] The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included.
[c] The impedance measurement identifies the impedance mismatches present in the bulk cable or backplane when terminated in its characteristic impedance. This measurement excludes mated connectors at both ends of the bulk cable or backplane, when present, but includes any intermediate connectors or splices.
[d] Where the bulk cable or backplane has an electrical length of > 4 ns the procedure detailed in IEC/TS 62452, or an equivalent procedure, shall be used to determine the impedance.
[e] The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.
[f] The mated connectors measurement applies only to the mated connector pair at each end, as applicable.

Table 34 defines requirements for SAS internal cable assemblies using SAS 4i cable receptacle connectors or Mini SAS 4i cable plug connectors.

Table 34 — Requirements for SAS internal cable assemblies using SAS 4i or Mini SAS 4i

Requirement [a, b]	Units	1,5 Gbit/s	3,0 Gbit/s
Bulk cable: [c, d]			
Differential impedance	ohm	100 ± 10	
Maximum differential impedance imbalance [e]	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors: [d, f]			
Differential impedance	ohm	100 ± 15	
Maximum differential impedance imbalance [e]	ohm	5	
Cable assembly: [g]			
Maximum insertion loss [h]	dB	6	
Maximum intra-pair skew [i, j]	ps	10	

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

[b] The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included.

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

[d] Where the bulk cable has an electrical length of > 4 ns, the procedure detailed in IEC/TS 62452, or an equivalent procedure, shall be used to determine the impedance.

[e] The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.

[f] The mated connectors measurement applies only to the mated connector pair at each end, as applicable.

[g] The internal cable assembly is part of a TxRx connection that complies with the requirements for intra-enclosure compliance points defined in 5.3.

[h] The range for this frequency domain measurement is 10 MHz to 4 500 MHz.

[i] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.

[j] The procedure detailed in IEC/TS 62452, or an equivalent procedure, shall be used to determine the intra-pair skew.

Table 35 defines additional requirements for cable assemblies using SAS 4i cable receptacle connectors.

Table 35 — Additional requirements for cable assemblies using SAS 4i

Requirement [a, b, c, d]	Units	1,5 Gbit/s	3,0 Gbit/s
Maximum near-end crosstalk on any of the following (adjacent) signal pairs: RX0/TX0, TX0/RX1, RX1/TX1, RX2/TX2, TX2/RX3, and RX3/TX3	dB	-33	
Maximum near-end crosstalk on any of the following signal pairs: RX0/RX1, RX0/TX1, TX0/TX1, RX2/RX3, RX2/TX3, and TX2/TX3	dB	-45	
Maximum near-end crosstalk on all other signal pairs	dB	-50	
[a] All measurements are made through mated connector pairs. [b] The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included. [c] The range for this frequency domain measurement is 10 MHz to 4 500 MHz. [d] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

Table 36 defines additional requirements for cable assemblies using Mini SAS 4i cable plug connectors.

Table 36 — Additional requirements for cable assemblies using Mini SAS 4i

Requirement [a, b, c, d]	Units	1,5 Gbit/s	3,0 Gbit/s
Maximum near-end crosstalk for each receive pair	dB	-26	
[a] All measurements are made through mated connector pairs. [b] Determine all valid aggressor/victim near-end crosstalk transfer modes. Over the complete frequency range of this measurement, determine the sum of the crosstalk transfer ratios, measured in the frequency domain, of all crosstalk transfer modes. To remove unwanted bias due to test fixture noise, magnitudes less than -50 dB (e.g., -60 dB) at all frequencies may be ignored. The following equation details the summation process of the four valid near-end crosstalk sources. All NEXT values expressed in dB format in a passive transfer network shall have negative dB magnitude. $$\mathrm{TotalNEXT}(f) = 10 \times \log \sum_{1}^{4} 10^{\langle \mathrm{NEXT}(f)/10 \rangle}$$ [c] The range for this frequency domain measurement is 10 MHz to 4 500 MHz. [d] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.			

Table 37 defines requirements for external cable assemblies.

Table 37 — Requirements for external cable assemblies

Requirement [a, b]	Units	1,5 Gbit/s	3,0 Gbit/s
Bulk cable: [c, d]			
Differential impedance	ohm	100 ± 5	
Maximum differential impedance imbalance [e]	ohm	5	
Common-mode impedance	ohm	32,5 ± 7,5	
Mated connectors:			
Differential impedance [f, d]	ohm	100 ± 10	
Cable assembly: [g]			
Maximum insertion loss		See 5.3.3	
Maximum rise time [h, i]	ps	150	
Maximum ISI [j]	ps	60	
Maximum intra-pair skew [h, k]	ps	50	

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

[b] The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included.

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

[d] Where the bulk cable has an electrical length of > 4 ns, the procedure detailed in IEC/TS 62452, or an equivalent procedure, shall be used to determine the impedance.

[e] The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.

[f] The mated connectors measurement applies only to the mated connector pair at each end, as applicable.

[g] The external cable assembly is part of a TxRx connection that complies with the requirements for inter-enclosure compliance points as defined in 5.3.

[h] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.

[i] Connect the TDR step impulse response generators to the near end of the cable assembly and measure the output rise time at the far end. The input rise time shall be no higher than 35 ps.

[j] Measured DJ at the far end of the cable assembly under test using a lone bit pattern at 3,0 Gbit/s.

[k] The procedure detailed in IEC/TS 62452, or an equivalent procedure, shall be used to determine the intra-pair skew.

Table 38 defines additional requirements for external cable assemblies using SAS 4x cable plug connectors.

Table 38 — Additional requirements for external cable assemblies using SAS 4x

Requirement [a, b, c]	Units	1,5 Gbit/s	3,0 Gbit/s
Frequency domain measurement range 10 MHz to 2 250 MHz:			
Maximum near-end crosstalk from any single aggressor pair offset by one position (i.e., adjacent)(e.g., RX1/RX2)	dB	-30	
Maximum near-end crosstalk from any single aggressor pair offset by two positions (e.g., RX1/RX3)	dB	-36	
Maximum near-end crosstalk from any single aggressor pair offset by more than two positions (e.g., RX1/RX4)	dB	-40	
Frequency domain measurement range 2 250 Mhz to 4 500 MHz:			
Maximum near-end crosstalk from any single aggressor pair offset by one position (i.e., adjacent)(e.g., RX1/RX2)	dB	-24	
Maximum near-end crosstalk from any single aggressor pair offset by two positions (e.g., RX1/RX3)	dB	-30	
Maximum near-end crosstalk from any single aggressor pair offset by more than two positions (e.g., RX1/RX4	dB	-40	

[a] All measurements are made through mated connector pairs.
[b] The equivalent maximum TDR rise time from 20 % to 80 % shall be 70 ps. Filtering may be used to obtain the equivalent rise time. The filter consists of the two-way launch/return path of the test fixture, the two-way launch/return path of the test cable, and the software or hardware filtering of the TDR scope. The equivalent rise time is the rise time of the TDR scope output after application of all filter components. When configuring software or hardware filters of the TDR scope to obtain the equivalent rise time, filtering effects of test cables and test fixtures shall be included.
[c] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.

Table 39 defines additional requirements for external cable assemblies using Mini SAS 4x cable plug connectors.

Table 39 — Additional requirements for external cable assemblies using Mini SAS 4x

Requirement [a, b, c, d]	Units	1,5 Gbit/s	3,0 Gbit/s
Maximum near-end crosstalk for each receive pair	dB	-26	

[a] All measurements are made through mated connector pairs.
[b] Determine all valid aggressor/victim near-end crosstalk transfer modes. Over the complete frequency range of this measurement, determine the sum of the crosstalk transfer ratios, measured in the frequency domain, of all crosstalk transfer modes. To remove unwanted bias due to test fixture noise, magnitudes less than -50 dB (e.g., -60 dB) at all frequencies may be ignored. The following equation details the summation process of the four valid near-end crosstalk sources. All NEXT values expressed in dB format in a passive transfer network shall have negative dB magnitude.

$$TotalNEXT(f) = 10 \times \log \sum_{1}^{4} 10^{\langle NEXT(f)/10 \rangle}$$

[c] The range for this frequency domain measurement is 10 MHz to 4 500 MHz.
[d] The far end of the mated cable assembly shall be terminated in its characteristic impedance. Insertion loss variations (i.e., cable length) may change the measurement result.

5.3 Transmitter and receiver device electrical characteristics

5.3.1 Compliance points

Signal behavior at separable connectors requires compliance with signal characteristics defined by this standard only if the connectors are identified as compliance points by the supplier of the parts that contain or comprise the candidate compliance point.

Signal characteristics for compliance points are measured at physical positions called probe points in a test load (see 5.3.2). Measurements at the probe points in a test load approximate measurements at the compliance point in the actual TxRx connection. Some components in the test load may be de-embedded as described in B.4.

Table 40 lists the compliance points.

Table 40 — Compliance points

Compliance point	Type	Description
IT	intra-enclosure (i.e., internal)	The signal from a transmitter device (see 3.1.244), as measured at probe points in a test load attached with an internal connector.
IR	intra-enclosure (i.e., internal)	The signal going to a receiver device (see 3.1.153), as measured at probe points in a test load attached with an internal connector.
CT	inter-enclosure (i.e., cabinet)	The signal from a transmitter device, as measured at probe points in a test load attached with an external connector.
CR	inter-enclosure (i.e., cabinet)	The signal going to a receiver device, as measured at probe points in a test load attached with an external connector.

Figure 89 shows the locations of the CT and CR compliance points using a SAS 4x or Mini SAS 4x cable assembly, and shows how two of the compliance points are tested using test loads (see 5.3.2).
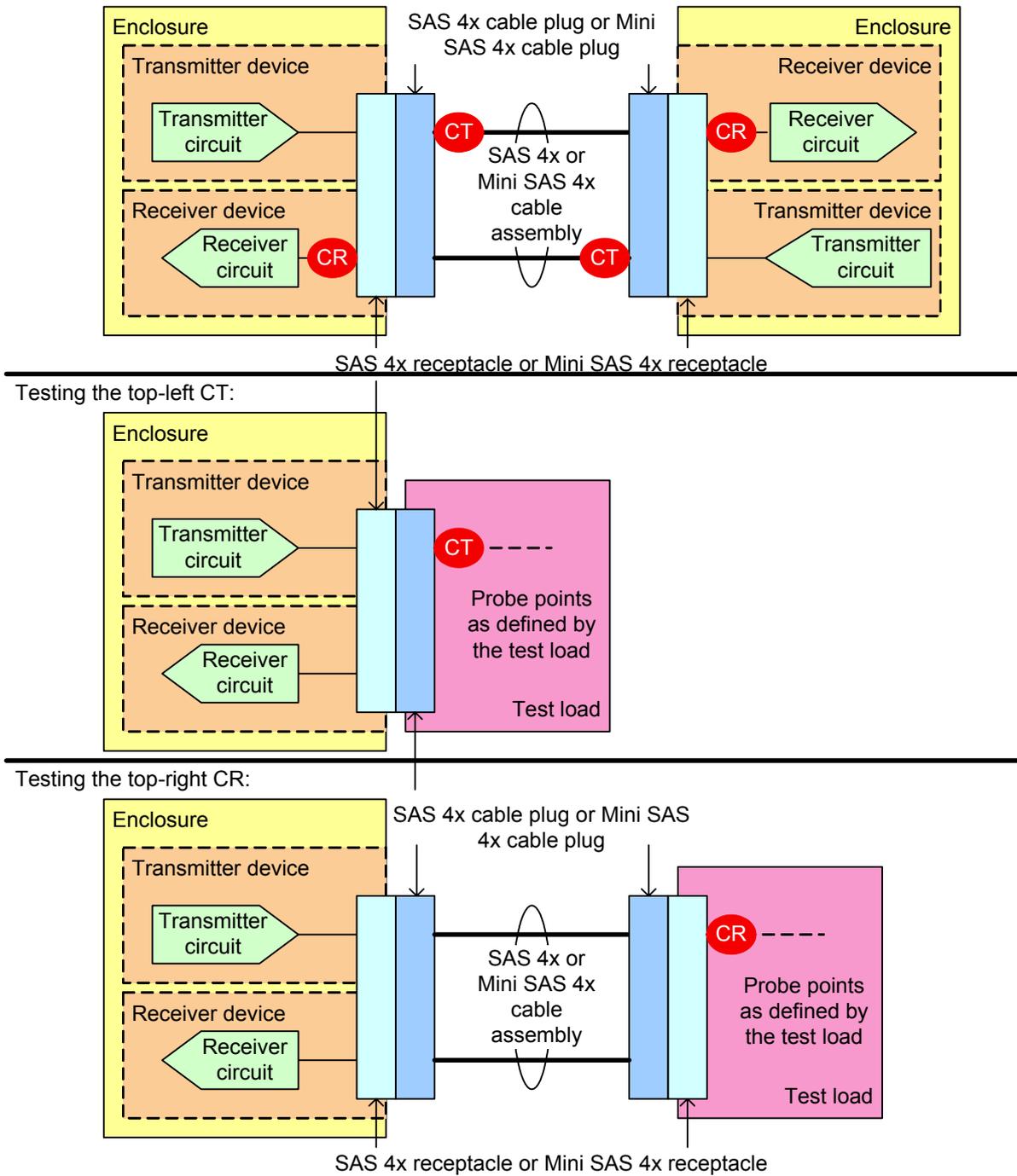
Figure 89 — SAS 4x and Mini SAS 4x cable assembly CT and CR compliance points

Figure 90 shows the locations of the IT and IR compliance points using a backplane with a SAS Drive backplane receptacle (see 5.2.3.2.1.3) that is not using SATA, and shows how the compliance points are tested using test loads (see 5.3.2).
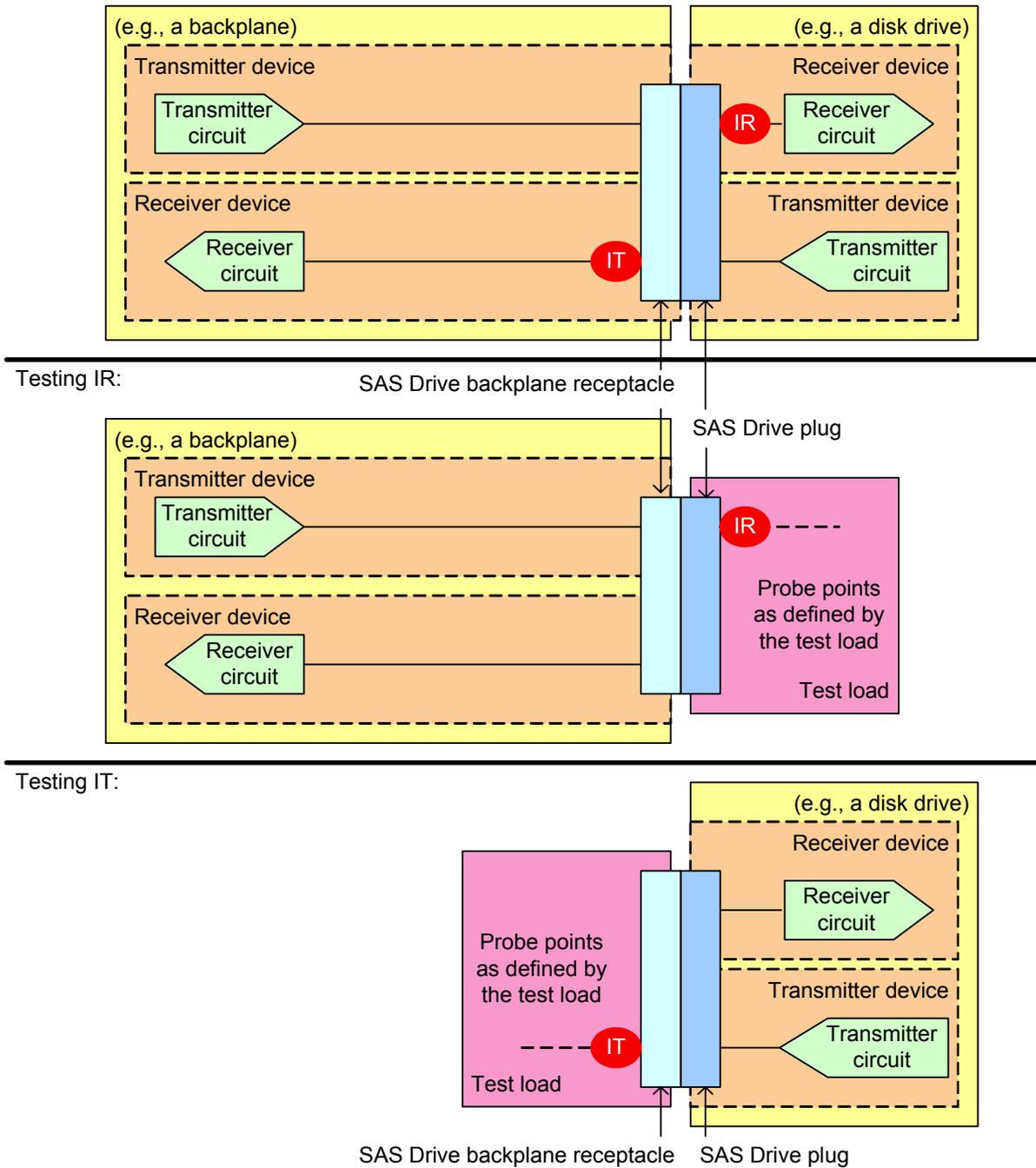
Figure 90 — Backplane IT and IR compliance points

If the backplane supports SATA, there are no IT or IR compliance points. SATA defines the signal characteristics that the SATA phy delivers and that the SAS backplane is required to deliver to the SATA device, as shown in figure 91.
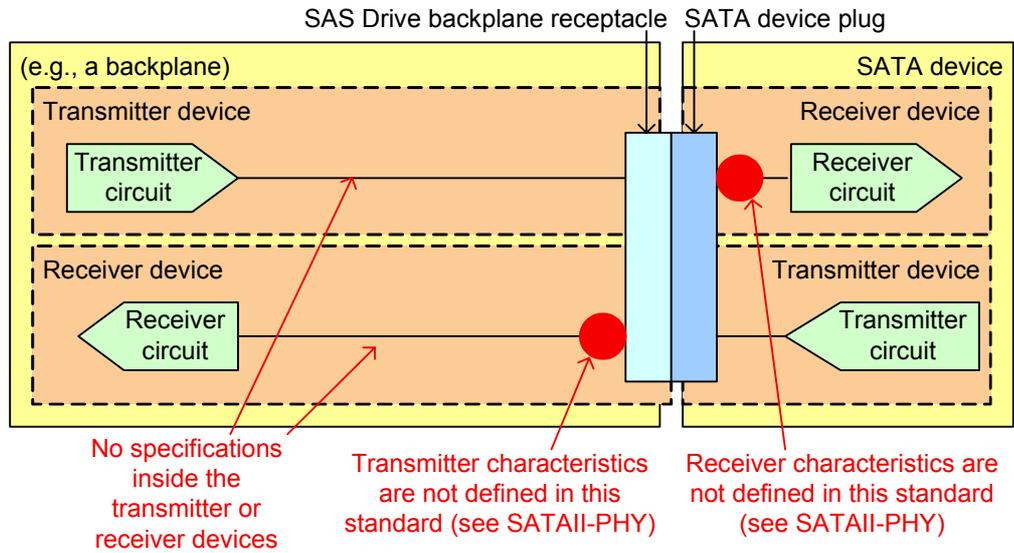
Figure 91 — Backplane compliance points with SATA phy attached

Figure 92 shows the locations of the IT and IR compliance points using a SAS 4i or Mini SAS 4i cable assembly, and shows how two of the compliance points are tested using test loads (see 5.3.2).

Figure 92 — SAS 4i and Mini SAS 4i cable assembly IT and IR compliance points

Figure 93 shows the locations of the IT and IR compliance points using a SAS 4i cable and a backplane, where the backplane is not attached to a SATA device, and shows how two of the compliance points are tested using test loads (see 5.3.2).
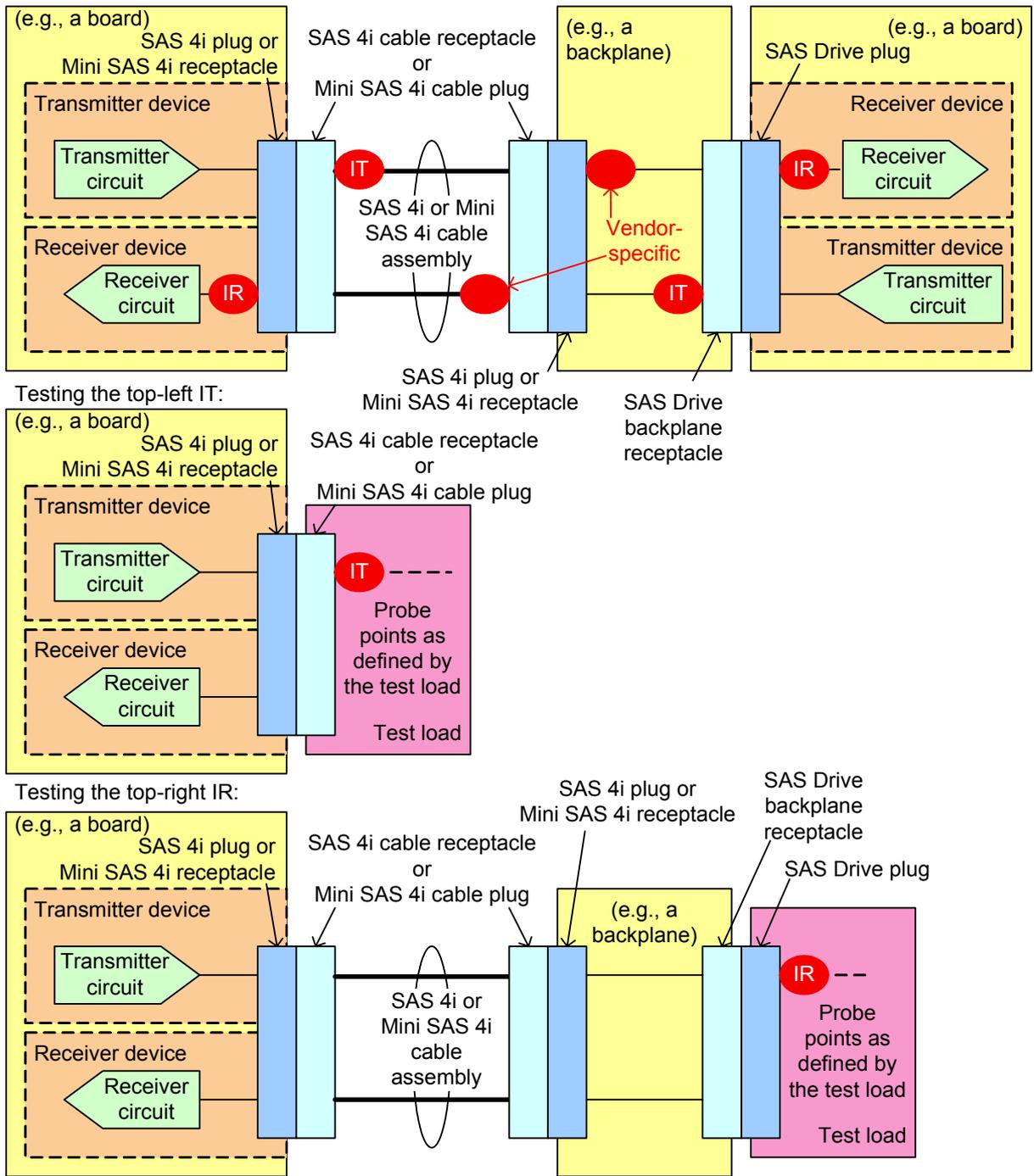
Figure 93 — SAS 4i and Mini SAS 4i cable and backplane IT and IR compliance points

Figure 94 shows the locations of the IT and IR compliance points using a SAS 4i cable and a backplane, where the backplane supports being attached to a SATA device. There are no IT and IR compliance points at the SAS Drive backplane receptacle connector when a SATA device is attached; SATA defines the signal characteristics that the SATA device delivers and that the SAS backplane is required to deliver to the SATA device. There are compliance points at the SAS 4i connector, however.
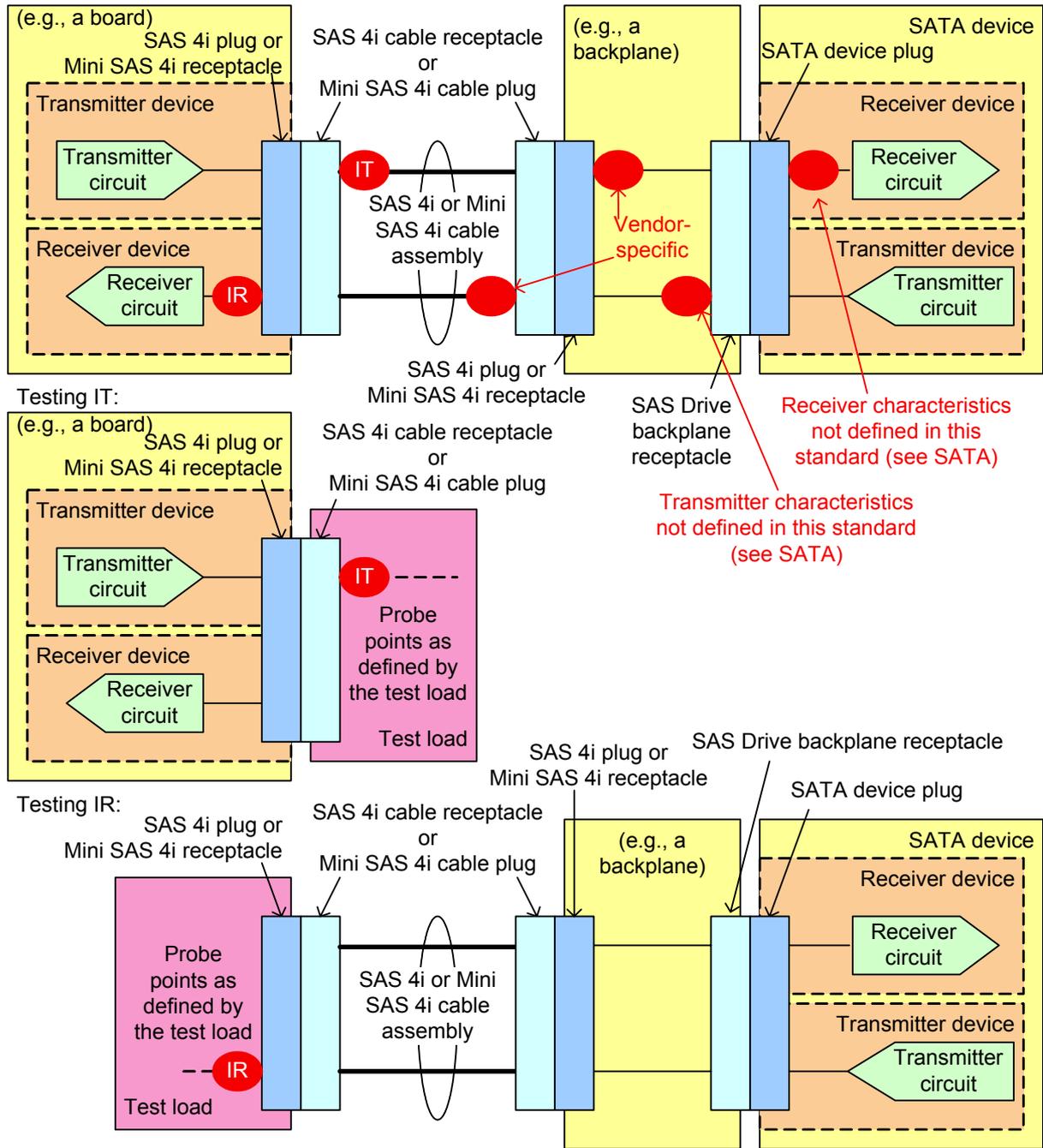
Figure 94 — Internal cable and backplane IT and IR compliance points with SATA device attached

Figure 95 shows the locations of the IT and IR compliance points using an internal cable. It also shows how two of the compliance points are tested using test loads (see 5.3.2).
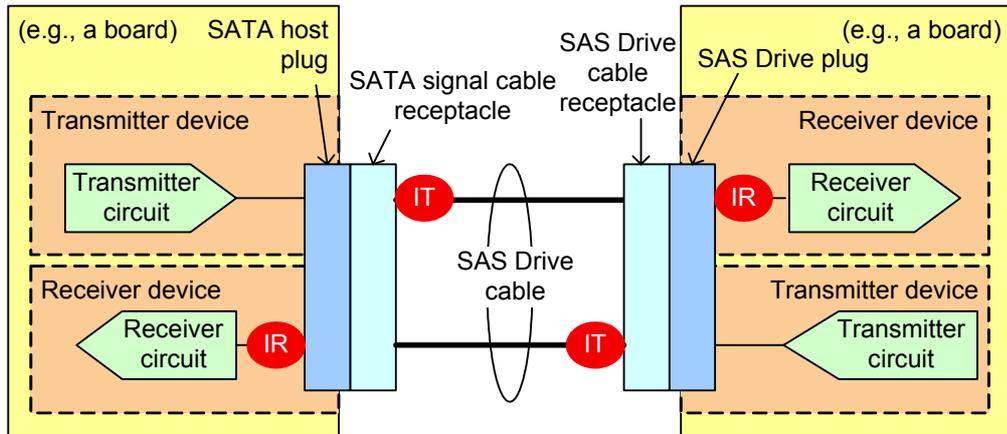
Testing the top-left IT:
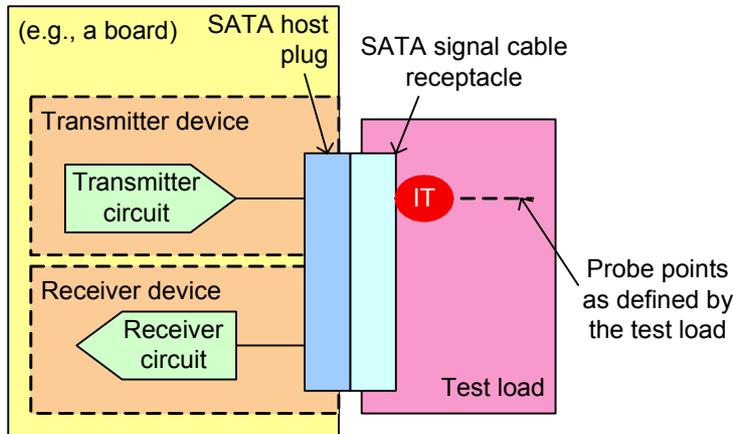
Testing the top-right IR:
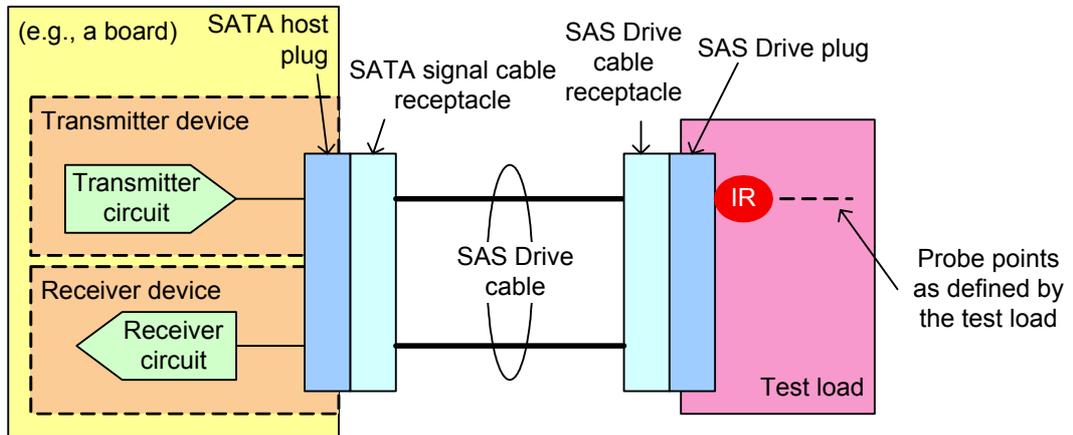
Figure 95 — Internal cable IT and IR compliance points

5.3.2 Test loads

5.3.2.1 Test loads overview

A test load methodology is used for the specification of transmitter device signal output characteristics (see 5.3.6.2 and 5.3.6.3) and delivered signal characteristics (see 5.3.7.2). This methodology specifies the signal as measured at specified probe points in specified test loads.

The test loads used by the methodology are:

    a) zero-length test load (see 5.3.2.2): used for testing transmitter device compliance points and receiver device compliance points;

    b) transmitter compliance transfer function (TCTF) test load (see 5.3.2.3): used for testing transmitter device compliance points; and

    c) low-loss TCTF test load (see 5.3.2.3): used for testing transmitter device compliance points when SATA devices using Gen2i levels (see SATAII-PHY) are supported and the SAS receiver device does not support the signal levels received through a full TCTF test load (see 5.3.2.3).

Physical positions denoted as probe points identify the position in the test load where the signal properties are measured, but do not imply that physical probing is used for the measurement. Physical probing may be disruptive to the signal and should not be used unless verified to be non-disruptive.

5.3.2.2 Zero-length test load

Figure 96 shows the zero-length test load as used for testing a transmitter device compliance point.
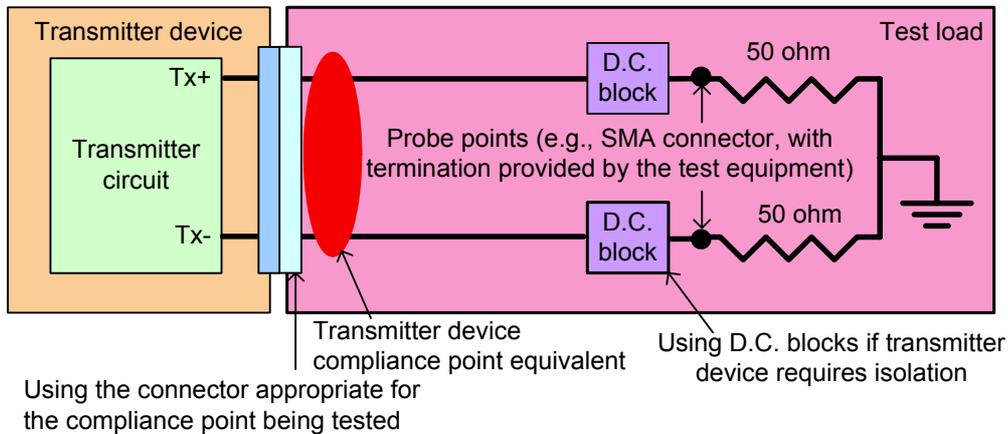
Figure 96 — Zero-length test load for transmitter device compliance point

Figure 97 shows the zero-length test load as used for testing a receiver device compliance point.
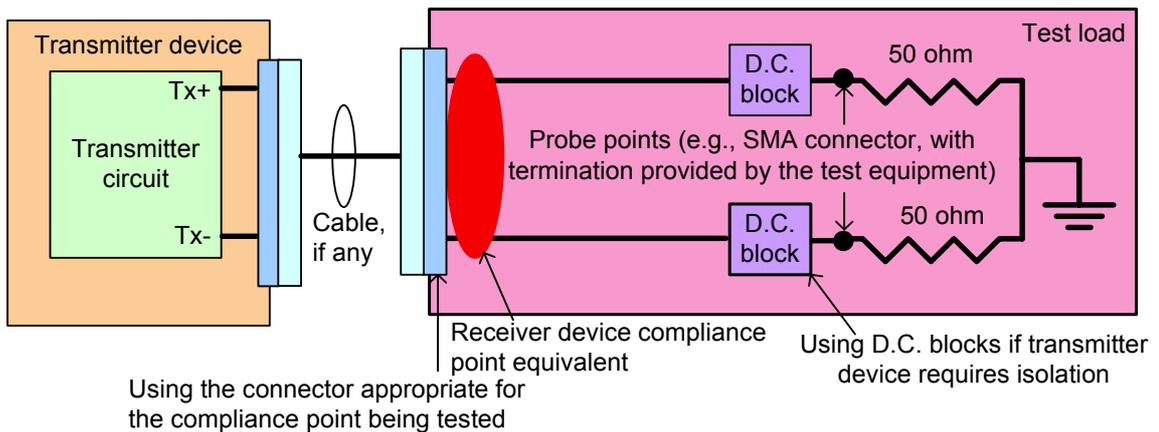
Figure 97 — Zero-length test load for receiver device compliance point

5.3.2.3 TCTF test load

Figure 98 shows the TCTF test load.
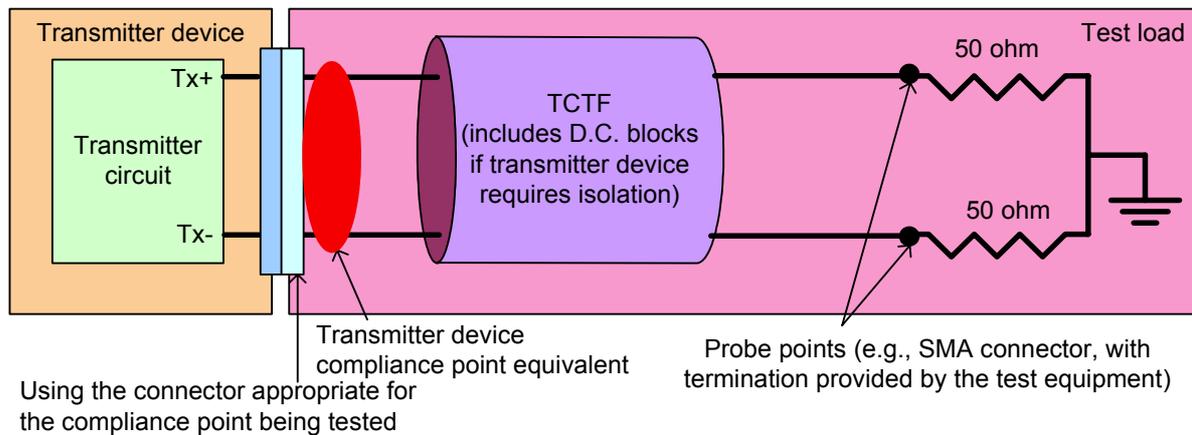
Figure 98 — TCTF test load

The TCTF test load shall meet the requirements in table 33 (see 5.2.6). The nominal impedance shall be the target impedance.

The TCTF is defined by a set of S-parameters (see B.9). Only the magnitude of $S_{DD21}(f)$ (i.e., insertion loss) is specified by this standard.

For testing a 3,0 Gbit/s transmitter device at IT, the TCTF test load shall comply with the following equations:

For 50 MHz < f $\leq$ 3,0 GHz:

$|S_{DD21}(f)| \leq$ -20 $\log_{10}(e)$ x ((6,5 x 10^{-6} x $f^{0,5}$) + (2,0 x 10^{-10} x f) + (3,3 x 10^{-20} x f^2)) dB

and for 3,0 GHz < f $\leq$ 5,0 GHz:

$|S_{DD21}(f)| \leq$ -10,9 dB

and, specifying a minimum ISI loss:

(($|S_{DD21}(f = 300$ MHz)$|$ of TCTF test load) - ($|S_{DD21}(f = 1\,500$ MHz)$|$) of TCTF test load)) > 3,9 dB

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f   signal frequency in Hz

For testing a 3,0 Gbit/s transmitter device at CT, the TCTF test load shall comply with the following equations:

For 50 MHz < f $\leq$ 3,0 GHz:

$|S_{DD21}(f)| \leq$ -20 $\log_{10}(e)$ x ((1,7 x 10^{-5} x $f^{0,5}$) + (1,0 x 10^{-10} x f)) dB

and for 3,0 GHz < f $\leq$ 5,0 GHz:

$|S_{DD21}(f)| \leq$ -10,7 dB

and, specifying a minimum ISI loss:

$|S_{DD21}(f = 300$ MHz)$|$ - $|S_{DD21}(f = 1\,500$ MHz)$|$ > 3,9 dB

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f   signal frequency in Hz

Figure 99 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at 3,0 Gbit/s.
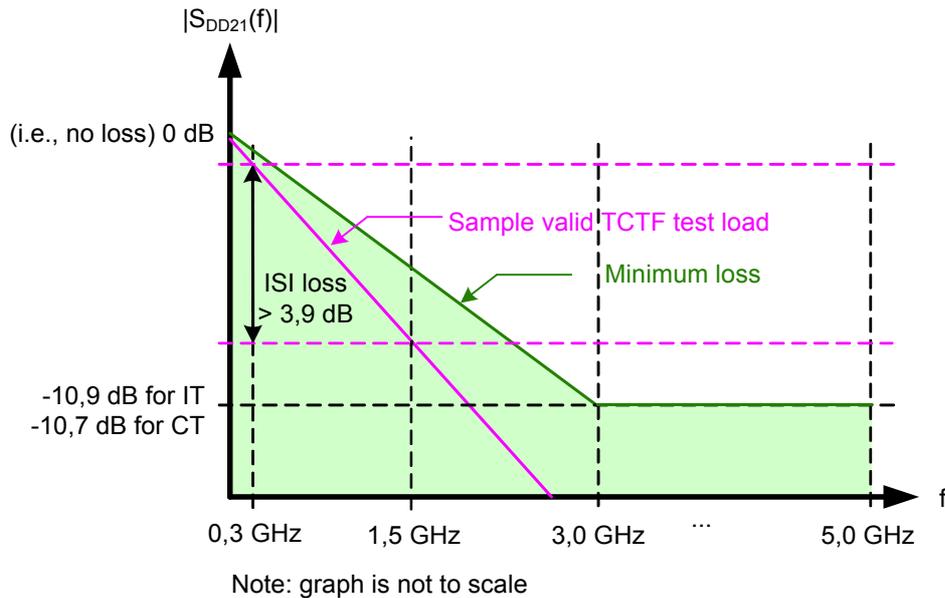
Figure 99 — TCTF test load S_{DD21} and ISI loss requirements at 3,0 Gbit/s

For testing a 1,5 Gbit/s transmitter device at IT, the TCTF test load shall comply with the following equations:

For 50 MHz < f ≤ 1,5 GHz:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((6,5 \times 10^{-6} \times f^{0,5}) + (2,0 \times 10^{-10} \times f) + (3,3 \times 10^{-20} \times f^2)) \text{ dB}$$

and for 1,5 GHz < f ≤ 5,0 GHz:

$$|S_{DD21}(f)| \leq -5,4 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 150 \text{ MHz})| - |S_{DD21}(f = 750 \text{ MHz})| > 2,0 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f   signal frequency in Hz

For testing a 1,5 Gbit/s transmitter device at CT, the TCTF test load shall comply with the following equations:

For 50 MHz < f ≤ 1,5 GHz:

$$|S_{DD21}(f)| \leq -20 \log_{10}(e) \times ((1,7 \times 10^{-5} \times f^{0,5}) + (1,0 \times 10^{-10} \times f)) \text{ dB}$$

and for 1,5 GHz < f ≤ 5,0 GHz:

$$|S_{DD21}(f)| \leq -7,0 \text{ dB}$$

and, specifying a minimum ISI loss:

$$|S_{DD21}(f = 150 \text{ MHz})| - |S_{DD21}(f = 750 \text{ MHz})| > 2,0 \text{ dB}$$

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f   signal frequency in Hz

Figure 100 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a TCTF test load and the $|S_{DD21}(f)|$ of a sample TCTF test load at 1,5 Gbit/s.
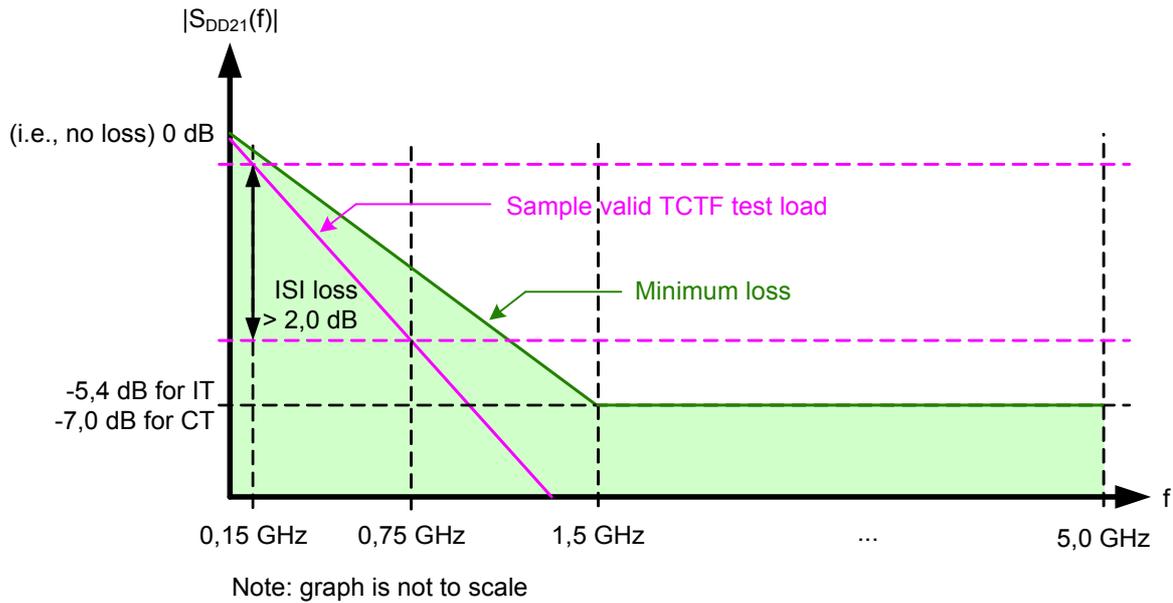
Figure 100 — TCTF test load S_{DD21} and ISI loss requirements at 1,5 Gbit/s

5.3.2.4 Low-loss TCTF test load

Figure 101 shows the low-loss TCTF test load.
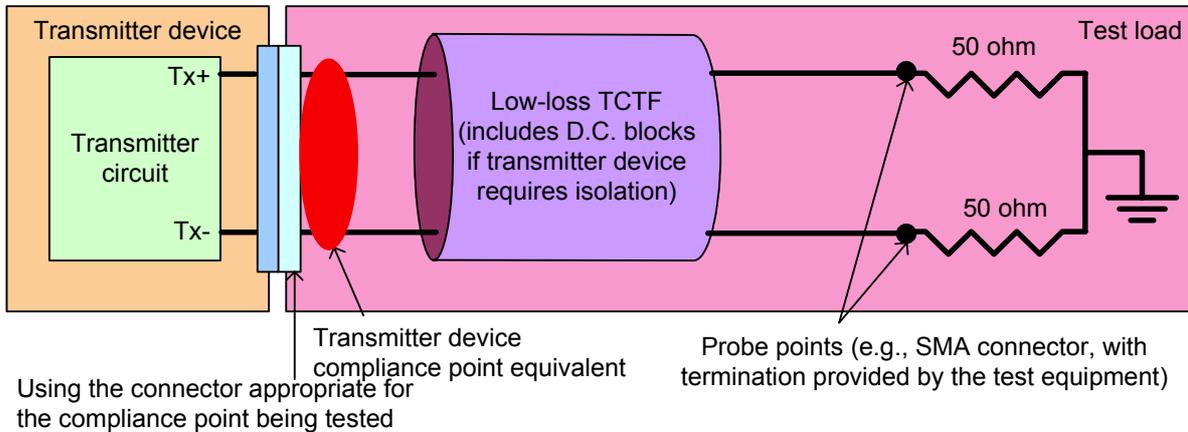
Figure 101 — Low-loss TCTF test load

The low-loss TCTF test load shall meet the requirements in table 33 (see 5.2.6). The nominal impedance shall be the target impedance.

The low-loss TCTF is defined by a set of S-parameters (see B.9). Only the magnitude of $S_{DD21}(f)$ (i.e., insertion loss) is specified by this standard.

The low-loss TCTF test load shall comply with the following equations:

For 50 MHz < f $\leq$ 3,0 GHz:

$|S_{DD21}(f)| \leq$ -20 $\log_{10}(e)$ x $((2,2 \times 10^{-6} \times f^{0,5}) + (6,9 \times 10^{-11} \times f) + (1,1 \times 10^{-20} \times f^2))$ dB

for 3,0 GHz < f $\leq$ 5,0 GHz:

$|S_{DD21}(f)| \leq$ 3,8 dB

and, specifying a minimum ISI loss:

$|S_{DD21}(f = 300$ MHz$)|$ - $|S_{DD21}(f = 1\ 500$ MHz$)|$ > 1,3 dB

where:

$|S_{DD21}(f)|$ magnitude of $S_{DD21}(f)$

f signal frequency in Hz

Figure 102 shows the allowable $|S_{DD21}(f)|$ and minimum ISI loss of a low-loss TCTF test load and the $|S_{DD21}(f)|$ of a sample low-loss TCTF test load at 3,0 Gbit/s.

Note: graph is not to scale
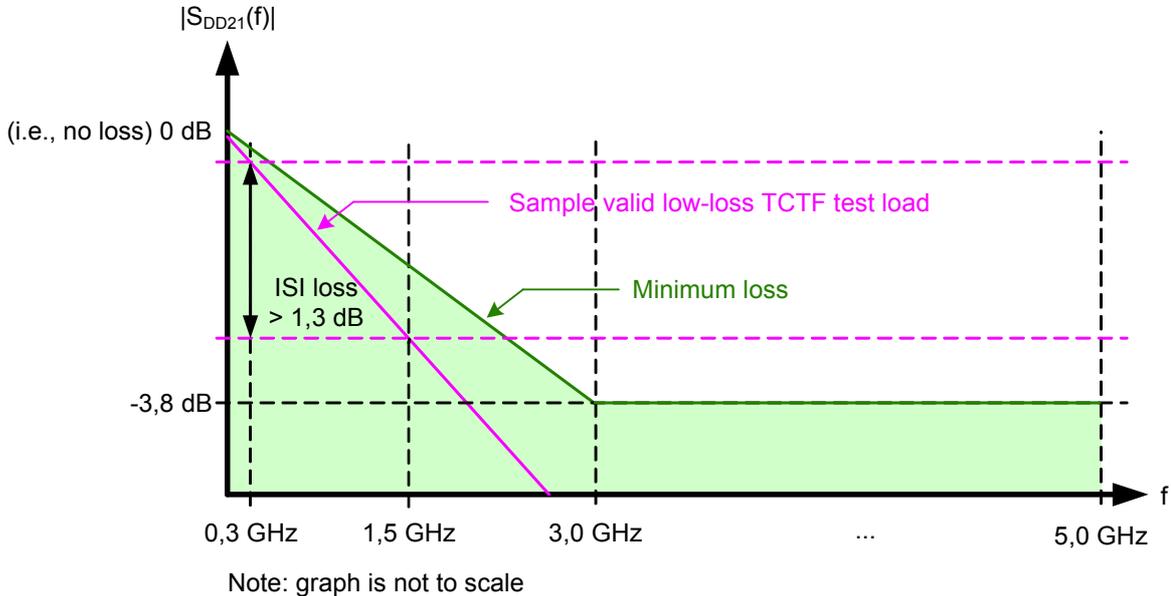
Figure 102 — Low-loss TCTF test load S_{DD21} and ISI loss requirements at 3,0 Gbit/s

5.3.3 General electrical characteristics

A TxRx connection is the complete simplex signal path between the transmitter circuit (see 3.1.242) and receiver circuit (see 3.1.152).

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

This standard defines the electrical requirements of the signal at the compliance points IT, IR, CT, and CR in a TxRx connection. Each compliant phy shall be compatible with these electrical requirements to allow interoperability within a SAS environment.

Each TxRx connection shall support a bit error ratio (BER) that is less than 10^{-12} (i.e., fewer than one bit error per 10^{12} bits). The parameters specified in this standard support meeting this requirement under all conditions including the minimum input and output amplitude levels.

Each TxRx connection shall be designed such that its loss characteristics are less than:

a)   the loss of the TCTF test load plus ISI at 3,0 Gbit/s (see figure 100 in 5.3.2.3) over the frequency range of 50 MHz to 3 000 MHz; or

b)   the loss of the low-loss TCTF test load plus ISI at 3,0 Gbit/s (see figure 102 in 5.3.2.4) over the frequency range of 50 MHz to 3 000 MHz, if the system supports SATA devices using Gen2i levels (see SATAII-PHY) but the receiver device does not support SATA Gen2i levels through the TCTF test load.

Each TxRx connection shall meet the delivered signal specifications in table 47 (see 5.3.7.2).

> NOTE 8 - A TxRx connection is constructed from multiple components. It is possible that a TxRx connection does not meet the delivered signal requirements of table 47 (see 5.3.7.2) when the combined losses and noise introduced by those components is considered, even if each individual component is compliant with the requirements of this standard. Such a TxRx connection is not compliant with this standard.

For external cable assemblies, these electrical requirements are consistent with using good quality passive cable assemblies constructed with shielded twinaxial cable with 0,205 mm^2 (24 AWG) solid wire up to 6 meters in length.

Each TxRx connection segment shall comply with the impedance requirements detailed in 5.2.6 for the conductive material from which they are formed. An equalizer network, if present, shall be considered part of the TxRx connection.

TxRx connections shall be applied only to homogenous ground applications (e.g., between devices within an enclosure or rack, or between enclosures interconnected by a common ground return or ground plane).

Table 41 defines the general electrical characteristics.

Table 41 — General electrical characteristics

Characteristic	Units	1,5 Gbit/s (i.e., G1)	3,0 Gbit/s (i.e., G2)
Physical link rate	MBps	150	300
Bit rate (nominal)	Mbaud	1 500	3 000
Unit interval (UI)(nominal)	ps	$666,\overline{6}$	$333,\overline{3}$
Differential TxRx connection impedance (nominal)	ohm	100	
A.C. coupling capacitor, maximum [a]	nF	12	
Maximum noise during OOB idle time [b]	mV(P-P)	120	

[a] The coupling capacitor value for A.C. coupled transmit and receive pairs. A.C. coupling requirements for transmitter devices are described in 5.3.6.1. A.C. coupling requirements for receiver devices are described in 5.3.7.1.
[b] With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3,0 Gbit/s), no signal level during the idle time shall exceed the specified maximum differential amplitude.

Table 42 defines the transmitter device general electrical characteristics.

Table 42 — General transmitter device electrical characteristics

Characteristic	Units	1,5 Gbit/s	3,0 Gbit/s
Physical link rate tolerance at IT and CT	ppm	± 100	
Maximum transmitter device transients [a]	V	± 1,2	
Transmitter device source termination:			
Differential impedance [b]	ohm	60 min/115 max	
Maximum differential impedance imbalance [b, c]	ohm	5	
Common-mode impedance [b]	ohm	15 min/40 max	

[a] See 5.3.4 for transient test circuits and conditions.
[b] All transmitter device termination measurements are made through mated connector pairs.
[c] The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.

Table 43 defines the receiver device general electrical characteristics.

Table 43 — Receiver device general electrical characteristics

Characteristic	Units	1,5 Gbit/s	3,0 Gbit/s
Physical link rate tolerance at IR if SATA is supported [a]	ppm	+350 / -5 350	
Physical link rate tolerance at IR if SATA is not supported and at CR	ppm	± 100	
Physical link rate tolerance at IT and CT	ppm	± 100	
Maximum receiver device transients [b]	V	± 1,2	
Receiver A.C. common-mode voltage tolerance V_{CM}, minimum [c]	mV(P-P)	150	
Receiver A.C. common-mode frequency tolerance range F_{CM} [c]	MHz	2 to 200	
Receiver device termination:			
Differential impedance [d, e, f]	ohm	100 ± 15	
Maximum differential impedance imbalance [d, e, f, g]	ohm	5	
Maximum receiver termination time constant [d, e, f]	ps	150	100
Common-mode impedance [d, e]	ohm	20 min/40 max	

[a] Allows support for SATA devices with spread spectrum clocking (see ATA/ATAPI-7 V3 and SATAII-PHY).

[b] See 5.3.4 for transient test circuits and conditions.

[c] Receiver devices shall tolerate sinusoidal common-mode noise components within the peak-to-peak amplitude (V_{CM}) and the frequency range (F_{CM}).

[d] All receiver device termination measurements are made through mated connector pairs.

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

[f] At the time point corresponding to the connection of the receiver device to the transmission line, the input capacitance of the receiver device and its connection to the transmission line may cause the measured impedance to fall below the minimum impedances specified in this table. With impedance measured using amplitude in units of ρ (i.e., the reflection coefficient, a dimensionless unit) and duration in units of time, the area of the impedance dip caused by this capacitance is the receiver termination time constant. The receiver termination time constant shall not be greater than the values shown in this table.

An approximate value for the receiver termination time constant is given by the product of the amplitude of the dip in units of ρ and the width of the dip in units of time, as measured at the half amplitude point. The amplitude is defined as the difference in the reflection coefficient between the reflection coefficient at the nominal impedance and the reflection coefficient at the minimum impedance point.

The value of the receiver device excess input capacitance is given by the following equation:

$$C = \frac{\text{receiver termination time constant}}{(R0 \parallel RR)}$$

where (R0 || RR) is the parallel combination of the transmission line characteristic impedance and termination resistance at the receiver device.

[g] The difference in measured impedance to SIGNAL GROUND on the plus and minus terminals on the interconnect, transmitter device, or receiver device, with a differential test signal applied to those terminals.

5.3.4 Transmitter and receiver device transients

Transients may occur at transmitter devices or receiver devices as a result of changes in supply power conditions or mode transitions.

A mode transition is an event that may result in a measurable transient due to the response of the transmitter device or receiver device. The following conditions constitute a mode transition:

    a) enabling or disabling driver circuitry;
    b) enabling or disabling receiver common-mode circuitry;
    c) hot plug event;
    d) adjusting driver amplitude;
    e) enabling or disabling pre-emphasis (i.e., de-emphasis); and
    f) adjusting terminator impedance.

Transmitter device transients are measured at nodes V_P and V_N with respect to GROUND on the test circuit shown in figure 103 during all power state and mode transitions. Receiver device transients are measured at nodes V_P and V_N with respect to GROUND on the test circuit shown in figure 104 during all power state and mode transitions. Test conditions shall include power supply power on and power off conditions, voltage sequencing, and mode transitions.

Figure 103 shows the test circuit attached to IT or CT to test transmitter device transients.
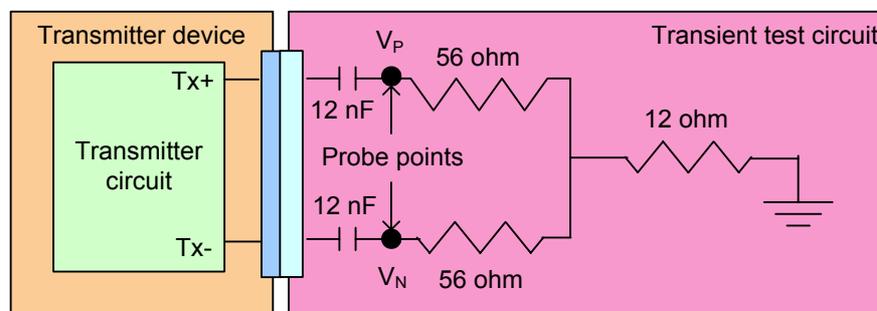
Figure 103 — Transmitter device transient test circuit

Figure 104 shows the test circuit attached to IR or CR to test receiver device transients.
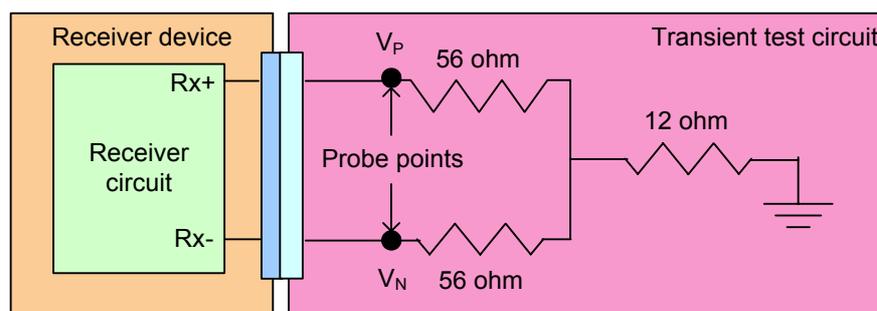
Figure 104 — Receiver device transient test circuit

5.3.5 Eye masks

5.3.5.1 Eye masks overview

The eye masks shown in this subclause shall be interpreted as graphical representations of the voltage and time limits of the signal. The eye mask boundaries define the eye contour of the 10^{-12} jitter population at all signal levels. Current equivalent time sampling oscilloscope technology is not practical for measuring compliance to this eye contour. See MJSQ for methods that are suitable for verifying compliance to these eye masks.

5.3.5.2 Transmitter device eye mask

Figure 105 describes the eye mask used for testing the signal output of the transmitter device at IT, CT, IR, and CR. This eye mask applies to jitter after the application of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667).
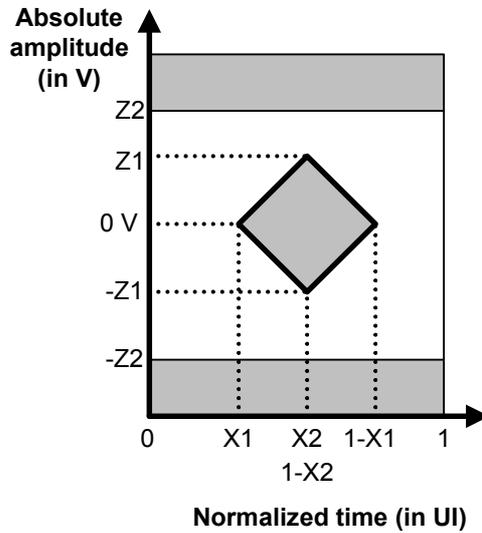
Figure 105 — Transmitter device eye mask

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

5.3.5.3 Receiver device eye mask

Figure 106 describes the eye mask used for testing the signal delivered to the receiver device at IR and CR. The signal shall be measured using a jitter timing reference (e.g., a golden PLL) that approximates a single pole (i.e., 20 dB per decade) low-pass filter with corner frequency of ((bit rate) / 1 667). This requirement accounts for the low frequency tracking properties and response time of the CDRs in receiver devices.
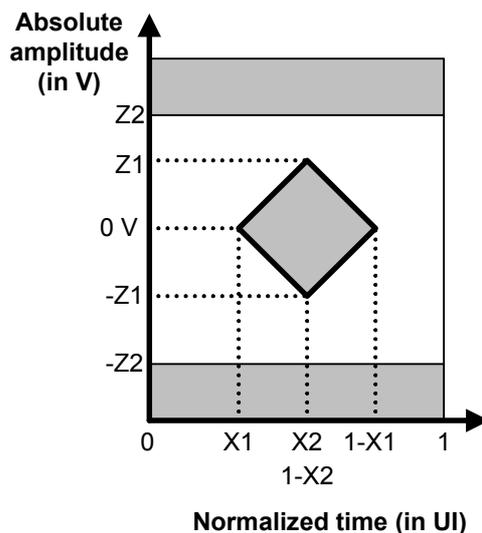
Figure 106 — Receiver device eye mask

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

5.3.5.4 Receiver device jitter tolerance eye mask

Figure 107 describes the eye mask used to test the jitter tolerance of the receiver device at IR and CR. Figure 107 shall be constructed using the following values:

a) X2 and Z2 shall be the values for the delivered signal listed in table 47 (see 5.3.7.2);
b) $X1_{OP}$ shall be half the value of TJ for maximum delivered jitter listed in table 48 (see 5.3.7.3); and
c) $X1_{TOL}$ shall be half the value of TJ for receiver device jitter tolerance listed in table 49 (see 5.3.7.4), for applied sinusoidal jitter frequencies above ((bit rate) / 1 667).
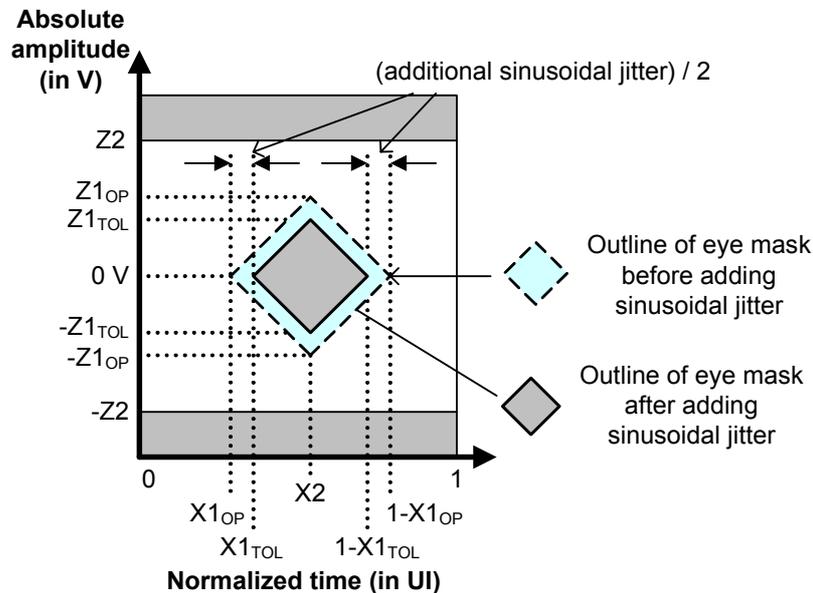
Figure 107 — Deriving a receiver device jitter tolerance eye mask

The leading and trailing edge slopes of the receiver device eye mask in figure 106 (see 5.3.5.3) shall be preserved. As a result, the amplitude value of Z1 is less than that given for the delivered signal in table 47 (see 5.3.7.2), and $Z1_{TOL}$ and $Z1_{OP}$ shall be defined from those slopes by the following equation:

$$Z1_{TOL} = Z1_{OP} \times \frac{X2 - \left(\frac{ASJ}{2}\right) - X1_{OP}}{X2 - X1_{OP}}$$

where:

$Z1_{TOL}$ is the value for Z1 to be used for the receiver device jitter tolerance eye mask

$Z1_{OP}$ is the Z1 value for the delivered signal in table 47

$X1_{OP}$ is the X1 value for the delivered signal in table 47

$X2$ is the X2 value for the delivered signal in table 47

ASJ is the additional sinusoidal jitter defined in figure 108

The X1 points in the receiver device jitter tolerance eye mask (see figure 107) are greater than the X1 points in the receiver device eye mask (see figure 106) due to the addition of sinusoidal jitter.

Figure 108 defines the applied sinusoidal jitter.
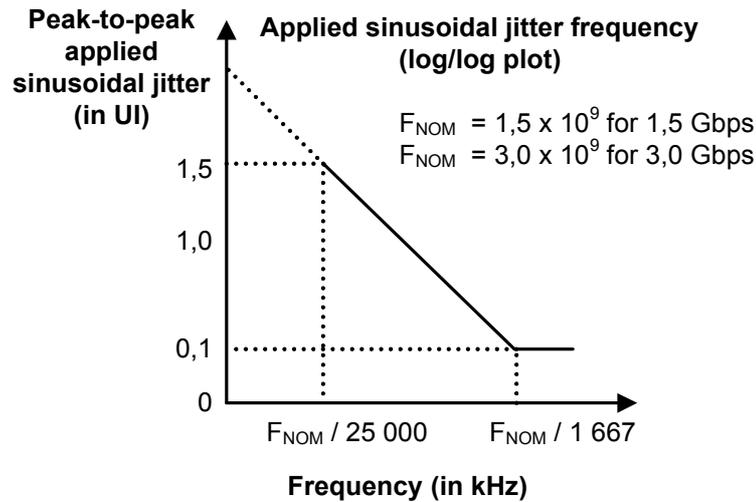
Figure 108 — Applied sinusoidal jitter

CJTPAT shall be used for all jitter testing unless otherwise specified. Annex A defines the required pattern on the physical link and provides information regarding special considerations for running disparity (see 6.2) and scrambling (see 7.6).

5.3.6 Transmitter device characteristics

5.3.6.1 Transmitter device characteristics overview

A.C. coupling requirements for transmitter devices are as follows:

a) transmitter devices using inter-enclosure TxRx connections (i.e., attached to CT compliance points) shall be A.C. coupled to the interconnect through a transmission network;

b) transmitter devices using intra-enclosure TxRx connections (i.e., attached to IT compliance points) that support SATA shall be A.C. coupled to the interconnect through a transmission network; and

c) transmitter devices using intra-enclosure TxRx connections (i.e., attached to IT compliance points) that do not support SATA may be A.C. or D.C. coupled.

Transmitter devices may or may not incorporate pre-emphasis (i.e., de-emphasis) and other forms of compensation. The transmitter device shall use the same settings (e.g., pre-emphasis and voltage swing) with both the zero-length test load and the appropriate TCTF test load.

See B.5 for a methodology for measuring transmitter device signal output.

5.3.6.2 Transmitter device signal output characteristics as measured with the zero-length test load

Table 44 specifies the signal output characteristics for the transmitter device as measured with the zero-length test load (see 5.3.2.2) attached at a transmitter device compliance point (i.e., IT or CT). All specifications are based on differential measurements.

Table 44 — Transmitter device signal output characteristics as measured with the zero-length test load at transmitter device compliance points IT and CT

Signal characteristic [a]	Units	1,5 Gbit/s	3,0 Gbit/s
Maximum intra-pair skew [b]	ps	20	15
Maximum transmitter device off voltage [c]	mV(P-P)	50	
Maximum rise/fall time [d]	ps	273	137
Minimum rise/fall time [d]	ps	67	
Maximum transmitter output imbalance [e]	%	10	
OOB offset delta [f]	mV	$\pm$ 25	
OOB common-mode delta [g]	mV	$\pm$ 50	

[a] All tests in this table shall be performed with zero-length test load (see 5.3.2.2).
[b] The intra-pair skew measurement shall be made at the midpoint of the transition with a repeating 0101b pattern on the physical link. The same stable trigger, coherent to the data stream, shall be used for both the Tx+ and Tx- signals. Intra-pair skew is defined as the time difference between the means of the midpoint crossing times of the Tx+ signal and the Tx- signal.
[c] The transmitter device off voltage is the maximum A.C. voltage measured at compliance points IT and CT when the transmitter is unpowered or transmitting D.C. idle (e.g., during idle time of an OOB signal).
[d] Rise/fall times are measured from 20 % to 80 % of the transition with a repeating 0101b pattern on the physical link.
[e] The maximum difference between the V+ and V- A.C. rms transmitter device amplitudes measured with CJTPAT (see A.2) into the zero-length test load shown in figure 96 (see 5.3.2.2), as a percentage of the average of the V+ and V- A.C. rms amplitudes.
[f] The maximum difference in the average differential voltage (D.C. offset) component between the burst times and the idle times of an OOB signal.
[g] The maximum difference in the average of the common-mode voltage between the burst times and the idle times of an OOB signal.

5.3.6.3 Transmitter device signal output characteristics as measured with each test load

Table 45 specifies the signal output characteristics for the transmitter device as measured with each test load (i.e., the zero-length test load (see 5.3.2.2) and either the TCTF test load (see 5.3.2.3) or the low-loss TCTF test load (see 5.3.2.4)) attached at a transmitter device compliance point (i.e., IT or CT). All specifications are based on differential measurements.

Table 45 — Transmitter device signal output characteristics as measured with each test load at transmitter device compliance points IT and CT

Signal characteristic	Units	IT		CT	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Maximum jitter (see figure 105 in 5.3.5.2) [a]	N/A	See table 48 in 5.3.7.3			
Maximum peak to peak voltage (i.e., 2 x Z2 in figure 105) if SATA is not supported	mV(P-P)	1 600		1 600	
Maximum peak to peak voltage (i.e., 2 x Z2 in figure 105) if SATA is supported	mV(P-P)	see SATAII-PHY [e]		N/A	
Minimum eye opening (i.e., 2 x Z1 in figure 105), if SATA is not supported	mV(P-P)	325	275	275	
Minimum eye opening (i.e., 2 x Z1 in figure 105), if SATA is supported	mV(P-P)	see SATAII-PHY [e]		N/A	
Half of maximum jitter (i.e., X1 in figure 105) [b]	UI	0,275			
Center of bit time (i.e., X2 in figure 105)	UI	0,50			
Maximum intra-pair skew [c]	ps	80	75	80	75
Maximum voltage (non-operational)	mV(P-P)	2 000			
Minimum OOB burst amplitude [d], if SATA is not supported	mV(P-P)	240 [f]			
Minimum OOB burst amplitude [d], if SATA is supported	mV(P-P)	240 mV [e] [g]		N/A	

[a] The value for X1 applies at a total jitter probability of 10^{-12}. At this level of probability, direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter requirements.

[b] The value for X1 shall be half the value of TJ for maximum delivered jitter listed in table 48. The test or analysis shall include the effects of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667).

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

[d] With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3,0 Gbit/s), each signal level during the OOB burst shall exceed the specified minimum differential amplitude before transitioning to the opposite bit value or before termination of the OOB burst.

[e] Amplitude measurement methodologies of SATA and this standard differ. Under conditions of maximum rise/fall time and jitter, eye diagram methodologies used in this standard may indicate less signal amplitude than the technique specified by SATAII-PHY. Implementors of designs supporting SATA are required to ensure interoperability and should perform additional system characterization with an eye diagram methodology using SATA devices.

[f] The OOB burst is comprised of either 1,5 Gbit/s ALIGN (0) primitives or 3,0 Gbit/s ALIGN (0) primitives (see 6.6).

[g] The OOB burst is comprised of 1,5 Gbit/s ALIGN (0) primitives (see 6.6 and SATAII-PHY).

5.3.6.4 Transmitter device maximum jitter

Table 46 defines the maximum jitter the transmitter device shall deliver as measured with each test load (i.e., the zero-length test load (see 5.3.2.2) and either the TCTF test load (see 5.3.2.3) or the low-loss TCTF test load (see 5.3.2.4)) at a transmitter device compliance point (i.e., IT or CT).

Table 46 — Transmitter device maximum jitter as measured with each test load at transmitter device compliance points IT and CT

Signal characteristic [a, b]	Units	IT		CT	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Deterministic jitter (DJ) [d]	UI	0,35			
Total jitter (TJ) [c, d, e]	UI	0,55			

[a] All DJ and TJ values are level 1 (see MJSQ).
[b] The values for jitter in this table are measured at the average signal amplitude point.
[c] TJ is specified at a CDF level of 10^{-12}.
[d] The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.2. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ.
[e] If TJ received at any point is less than the maximum allowed, then the jitter distribution of the signal is allowed to be asymmetric. The TJ plus the magnitude of the asymmetry shall not exceed the allowed maximum TJ. The numerical difference between the average of the peaks with a BER < 10^{-12} and the average of the individual events is the measure of the asymmetry. Jitter peak-to-peak measured < (maximum TJ - |Asymmetry|).

5.3.6.5 Transmitter device signal output levels for OOB signals

Transmitter devices supporting SATA shall use SATA Gen1i or Gen2i signal output levels (see SATAII-PHY) during the first OOB sequence (see 6.7) after a power on or hard reset. If the phy does not receive COMINIT within a hot-plug timeout (see 6.7.5), the transmitter device shall increase its transmit levels to the SAS signal output levels specified in table 44 (see 5.3.6) and table 45 (see 5.3.6.3) and perform the OOB sequence again. If no COMINIT is received within a hot-plug timeout of the second OOB sequence, the transmitter device shall initiate another OOB sequence using SATA Gen1i or Gen2i signal output levels. The transmitter device shall continue alternating between transmitting COMINIT using SATA Gen1i or Gen2i signal output levels and transmitting COMINIT with SAS signal output levels until the phy receives COMINIT.

If the phy both transmits and receives COMSAS (i.e., a SAS phy or expander phy is attached), the transmitter device shall set its transmit levels to the SAS signal output levels. If it had been using SATA Gen1i or Gen2i signal output levels, this mode transition (i.e., output voltage change) may result in a transient (see 5.3.4) during the idle time between COMSAS and the SAS speed negotiation sequence (see 6.7.4.2).

If the transmitter device is using SAS signal output levels and the phy does not receive COMSAS (i.e., a SATA phy is attached), the transmitter device shall set its transmit levels to the SATA Gen1i or Gen2i signal output levels and restart the OOB sequence.

Transmitter devices that do not support SATA shall transmit OOB signals using SAS signal output levels.

5.3.7 Receiver device characteristics

5.3.7.1 Receiver device characteristics overview

A.C. coupling requirements for receiver devices are as follows:

   a)   all receiver devices (i.e., attached to IR or CR compliance points) shall be A.C. coupled to the interconnect through a receive network.

The receive network shall terminate the TxRx connection by a 100 ohm equivalent impedance as specified in table 33 (see 5.2.6).

The receiver device shall operate within the required BER (see 5.3.3) when a signal with valid voltage and timing characteristics is delivered to the receiver device compliance point from a nominal 100 ohm source. The received signal shall be considered valid if it meets the voltage and timing limits specified in table 47 (see 5.3.7.2).

Additionally, the receiver device shall operate within the required BER (see 5.3.3) when the signal has additional sinusoidal jitter present as specified in table 49 (see 5.3.7.4) with the common-mode signal V_{CM} as specified in table 43 (see 5.3.3). Jitter tolerance for receiver device compliance points is illustrated in figure 107 (see 5.3.5.4). Figure 107 assumes that any external interference occurs prior to the point at which the test is applied. When testing the jitter tolerance capability of a receiver device, the additional 0,1 UI of sinusoidal jitter may be reduced by an amount proportional to the actual externally induced interference between the application point of the test and the input to the receiver device. The additional jitter reduces the eye opening in both voltage and time.

See B.8 for a methodology for measuring receiver device signal tolerance.

5.3.7.2 Delivered signal (receiver device signal tolerance) characteristics

Table 47 specifies the requirements of the signal delivered by the system with the zero-length test load (see 5.3.2.2) at the receiver device compliance point (i.e., IR or CR). These imply the required signal tolerance characteristics of the receiver device.

Table 47 — Delivered signal characteristics as measured with the zero length test load at receiver device compliance points IR and CR (Sheet 1 of 2)

Signal characteristic	Units	IR		CR	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Maximum peak to peak voltage (i.e., 2 x Z2 in figure 106) if a SATA phy is not attached	mV(P-P)	1 600		1 600	
Maximum peak to peak voltage (i.e., 2 x Z2 in figure 106) if a SATA phy is attached	mV(P-P)	see SATAII-PHY [e]		N/A	
Minimum eye opening (i.e., 2 x Z1 in figure 106), if a SATA phy is not attached	mV(P-P)	325	275	275	
Minimum eye opening (i.e., 2 x Z1 in figure 106), if a SATA phy using Gen1i or Gen1x levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	225 [e]	N/A	N/A	
Minimum eye opening (i.e., 2 x Z1 in figure 106), if a SATA phy using Gen2i levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	N/A	175 [e]	N/A	
Minimum eye opening (i.e., 2 x Z1 in figure 106), if a SATA phy using Gen2x levels is attached and the interconnect is characterized with the TCTF test load (see 5.3.2.3)	mV(P-P)	N/A	275 [e]	N/A	
Minimum eye opening (i.e., 2 x Z1 in figure 106), if a SATA phy is attached and the interconnect is characterized with the low-loss TCTF test load (see 5.3.2.4)	mV(P-P)	275 [e]		N/A	

Table 47 — Delivered signal characteristics as measured with the zero length test load at receiver device compliance points IR and CR (Sheet 2 of 2)

Signal characteristic	Units	IR		CR	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Jitter tolerance (see figure 107 in 5.3.5.4) [a]	N/A	See table 49 in 5.3.7.4			
Half of maximum jitter (i.e., X1 in figure 106) [b]	UI	0,275			
Center of bit time (i.e., X2 in figure 106)	UI	0,50			
Maximum intra-pair skew [c]	ps	80	75	80	75
Maximum voltage (non-operational)	mV(P-P)	2 000			
Minimum OOB burst amplitude [d], if SATA is not supported	mV(P-P)	240 [f]			
Minimum OOB burst amplitude [d], if SATA is supported	mV(P-P)	225 [g]		N/A	

[a] The value for X1 applies at a total jitter probability of 10^{-12}. At this level of probability direct visual comparison between the mask and actual signals is not a valid method for determining compliance with the jitter requirements.

[b] The value for X1 shall be half the value given for TJ in table 48. The test or analysis shall include the effects of a single pole high-pass frequency-weighting function that progressively attenuates jitter at 20 dB/decade below a frequency of ((bit rate) / 1 667).

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

[d] With a measurement bandwidth of 1,5 times the highest supported baud rate (e.g., 4,5 GHz for 3,0 Gbit/s), to be detected as an OOB burst, each signal level during the OOB burst shall exceed the specified minimum differential amplitude before transitioning to the opposite bit value or before termination of the OOB burst.

[e] Amplitude measurement methodologies of SATA and this standard differ. Under conditions of maximum rise/fall time and jitter, eye diagram methodologies used in this standard may indicate less signal amplitude than the technique specified by SATAII-PHY. Implementors of designs supporting attachment to SATA devices are required to ensure interoperability and should perform additional system characterization with an eye diagram methodology using SATA devices.

[f] The OOB burst is comprised of either 1,5 Gbit/s ALIGN (0) primitives or 3,0 Gbit/s ALIGN (0) primitives (see 6.6).

[g] The OOB burst is comprised of either 1,5 Gbit/s D24.3 characters, 1,5 Gbit/s ALIGN (0) primitives, or 3,0 Gbit/s ALIGN 0) primitives (see 6.6 and SATAII-PHY).

5.3.7.3 Maximum delivered jitter

Table 48 defines the maximum jitter the system shall deliver to the receiver device at the receiver device compliance point (i.e., IR or CR).

Table 48 — Maximum delivered jitter at receiver device compliance points IR and CR

Signal characteristic [a, b]	Units	IR		CR	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Deterministic jitter (DJ) [d]	UI	0,35			
Total jitter (TJ) [c, d, e]	UI	0,55			

[a] All DJ and TJ values are level 1 (see MJSQ).
[b] The values for jitter in this table are measured at the average signal amplitude point.
[c] TJ is specified at a CDF level of 10^{-12}.
[d] The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.2. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ.
[e] If TJ received at any point is less than the maximum allowed, then the jitter distribution of the signal is allowed to be asymmetric. The TJ plus the magnitude of the asymmetry shall not exceed the allowed maximum TJ. The numerical difference between the average of the peaks with a BER $< 10^{-12}$ and the average of the individual events is the measure of the asymmetry. Jitter peak-to-peak measured < (maximum TJ - |Asymmetry|).

5.3.7.4 Receiver device jitter tolerance

Table 49 defines the amount of jitter the receiver device shall tolerate at the receiver device compliance point (i.e., IR or CR). Receiver device jitter testing shall be performed with the maximum (i.e., slowest) rise/fall times, minimum signal amplitude, and maximum total jitter, and should be performed with normal activity in the receiver device (e.g., with other transmitter circuits and receiver circuits on the same board as the receiver device performing normal activity).

Table 49 — Receiver device jitter tolerance at receiver device compliance points IR and CR

Signal characteristic	Units	IR		CR	
		1,5 Gbit/s	3,0 Gbit/s	1,5 Gbit/s	3,0 Gbit/s
Applied sinusoidal jitter (SJ) [b]	UI	0,10 [c]		0,10 [d]	
Deterministic jitter (DJ) [a, h]	UI	0,35 [f]		0,35 [g]	
Total jitter (TJ) [a, e, h]	UI	0,65			

[a] All DJ and TJ values are level 1 (see MJSQ).
[b] The jitter values given are normative for a combination of applied SJ, DJ, and TJ that receiver devices shall be able to tolerate without exceeding the required BER (see 5.3.3). Receiver devices shall tolerate applied SJ of progressively greater amplitude at lower frequencies, according to figure 108 (see 5.3.5.4), with the same DJ and RJ levels as were used in the high frequency sweep.
[c] Applied sinusoidal swept frequency: 900 kHz to the minimum of 5 MHz and $(3{,}75 \times 2^{(generation - 1)}$ MHz) (e.g., 5 MHz for 1,5 Gbit/s and 7,5 MHz for 3,0 Gbit/s).
[d] Applied sinusoidal swept frequency: 1 800 kHz to the minimum of 5 MHz and $(3{,}75 \times 2^{(generation - 1)}$ MHz) (e.g., 5 MHz for 1,5 Gbit/s and 7,5 MHz for 3,0 Gbit/s).
[e] No value is given for RJ. For compliance with this standard, the actual RJ amplitude shall be the value that brings TJ to the stated value at a probability of 10^{-12}. The additional 0,1 UI of applied SJ is added to ensure the receiver device has sufficient operating margin in the presence of external interference.
[f] The measurement bandwidth shall be 900 kHz to 750 MHz.
[g] The measurement bandwidth shall be 1 800 kHz to 1 500 MHz.
[h] The DJ and TJ values in this table apply to jitter measured as described in 5.3.5.3. Values for DJ and TJ shall be calculated from the CDF for the jitter population using the calculation of level 1 jitter compliance levels method in MJSQ.

5.3.8 Spread spectrum clocking

Transmitter devices shall not transmit with spread spectrum clocking.

Receiver devices that support SATA shall support receiving with spread spectrum clocking (see ATA/ATAPI-7 V3 and SATAII-PHY). Receiver devices that do not support SATA are not required to support receiving with spread spectrum clocking.

An expander device shall retime data received from a SATA phy with an internal clock before forwarding to the rest of the SAS domain.

5.3.9 Non-tracking clock architecture

Transceivers shall be designed with a non-tracking clock architecture (i.e., the receive clock derived from the bit stream received by the receiver device shall not be used as the transmit clock by the transmitter device).

Receiver devices that support SATA shall tolerate clock tracking by the SATA device. Receiver devices that do not support SATA are not required to tolerate clock tracking by the SATA device.

5.4 READY LED signal electrical characteristics

A SAS target device uses the READY LED signal to activate an externally visible LED that indicates the state of readiness and activity of the SAS target device.

All SAS target devices using the SAS Drive plug connector (see 5.2.3.2.1.1) shall support the READY LED signal.

The READY LED signal is designed to pull down the cathode of an LED using an open collector or open drain transmitter circuit. The LED and the current limiting circuitry shall be external to the SAS target device.

Table 50 describes the output characteristics of the READY LED signal.

Table 50 — Output characteristics of the READY LED signal

State	Test condition	Requirement
Negated (LED off)	$0\text{ V} \leq V_{OH} \leq 3,6\text{ V}$	$-100\text{ }\mu\text{A} < I_{OH} < 100\text{ }\mu\text{A}$
Asserted (LED on)	$I_{OL} = 15\text{ mA}$	$0 \leq V_{OL} \leq 0,225\text{ V}$

The READY LED signal behavior is defined in 10.4.1.

> NOTE 9 - SATA devices use the pin used by the READY LED signal (i.e., P11) for activity indication and staggered spin-up disable (see SATAII-EXT). The output characteristics differ from those in table 50.

6 Phy layer

6.1 Phy layer overview

The phy layer defines 8b10b coding and OOB signals. Phy layer state machines interface between the link layer and the physical layer to perform the phy reset sequence and keep track of dword synchronization.

6.2 8b10b coding

6.2.1 8b10b coding overview

All information transferred in SAS is encoded into 10-bit characters using 8b10b encoding. Information includes data bytes (e.g., representing data in a frame) and control bytes (e.g., used for frame delimiters).

Running disparity (RD) shall be maintained separately on each physical link in each direction. During a connection (see 4.1.10), expander devices shall convert incoming 10-bit characters to 8-bit bytes and generate the 10-bit character with correct disparity for the output physical link. Phys within a device may or may not begin operation with the same disparity after the reset sequence.

6.2.2 8b10b coding introduction

Information to be transmitted across a physical link shall be encoded eight bits at a time into a 10-bit character and then transmitted serially bit-by-bit across the physical link. Information received over the physical link shall be collected ten bits at a time, and those characters that are used for data, called data characters, shall be decoded into the correct 8-bit data bytes. The 10-bit characters support all 256 8-bit combinations. Some of the remaining 10-bit characters, referred to as control characters, are used for functions that are to be distinguishable from the contents of a frame. The rest of the 10-bit characters are invalid characters.

8b10b coding ensures that sufficient transitions are present in the serial bit stream to make clock recovery possible at the receiver. Such encoding also greatly increases the likelihood of detecting any single or multiple bit errors that may occur during transmission and reception of information. In addition, some of the control characters of the transmission code contain a distinct and easily recognizable bit pattern called a comma pattern which assists a receiver in achieving character and dword alignment on the incoming bit stream.

6.2.3 8b10b coding notation conventions

This subclause uses letter notation for describing information bits and control variables. Such notation differs from the bit notation specified by the remainder of this standard. The following text describes the translation process between these notations and provides a translation example. It also describes the conventions used to name valid characters. This text is provided for the purposes of terminology clarification only.

An unencoded information byte is composed of eight information bits A, B, C, D, E, F, G, H and the control variable Z. This information is encoded into the bits a, b, c, d, e, i, f, g, h, j of a 10-bit character.

An information bit contains either a binary zero or a binary one. A control variable has either the value D or the value K. When the control variable associated with an unencoded information byte contains the value D, that byte is referred to as a data byte. When the control variable associated with an unencoded information byte contains the value K, that byte is referred to as a control byte.

The information bit labeled A corresponds to bit 0 in the numbering scheme of this standard, B corresponds to bit 1, and so on, as shown in table 51. Bit H is the most significant bit of the byte and bit A is the least significant bit of the byte.

Table 51 — Bit designations

Bit notation:	7	6	5	4	3	2	1	0	Control variable
Unencoded bit notation:	H	G	F	E	D	C	B	A	Z

Each valid character has been given a name using the following convention:

Zxx.y

where:

Z	is the control variable of the unencoded information byte. The value of Z is used to indicate whether the character is a data character (Z = D) or a control character (Z = K).
xx	is the decimal value of the binary number composed of the bits E, D, C, B, and A of the unencoded information byte in that order.
y	is the decimal value of the binary number composed of the bits H, G, and F of the unencoded information byte in that order.

Table 52 shows the conversion from byte notation to the character naming convention.

Table 52 — Conversion from byte notation to character name example

Byte notation	BCh			
Bit notation	7 6 5 4	3 2 1 0	Control	
	1 0 1 1	1 1 0 0	K	
Unencoded bit notation	H G F	E D C B A	Z	
	1 0 1	1 1 1 0 0	K	
Unencoded bit notation reordered to conform with Zxx.y naming convention	Z	E D C B A	H G	F
	K	1 1 1 0 0	1 0	1
Character name	K	28	.	5

Most Kxx.y combinations do not result in valid characters within the 8b10b coding scheme. Only those combinations that result in control characters as specified by table 54 are considered valid.

6.3 Character encoding and decoding

6.3.1 Introduction

This subclause describes how to select valid characters (i.e., 8b10b encoding) and check the validity of received characters (i.e., 10b8b decoding). It also specifies the ordering rules to be followed when transmitting the bits within a character.

6.3.2 Transmission order

Within the definition of the 8b10b code, the bit positions of the characters are labeled a, b, c, d, e, i, f, g, h, and j. Bit a shall be transmitted first, followed by bits b, c, d, e, i, f, g, h, and j, in that order.

> NOTE 10 - Bit i is transmitted between bit e and bit f, rather than in the order that would be indicated by the letters of the alphabet.

Characters within primitives shall be transmitted sequentially beginning with the control character used to distinguish the primitive (e.g., K28.3 or K28.5) and proceeding character by character from left to right within the definition of the primitive until all characters of the primitive are transmitted.

The contents of a frame shall be transmitted sequentially beginning with the primitive used to denote the start of frame (e.g., SOAF, SOF, or SATA_SOF) and proceeding character-by-character from left to right within the definition of the frame until the primitive used to denote the end of frame (e.g., EOAF, EOF, or SATA_EOF) is transmitted.

6.3.3 Data and control characters

Table 53 and table 54 define the data characters (i.e., Dxx.y characters) and control characters (i.e., Kxx.y characters), respectively, and shall be used for both generating characters (i.e., encoding) and checking the validity of received characters (i.e., decoding). Each data character and control character entry has two columns that represent two (not necessarily different) characters, corresponding to the current value of the running disparity (current RD - or current RD +). RD is a binary parameter with a negative (-) or positive (+) value.

After power on, the transmitter may initialize the current RD to either positive or negative. Upon transmission of any character, the transmitter shall calculate a new value for its RD based on the contents of the transmitted character.

After power on, the receiver shall assume either the positive or negative value for its initial RD. Upon reception of any character, the receiver shall determine whether the character is valid or invalid and shall calculate a new value for its RD based on the contents of the received character.

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

RD for a character shall be calculated on the basis of sub-blocks, where the first six bits (i.e., bits a, b, c, d, e, and i) form one sub-block (i.e., the six-bit sub-block) and the second four bits (i.e., bits f, g, h, and j) form the other sub-block (i.e., the four-bit sub-block):

   a) RD at the beginning of the six-bit sub-block is the RD at the end of the preceding character;
   b) RD at the beginning of the four-bit sub-block is the RD at the end of the preceding six-bit sub-block; and
   c) RD at the end of the character is the RD at the end of the four-bit sub-block.

RD for the sub-blocks shall be calculated as follows:

   a) If the sub-block contains more ones than zeros, then RD at the end of a sub-block is positive;
   b) If the sub-block contains more zeros than ones, then RD at the end of a sub-block is negative;
   c) If the sub-block contains equal numbers of zeros and ones, then:
      A) if it is a six-bit sub-block containing 000111b, then RD at the end of the sub-clock is positive;
      B) if it is a six-bit sub-block containing 111000b, then RD at the end of the sub-clock is negative;
      C) if it is a four-bit sub-block containing 0011b, then RD at the end of the sub-clock is positive;
      D) if it is a four-bit sub-block containing 1100b, then RD at the end of the sub-clock is negative; and
      E) otherwise, RD at the end of the sub-block is the same as at the beginning of the sub-block

All sub-blocks with equal numbers of zeros and ones have neutral disparity (i.e., the ending disparity is the same as the beginning disparity). In order to limit the run length of zeros or ones across adjacent sub-blocks, the 8b10b code rules specify that sub-blocks encoded as 000111b or 0011b are generated only when the RD at the beginning of the sub-block is positive, ensuring that RD at the end of these sub-blocks is also positive. Likewise, sub-blocks containing 111000b or 1100b are generated only when the RD at the beginning of the sub-block is negative, ensuring that RD at the end of these sub-blocks is also negative.

Table 53 defines the data characters.

Table 53 — Data characters (Sheet 1 of 6)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D00.0	000 00000	00h	100111 0100	011000 1011
D01.0	000 00001	01h	011101 0100	100010 1011
D02.0	000 00010	02h	101101 0100	010010 1011
D03.0	000 00011	03h	110001 1011	110001 0100
D04.0	000 00100	04h	110101 0100	001010 1011
D05.0	000 00101	05h	101001 1011	101001 0100
D06.0	000 00110	06h	011001 1011	011001 0100
D07.0	000 00111	07h	111000 1011	000111 0100
D08.0	000 01000	08h	111001 0100	000110 1011
D09.0	000 01001	09h	100101 1011	100101 0100
D10.0	000 01010	0Ah	010101 1011	010101 0100
D11.0	000 01011	0Bh	110100 1011	110100 0100
D12.0	000 01100	0Ch	001101 1011	001101 0100
D13.0	000 01101	0Dh	101100 1011	101100 0100
D14.0	000 01110	0Eh	011100 1011	011100 0100
D15.0	000 01111	0Fh	010111 0100	101000 1011
D16.0	000 10000	10h	011011 0100	100100 1011
D17.0	000 10001	11h	100011 1011	100011 0100
D18.0	000 10010	12h	010011 1011	010011 0100
D19.0	000 10011	13h	110010 1011	110010 0100

Table 53 — Data characters (Sheet 2 of 6)

Name	Data byte Binary representation (HGF EDCBA)	Data byte Hexadecimal representation	Data character (binary representation) Current RD - abcdei fghj	Data character (binary representation) Current RD + abcdei fghj
D20.0	000 10100	14h	001011 1011	001011 0100
D21.0	000 10101	15h	101010 1011	101010 0100
D22.0	000 10110	16h	011010 1011	011010 0100
D23.0	000 10111	17h	111010 0100	000101 1011
D24.0	000 11000	18h	110011 0100	001100 1011
D25.0	000 11001	19h	100110 1011	100110 0100
D26.0	000 11010	1Ah	010110 1011	010110 0100
D27.0	000 11011	1Bh	110110 0100	001001 1011
D28.0	000 11100	1Ch	001110 1011	001110 0100
D29.0	000 11101	1Dh	101110 0100	010001 1011
D30.0	000 11110	1Eh	011110 0100	100001 1011
D31.0	000 11111	1Fh	101011 0100	010100 1011
D00.1	001 00000	20h	100111 1001	011000 1001
D01.1	001 00001	21h	011101 1001	100010 1001
D02.1	001 00010	22h	101101 1001	010010 1001
D03.1	001 00011	23h	110001 1001	110001 1001
D04.1	001 00100	24h	110101 1001	001010 1001
D05.1	001 00101	25h	101001 1001	101001 1001
D06.1	001 00110	26h	011001 1001	011001 1001
D07.1	001 00111	27h	111000 1001	000111 1001
D08.1	001 01000	28h	111001 1001	000110 1001
D09.1	001 01001	29h	100101 1001	100101 1001
D10.1	001 01010	2Ah	010101 1001	010101 1001
D11.1	001 01011	2Bh	110100 1001	110100 1001
D12.1	001 01100	2Ch	001101 1001	001101 1001
D13.1	001 01101	2Dh	101100 1001	101100 1001
D14.1	001 01110	2Eh	011100 1001	011100 1001
D15.1	001 01111	2Fh	010111 1001	101000 1001
D16.1	001 10000	30h	011011 1001	100100 1001
D17.1	001 10001	31h	100011 1001	100011 1001
D18.1	001 10010	32h	010011 1001	010011 1001
D19.1	001 10011	33h	110010 1001	110010 1001
D20.1	001 10100	34h	001011 1001	001011 1001
D21.1	001 10101	35h	101010 1001	101010 1001
D22.1	001 10110	36h	011010 1001	011010 1001
D23.1	001 10111	37h	111010 1001	000101 1001
D24.1	001 11000	38h	110011 1001	001100 1001
D25.1	001 11001	39h	100110 1001	100110 1001
D26.1	001 11010	3Ah	010110 1001	010110 1001
D27.1	001 11011	3Bh	110110 1001	001001 1001
D28.1	001 11100	3Ch	001110 1001	001110 1001
D29.1	001 11101	3Dh	101110 1001	010001 1001
D30.1	001 11110	3Eh	011110 1001	100001 1001
D31.1	001 11111	3Fh	101011 1001	010100 1001
D00.2	010 00000	40h	100111 0101	011000 0101
D01.2	010 00001	41h	011101 0101	100010 0101
D02.2	010 00010	42h	101101 0101	010010 0101
D03.2	010 00011	43h	110001 0101	110001 0101
D04.2	010 00100	44h	110101 0101	001010 0101
D05.2	010 00101	45h	101001 0101	101001 0101
D06.2	010 00110	46h	011001 0101	011001 0101
D07.2	010 00111	47h	111000 0101	000111 0101
D08.2	010 01000	48h	111001 0101	000110 0101
D09.2	010 01001	49h	100101 0101	100101 0101
D10.2	010 01010	4Ah	010101 0101	010101 0101
D11.2	010 01011	4Bh	110100 0101	110100 0101
D12.2	010 01100	4Ch	001101 0101	001101 0101

Table 53 — Data characters (Sheet 3 of 6)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D13.2	010 01101	4Dh	101100 0101	101100 0101
D14.2	010 01110	4Eh	011100 0101	011100 0101
D15.2	010 01111	4Fh	010111 0101	101000 0101
D16.2	010 10000	50h	011011 0101	100100 0101
D17.2	010 10001	51h	100011 0101	100011 0101
D18.2	010 10010	52h	010011 0101	010011 0101
D19.2	010 10011	53h	110010 0101	110010 0101
D20.2	010 10100	54h	001011 0101	001011 0101
D21.2	010 10101	55h	101010 0101	101010 0101
D22.2	010 10110	56h	011010 0101	011010 0101
D23.2	010 10111	57h	111010 0101	000101 0101
D24.2	010 11000	58h	110011 0101	001100 0101
D25.2	010 11001	59h	100110 0101	100110 0101
D26.2	010 11010	5Ah	010110 0101	010110 0101
D27.2	010 11011	5Bh	110110 0101	001001 0101
D28.2	010 11100	5Ch	001110 0101	001110 0101
D29.2	010 11101	5Dh	101110 0101	010001 0101
D30.2	010 11110	5Eh	011110 0101	100001 0101
D31.2	010 11111	5Fh	101011 0101	010100 0101
D00.3	011 00000	60h	100111 0011	011000 1100
D01.3	011 00001	61h	011101 0011	100010 1100
D02.3	011 00010	62h	101101 0011	010010 1100
D03.3	011 00011	63h	110001 1100	110001 0011
D04.3	011 00100	64h	110101 0011	001010 1100
D05.3	011 00101	65h	101001 1100	101001 0011
D06.3	011 00110	66h	011001 1100	011001 0011
D07.3	011 00111	67h	111000 1100	000111 0011
D08.3	011 01000	68h	111001 0011	000110 1100
D09.3	011 01001	69h	100101 1100	100101 0011
D10.3	011 01010	6Ah	010101 1100	010101 0011
D11.3	011 01011	6Bh	110100 1100	110100 0011
D12.3	011 01100	6Ch	001101 1100	001101 0011
D13.3	011 01101	6Dh	101100 1100	101100 0011
D14.3	011 01110	6Eh	011100 1100	011100 0011
D15.3	011 01111	6Fh	010111 0011	101000 1100
D16.3	011 10000	70h	011011 0011	100100 1100
D17.3	011 10001	71h	100011 1100	100011 0011
D18.3	011 10010	72h	010011 1100	010011 0011
D19.3	011 10011	73h	110010 1100	110010 0011
D20.3	011 10100	74h	001011 1100	001011 0011
D21.3	011 10101	75h	101010 1100	101010 0011
D22.3	011 10110	76h	011010 1100	011010 0011
D23.3	011 10111	77h	111010 0011	000101 1100
D24.3	011 11000	78h	110011 0011	001100 1100
D25.3	011 11001	79h	100110 1100	100110 0011
D26.3	011 11010	7Ah	010110 1100	010110 0011
D27.3	011 11011	7Bh	110110 0011	001001 1100
D28.3	011 11100	7Ch	001110 1100	001110 0011
D29.3	011 11101	7Dh	101110 0011	010001 1100
D30.3	011 11110	7Eh	011110 0011	100001 1100
D31.3	011 11111	7Fh	101011 0011	010100 1100
D00.4	100 00000	80h	100111 0010	011000 1101
D01.4	100 00001	81h	011101 0010	100010 1101
D02.4	100 00010	82h	101101 0010	010010 1101
D03.4	100 00011	83h	110001 1101	110001 0010
D04.4	100 00100	84h	110101 0010	001010 1101
D05.4	100 00101	85h	101001 1101	101001 0010

Table 53 — Data characters (Sheet 4 of 6)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D06.4	100 00110	86h	011001 1101	011001 0010
D07.4	100 00111	87h	111000 1101	000111 0010
D08.4	100 01000	88h	111001 0010	000110 1101
D09.4	100 01001	89h	100101 1101	100101 0010
D10.4	100 01010	8Ah	010101 1101	010101 0010
D11.4	100 01011	8Bh	110100 1101	110100 0010
D12.4	100 01100	8Ch	001101 1101	001101 0010
D13.4	100 01101	8Dh	101100 1101	101100 0010
D14.4	100 01110	8Eh	011100 1101	011100 0010
D15.4	100 01111	8Fh	010111 0010	101000 1101
D16.4	100 10000	90h	011011 0010	100100 1101
D17.4	100 10001	91h	100011 1101	100011 0010
D18.4	100 10010	92h	010011 1101	010011 0010
D19.4	100 10011	93h	110010 1101	110010 0010
D20.4	100 10100	94h	001011 1101	001011 0010
D21.4	100 10101	95h	101010 1101	101010 0010
D22.4	100 10110	96h	011010 1101	011010 0010
D23.4	100 10111	97h	111010 0010	000101 1101
D24.4	100 11000	98h	110011 0010	001100 1101
D25.4	100 11001	99h	100110 1101	100110 0010
D26.4	100 11010	9Ah	010110 1101	010110 0010
D27.4	100 11011	9Bh	110110 0010	001001 1101
D28.4	100 11100	9Ch	001110 1101	001110 0010
D29.4	100 11101	9Dh	101110 0010	010001 1101
D30.4	100 11110	9Eh	011110 0010	100001 1101
D31.4	100 11111	9Fh	101011 0010	010100 1101
D00.5	101 00000	A0h	100111 1010	011000 1010
D01.5	101 00001	A1h	011101 1010	100010 1010
D02.5	101 00010	A2h	101101 1010	010010 1010
D03.5	101 00011	A3h	110001 1010	110001 1010
D04.5	101 00100	A4h	110101 1010	001010 1010
D05.5	101 00101	A5h	101001 1010	101001 1010
D06.5	101 00110	A6h	011001 1010	011001 1010
D07.5	101 00111	A7h	111000 1010	000111 1010
D08.5	101 01000	A8h	111001 1010	000110 1010
D09.5	101 01001	A9h	100101 1010	100101 1010
D10.5	101 01010	AAh	010101 1010	010101 1010
D11.5	101 01011	ABh	110100 1010	110100 1010
D12.5	101 01100	ACh	001101 1010	001101 1010
D13.5	101 01101	ADh	101100 1010	101100 1010
D14.5	101 01110	AEh	011100 1010	011100 1010
D15.5	101 01111	AFh	010111 1010	101000 1010
D16.5	101 10000	B0h	011011 1010	100100 1010
D17.5	101 10001	B1h	100011 1010	100011 1010
D18.5	101 10010	B2h	010011 1010	010011 1010
D19.5	101 10011	B3h	110010 1010	110010 1010
D20.5	101 10100	B4h	001011 1010	001011 1010
D21.5	101 10101	B5h	101010 1010	101010 1010
D22.5	101 10110	B6h	011010 1010	011010 1010
D23.5	101 10111	B7h	111010 1010	000101 1010
D24.5	101 11000	B8h	110011 1010	001100 1010
D25.5	101 11001	B9h	100110 1010	100110 1010
D26.5	101 11010	BAh	010110 1010	010110 1010
D27.5	101 11011	BBh	110110 1010	001001 1010
D28.5	101 11100	BCh	001110 1010	001110 1010
D29.5	101 11101	BDh	101110 1010	010001 1010
D30.5	101 11110	BEh	011110 1010	100001 1010

Table 53 — Data characters (Sheet 5 of 6)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D31.5	101 11111	BFh	101011 1010	010100 1010
D00.6	110 00000	C0h	100111 0110	011000 0110
D01.6	110 00001	C1h	011101 0110	100010 0110
D02.6	110 00010	C2h	101101 0110	010010 0110
D03.6	110 00011	C3h	110001 0110	110001 0110
D04.6	110 00100	C4h	110101 0110	001010 0110
D05.6	110 00101	C5h	101001 0110	101001 0110
D06.6	110 00110	C6h	011001 0110	011001 0110
D07.6	110 00111	C7h	111000 0110	000111 0110
D08.6	110 01000	C8h	111001 0110	000110 0110
D09.6	110 01001	C9h	100101 0110	100101 0110
D10.6	110 01010	CAh	010101 0110	010101 0110
D11.6	110 01011	CBh	110100 0110	110100 0110
D12.6	110 01100	CCh	001101 0110	001101 0110
D13.6	110 01101	CDh	101100 0110	101100 0110
D14.6	110 01110	CEh	011100 0110	011100 0110
D15.6	110 01111	CFh	010111 0110	101000 0110
D16.6	110 10000	D0h	011011 0110	100100 0110
D17.6	110 10001	D1h	100011 0110	100011 0110
D18.6	110 10010	D2h	010011 0110	010011 0110
D19.6	110 10011	D3h	110010 0110	110010 0110
D20.6	110 10100	D4h	001011 0110	001011 0110
D21.6	110 10101	D5h	101010 0110	101010 0110
D22.6	110 10110	D6h	011010 0110	011010 0110
D23.6	110 10111	D7h	111010 0110	000101 0110
D24.6	110 11000	D8h	110011 0110	001100 0110
D25.6	110 11001	D9h	100110 0110	100110 0110
D26.6	110 11010	DAh	010110 0110	010110 0110
D27.6	110 11011	DBh	110110 0110	001001 0110
D28.6	110 11100	DCh	001110 0110	001110 0110
D29.6	110 11101	DDh	101110 0110	010001 0110
D30.6	110 11110	DEh	011110 0110	100001 0110
D31.6	110 11111	DFh	101011 0110	010100 0110
D00.7	111 00000	E0h	100111 0001	011000 1110
D01.7	111 00001	E1h	011101 0001	100010 1110
D02.7	111 00010	E2h	101101 0001	010010 1110
D03.7	111 00011	E3h	110001 1110	110001 0001
D04.7	111 00100	E4h	110101 0001	001010 1110
D05.7	111 00101	E5h	101001 1110	101001 0001
D06.7	111 00110	E6h	011001 1110	011001 0001
D07.7	111 00111	E7h	111000 1110	000111 0001
D08.7	111 01000	E8h	111001 0001	000110 1110
D09.7	111 01001	E9h	100101 1110	100101 0001
D10.7	111 01010	EAh	010101 1110	010101 0001
D11.7	111 01011	EBh	110100 1110	110100 1000
D12.7	111 01100	ECh	001101 1110	001101 0001
D13.7	111 01101	EDh	101100 1110	101100 1000
D14.7	111 01110	EEh	011100 1110	011100 1000
D15.7	111 01111	EFh	010111 0001	101000 1110

Table 53 — Data characters (Sheet 6 of 6)

Name	Data byte		Data character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
D16.7	111 10000	F0h	011011 0001	100100 1110
D17.7	111 10001	F1h	100011 0111	100011 0001
D18.7	111 10010	F2h	010011 0111	010011 0001
D19.7	111 10011	F3h	110010 1110	110010 0001
D20.7	111 10100	F4h	001011 0111	001011 0001
D21.7	111 10101	F5h	101010 1110	101010 0001
D22.7	111 10110	F6h	011010 1110	011010 0001
D23.7	111 10111	F7h	111010 0001	000101 1110
D24.7	111 11000	F8h	110011 0001	001100 1110
D25.7	111 11001	F9h	100110 1110	100110 0001
D26.7	111 11010	FAh	010110 1110	010110 0001
D27.7	111 11011	FBh	110110 0001	001001 1110
D28.7	111 11100	FCh	001110 1110	001110 0001
D29.7	111 11101	FDh	101110 0001	010001 1110
D30.7	111 11110	FEh	011110 0001	100001 1110
D31.7	111 11111	FFh	101011 0001	010100 1110

Table 54 defines the control characters. Comma patterns, which are two bits of one polarity followed by five bits of the opposite polarity (i.e., 0011111b or 1100000b), are underlined.

Table 54 — Control characters

Name	Control byte		Control character (binary representation)	
	Binary representation (HGF EDCBA)	Hexadecimal representation	Current RD - abcdei fghj	Current RD + abcdei fghj
K28.0	000 11100	1Ch	001111 0100	110000 1011
K28.1	001 11100	3Ch	<u>001111 1</u>001	<u>110000 0</u>110
K28.2	010 11100	5Ch	001111 0101	110000 1010
K28.3	011 11100	7Ch	001111 0011	110000 1100
K28.4	100 11100	9Ch	001111 0010	110000 1101
K28.5	101 11100	BCh	<u>001111 1</u>010	<u>110000 0</u>101
K28.6	110 11100	DCh	001111 0110	110000 1001
K28.7	111 11100	FCh	<u>001111 1</u>000	<u>110000 0</u>111
K23.7	111 10111	F7h	111010 1000	000101 0111
K27.7	111 11011	FBh	110110 1000	001001 0111
K29.7	111 11101	FDh	101110 1000	010001 0111
K30.7	111 11110	FEh	011110 1000	100001 0111

NOTE 11 - K28.1, K28.5, and K28.7 are the only characters which contain comma patterns. Comma patterns do not appear in any data characters and do not appear across any adjacent data characters. The K28.7 control character introduces a comma pattern when followed by any of the following characters: K28.y, D3.y, D11.y, D12.y, D19.y, D20.y, or D28.y, where y is a value in the range 0 to 7, inclusive. None of the other control characters introduce a comma pattern when adjacent to any other character. Therefore, K28.7 is not used, ensuring that comma patterns do not appear in any sequence of characters except the first 7 bits of K28.1 or K28.5.

The only control characters used in this standard are K28.3, K28.5, and K28.6, as defined in table 55.

Table 55 — Control character usage

First character of a dword	Usage in SAS physical links	Usage in SATA physical links
K28.3	Primitives used only inside STP connections	All primitives except ALIGN
K28.5	ALIGN and most primitives defined in this standard	ALIGN
K28.6	Not used	SATA_ERROR

See 7.2 for details on primitives, which use those control characters.

6.3.4 Encoding characters in the transmitter

To transmit a data byte, the transmitter shall select the appropriate character from table 53 based on the current value of the transmitter's RD. To transmit a control byte, the transmitter shall select the appropriate character from table 54 based on the current value of the transmitter's RD. After the transmitting the character, the transmitter shall calculate a new value for its RD based on that character. This new value shall be used as the transmitter's current RD for the next character transmitted. This process is called 8b10b encoding.

6.3.5 Decoding characters in the receiver

After receiving a character, the receiver shall search the character column in table 53 and table 54 corresponding to its current RD to determine the data byte or control byte to which the character corresponds. This process is called 10b8b decoding. If the received character is not found in the proper column, then the character shall be considered invalid and the dword containing the character shall be considered an invalid dword.

Regardless of the received character's validity, the received character shall be used to calculate a new value of RD in the receiver. This new value shall be used as the receiver's current RD for the next received character.

Detection of a code violation does not necessarily indicate that the character in which the code violation was detected is in error. Code violations may result from a prior error that altered the RD of the bit stream but did not result in a detectable error at the character in which the error occurred. The example shown in table 56 exhibits this behavior. These errors may span dword boundaries. Expanders forwarding such a dword forward it as an ERROR (see 7.2.5.7).

Table 56 — Delayed code violation example

	RD	First character	RD	Second character	RD	Third character	RD
Transmitted character stream	-	D21.1	-	D10.2	-	D23.5	+
Transmitted bit stream	-	101010 1001	-	010101 0101	-	111010 1010	+
Bit stream after error	-	101010 1011 (error in second to last bit)	+	010101 0101	+	111010 1010	+
Decoded character stream	-	D21.0 (rather than D21.1) (not detected as an error)	+	D10.2 (no error)	+	Code violation (although D23.5 was properly received)	+

6.4 Dwords, primitives, data dwords, and invalid dwords

All characters transferred in SAS are grouped into four-character sequences called dwords.

A primitive is a dword whose first character is K28.3 or K28.5 and whose remaining three characters are data characters with correct disparity.

Primitives are defined with both negative and positive starting RD (see 6.3.3). SAS defines primitives starting with K28.5 (see 7.2.5 and 7.2.6). SATA defines primitives starting with K28.3 and K28.5, which are used in SAS during STP connections (see 7.2.7).

A data dword is a dword that contains four data characters with correct disparity.

A dword containing an invalid character shall be considered an invalid dword.

6.5 Bit order

Dwords transmitted in a STP connection shall be transmitted in the bit order specified by SATA.

Dwords for other types of connections and outside of connections shall be transmitted in the bit order in figure 109.
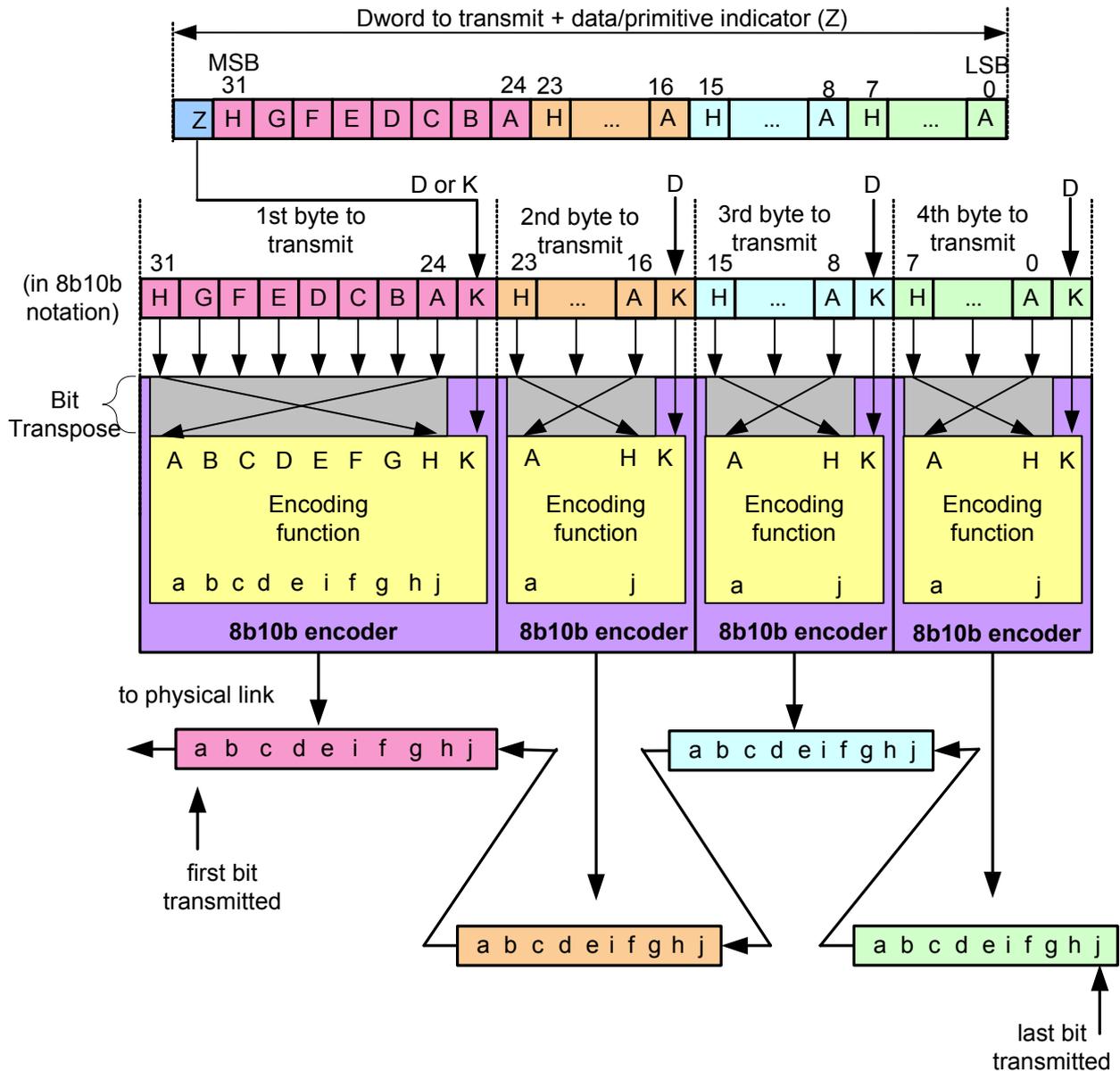
Figure 109 — SAS bit transmission logic

Figure 110 shows the SAS bit reception order.
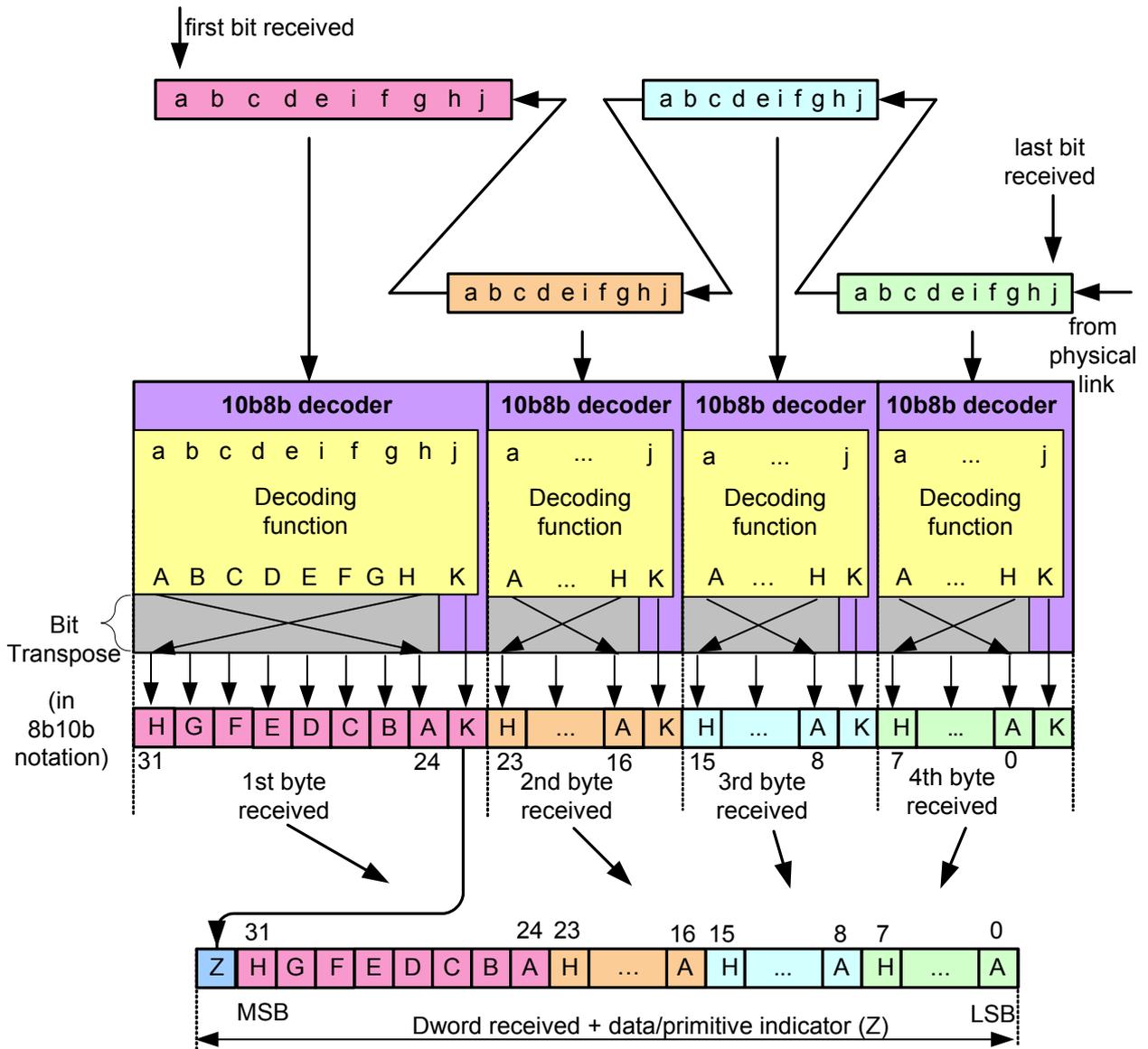
Figure 110 — SAS bit reception logic

6.6 Out of band (OOB) signals

6.6.1 OOB signals overview

Out of band (OOB) signals are low-speed signal patterns that do not appear in normal data streams. OOB signals consist of defined amounts of idle time followed by defined amounts of burst time. During the idle time, the physical link carries D.C. idle (see 3.1.38). During the burst time, the physical link carries signal transitions. The signals are differentiated by the length of idle time between the burst times.

SATA defines two OOB signals: COMINIT/COMRESET and COMWAKE. COMINIT and COMRESET are used in this standard interchangeably. Phys compliant with this standard identify themselves with an additional SAS-specific OOB signal called COMSAS.

Table 57 defines the timing specifications for OOB signals.

Table 57 — OOB signal timing specifications

Parameter	Minimum	Nominal	Maximum	Comments
OOB Interval (OOBI) [a]	666,600 ps	666,$\overline{6}$ ps	666,733 ps	The time basis for burst times and idle times used to create OOB signals. Based on 1,5 Gbit/s clock tolerance (see table 41 in 5.3.3).
COMSAS detect timeout	13,65 μs			The minimum time a receiver device shall allow to detect COMSAS after transmitting COMSAS. Derived from: OOBI x 512 x 40
[a] OOBI is different than UI(OOB) defined in SATA (e.g., SAS has tighter clock tolerance). This is a fixed value equal to the UI for G1, regardless of the actual transfer rate being used to create the burst time.				

6.6.2 Transmitting OOB signals

Table 58 describes the OOB signal transmitter requirements for the burst time, idle time, and negation times that comprise each OOB signal.

Table 58 — OOB signal transmitter device requirements

Signal	Burst time	Idle time	Negation time
COMWAKE	160 OOBI	160 OOBI	280 OOBI
COMINIT/RESET	160 OOBI	480 OOBI	800 OOBI
COMSAS	160 OOBI	1 440 OOBI	2 400 OOBI

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

1) transmit D.C. idle for an idle time; and
2) transmit an OOB burst consisting of ALIGN (0) primitives for a burst time.

The transmitter device shall then transmit D.C. idle for an OOB signal negation time.

The transmitter device shall use signal output levels during burst time and idle time as described in 5.3.6.5.

The ALIGN (0) primitives used in OOB signals should be at the generation 1 (G1) physical link rate (i.e., 1,5 Gbit/s). The ALIGN (0) primitives are only required to generate an envelope for the detection circuitry, as required for any signaling that may be A.C. coupled. If G2 ALIGN (0) primitives are used, the number of ALIGN (0) primitives doubles compared with G1 ALIGN (0) primitives.

When transmitting an OOB burst, if the phy supports SATA, the transmitter device shall transmit ALIGN (0) primitives at G1. If the phy does not support SATA, the transmitter device:

a) should transmit ALIGN (0) primitives at G1;
b) may transmit ALIGN (0) primitives at its lowest supported physical link rate if it is not able to transmit at G1; and
c) shall not transmit ALIGN (0) primitives at a physical link rate faster than its lowest supported physical link rate.

Figure 111 describes OOB signal transmission by the SP transmitter (see 6.8). The COMWAKE Transmitted, COMINIT Transmitted, and COMSAS Transmitted messages are sent to the SP state machine (see 6.8).
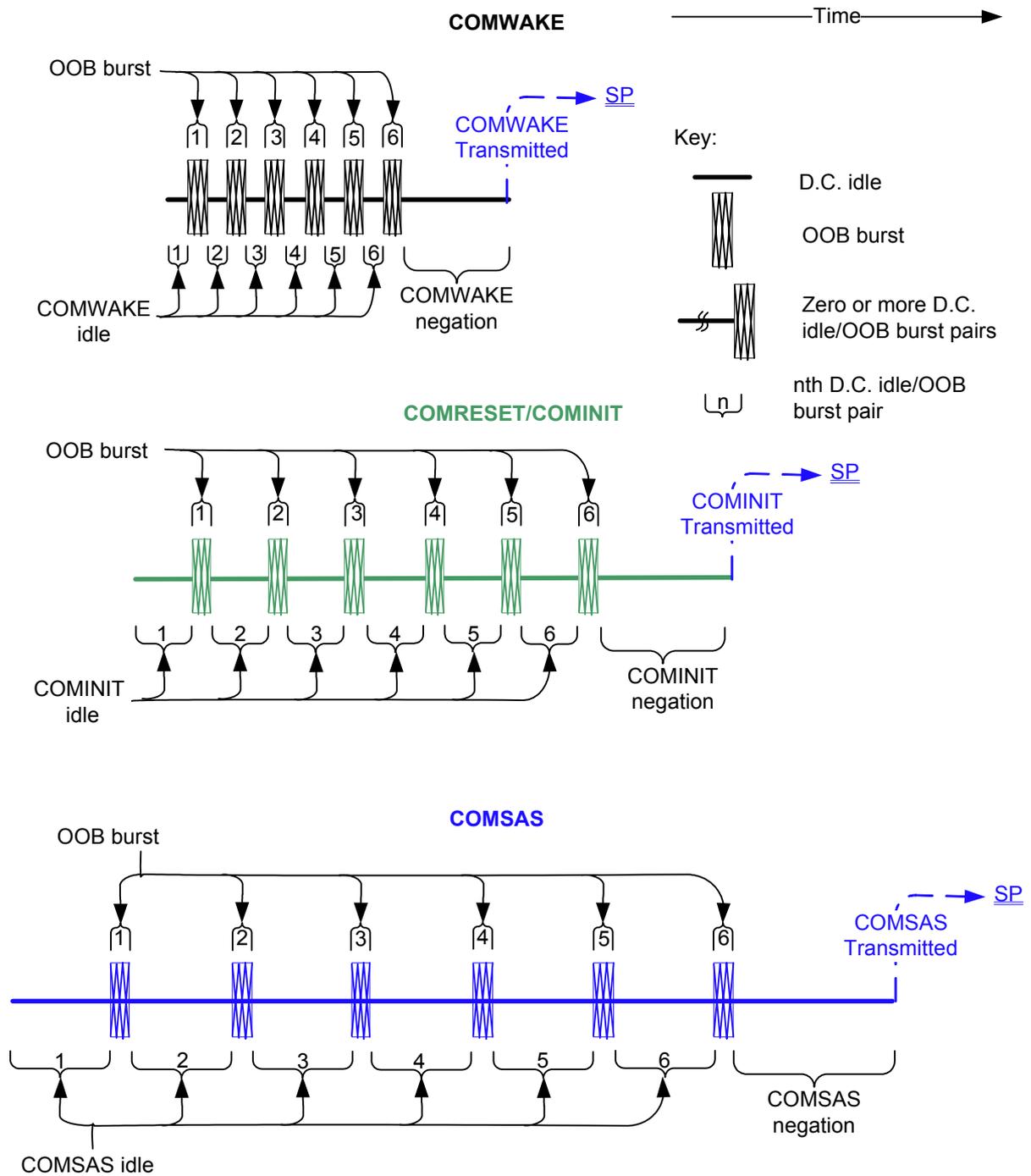
Note: D.C. idle is shown here as a neutral signal for visual clarity only.

Figure 111 — OOB signal transmission

6.6.3 Receiving OOB signals

Table 59 describes the OOB signal receiver device requirements for detecting burst times, assuming T_{burst} is the length of the detected burst time. The burst time is not used to distinguish between signals.

Table 59 — OOB signal receiver device burst time detection requirements

Signal	may detect	shall detect
COMWAKE	$T_{burst} \leq 100$ ns	$T_{burst} > 100$ ns
COMINIT/COMRESET	$T_{burst} \leq 100$ ns	$T_{burst} > 100$ ns
COMSAS	$T_{burst} \leq 100$ ns	$T_{burst} > 100$ ns

Table 60 describes the OOB signal receiver device requirements for detecting idle times, assuming T_{idle} is the length of the detected idle time.

Table 60 — OOB signal receiver device idle time detection requirements

Signal	may detect	shall detect	shall not detect
COMWAKE	55 ns $\leq T_{idle} < 175$ ns	101,3 ns $\leq T_{idle} \leq 112$ ns	$T_{idle} < 55$ ns or $T_{idle} \geq 175$ ns
COMINIT/ COMRESET	175 ns $\leq T_{idle} < 525$ ns	304 ns $\leq T_{idle} \leq 336$ ns	$T_{idle} < 175$ ns or $T_{idle} \geq 525$ ns
COMSAS	525 ns $\leq T_{idle} < 1\ 575$ ns	911,7 ns $\leq T_{idle} \leq 1\ 008$ ns	$T_{idle} < 525$ ns or $T_{idle} \geq 1\ 575$ ns

Table 61 describes the OOB signal receiver device requirements for detecting negation times, assuming T_{idle} is the length of the detected idle time.

Table 61 — OOB signal receiver device negation time detection requirements

Signal	shall detect
COMWAKE	$T_{idle} > 175$ ns
COMINIT/COMRESET	$T_{idle} > 525$ ns
COMSAS	$T_{idle} > 1\ 575$ ns

A receiver device shall detect an OOB signal after receiving four consecutive idle time/burst time pairs (see figure 112) while the SP_DWS state machine (see 6.9) has not achieved dword synchronization (see 6.8.4.9 and 6.8.5.8), and may but should not detect an OOB signal after receiving four consecutive idle time/burst time pairs while the SP_DWS state machine has achieved dword synchronization. It is not an error to receive more than four idle time/burst time pairs. A receiver device shall not detect the same OOB signal again until it has detected the corresponding negation time (e.g., a COMINIT negation time for a COMINIT) or has detected a different OOB signal (e.g., if a receiver device previously detected COMINIT, then receives four sets of COMWAKE idle times followed by burst times, the receiver device detects COMWAKE. The receiver device may then detect COMINIT again).

A SAS receiver device shall detect OOB bursts comprised of ALIGN (0) primitives 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 device supporting only 3,0 Gbit/s detects 1,5 Gbit/s based ALIGN (0) primitives, providing interoperability with a SAS transmitter device supporting both 1,5 Gbit/s and 3,0 Gbit/s).

> NOTE 12 - SAS transmitter devices compliant with future versions of this standard may not support transmitting OOB signals using the G1 physical link rate.

A SAS receiver device that supports SATA shall also detect OOB bursts comprised of D24.3 characters transmitted at 1,5 Gbit/s (see SATAII-PHY).

A SAS receiver device shall not check the characters used to form the OOB burst; only the frequency content of the burst matters.

Figure 112 describes SAS OOB signal detection by the SP receiver (see 6.8). The COMWAKE Detected, COMWAKE Completed, COMINIT Detected, COMSAS Detected, and COMSAS Completed messages are sent to the SP state machine (see 6.8) to indicate that an OOB signal has been partially or fully detected.
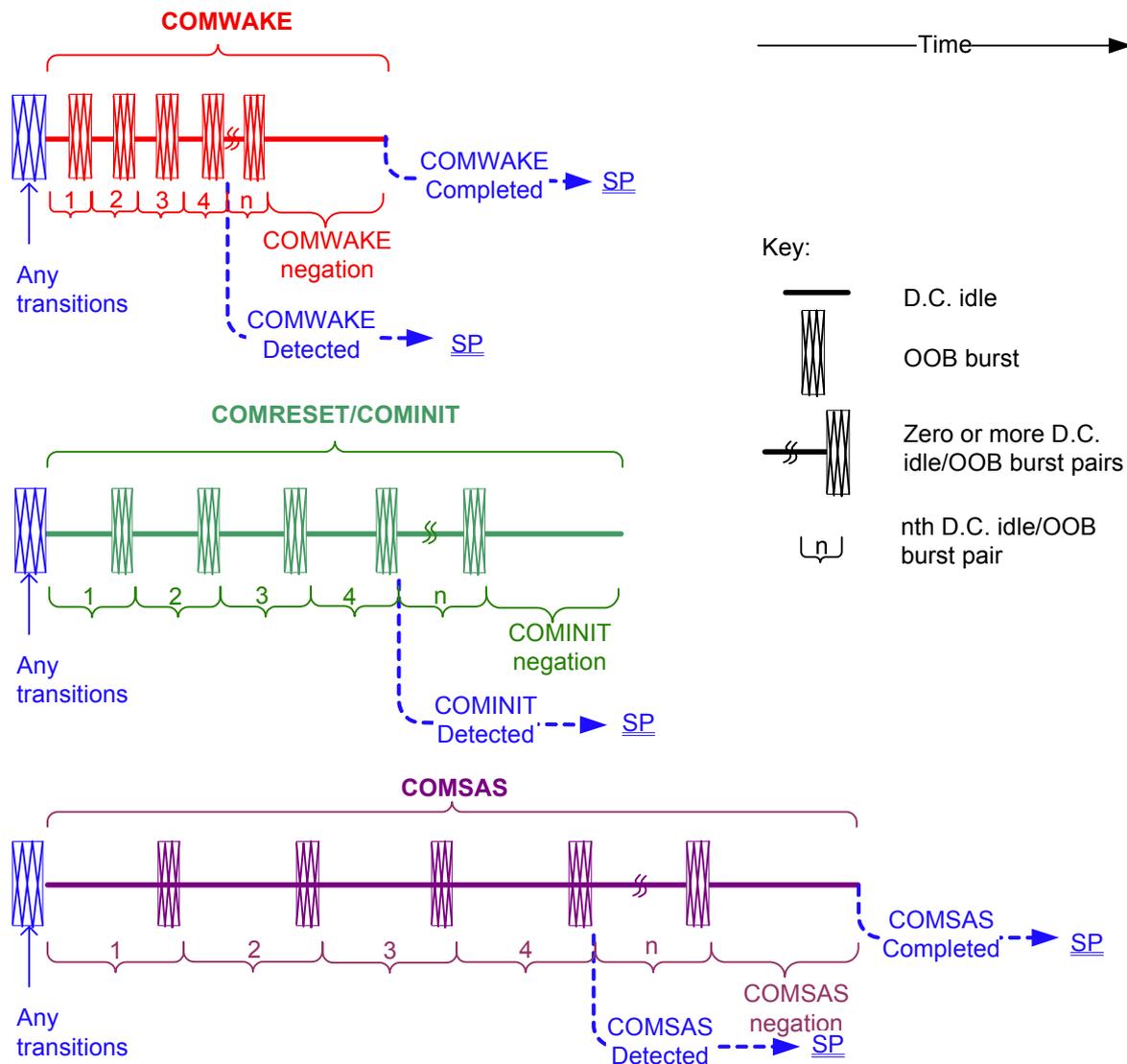
Note: D.C. idle is shown here as a neutral signal for visual clarity only.

Figure 112 — OOB signal detection

Expander devices shall not forward OOB signals. An expander device shall run the link reset sequence independently on each physical link.

6.6.4 Transmitting the SATA port selection signal

The SATA port selection signal shown in figure 113 causes the attached SATA port selector to select the attached phy (one of the port selector's host phys) as the active phy (see SATAII-PS).
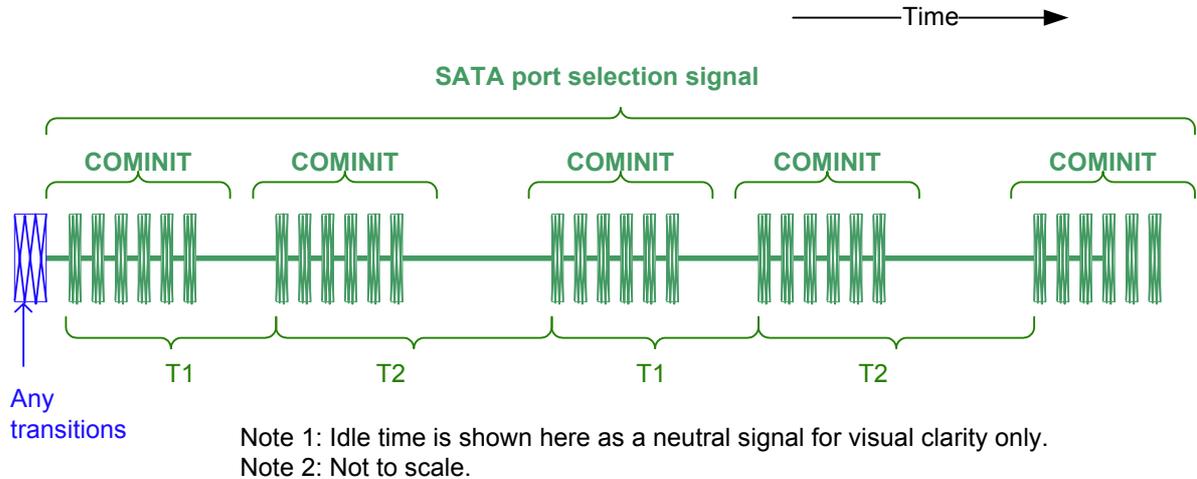
Figure 113 — SATA port selection signal

The SATA port selection signal shall be composed of 5 COMINIT signals, each starting a specified time interval, T1 or T2, as shown in figure 113, after the start of the OOB burst portion of the previous COMINIT signal. The values of T1 and T2 shall be as shown in table 62.

Table 62 — SATA port selection signal transmitter device requirements

Parameter	Time
T1	3×10^6 OOBI
T2	12×10^6 OOBI

See 6.8.6 and 10.4.3.10 for information on usage of the SATA port selection signal.

6.7 Phy reset sequences

6.7.1 Phy reset sequences overview

The phy reset sequence consists of:

1) an OOB sequence; and
2) a speed negotiation sequence.

The phy reset sequence shall only affect the phy, not the port or device containing the phy or other phys in the same port or device.

A phy shall originate a phy reset sequence after:

a) power on;
b) hard reset (i.e., receiving a HARD_RESET primitive sequence before an IDENTIFY address frame) (see 4.4.2);
c) management application layer request (see 6.8.1);
d) losing dword synchronization and not attempting to re-acquire dword synchronization (see 6.8.4.9 and 6.8.5.8);
e) Receive Identify Timeout timer expires (see 7.9.5); and
f) for expander phys, after a hot-plug timeout (see 6.7.5).

A SAS phy may originate a phy reset sequence after a hot-plug timeout (see 6.7.5).

After receiving a HARD_RESET primitive sequence before an IDENTIFY address frame, a phy should start the phy reset sequence within 250 ms.

Phys shall not originate a phy reset sequence until 10 ms have elapsed since the previous attempt at running a phy reset sequence (e.g., if a reply to COMINIT is not detected in an OOB sequence, or after a speed negotiation sequence fails).

Table 63 defines phy reset sequence timing parameters used by the SP state machine (see 6.8).

Table 63 — Phy reset sequence timing specifications

Parameter	Time	Comments
Hot-plug timeout	500 ms	The maximum time after which an expander phy shall retry an unsuccessful phy reset sequence, and after which a SAS initiator phy should retry an unsuccessful phy reset sequence (see 6.7.5).

6.7.2 SATA phy reset sequence

6.7.2.1 SATA OOB sequence

Figure 114 shows the SATA OOB sequence between a SATA host and SATA device. The SATA OOB sequence is defined by SATA (see ATA/ATAPI-7 V3 and SATAII-PHY for detailed requirements).
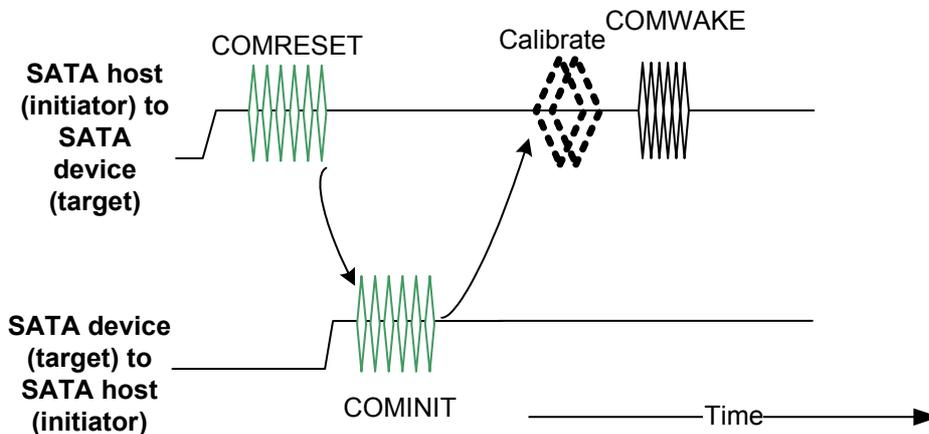
Figure 114 — SATA OOB sequence

6.7.2.2 SATA speed negotiation sequence

Figure 115 shows the speed negotiation sequence between a SATA host and SATA device. The SATA speed negotiation sequence is defined by SATA; see ATA/ATAPI-7 V3 and SATAII-PHY for detailed requirements.
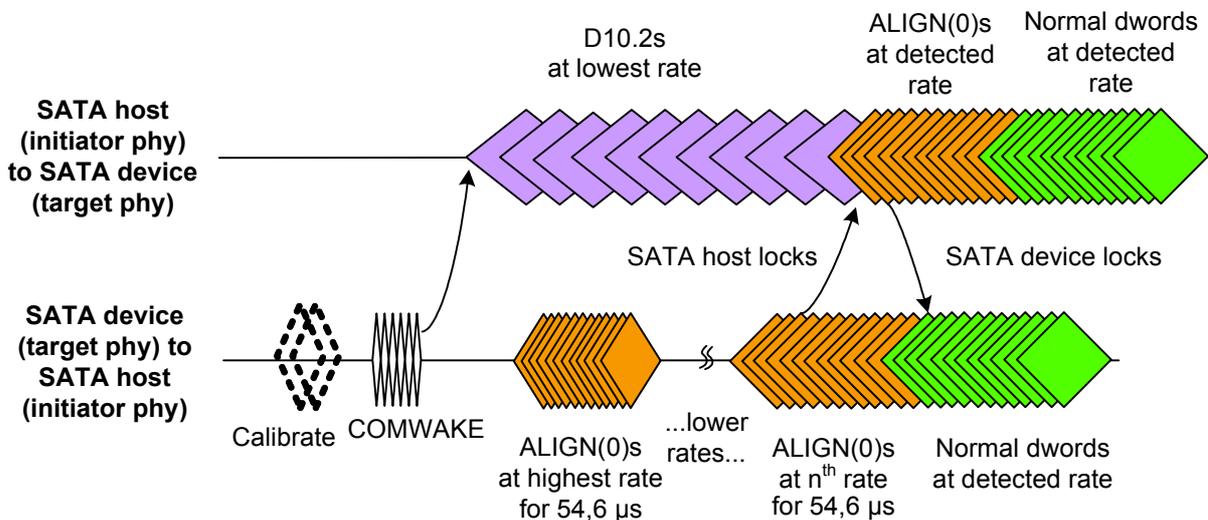
Figure 115 — SATA speed negotiation sequence

Table 64 defines SATA speed negotiation sequence timing parameters used by the SP state machine (see 6.8).

Table 64 — SATA speed negotiation sequence timing specifications

Parameter	Time	Comments
Await ALIGN timeout	1 310 720 OOBI	The minimum time during SATA speed negotiation that a phy shall allow for an ALIGN (0) to be received after detecting COMWAKE Completed.
COMWAKE response time	533 ns	The maximum time during SATA speed negotiation after detecting COMWAKE Completed before which a phy shall start transmitting D10.2 characters.

The transmitter device shall use SATA signal output levels during the SATA speed negotiation sequence as described in 5.3.6.5.

6.7.3 SAS to SATA phy reset sequence

SAS initiator phys and expander phys may support SATA (e.g., support being directly attached to a SATA device or a SATA port selector).

To initiate a phy reset sequence a phy shall:

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

The COMSAS identifies the phy as a SAS phy or expander phy instead of a SATA phy.

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

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 phy that a SATA phy is attached. As a result, the phy shall transmit COMWAKE and enter the SATA speed negotiation sequence (see 6.7.2.2).

Figure 116 shows a reset sequence between a SAS phy or expander phy (i.e., a phy compliant with this standard) and a SATA phy (i.e., a phy in a SATA device, defined by SATA). The two possible cases are presented. The first case is that the SATA phy ignores the COMSAS and provides no response within a COMSAS detect timeout. The second case is that the SATA phy misinterprets the COMSAS to be a COMRESET and responds with a COMINIT.

The SP state machine treats these two cases the same, and determines that a SATA phy is attached after a COMSAS detect timeout. The SATA speed negotiation sequence shall be entered after COMWAKE.

X : Power on

Time 0: OOB sequence begins
Time z: Speed negotiation sequence begins
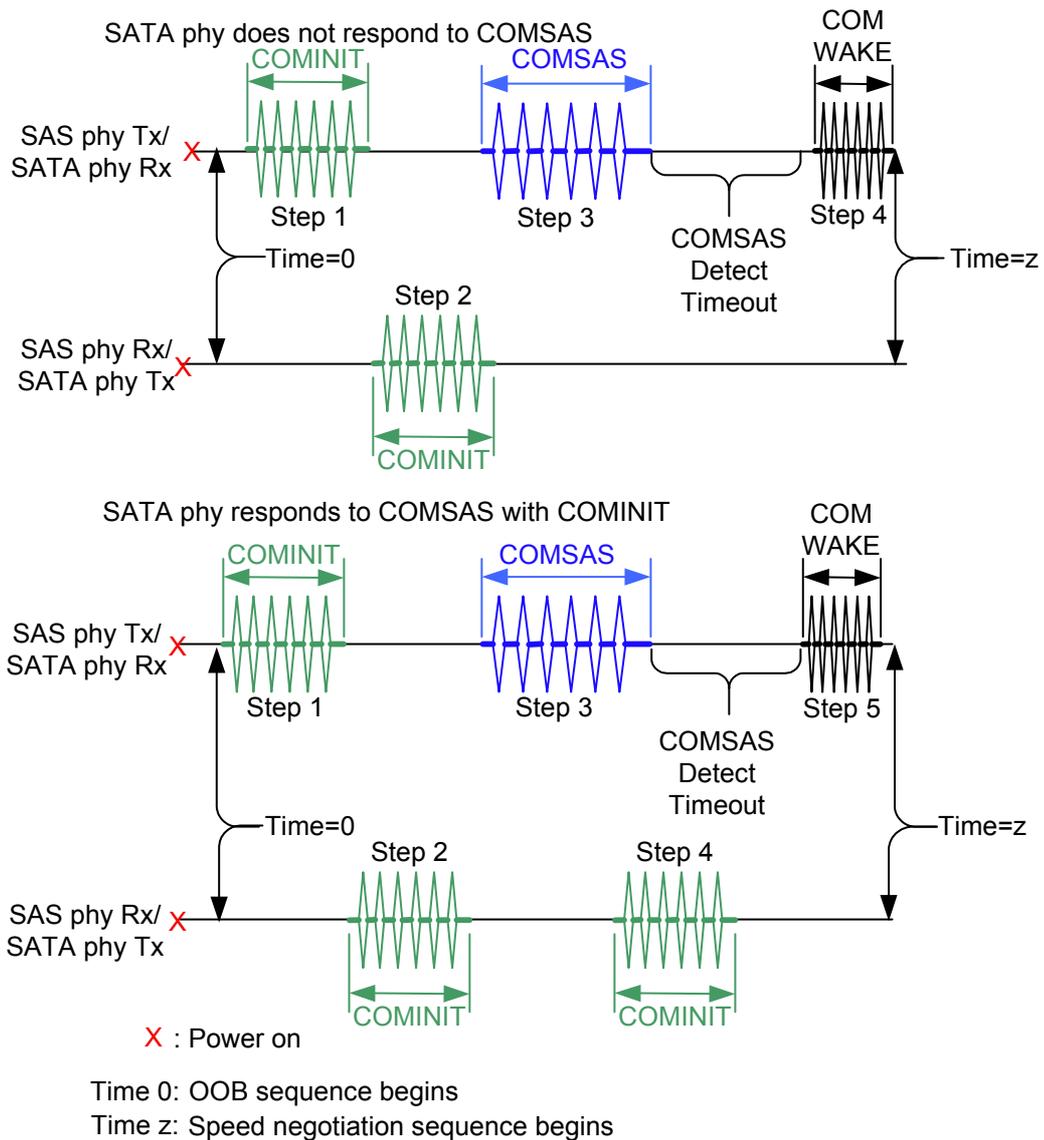
Figure 116 — SAS to SATA OOB sequence

6.7.4 SAS to SAS phy reset sequence

6.7.4.1 SAS OOB sequence

To initiate a SAS OOB sequence a phy shall transmit a COMINIT.

On receipt of a COMINIT a phy shall either:

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

On receipt of a COMSAS, if the receiving phy has not yet transmitted a COMSAS, the phy shall transmit a COMSAS.

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

A phy shall distinguish between COMINIT and COMSAS and continue with a SAS speed negotiation sequence (see 6.7.4.2) after completing the SAS OOB sequence.

Figure 117 shows several different SAS OOB sequences between phy A and phy B, with phy A starting the SAS OOB sequence at the same time as phy B, before phy B, and before phy B powers on.
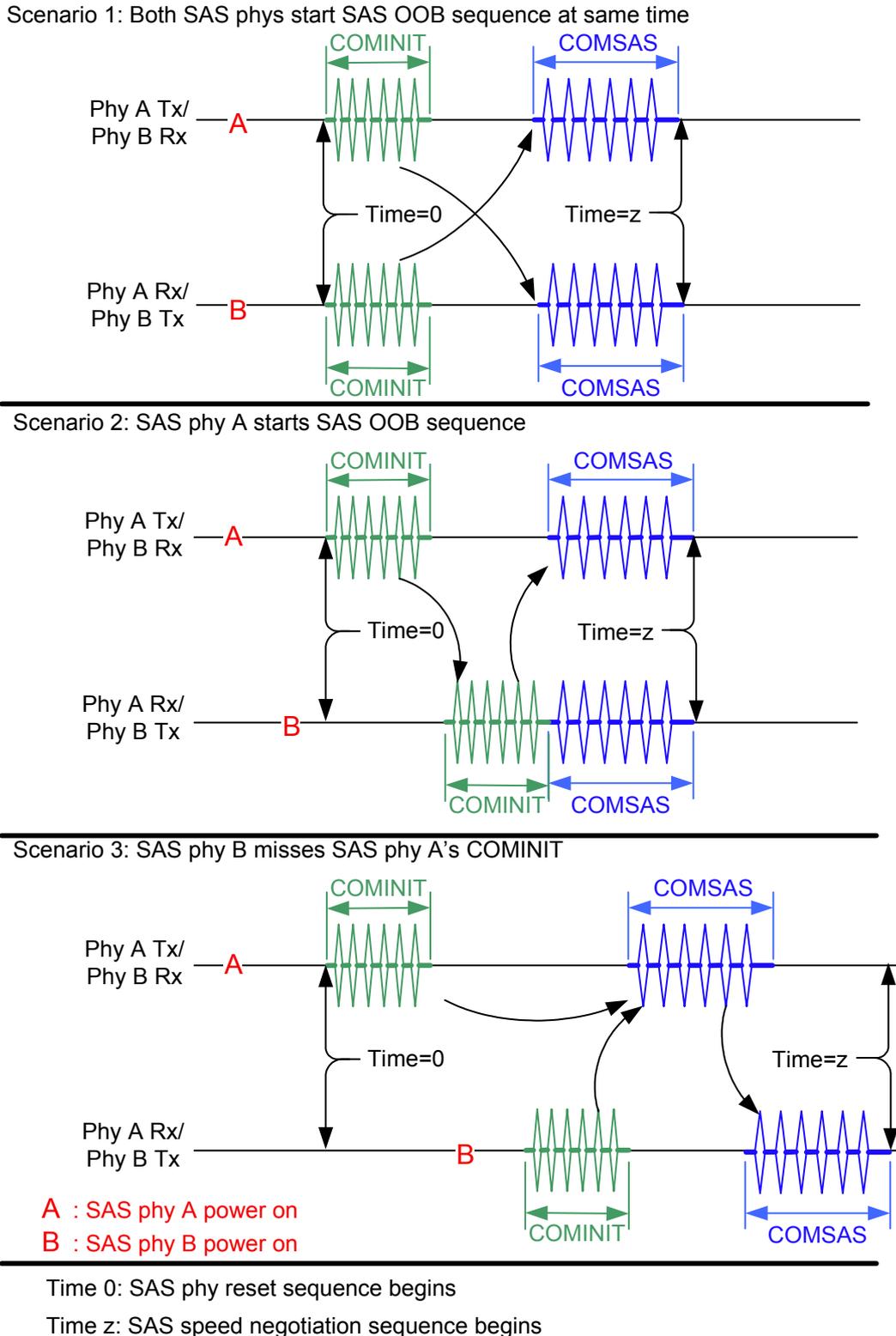
Figure 117 — SAS to SAS OOB sequence

6.7.4.2 SAS speed negotiation sequence

6.7.4.2.1 SAS speed negotiation sequence overview

The SAS speed negotiation sequence is a peer-to-peer negotiation technique that does not assume initiator and target (i.e., host and device) roles. The sequence consists of a set of speed negotiation windows (see 6.7.4.2.2) for

each physical link rate, starting with 1,5 Gbit/s, then 3,0 Gbit/s, then the next physical link rate. The length of the speed negotiation sequence (i.e., the number of speed negotiation windows) is determined by the number of physical link rates supported by the phys.

The transmitter device shall use SAS signal output levels during the SAS speed negotiation sequence as described in 5.3.6.5.

6.7.4.2.2 Speed negotiation window

Figure 118 defines the speed negotiation window, including:

- a) speed negotiation window time;
- b) speed negotiation window rate;
- c) rate change delay time (RCDT);
- d) speed negotiation transmit time (SNTT); and
- e) speed negotiation lock time (SNLT).
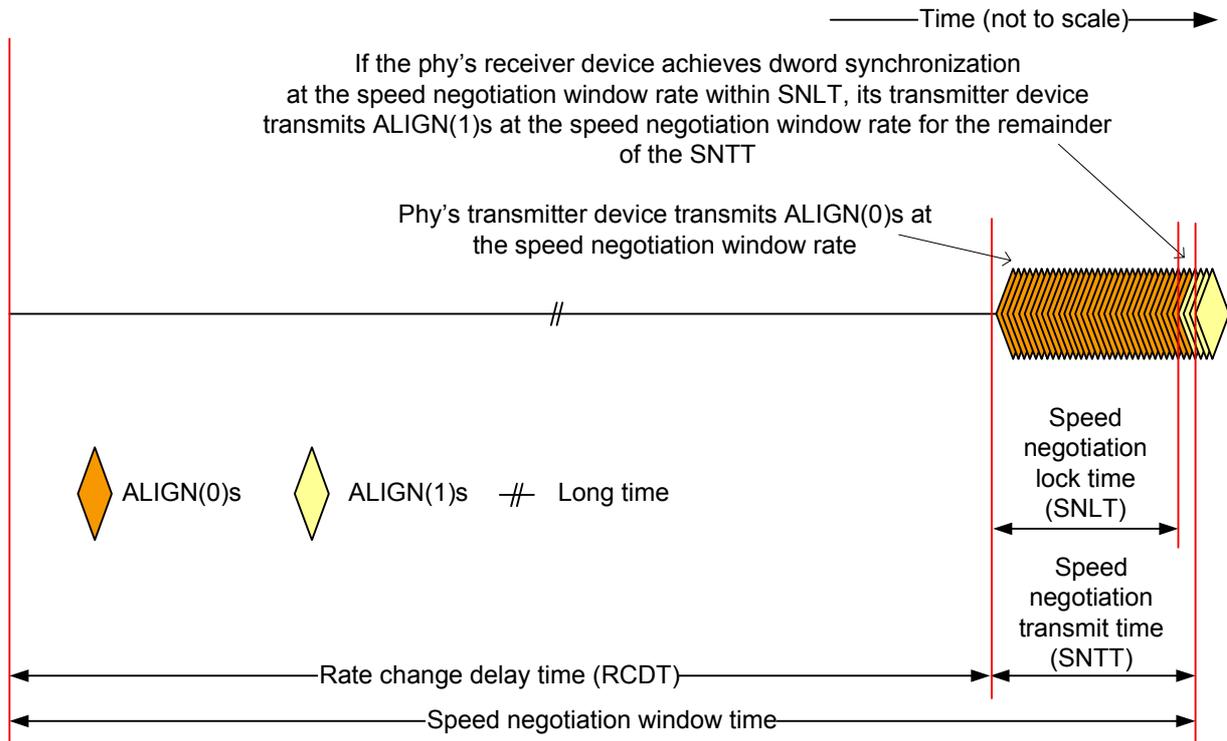
Figure 118 — SAS speed negotiation window

Table 65 defines the timing specifications for the SAS speed negotiation sequence.

Table 65 — SAS speed negotiation sequence timing specifications

Parameter	Time	Comments
Rate change delay time (RCDT)	750 000 OOBI	The time the transmitter device shall transmit D.C. idle between rates during speed negotiation.
Speed negotiation transmit time (SNTT)	163 840 OOBI	The time during which ALIGN (0) or ALIGN (1) is transmitted at each physical link rate during the speed negotiation sequence. Derived from: OOBI x 4 096 x 40.
Speed negotiation lock time (SNLT)	153 600 OOBI	The maximum time during the speed negotiation window for a transmitter device to reply with ALIGN (1). Derived from: OOBI x 3 840 x 40
Speed negotiation window time	913 840 OOBI	The duration of a speed negotiation window. Derived from: RCDT + SNTT.

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

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

If the phy supports the speed negotiation window 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 time). If the phy does not achieve dword synchronization within the SNLT, it shall continue transmitting ALIGN (0) primitives for the remainder of the SNTT (i.e., the remainder of the speed negotiation window time).

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.

6.7.4.2.3 SAS speed negotiation sequence

The SAS speed negotiation sequence consists of a set of speed negotiation windows (see 6.7.4.2.2) for each physical link rate in this order:

1) G1 (i.e., 1,5 Gbit/s);
2) G2 (i.e., 3,0 Gbit/s);
3) G3, if needed;
4) etc.

The number of speed negotiation windows is determined by the number of physical link rates supported by the phys.

A phy shall participate in all speed negotiation windows:

a) up to its highest supported physical link rate plus one (e.g., a phy supporting G2 participates in G1, G2, and G3 speed negotiation windows). This is the maximum speed negotiation window; 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.

Once a phy reaches its limit, if the phy detected a valid physical link rate in the previous speed negotiation window, it shall participate in a final speed negotiation window using the highest previously successful physical link rate.

Figure 119 shows speed negotiation between a phy A that supports G1 through G3 and a phy B that only supports G2. Both phys run:

1) the G1 speed negotiation window, supported by phy A but not by phy B;
2) the G2 speed negotiation window, supported by both phys; and

3) the G3 speed negotiation window, supported by phy A but not by phy B.
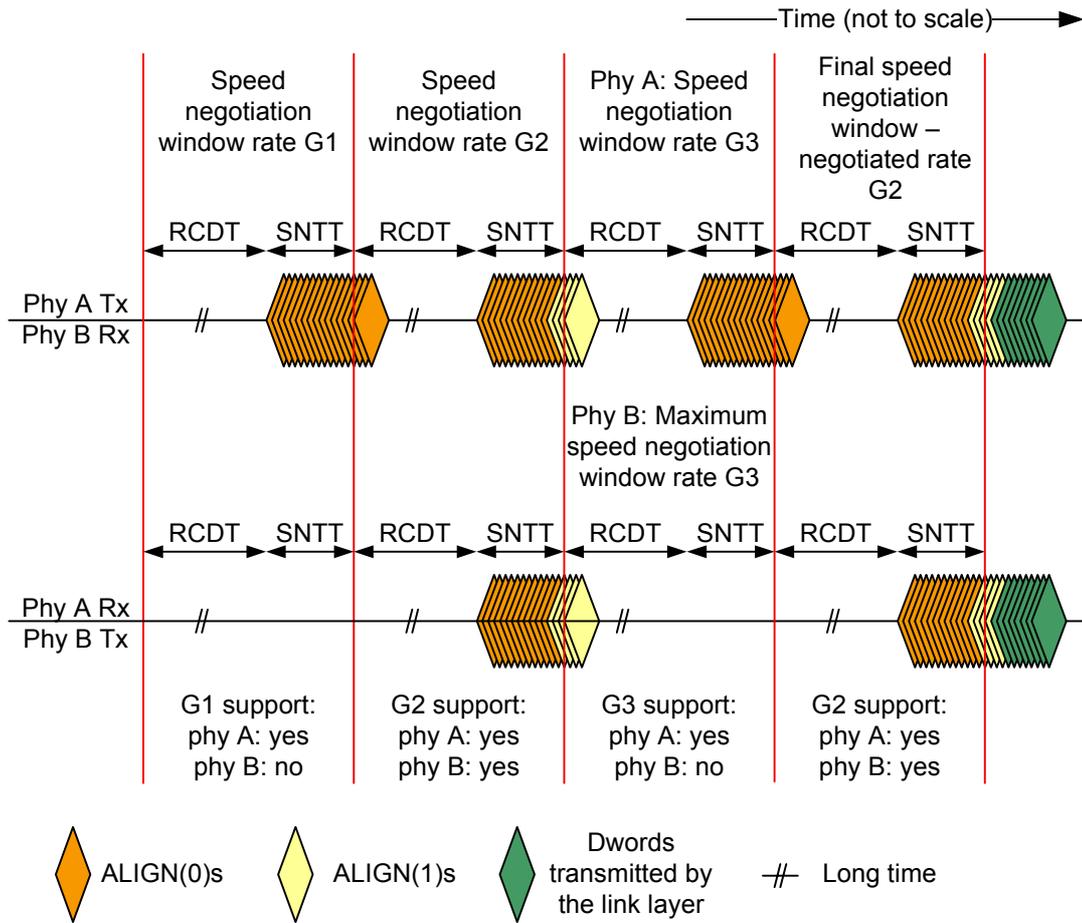
Figure 119 — SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G2 only)

Phy A reaches its limit because it has run a speed negotiation window that does not detect a valid physical link rate (i.e., G3) after having detected a valid physical link rate in a previous speed negotiation window (i.e., G2). Phy B reaches its limit because it has reached its highest support physical link rate plus one (i.e., G3). Both phys select G2 for the final speed negotiation window to establish the negotiated physical link rate.

If the phy does not achieve dword synchronization during the final speed negotiation window, the SAS speed negotiation sequence fails. This is called a phy reset problem and may be counted and reported in the PHY RESET PROBLEM COUNT field in the SMP REPORT PHY ERROR LOG page (see 10.4.3.6) and the REPORT PHY ERROR LOG log page (see 10.2.8.1).

Figure 120 shows a speed negotiation sequence where phy B does not achieve dword synchronization during the final speed negotiation window. If this occurs, the handshake is not complete and the phy reset sequence is retried.
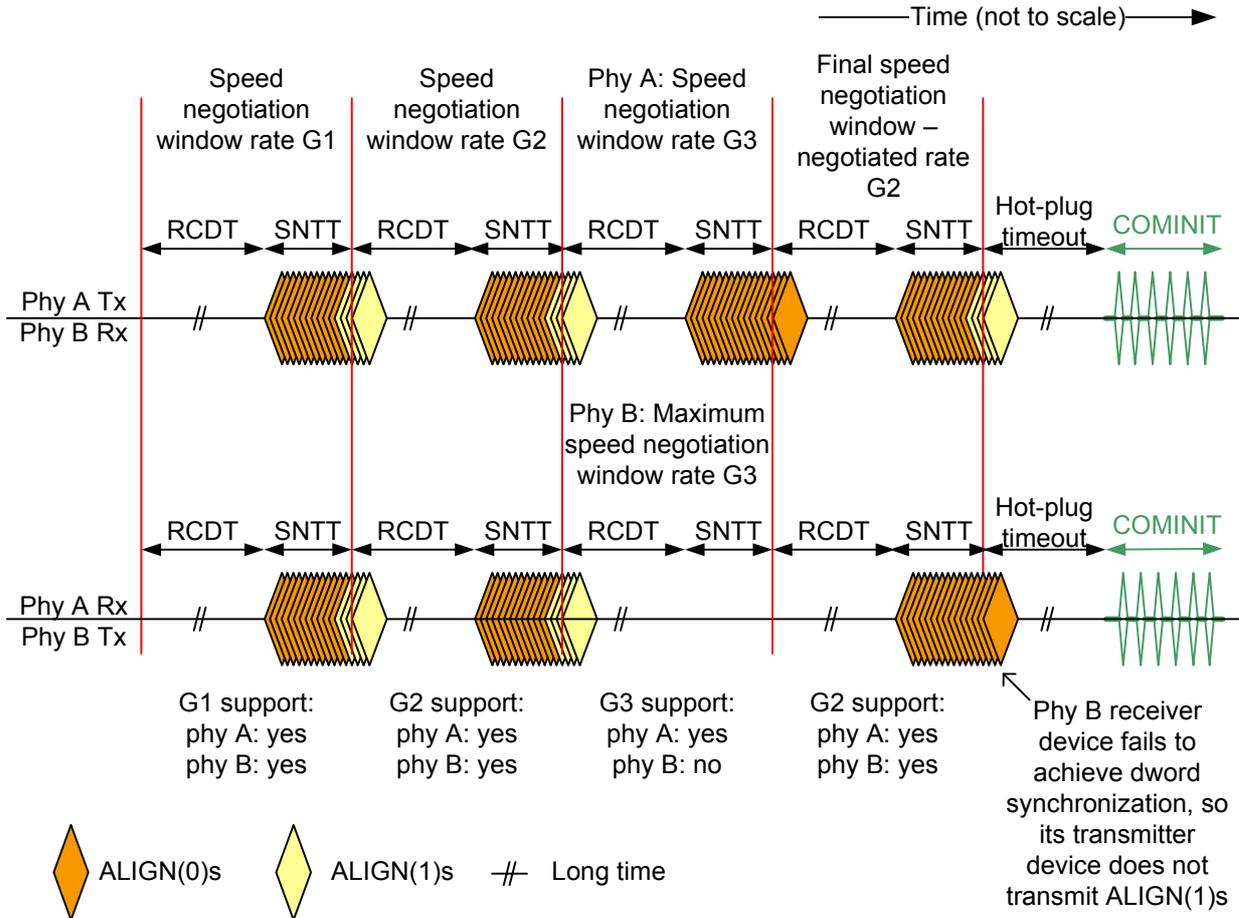
Figure 120 — SAS speed negotiation sequence - phy reset problem

For more examples of speed negotiations between phys that support various speeds, see Annex C.

6.7.5 Phy reset sequence after devices are attached

Since SATA and SAS signal cable connectors do not include power lines, it is not possible to detect the physical insertion of the signal cable connector onto a plug. Non-cabled environments may similarly not have a way to detect physical insertion of a device. As a result, every time a phy reset sequence is originated:

a) expander phys that are enabled but not active shall originate a new phy reset sequence repeatedly, with no more than a hot-plug timeout (see table 63) between each attempt, until a speed negotiation sequence completes successfully;

b) SAS initiator phys should originate a new phy reset sequence after every hot-plug timeout; and

c) SAS target phys should not originate a new phy reset sequence after their first attempt.

Figure 121 shows how two phys complete the phy reset sequence if the phys are not attached at power on. In this example, phy A and phy B are attached some time before phy B's second hot-plug timeout occurs. Phy B's OOB detection circuitry detects a COMINIT after the attachment, and therefore phy B transmits COMSAS, since it has both transmitted and received a COMINIT. Upon receiving COMSAS, phy A transmits its own COMSAS. The SAS speed negotiation sequence follows.
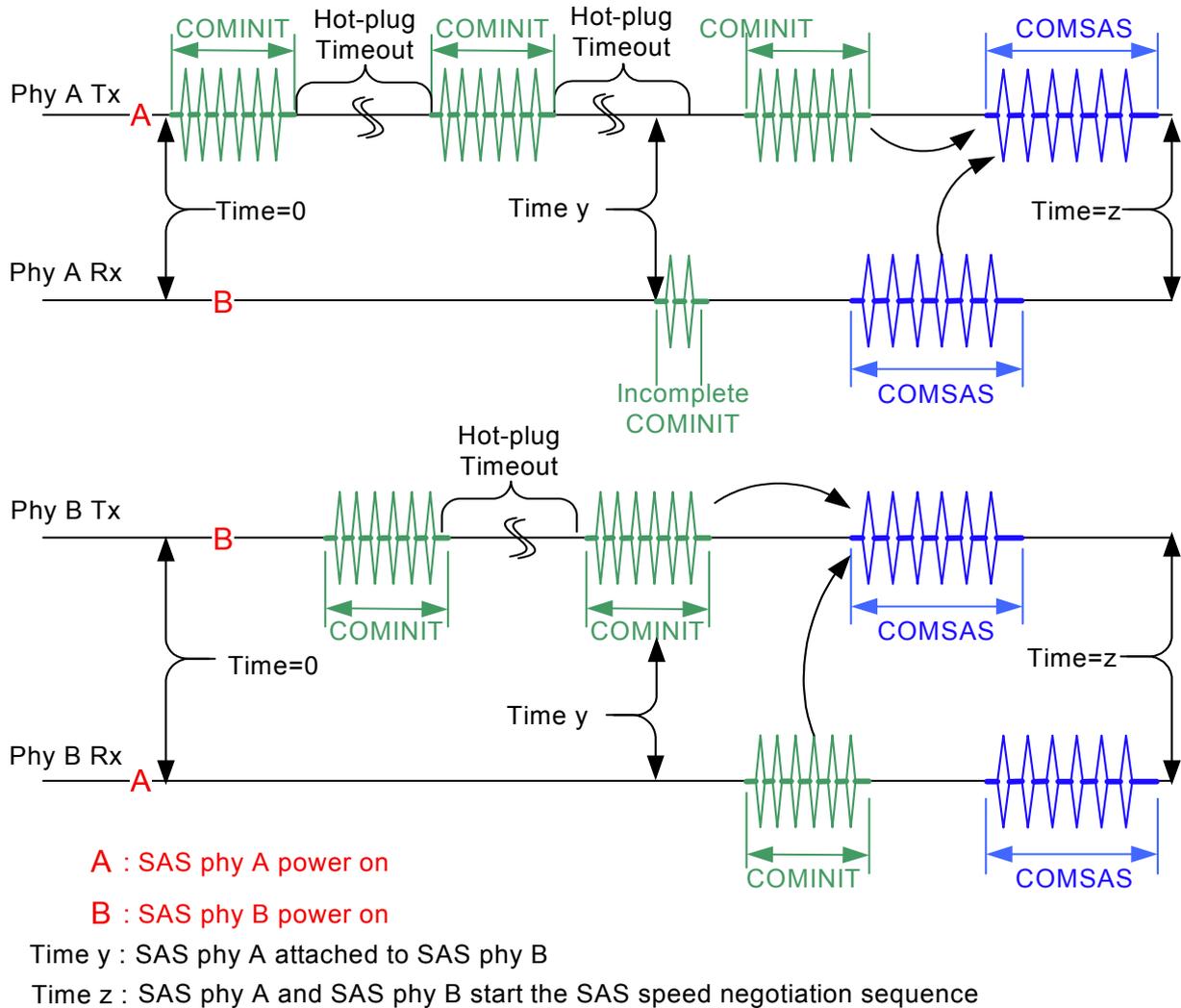
Figure 121 — Hot-plug and the phy reset sequence

6.8 SP (phy layer) state machine

6.8.1 SP state machine overview

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

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

This state machine consists of the following states:

   a) SP0:OOB_COMINIT (see 6.8.3.2)(initial state);
   b) SP1:OOB_AwaitCOMX (see 6.8.3.3);
   c) SP2:OOB_NoCOMSASTimeout;
   d) SP3:OOB_AwaitCOMINIT_Sent (see 6.8.3.5);
   e) SP4:OOB_COMSAS (see 6.8.3.6);
   f) SP5:OOB_AwaitCOMSAS_Sent (see 6.8.3.7);
   g) SP6:OOB_AwaitNoCOMSAS (see 6.8.3.8);
   h) SP7:OOB_AwaitCOMSAS (see 6.8.3.9);

    i)   SP8:SAS_Start (see 6.8.4.2);
    j)   SP9:SAS_RateNotSupported (see 6.8.4.3);
    k)   SP10:SAS_AwaitALIGN (see 6.8.4.4);
    l)   SP11:SAS_AwaitALIGN1 (see 6.8.4.5);
    m) SP12:SAS_AwaitSNW (see 6.8.4.6);
    n)   SP13:SAS_Pass (see 6.8.4.7);
    o)   SP14 SAS_Fail (see 6.8.4.8);
    p)   SP15:SAS_PHY_Ready (see 6.8.4.9);
    q)   SP16:SATA_COMWAKE (see 6.8.5.2);
    r)   SP17:SATA_AwaitCOMWAKE (see 6.8.5.3);
    s)   SP18:SATA_AwaitNoCOMWAKE (see 6.8.5.4);
    t)   SP19:SATA_AwaitALIGN (see 6.8.5.5);
    u)   SP20:SATA_AdjustSpeed (see 6.8.5.6);
    v)   SP21:SATA_Transmit_ALIGN (see 6.8.5.7);
    w) SP22:SATA_PHY_Ready (see 6.8.5.8);
    x)   SP23:SATA_PM_Partial (see 6.8.5.9);
    y)   SP24:SATA_PM_Slumber (see 6.8.5.10);
    z)   SP25:SATA_PortSel (see 6.8.6.2); and
  aa) SP26:SATA_SpinupHold (see 6.8.7.2).

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

    a)   a power on;
    b)   a hard reset;
    c)   receiving a Management Reset request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET in an expander device); or
    d)   receiving a Disable Phy request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of DISABLE in an expander device).

The SP state machine shall maintain a MgmtReset state machine variable to determine whether a Management Reset request has been received. Any SP state that receives a Management Reset request shall set the MgmtReset state machine variable to one before making a transition to the SP7:OOB_AwaitCOMSAS state (see 6.8.3.2). Any SP state that receives a power on, or a hard reset shall set the MgmtReset state machine variable to zero before making a transition to the SP7:OOB_AwaitCOMSAS state.

If the phy supports SATA port selectors, the SP state machine shall maintain a COMWAKE_Received state machine variable to indicate whether a COMWAKE detected message was received in the SP0:OOB_COMINIT state or the SP1:OOB_AwaitCOMX state since the last time the SP0:OOB_COMINIT state was entered, and the SP state machine shall transition to the SP25:SATA_PortSel state whenever it receives a Transmit SATA Port Selection Signal request.

The SP state machine sends the following messages to the SP_DWS state machine (see 6.9):

    a)   Start DWS; and
    b)   Stop DWS.

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

    a)   DWS Lost; and
    b)   DWS Reset.

The SP state machine shall maintain the timers listed in table 66.

Table 66 — SP state machine timers

Timer	Initial value
COMSAS Detect Timeout timer	COMSAS detect timeout (see table 57 in 6.6.1)
Await ALIGN Timeout timer	Await ALIGN timeout (see table 64 in 6.7.2.2)
Hot-Plug Timeout timer	Hot plug timeout (see table 63 in 6.7.1)
RCDT timer	RCDT (see table 65 in 6.7.4.2)
SNLT timer	SNLT (see table 65 in 6.7.4.2)
SNTT timer	SNTT (see table 65 in 6.7.4.2)

6.8.2 SP transmitter and receiver

The SP transmitter transmits OOB signals and dwords on the physical link based on messages from the SP state machine (see 6.8).

The SP transmitter receives the following messages from the SP state machine:

  a) Transmit COMINIT;
  b) Transmit COMSAS;
  c) Transmit COMWAKE;
  d) Transmit SATA Port Selection Signal;
  e) Transmit D10.2;
  f) Set Rate (Physical Link Rate); and
  g) Transmit ALIGN with an argument indicating the specific type (e.g., Transmit ALIGN (0)).

When not otherwise instructed, the SP transmitter transmits D.C. idle.

The SP transmitter shall complete any physical link rate change requested with the Set Rate message within RCDT (see table 65 in 6.7.4.2).

.The SP transmitter sends the following messages to the SP state machine:

  a) COMINIT Transmitted;
  b) COMSAS Transmitted;
  c) COMWAKE Transmitted; and
  d) SATA Port Selection Signal Transmitted.

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

The SP receiver sends the following messages to the SP state machine:

  a) COMINIT Detected;
  b) COMSAS Detected;
  c) COMWAKE Detected;
  d) COMSAS Completed;
  e) COMWAKE Completed;
  f) ALIGN Received with an argument indicating the specific type (e.g., ALIGN Received (0)); and
  g) Dword Received.

The ALIGN Received and Dword Received messages are only sent when the SP_DWS state machine has achieved dword synchronization.

For SATA speed negotiation, the ALIGN Received (0) message includes an argument containing the physical link rate at which the ALIGN (0) primitives were detected. For SAS speed negotiation, only ALIGNs at the physical link rate specified by the last Set Rate message received by the SP transmitter cause ALIGN Received messages.

6.8.3 OOB sequence states

6.8.3.1 OOB sequence states overview

Figure 122 shows the OOB sequence states. These states are indicated by state names with a prefix of OOB.

Figure 122 — SP (phy layer) state machine - OOB sequence states

6.8.3.2 SP0:OOB_COMINIT state

6.8.3.2.1 State description

This state is the initial state for this state machine.

This state machine waits for receipt of a COMINIT Transmitted message and/or a COMINIT Detected message.

Upon entry into this state, this state shall:

    a)   set the COMWAKE_Received state machine variable to zero;
    b)   send a Stop DWS message to the SP_DWS state machine; and
    c)   send a Phy Layer Not Ready confirmation to the link layer.

If this state was entered due to power on, the phy shall set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero.

If this state was not entered because of a Disable Phy request, this state shall send a Transmit COMINIT message to the SP transmitter upon entry into this state.

If this state was entered because of a Disable Phy request, this state shall ignore COMINIT Detected messages until this state is re-entered due to a power on, hard reset, or Management Reset request.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

    a)   set the COMWAKE_Received state machine variable to one; and
    b)   if the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to zero:
        A)   set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to one; and
        B)   send a SATA Port Selector Change confirmation to the link layer.

6.8.3.2.2 Transition SP0:OOB_COMINIT to SP1:OOB_AwaitCOMX

This transition shall occur if this state receives a COMINIT Transmitted message and has not received a COMINIT Detected message.

6.8.3.2.3 Transition SP0:OOB_COMINIT to SP3:OOB_AwaitCOMINIT_Sent

This transition shall occur if this state receives a COMINIT Detected message and has not received a COMINIT Transmitted message.

6.8.3.2.4 Transition SP0:OOB_COMINIT to SP4:OOB_COMSAS

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

6.8.3.3 SP1:OOB_AwaitCOMX state

6.8.3.3.1 State description

Upon entry into this state, the Hot-Plug Timeout timer shall be initialized and started if this phy is:

    a)   an expander phy; or
    b)   an initiator phy or target phy implementing the Hot-Plug Timeout timer.

If the phy supports SATA port selectors and this state receives a COMWAKE Detected message, this state shall:

    a)   set the COMWAKE_Received state machine variable to one; and
    b)   if the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to zero:
        A)   set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to one; and
        B)   send a SATA Port Selector Change confirmation to the link layer.

6.8.3.3.2 Transition SP1:OOB_AwaitCOMX to SP0:OOB_COMINIT

This transition shall occur if the Hot-Plug Timeout timer expires.

If the COMWAKE_Received state machine variable is set to zero and the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to one, the state machine shall, before the transition:

    a)   set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero; and
    b)   send a SATA Port Selector Change confirmation to the link layer.

6.8.3.3.3 Transition SP1:OOB_AwaitCOMX to SP4:OOB_COMSAS

This transition shall occur after receiving either a COMINIT Detected message or a COMSAS Detected message. If COMSAS Detected was received, this transition shall include a COMSAS Detected argument.

If the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to one, the state machine shall, before the transition:

a) set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero; and
b) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.4 SP2:OOB_NoCOMSASTimeout state

6.8.3.4.1 State description

Upon entry into this state, the Hot-Plug Timeout timer shall be initialized and started if this phy is:

a) this phy is an expander phy; or
b) this phy is an initiator phy or target phy implementing the Hot-Plug Timeout timer.

6.8.3.4.2 Transition SP2:OOB_NoCOMSASTimeout to SP0:OOB_COMINIT

This transition shall occur if the Hot-Plug Timeout timer expires.

6.8.3.4.3 Transition SP2:OOB_NoCOMSASTimeout to SP4:OOB_COMSAS

This transition shall occur after receiving a COMINIT Detected message.

6.8.3.5 SP3:OOB_AwaitCOMINIT_Sent state

6.8.3.5.1 State description

This state waits for a COMINIT Transmitted message.

If the phy supports SATA port selectors, the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to zero, and this state receives a COMWAKE Detected message, this state shall:

a) set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to one; and
b) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.5.2 Transition SP3:OOB_AwaitCOMINIT_Sent to SP4:OOB_COMSAS

This transition shall occur after receiving a COMINIT Transmitted message.

6.8.3.6 SP4:OOB_COMSAS state

6.8.3.6.1 State description

Upon entry into this state, this state shall send a Transmit COMSAS message to the SP transmitter.

This state waits for receipt of a COMSAS Transmitted message and/or a COMSAS Detected message.

If the phy supports SATA port selectors, the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to zero, and this state receives a COMWAKE Detected message, this state shall:

a) set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to one; and
b) send a SATA Port Selector Change confirmation to the link layer.

6.8.3.6.2 Transition SP4:OOB_COMSAS to SP5:OOB_AwaitCOMSAS_Sent

This transition shall occur if this state receives a COMSAS Detected message or this state was entered with a COMSAS Detected argument, and this state has not received a COMSAS Transmitted message.

If the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to one, the state machine shall set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero and send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.6.3 Transition SP4:OOB_COMSAS to SP6:OOB_AwaitNoCOMSAS

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

If the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is set to one, the state machine shall set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero and send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.6.4 Transition SP4:OOB_COMSAS to SP7:OOB_AwaitCOMSAS

This transition shall occur if this state receives a COMSAS Transmitted message and has not received a COMSAS Detected message.

6.8.3.7 SP5:OOB_AwaitCOMSAS_Sent state

6.8.3.7.1 State description

This state waits for receipt of a COMSAS Transmitted message.

6.8.3.7.2 Transition SP5:OOB_AwaitCOMSAS_Sent to SP6:OOB_AwaitNoCOMSAS

This transition shall occur after receiving a COMSAS Transmitted message.

If this state received a COMSAS Completed message, it shall include a COMSAS Completed argument with the transition.

6.8.3.8 SP6:OOB_AwaitNoCOMSAS state

6.8.3.8.1 State description

This state machine waits for a COMSAS Completed message, which indicates that COMSAS has been completely received.

6.8.3.8.2 Transition SP6:OOB_AwaitNoCOMSAS to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.3.8.3 Transition SP6:OOB_AwaitNoCOMSAS to SP8:SAS_Start

This transition shall occur after receiving a COMSAS Completed message, or shall occur if a COMSAS Completed argument was received in the transition.

6.8.3.9 SP7:OOB_AwaitCOMSAS state

6.8.3.9.1 State description

Upon entry into this state the COMSAS Detect Timeout timer shall be initialized and started.

6.8.3.9.2 Transition SP7:OOB_AwaitCOMSAS to SP2:OOB_NoCOMSASTimeout

This transition shall occur if the phy does not support SATA and the COMSAS Detect Timeout timer expires.

The state machine shall set the MgmtReset state machine variable to zero before the transition.

6.8.3.9.3 Transition SP7:OOB_AwaitCOMSAS to SP6:OOB_AwaitNoCOMSAS

This transition shall occur after receiving a COMSAS Detected message.

The state machine shall set the MgmtReset state machine variable to zero before the transition.

The state machine shall set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero. If the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response was set to one prior to this transition, the state machine shall send a SATA Port Selector Change confirmation to the link layer before the transition.

6.8.3.9.4 Transition SP7:OOB_AwaitCOMSAS to SP16:SATA_COMWAKE

This transition shall occur if:

   a) the phy supports SATA;
   b) the COMSAS Detect Timeout timer expires; and
       A) the MgmtReset state machine variable is set to one; or

B) the phy does not implement SATA spinup hold.

The state machine shall set the MgmtReset state machine variable to zero before the transition.

6.8.3.9.5 Transition SP7:OOB_AwaitCOMSAS to SP26:SATA _SpinupHold

This transition shall occur if:

a) the phy supports SATA;
b) the COMSAS Detect Timeout timer expires;
c) the phy implements SATA spinup hold; and
d) the MgmtReset state machine variable is set to zero.

6.8.4 SAS speed negotiation states

6.8.4.1 SAS speed negotiation states overview

Figure 123 shows the SAS speed negotiation states, in which the phy has detected that it is attached to a SAS phy or expander phy rather than a SATA phy, and performs the SAS speed negotiation sequence. These states are indicated by state names with a prefix of SAS.

Figure 123 — SP (phy layer) state machine - SAS speed negotiation states

6.8.4.2 SP8:SAS_Start state

6.8.4.2.1 State description

This is the initial state for the SAS speed negotiation sequence.

Upon entry into this state, this state shall:

   a)   initialize and start the RCDT timer;
   b)   send a Set Rate message to the SP transmitter with the argument set to:
        A)   1.5 Gbit/s, if the transition into this state was from the SP6:OOB_AwaitNoCOMSAS state (i.e., if this is
             the first speed negotiation window); or
        B)   the value of the SAS Speed Negotiation Window Rate argument.

During this state D.C. idle shall be transmitted.

6.8.4.2.2 Transition SP8:SAS_Start to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.4.2.3 Transition SP8:SAS_Start to SP9:SAS_RateNotSupported

This transition shall occur after the RCDT timer expires if the current speed negotiation window rate is not
supported.

6.8.4.2.4 Transition SP8:SAS_Start to SP10:SAS_AwaitALIGN

This transition shall occur after the RCDT timer expires if the current speed negotiation window rate is supported.

6.8.4.3 SP9:SAS_RateNotSupported state

6.8.4.3.1 State description

Upon entry into this state the SNTT timer shall be initialized and started.

During this state D.C. idle shall be transmitted.

6.8.4.3.2 Transition SP9:SAS_RateNotSupported to SP14:SAS_Fail

This transition shall occur after the SNTT timer expires.

6.8.4.4 SP10:SAS_AwaitALIGN state

6.8.4.4.1 State description

Upon entry into this state, the SNTT timer and SNLT timer shall be initialized and started and this state shall
repeatedly send Transmit ALIGN (0) messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS
state machine to re-acquire dword synchronization without running a new link reset sequence.

Upon entry into this state, this state shall send a Start DWS message to the SP_DWS state machine.

6.8.4.4.2 Transition SP10:SAS_AwaitALIGN to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or
after receiving a COMINIT Detected message.

6.8.4.4.3 Transition SP10:SAS_AwaitALIGN to SP11:SAS_AwaitALIGN1

This transition shall occur if this state receives an ALIGN Received (0) message before the SNLT timer expires.

6.8.4.4.4 Transition SP10:SAS_AwaitALIGN to SP12:SAS_AwaitSNW

This transition shall occur if this state receives an ALIGN Received (1) message before the SNLT timer expires.

6.8.4.4.5 Transition SP10:SAS_AwaitALIGN to SP14:SAS_Fail

This transition shall occur if the SNTT timer expires.

6.8.4.5 SP11:SAS_AwaitALIGN1 state

6.8.4.5.1 State description

This state shall repeatedly send Transmit ALIGN (1) messages to the SP transmitter.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.5.2 Transition SP11:SAS_AwaitALIGN1 to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

6.8.4.5.3 Transition SP11:SAS_AwaitALIGN1 to SP12:SAS_AwaitSNW

This transition shall occur if this state receives an ALIGN Received (1) message before the SNLT timer expires. This indicates that the other phy has been able to achieve dword synchronization in the current speed negotiation window.

6.8.4.5.4 Transition SP11:SAS_AwaitALIGN1 to SP14:SAS_Fail

This transition shall occur if the SNTT timer expires. This indicates that the other phy has not been able to achieve dword synchronization in the current speed negotiation window.

6.8.4.6 SP12:SAS_AwaitSNW state

6.8.4.6.1 State description

This state shall repeatedly send Transmit ALIGN (1) messages to the SP transmitter.

If this is the final speed negotiation window, this state shall send a Start SL_IR Receiver confirmation to the link layer.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

This state waits for the SNTT timer to expire or for a Stop SNTT request.

6.8.4.6.2 Transition SP12:SAS_AwaitSNW to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

6.8.4.6.3 Transition SP12:SAS_AwaitSNW to SP13:SAS_Pass

This transition shall occur after the SNTT timer expires or after receiving a Stop SNTT request.

6.8.4.7 SP13:SAS_Pass state

6.8.4.7.1 State description

This state determines if:

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

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.7.2 Transition SP13:SAS_Pass to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message if this state does not send a Start DWS message, or after receiving a COMINIT Detected message.

6.8.4.7.3 Transition SP13:SAS_Pass to SP8:SAS_Start

This transition shall occur if this is not the final speed negotiation window.

This transition shall include a SAS Speed Negotiation Window Rate argument with the transition set to the next higher speed negotiation window rate.

6.8.4.7.4 Transition SP13:SAS_Pass to SP15:SAS_PHY_Ready

This transition shall occur if this is the final speed negotiation window.

6.8.4.8 SP14:SAS_Fail state

6.8.4.8.1 State description

This state determines if:

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

6.8.4.8.2 Transition SP14:SAS_Fail to SP1:OOB_AwaitCOMX

This transition shall occur if the current speed negotiation window is:

   a)   the maximum SAS speed negotiation window; or
   b)   the final SAS speed negotiation window.

6.8.4.8.3 Transition SP14:SAS_Fail to SP8:SAS_Start

If the previous SAS speed negotiation window was successful, this transition shall occur and shall include:

   a)   a SAS Speed Negotiation Window Rate argument set to the previous speed negotiation window rate; and
   b)   a Final SAS Speed Negotiation Window argument.

If the previous SAS speed negotiation window failed and the current speed negotiation window is not the maximum SAS speed negotiation window, this transition shall occur and shall include a SAS Speed Negotiation Window Rate argument set to the next higher speed negotiation window rate.

6.8.4.9 SP15:SAS_PHY_Ready state

6.8.4.9.1 State description

This state waits for a COMINIT Detected message, a DWS Lost message, or a DWS Reset message.

While in this state dwords from the link layer are transmitted at the negotiated physical link rate at the rate established in the previous speed negotiation window.

Upon entry into this state, this state shall send a Phy Layer Ready (SAS) confirmation to the link layer to indicate that the physical link has been brought up successfully in SAS mode.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.4.9.2 Transition SP15:SAS_PHY_Ready to SP0:OOB_COMINIT

This transition shall occur after:

   a)   receiving a DWS Lost message, if this state does not send a Start DWS message;
   b)   receiving a DWS Lost message followed by a COMINIT Detected message, if this state does not send a Start DWS message; or
   c)   receiving a DWS Reset message.

This transition may but should not occur after receiving a COMINIT Detected message before receiving a DWS Lost message, or after receiving a COMINIT Detected message after sending a Start DWS message (i.e., the SP state machine should ignore COMINIT Detected messages unless the SP_DWS state machine has indicated loss of dword synchronization).

6.8.5 SATA host emulation states

6.8.5.1 SATA host emulation states overview

Figure 124 shows the SATA host emulation states, in which the phy has detected that it is attached to a SATA phy and behaves as if it were a SATA host phy, initiating the SATA speed negotiation sequence. These states are indicated by state names with a prefix of SATA.

The power management states defined in this standard are for SAS initiator phys that support SATA; expander devices that support SATA do not support power management in this standard.
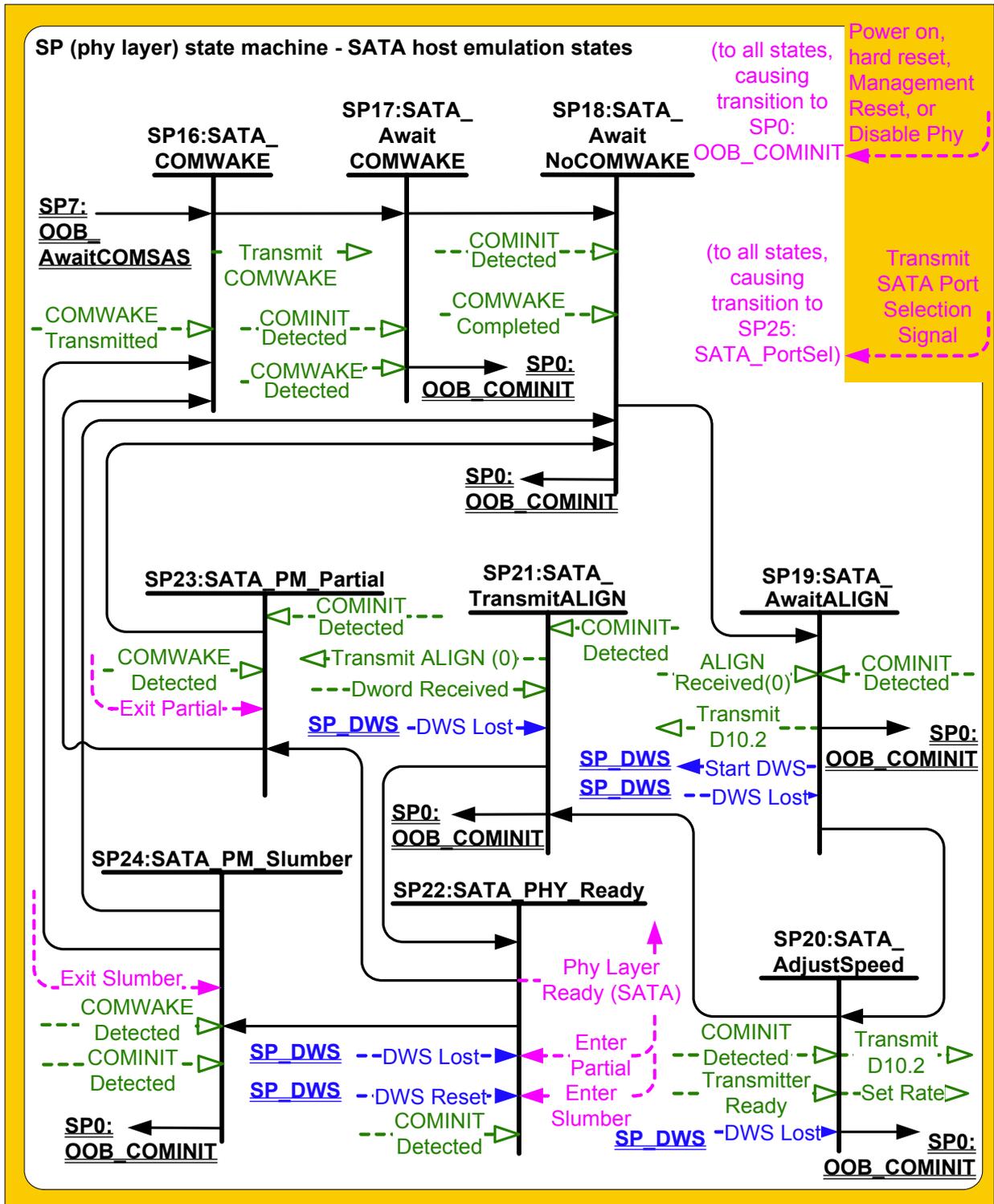
Figure 124 — SP (phy layer) state machine - SATA host emulation states

6.8.5.2 SP16:SATA_COMWAKE state

6.8.5.2.1 State description

This state shall send a Transmit COMWAKE message to the SP transmitter and wait for a COMWAKE Transmitted message.

6.8.5.2.2 Transition SP16:SATA_COMWAKE to SP17:SATA_AwaitCOMWAKE

This transition shall occur after receiving a COMWAKE Transmitted message.

6.8.5.3 SP17:SATA_AwaitCOMWAKE state

6.8.5.3.1 State description

This state waits for COMWAKE to be received.

6.8.5.3.2 Transition SP17:SATA_AwaitCOMWAKE to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.3.3 Transition SP17:SATA_AwaitCOMWAKE to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur after receiving a COMWAKE Detected message.

6.8.5.4 SP18:SATA_AwaitNoCOMWAKE state

6.8.5.4.1 State description

This state waits for a COMWAKE Completed message.

6.8.5.4.2 Transition SP18:SATA_AwaitNoCOMWAKE to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.4.3 Transition SP18:SATA_AwaitNoCOMWAKE to SP19:SATA_AwaitALIGN

This transition shall occur after receiving a COMWAKE Completed message.

6.8.5.5 SP19:SATA_AwaitALIGN state

6.8.5.5.1 State description

Upon entry into this state, this state shall send a Start DWS message to the SP_DWS state machine.

This state shall:

  a) repeatedly send Transmit D10.2 messages to the SP transmitter;
  b) initialize and start the Await ALIGN Timeout timer; and
  c) wait for an ALIGN Received (0) message to be received or for the Await ALIGN Timeout timer to expire.

The phy shall start transmitting D10.2 characters no later than a COMWAKE response time (see 6.7.2.2) after entry into this state.

6.8.5.5.2 Transition SP19:SATA_AwaitALIGN to SP0:OOB_COMINIT

This transition shall occur if the Await ALIGN Timeout timer expires or after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

6.8.5.5.3 Transition SP19:SATA_AwaitALIGN to SP20:SATA_AdjustSpeed

This transition shall occur if this state receives an ALIGN Received (0) message before the Await ALIGN Timeout timer expires. The ALIGN Received (0) message indicates an ALIGN (0) was received at any of the physical link rates supported by this phy.

6.8.5.6 SP20:SATA_AdjustSpeed state

6.8.5.6.1 State description

This state waits for the SP transmitter to adjust to the same physical link rate of the ALIGNs that were detected by the receiver circuitry.

This state shall:

  1) send a Set Rate message to the SP transmitter; and

2) repeatedly send Transmit D10.2 messages to the SP transmitter.

6.8.5.6.2 Transition SP20:SATA_AdjustSpeed to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

6.8.5.6.3 Transition SP20:SATA_AdjustSpeed to SP21:SATA_TransmitALIGN

This transition shall occur after receiving a Transmitter Ready message.

6.8.5.7 SP21:SATA_TransmitALIGN state

6.8.5.7.1 State description

This state shall repeatedly send Transmit ALIGN (0) messages to the SP transmitter.

6.8.5.7.2 Transition SP21:SATA_TransmitALIGN to SP0:OOB_COMINIT

This transition shall occur after receiving a DWS Lost message, or after receiving a COMINIT Detected message.

6.8.5.7.3 Transition SP21:SATA_TransmitALIGN to SP22:SATA_PHY_Ready

This transition shall occur after receiving three consecutive Dword Received messages containing primitives other than ALIGN (0).

6.8.5.8 SP22:SATA_PHY_Ready state

6.8.5.8.1 State description

While in this state dwords from the link layer are transmitted at the negotiated physical link rate at the rate established in the previous speed negotiation window.

This state shall send a Phy Layer Ready (SATA) confirmation to the link layer to indicate that the physical link has been brought up successfully in SATA mode.

This state waits for a COMINIT Detected message, a DWS Lost message, or a DWS Reset message.

Each time this state receives a DWS Lost message, this state may send a Start DWS message to the SP_DWS state machine to re-acquire dword synchronization without running a new link reset sequence.

6.8.5.8.2 Transition SP22:SATA_PHY_Ready to SP0:OOB_COMINIT

This transition shall occur after:

   a) receiving a DWS Lost message, if this state does not send a Start DWS message;
   b) receiving a DWS Lost message followed by a COMINIT Detected message, if this state does not send a Start DWS message; or
   c) receiving a DWS Reset message.

This transition may but should not occur after receiving a COMINIT Detected message before receiving a DWS Lost message, or after receiving a COMINIT Detected message after sending a Start DWS message (i.e., the SP state machine should ignore COMINIT Detected messages unless the SP_DWS state machine has indicated loss of dword synchronization).

6.8.5.8.3 Transition SP22:SATA_PHY_Ready to SP23:SATA_PM_Partial

This transition shall occur after receiving an Enter Partial request.

6.8.5.8.4 Transition SP22:SATA_PHY_Ready to SP24:SATA_PM_Slumber

This transition shall occur after receiving an Enter Slumber request.

6.8.5.9 SP23:SATA_PM_Partial state

6.8.5.9.1 State description

This state waits for a COMWAKE Detected message or an Exit Partial request.

6.8.5.9.2 Transition SP23:SATA_PM_Partial to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.9.3 Transition SP23:SATA_PM_Partial to SP16:SATA_COMWAKE

This transition shall occur after receiving a Exit Partial request.

6.8.5.9.4 Transition SP23:SATA_PM_Partial to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur after receiving a COMWAKE Detected message.

6.8.5.10 SP24:SATA_PM_Slumber state

6.8.5.10.1 State description

This state waits for a COMWAKE Detected message or an Exit Slumber request.

6.8.5.10.2 Transition SP24:SATA_PM_Slumber to SP0:OOB_COMINIT

This transition shall occur after receiving a COMINIT Detected message.

6.8.5.10.3 Transition SP24:SATA_PM_Slumber to SP16:SATA_COMWAKE

This transition shall occur after receiving a Exit Slumber request.

6.8.5.10.4 Transition SP24:SATA_PM_Slumber to SP18:SATA_AwaitNoCOMWAKE

This transition shall occur after receiving a COMWAKE Detected message.

6.8.6 SATA port selector state

6.8.6.1 State description

Figure 125 shows the SP25:SATA_PortSel state. This state controls transmission of the SATA port selection signal when a specified phy processes a Transmit SATA Port Selection Signal request.
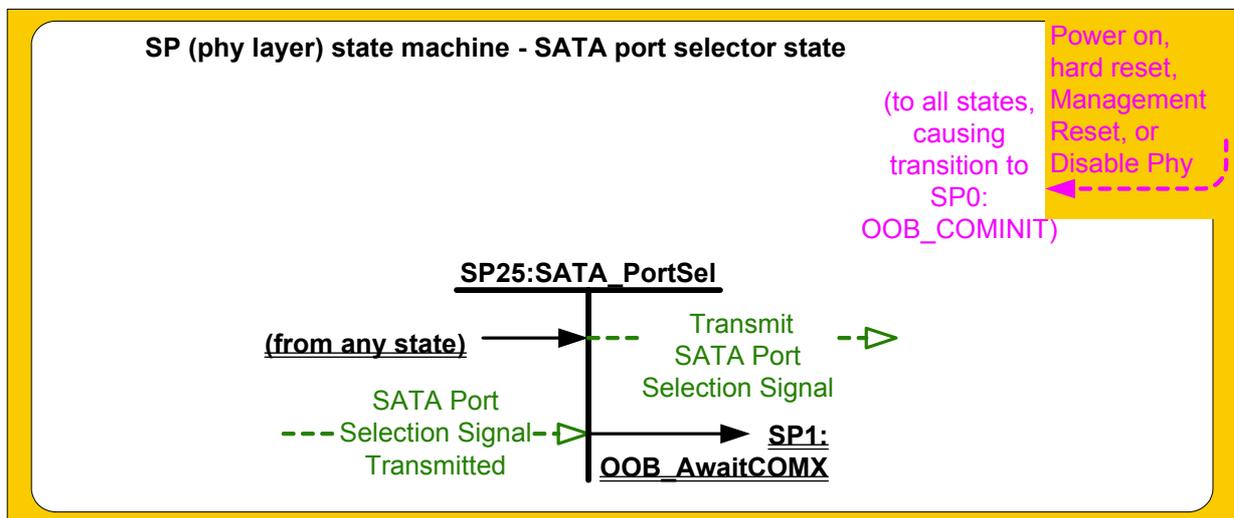
Figure 125 — SP (phy layer) state machine – SATA port selector state

Upon entry into this state, the phy shall:

   a) send a Transmit SATA Port Selection Signal message to the SP transmitter; and
   b) set the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response to zero.

6.8.6.2 Transition SP25:SATA_PortSel to SP1:OOB_AwaitCOMX

This transition shall occur after receiving a SATA Port Selection Signal Transmitted message.

6.8.7 SATA spinup hold state

6.8.7.1 State description

Figure 126 shows the SP26:SATA_SpinupHold state.
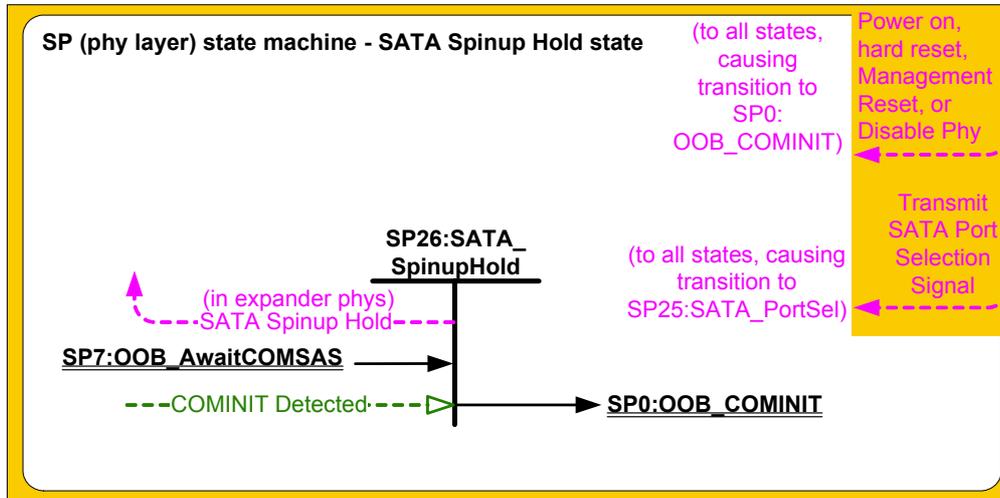
Figure 126 — SP (phy layer) state machine - SATA spinup hold state

If this state machine is in an expander phy, this state shall send a SATA Spinup Hold confirmation to the link layer.

6.8.7.2 Transition SP26:SATA_SpinupHold to SP0:OOB_COMINIT

This transition shall occur if this state receives a COMINIT Detected message.

6.9 SP_DWS (phy layer dword synchronization) state machine

6.9.1 SP_DWS state machine overview

Each phy includes an SP_DWS state machine and an SP_DWS receiver.

The SP_DWS state machine establishes the same dword boundaries at the receiver as at the attached transmitter by searching for control characters. The SP_DWS receiver monitors and decodes the incoming data stream and forces K28.5 characters into the first character position to effectively perform dword alignment when requested by the SP_DWS state machine. K28.5 characters with either positive or negative disparity shall be accepted. The SP_DWS receiver continues to reestablish dword alignment by forcing received K28.5 characters into the first character position until a K28.5-based primitive (i.e., K28.5, Dxx.y, Dxx.y, Dxx.y) with correct disparity on each data character is detected. The resultant primitives, dwords and valid dword indicators (e.g., encoding error indicators) are sent to this state machine to enable it to determine the dword synchronization policy.

After dword synchronization has been achieved, this state machine monitors invalid dwords that are received. When an invalid dword is detected, it requires two valid dwords to nullify its effect. When four invalid dwords are detected without nullification, dword synchronization is considered lost.

While dword synchronization is lost, the data stream received is invalid and dwords shall not be passed to the link layer.

This state machine consists of the following states:

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

j)   SP_DWS9:Lost3Recovered (see 6.9.12).

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

a)   power on;
b)   hard reset;
c)   receiving a Management Reset request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET in an expander device);
d)   receiving a Disable Phy request from the management layer (e.g., from the SMP PHY CONTROL function requesting a phy operation of DISABLE in an expander device); or
e)   receiving a Stop DWS message from the SP state machine.

This state machine receives the following messages from the SP state machine (see 6.8):

a)   Start DWS; and
b)   Stop DWS.

This state machine sends the following messages to the SP state machine:

a)   DWS Lost; and
b)   DWS Reset.

The SP_DWS state machine shall maintain the timers listed in table 67.

Table 67 — SP_DWS timers

Timer	Initial value
DWS Reset Timeout timer	1 ms

Figure 127 shows the SP_DWS state machine.
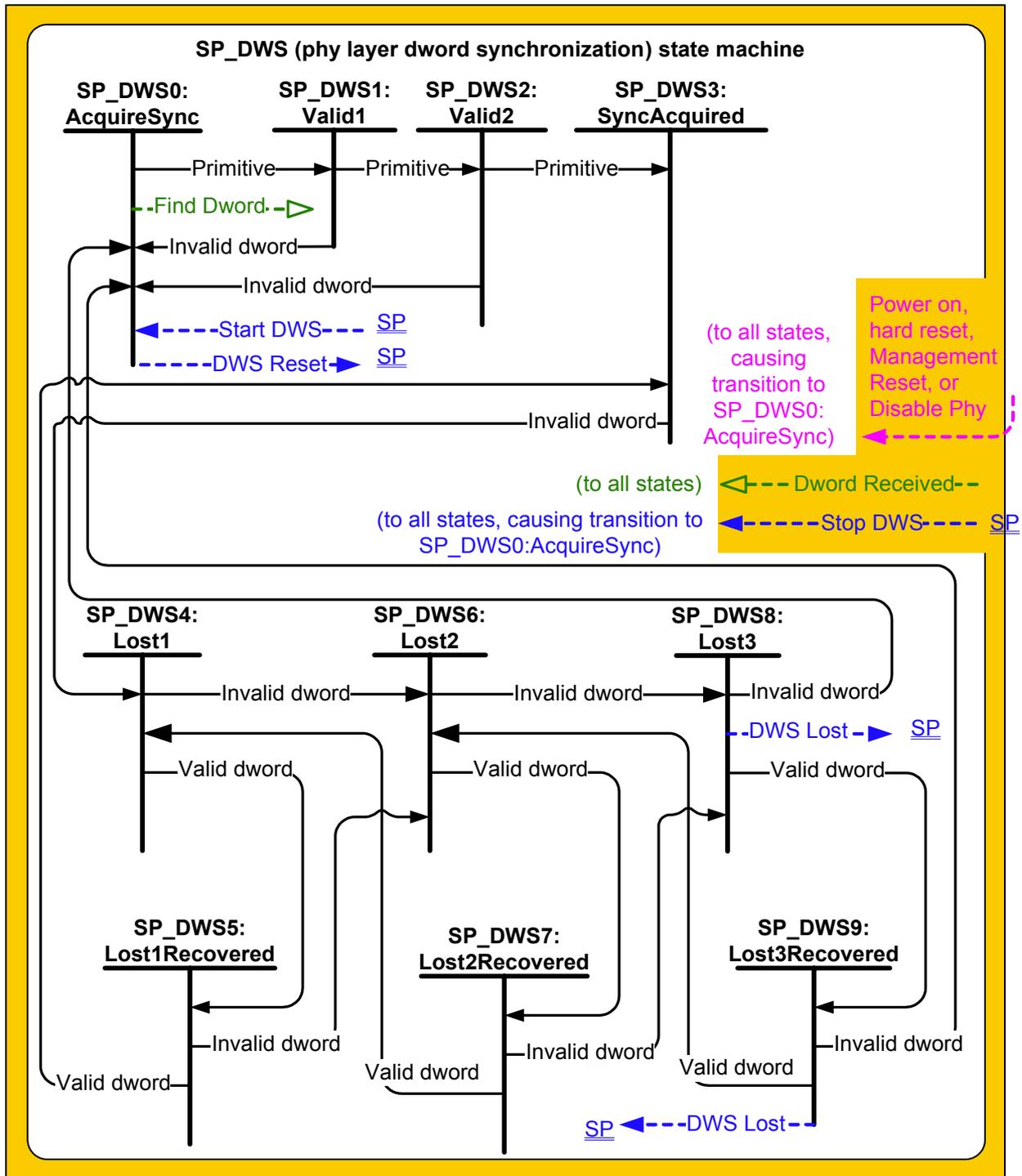
Figure 127 — SP_DWS (phy layer dword synchronization) state machine

6.9.2 SP_DWS receiver

The SP_DWS receiver receives the following messages from the SP_DWS state machine:

  a) Find Dword.

The SP_DWS receiver sends the following messages to the SP_DWS state machine:

  a) Dword Received (Primitive);
  b) Dword Received (Data Dword); and
  c) Dword Received (Invalid).

The SP_DWS receiver also sends Dword Received confirmations to the link layer state machine receivers (e.g., SL_IR, SL, SSP, SMP, and XL).

Upon receiving a Find Dword message, the SP_DWS receiver shall monitor the input data stream and force each K28.5 character detected into the first character position of a possible dword. If the next three characters are data characters with correct disparity, it shall send the dword as a Dword Received (Primitive) message to the SP_DWS state machine. Until it receives another Find Dword message, for every four characters it receives it shall:

  a)   send a Dword Received (Invalid) message to the SP_DWS state machine if the dword is an invalid dword (see 3.1.100);
  b)   send a Dword Received (Primitive) message to the SP_DWS state machine if the dword is a primitive (i.e., the dword contains a K28.5 character in the first character position followed by three data characters); or
  c)   send a Dword Received (Data Dword) message to the SP_DWS state machine if the dword is a data dword (i.e., it is not an invalid dword or a primitive).

6.9.3 SP_DWS0:AcquireSync state

6.9.3.1 State description

This is the initial state of this state machine.

After receiving a Start DWS message, this state shall:

  a)   send a Find Dword message to the SP_DWS receiver; and
  b)   initialize and start the DWS Reset Timeout timer;

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2, this state shall send a Find Dword message to the SP_DWS receiver. and the DWS Reset Timeout timer shall continue running.

If this state is entered from SP_DWS1:Valid1 or SP_DWS2:Valid2 and the DWS Reset Timeout timer has expired, this state may send a DWS Reset message to the SP state machine (e.g., if the phy chooses to initiate a new link reset sequence because dword synchronization has been lost for too long).

This state shall not send a DWS Reset message to the SP until the DWS Reset Timeout timer expires.

If the DWS Reset Timeout timer expires, this state may send a DWS Reset message to the SP state machine.

6.9.3.2 Transition SP_DWS0:AcquireSync to SP_DWS1:Valid1

This transition shall occur after sending a Find Dword message and receiving a Dword Received (Primitive) message.

6.9.4 SP_DWS1:Valid1 state

6.9.4.1 State description

This state is reached after one valid primitive has been received. This state waits for a second valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.9.4.2 Transition SP_DWS1:Valid1 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.9.4.3 Transition SP_DWS1:Valid1 to SP_DWS2:Valid2

This transition shall occur after receiving a Dword Received (Primitive) message.

6.9.5 SP_DWS2:Valid2 state

6.9.5.1 State description

This state is reached after two valid primitives have been received without adjusting the dword synchronization. This state waits for a third valid primitive or an invalid dword.

The DWS Reset Timeout timer shall continue running.

6.9.5.2 Transition SP_DWS2:Valid2 to SP_DWS0:AcquireSync

This transition shall occur after receiving a Dword Received (Invalid) message or after the DWS Reset Timeout timer expires.

6.9.5.3 Transition SP_DWS2:Valid2 to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (Primitive) message.

6.9.6 SP_DWS3:SyncAcquired state

6.9.6.1 State description

This state is reached after three valid primitives have been received without adjusting the dword synchronization.

The most recently received primitive and all subsequent dwords shall be forwarded for processing by the link layer.

This state waits for a Dword Received (Invalid) message, which indicates that dword synchronization might be lost.

6.9.6.2 Transition SP_DWS3:SyncAcquired to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.7 SP_DWS4:Lost1 state

6.9.7.1 State description

This state is reached when one invalid dword has been received and not nullified. This state waits for a Dword Received message.

6.9.7.2 Transition SP_DWS4:Lost1 to SP_DWS5:Lost1Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.7.3 Transition SP_DWS4:Lost1 to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.8 SP_DWS5:Lost1Recovered state

6.9.8.1 State description

This state is reached when a valid dword has been received after one invalid dword had been received. This state waits for a Dword Received message.

6.9.8.2 Transition SP_DWS5:Lost1Recovered to SP_DWS3:SyncAcquired

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.8.3 Transition SP_DWS5:Lost1Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.9 SP_DWS6:Lost2 state

6.9.9.1 State description

This state is reached when two invalid dwords have been received and not nullified. This state waits for a Dword Received message.

6.9.9.2 Transition SP_DWS6:Lost2 to SP_DWS7:Lost2Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.9.3 Transition SP_DWS6:Lost2 to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.10 SP_DWS7:Lost2Recovered state

6.9.10.1 State description

This state is reached when a valid dword has been received after two invalid dwords had been received. This state waits for a Dword Received message.

6.9.10.2 Transition SP_DWS7:Lost2Recovered to SP_DWS4:Lost1

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.10.3 Transition SP_DWS7:Lost2Recovered to SP_DWS8:Lost3

This transition shall occur after receiving a Dword Received (Invalid) message.

6.9.11 SP_DWS8:Lost3 state

6.9.11.1 State description

This state is reached when three invalid dwords have been received and not nullified. This state waits for a Dword Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

6.9.11.2 Transition SP_DWS8:Lost3 to SP_DWS9:Lost3Recovered

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.11.3 Transition SP_DWS8:Lost3 to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.9.12 SP_DWS9:Lost3Recovered state

6.9.12.1 State description

This state is reached when a valid dword has been received after three invalid dwords had been received. This state waits for a Dword Received message.

If a Dword Received (Invalid) message is received (i.e., the fourth non-nullified invalid dword is received), this state shall send a DWS Lost message to the SP state machine.

6.9.12.2 Transition SP_DWS9:Lost3Recovered to SP_DWS6:Lost2

This transition shall occur after receiving a Dword Received (Data Dword) message or a Dword Received (Primitive) message.

6.9.12.3 Transition SP_DWS9:Lost3Recovered to SP_DWS0:AcquireSync

This transition shall occur after sending a DWS Lost message.

6.10 Spin-up

If a SAS target device receives COMSAS during the reset sequence, it shall not spin-up until allowed by the SA_PC state machine (see 10.2.10).

Expander devices that detect an attached SATA phy may halt the automatic phy reset sequence after the COMSAS Detect Timeout (see 6.8) to delay spin-up; this is called SATA spinup hold. This is reported in the SMP DISCOVER function (see 10.4.3.5) and is released with the SMP PHY CONTROL function (see 10.4.3.10).

NOTE 13 - Enclosures supporting both SATA devices and SAS target devices may need to sequence power to each attached device to avoid excessive power consumption during power on, since the SATA devices may spin-up automatically after power on if staggered spin-up is not implemented (see SATAII-EXT).

7 Link layer

7.1 Link layer overview

The link layer defines primitives, address frames, and connections. Link layer state machines interface to the port layer and the phy layer and perform the identification and hard reset sequences, connection management, and SSP, STP, and SMP specific frame transmission and reception.

7.2 Primitives

7.2.1 Primitives overview

Primitives are dwords whose first character is a K28.3 or K28.5. Primitives are not considered big-endian or little-endian; they are just interpreted as first, second, third, and last characters. Table 68 defines the primitive format.

Table 68 — Primitive format

Character	Description
First	K28.5 control character (for primitives defined in this standard) or K28.3 control character (for primitives defined by SATA).
Second	Constant data character.
Third	Constant data character.
Last	Constant data character.

7.2.2 Primitive summary

Table 69 defines the primitives not specific to the type of connection.

Table 69 — Primitives not specific to type of connection (Sheet 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 (SES)	NoConn			T	I	E	T	Redundant
BROADCAST (RESERVED 1)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED 2)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED 3)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED 4)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED CHANGE 0)	NoConn				I	E	T	Redundant
BROADCAST (RESERVED CHANGE 1)	NoConn				I	E	T	Redundant

Table 69 — Primitives not specific to type of connection (Sheet 2 of 2)

Primitive	Use [a]	From [b]			To [b]			Primitive sequence type [c]
		I	E	T	I	E	T	
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
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	T	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 SAS initiator ports;
  b) E for expander ports; and
  c) T for SAS target ports.
Expander ports are not considered originators of primitives that are passing through from expander port to expander port.

[c] The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).

Table 70 defines the primitives used only inside SSP and SMP connections.

Table 70 — 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 SSP initiator ports and SMP initiator ports;
   b) E for expander ports; and
   c) T for SSP target ports and SMP target ports.
   Expander ports are not considered originators of primitives that are passing through from expander port to expander port.
[c] The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).

Table 71 lists the primitives used only inside STP connections and on SATA physical links.

Table 71 — Primitives used only inside STP connections and on SATA physical links

Primitive	Use [a]	From [b]			To [b]			Primitive sequence type [c]
		I	E	T	I	E	T	
SATA_CONT	STP, SATA	I		T	I		T	Single
SATA_DMAT	STP, SATA	I		T	I		T	Single
SATA_EOF	STP, SATA	I		T	I		T	Single
SATA_ERROR [d]	SATA		E				T	Single
SATA_HOLD	STP, SATA	I		T	I		T	Continued
SATA_HOLDA	STP, SATA	I		T	I		T	Continued
SATA_PMACK	STP, SATA							Repeated
SATA_PMNAK	STP, SATA	I	E				T	Repeated
SATA_PMREQ_P	STP, SATA							Continued
SATA_PMREQ_S	STP, SATA							Continued
SATA_R_ERR	STP, SATA	I		T	I		T	Continued
SATA_R_IP	STP, SATA	I		T	I		T	Continued
SATA_R_OK	STP, SATA	I		T	I		T	Continued
SATA_R_RDY	STP, SATA	I		T	I		T	Continued
SATA_SOF	STP, SATA	I		T	I		T	Single
SATA_SYNC	STP, SATA	I		T	I		T	Continued
SATA_WTRM	STP, SATA	I		T	I		T	Continued
SATA_X_RDY	STP, SATA	I		T	I		T	Continued

[a] The Use column indicates when the primitive is used:
  a) STP: SAS 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 STP initiator ports and SATA host ports;
  b) E for expander ports; and
  c) T for STP target ports and SATA device ports.
  Expander ports are not considered originators of primitives that are passing through from expander port to expander port.
[c] The Primitive sequence type columns indicate whether the primitive is sent as a single primitive sequence, a repeated primitive sequence, a continued primitive sequence, a triple primitive sequence, or a redundant primitive sequence (see 7.2.4).
[d] Although included in this table, SATA_ERROR is not a primitive (see 3.1.144) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword, used by expander devices forwarding an error onto a SATA physical link (see 7.2.7.1).

7.2.3 Primitive encodings

Table 72 defines the primitive encoding for primitives not specific to type of connection.

Table 72 — Primitive encoding for primitives not specific to type of connection (Sheet 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 (SES)	K28.5	D04.7	D07.3	D29.7
BROADCAST (RESERVED 1)	K28.5	D04.7	D01.4	D24.0
BROADCAST (RESERVED 2)	K28.5	D04.7	D04.7	D04.7
BROADCAST (RESERVED 3)	K28.5	D04.7	D16.7	D02.0
BROADCAST (RESERVED 4)	K28.5	D04.7	D29.7	D30.0
BROADCAST (RESERVED CHANGE 0)	K28.5	D04.7	D24.0	D31.4
BROADCAST (RESERVED CHANGE 1)	K28.5	D04.7	D27.4	D07.3
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

Table 72 — Primitive encoding for primitives not specific to type of connection (Sheet 2 of 2)

Primitive	Character			
	1st	2nd	3rd	4th (last)
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	D10.2	D10.2
OPEN_ACCEPT	K28.5	D16.7	D16.7	D16.7
OPEN_REJECT (BAD DESTINATION)	K28.5	D31.4	D31.4	D31.4
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 73 defines the primitive encodings for primitives used only inside SSP and SMP connections.

Table 73 — 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 74 lists the primitive encodings for primitives used only inside STP connections and on SATA physical links.

Table 74 — 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] [b]	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 (see ATA/ATAPI-7 V3). [b] Although included in this table, SATA_ERROR is not a primitive (see 3.1.144) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword, used by expander devices forwarding an error onto a SATA physical link (see 7.2.7.1).				

7.2.4 Primitive sequences

7.2.4.1 Primitive sequences overview

Table 75 summarizes the types of primitive sequences.

Table 75 — Primitive sequences

Primitive sequence type	Number of times the transmitter transmits the primitive to transmit the primitive sequence	Number of times the receiver receives the primitive to detect the primitive sequence
Single	1	1
Repeated	1 or more	1
Continued	2 followed by SATA_CONT	1
Triple	3	3
Redundant	6	3

Any number of ALIGNs and NOTIFYs may be sent inside primitive sequences without affecting the count or breaking the consecutiveness requirements. Rate matching ALIGNs and NOTIFYs shall be sent inside primitive sequences inside of connections if rate matching is enabled (see 7.13).

7.2.4.2 Single primitive sequence

Primitives labeled as single primitive sequences (e.g., RRDY, SATA_SOF) shall be transmitted one time to form a single primitive sequence.

Receivers count each primitive received that is labeled as a single primitive sequence as a distinct single primitive sequence.

7.2.4.3 Repeated primitive sequence

Primitives that form repeated primitive sequences (e.g., SATA_PMACK) shall be transmitted one or more times. Only STP primitives form repeated primitive sequences. ALIGNs and NOTIFYs may be sent inside repeated primitive sequences as described in 7.2.4.1.

Figure 128 shows an example of transmitting a repeated primitive sequence.
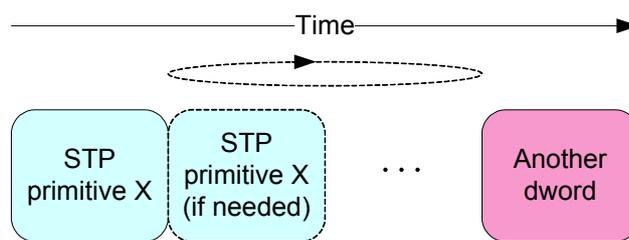
NOTE: Another dword is a dword other than ALIGN, NOTIFY, or STP primitive X

Figure 128 — Transmitting a repeated primitive sequence

Receivers do not count the number of times a repeated primitive is received (i.e., receivers are simply in the state of receiving the primitive).

Figure 129 shows an example of receiving a repeated primitive sequence.
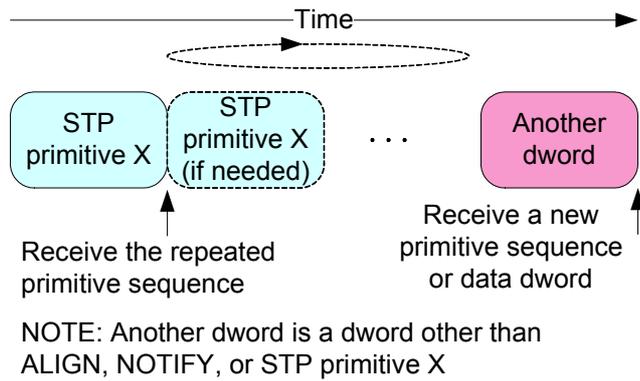
Figure 129 — Receiving a repeated primitive sequence

7.2.4.4 Continued primitive sequence

Primitives that form continued primitive sequences (e.g., SATA_HOLD) shall be transmitted as specified in Figure 7.17.4. ALIGNs and NOTIFYs may be sent inside continued primitive sequences as described in 7.2.4.1.

7.2.4.5 Triple primitive sequence

Primitives that form triple primitive sequences (e.g., CLOSE (NORMAL)) shall be sent three times consecutively. ALIGNs and NOTIFYs may be sent inside primitive sequences as described in 7.2.4.1.

Receivers shall detect a triple primitive sequence after the identical primitive is received in three consecutive dwords. After receiving a triple primitive sequence, a receiver shall not detect a second instance of the same triple primitive sequence until it has received three consecutive dwords that are not any of the following:

   a) the original primitive; or
   b) an ALIGN or NOTIFY.

Figure 130 shows examples of triple primitive sequences.
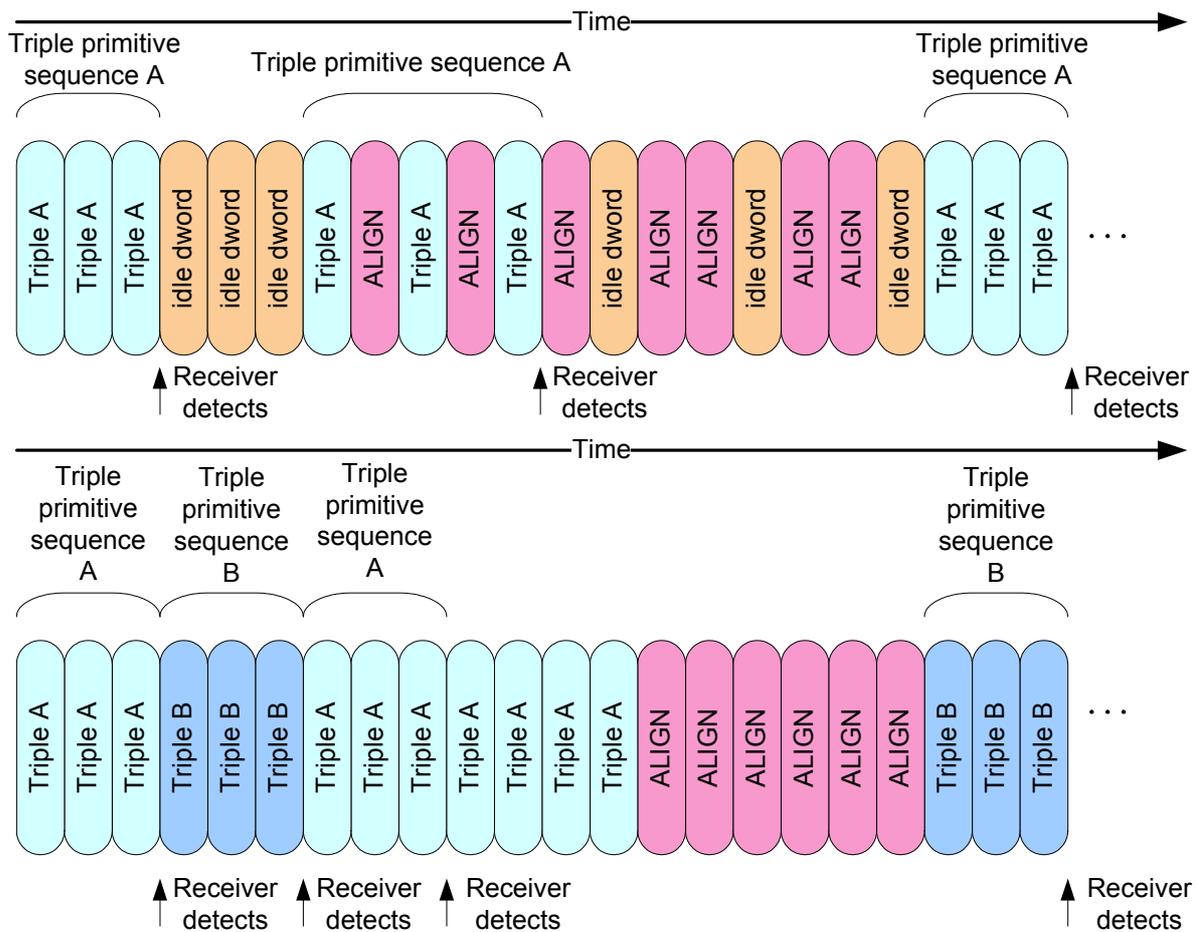
Figure 130 — Triple primitive sequence

7.2.4.6 Redundant primitive sequence

Primitives that form redundant primitive sequences (e.g., BROADCAST (CHANGE)) shall be sent six times consecutively. ALIGNs and NOTIFYs may be sent inside primitive sequences as described in 7.2.4.1.

A receiver shall detect a redundant primitive sequence after the identical primitive is received in three consecutive dwords. After receiving a redundant primitive sequence, a receiver shall not detect a second instance of the same redundant primitive sequence until it has received six consecutive dwords that are not any of the following:

a) the original primitive; or
b) an ALIGN or NOTIFY.

Figure 131 shows examples of redundant primitive sequences.
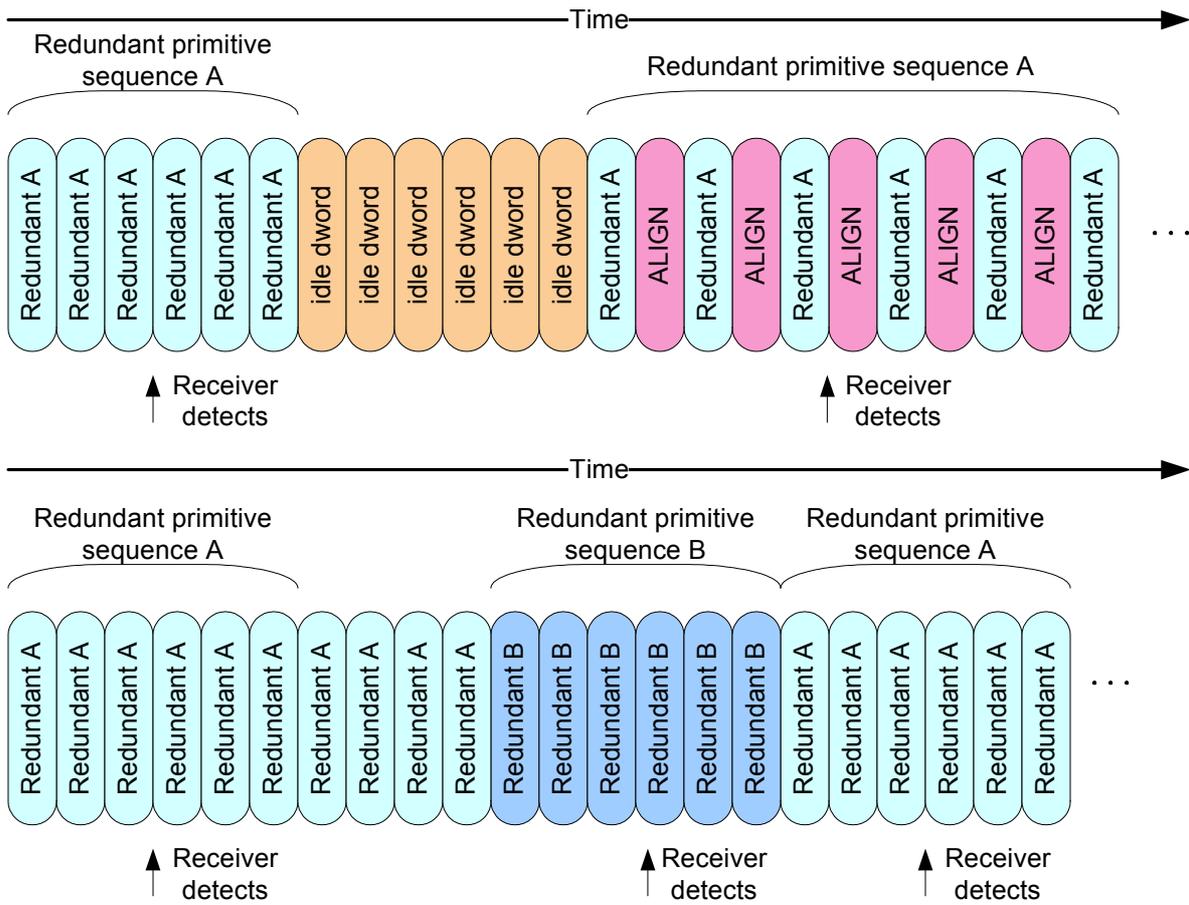
Figure 131 — Redundant primitive sequence

7.2.5 Primitives not specific to type of connections

7.2.5.1 AIP (Arbitration in progress)

AIP is sent by an expander device after a connection request to specify that the connection request is being processed and specify the status of the connection request.

The versions of AIP representing different statuses are defined in table 76.

Table 76 — AIP primitives

Primitive	Description
AIP (NORMAL)	Expander device has accepted the connection request. This may be sent multiple times (see 7.12.4.2).
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 either the destination phys are all being used for connections or there are insufficient routing resources to complete the connection request. This may be sent multiple times (see 7.12.4.2).
AIP (WAITING ON DEVICE)	Expander device has determined the routing for the connection request and forwarded it to the output physical link. This is sent one time (see 7.12.4.2).
AIP (WAITING ON PARTIAL)	Expander device has determined the routing for the connection request, but the destination phys are all busy with other partial pathways. This may be sent multiple times (see 7.12.4.2).
AIP (RESERVED WAITING ON PARTIAL)	Reserved. Processed the same as AIP (WAITING ON PARTIAL).

See 7.12 for details on connections.

7.2.5.2 ALIGN

ALIGNs are used for:

a) OOB signals;
b) character and dword alignment during the speed negotiation sequence;
c) clock skew management after the phy reset sequence (see 7.3);
d) rate matching during connections (see 7.13); and
e) STP initiator phy throttling during STP connections (see 7.17.2).

Table 77 defines the different versions of ALIGN primitives.

Table 77 — ALIGN primitives

Primitive	Description
ALIGN (0)	Used for OOB signals, the speed negotiation sequence, clock skew management, rate matching, and STP initiator phy throttling.
ALIGN (1)	Used for the speed negotiation sequence, clock skew management and rate matching, and STP initiator phy throttling.
ALIGN (2)	Used for clock skew management, rate matching, and STP initiator phy throttling.
ALIGN (3)	Used for clock skew management, rate matching, and STP initiator phy throttling.

Phys shall use ALIGN (0) to construct OOB signals as described in 6.6. Phys shall use ALIGN (0) and ALIGN (1) during the speed negotiation sequence as described in 6.7.4.2. Phys shall rotate through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3) for all ALIGNs sent after the phy reset sequence.

Phys receiving ALIGNs after the phy reset sequence shall not verify the rotation and shall accept any of the ALIGNs at any time.

Phys shall only detect an ALIGN after decoding all four characters in the primitive.

NOTE 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.

For clock skew management, rate matching, and STP initiator phy throttling, ALIGNs may be replaced by NOTIFYs (see 7.2.5.9). ALIGNs shall not be replaced by NOTIFYs during OOB signals and speed negotiation.

7.2.5.3 BREAK

BREAK is used to abort a connection request or break a connection.

See 7.12.5 and 7.12.7 for details on breaking connections.

7.2.5.4 BROADCAST

BROADCASTs are used to notify all SAS ports in a domain of an event.

The versions of BROADCAST representing different reasons are defined in table 78.

Table 78 — BROADCAST primitives

Primitive	Description
BROADCAST (CHANGE)	Notification of a configuration change.
BROADCAST (RESERVED CHANGE 0)	Reserved. Processed the same as BROADCAST (CHANGE) by SAS ports (i.e, SAS initiator ports and SAS target ports).
BROADCAST (RESERVED CHANGE 1)	Reserved. Processed the same as BROADCAST (CHANGE) by SAS ports (i.e., SAS initiator ports and SAS target ports).
BROADCAST (SES)	Notification of an asynchronous event from a logical unit with a peripheral device type set to 0Dh (i.e., enclosure services device) (see SPC-3 and SES-2) in the SAS domain.
BROADCAST (RESERVED 1)	Reserved.
BROADCAST (RESERVED 2)	Reserved.
BROADCAST (RESERVED 3)	Reserved.
BROADCAST (RESERVED 4)	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 phy reset sequence has completed.

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

BROADCAST (CHANGE) is sent by an expander device to notify SAS initiator ports and other expander devices that a configuration change has occurred. BROADCAST (CHANGE) may also be transmitted by SAS initiator ports. BROADCAST (CHANGE) shall be ignored by SAS target ports.

BROADCAST (SES) is sent by a SAS target port to notify SAS initiator ports that an asynchronous event has occurred in an enclosure, and SSP initiator ports should poll all the logical units with peripheral device types set to 0Dh (i.e., enclosure services devices)(see SPC-3 and SES-2) in the SAS domain. BROADCAST (SES) shall be ignored by SAS target ports.

BROADCAST (RESERVED CHANGE 0) and BROADCAST (RESERVED CHANGE 1) shall be processed the same as BROADCAST (CHANGE) by SAS ports. BROADCAST (RESERVED 1), BROADCAST (RESERVED 2), BROADCAST (RESERVED 3), and BROADCAST (RESERVED 4) shall be ignored by SAS ports.

See 7.11 for details on SAS domain changes. See 10.4.3.3 for details on counting BROADCAST (CHANGE) generation in an expander device.

7.2.5.5 CLOSE

CLOSE is used to close a connection. This primitive may be originated by a SAS initiator port or a SAS target port.

The versions of CLOSE representing different reasons are defined in table 79.

Table 79 — CLOSE primitives

Primitive	Description
CLOSE (CLEAR AFFILIATION)	Close an open STP connection and clear the affiliation (see 7.17.5). Processed the same as CLOSE (NORMAL) if: a) the connection is not an STP connection; b) the connection is an STP connection, but affiliations are not implemented by the STP target port; or c) the connection is an STP connection, but an affiliation is not present.
CLOSE (NORMAL)	Close a connection.
CLOSE (RESERVED 0)	Reserved. Processed the same as CLOSE (NORMAL).
CLOSE (RESERVED 1)	Reserved. Processed the same as CLOSE (NORMAL).

See 7.12.6 for details on closing connections.

7.2.5.6 EOAF (End of address frame)

EOAF specifies the end of an address frame.

See 7.8 for details on address frames.

7.2.5.7 ERROR

ERROR should be sent by an expander device when it is forwarding dwords from a SAS physical link or SATA physical link to a SAS physical link and it receives an invalid dword or an ERROR.

> NOTE 15 - Since an 8b10b coding error in one dword is sometimes not detected until the next dword (see table 56 in 6.3.5), expander devices should avoid deleting invalid dwords or ERRORs unless necessary (e.g., if the elasticity buffer is full) to avoid hiding evidence that an error has occurred.

See 7.15 for details on error handling by expander devices.

7.2.5.8 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 without an intervening identification sequence (see 4.4) and shall be ignored at other times.

7.2.5.9 NOTIFY

NOTIFY may be transmitted in place of any ALIGN (see 7.2.5.2) being transmitted for clock skew management (see 7.3), rate matching (see 7.13), or STP initiator phy throttling (see 7.17.2). Substitution of a NOTIFY may or may not affect the ALIGN rotation (i.e., the NOTIFY may take the place of one of the ALIGNs in the rotation through ALIGN (0), ALIGN (1), ALIGN (2), or ALIGN (3) or it may delay the rotation). A specific NOTIFY shall not be transmitted a second time until at least three ALIGNs or different NOTIFYs have been transmitted.

NOTIFY shall not be forwarded through expander devices. Expander devices shall substitute an ALIGN for a NOTIFY if necessary.

SAS target devices are not required to detect every transmitted NOTIFY.

The versions of NOTIFY representing different reasons are defined in table 80.

Table 80 — NOTIFY primitives

Primitive	Description
NOTIFY (ENABLE SPINUP)	Specify to an SAS target device that it may temporarily consume additional power while transitioning into the active or idle power condition state.
NOTIFY (RESERVED 0)	Reserved.
NOTIFY (RESERVED 1)	Reserved.
NOTIFY (RESERVED 2)	Reserved.

NOTIFY (ENABLE SPINUP) is transmitted by a SAS initiator port or expander port and is used to specify to an SAS target device that it may temporarily consume additional power (e.g., while spinning-up rotating media) while transitioning into the active or idle power condition state. The length of time the SAS target device consumes additional power and the amount of additional power is vendor specific. NOTIFY (ENABLE SPINUP) shall interact with the device's power condition state transitions, controlled by the Power Conditions mode page (see SPC-3) and/or the START STOP UNIT command (see SBC-2), as described in 10.2.10.

SAS initiator devices and expander devices shall use NOTIFY (ENABLE SPINUP) while attached to SAS 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 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 16 - The SAS initiator device or expander device uses NOTIFY (ENABLE SPINUP) to avoid exceeding enclosure power supply capabilities during spin-up of multiple SAS target devices. It may choose to rotate transmitting NOTIFY (ENABLE SPINUP) across all of its ports, distributing it to N ports at a time if the enclosure power supply is capable of powering N SAS target devices spinning up at a time. An expander device may allow this timing to be configured by an NVRAM programmed with enclosure-specific sequencing patterns, or may employ more complex, dynamic interaction with the enclosure power supply.

> NOTE 17 - NOTIFY (ENABLE SPINUP) should be transmitted as frequently as possible to avoid incurring application layer timeouts.

I_T nexus loss, logical unit reset, and hard reset shall not cause a SAS target device to spin-up automatically on receipt of NOTIFY (ENABLE SPINUP).

A SAS target device with multiple SAS target ports shall honor NOTIFY (ENABLE SPINUP) from all its SAS target ports equivalently (e.g., if a SAS target device contains two SSP target ports A and B, powers on in the Stopped state, and receives a START STOP UNIT command with the START bit set to one through SSP target port A, then a NOTIFY (ENABLE SPINUP) received on SSP target port B causes the SAS target device to spin up its rotating media (see 10.2.10)).

NOTIFY (RESERVED 0), NOTIFY (RESERVED 1), and NOTIFY (RESERVED 2) shall be ignored by all devices.

7.2.5.10 OPEN_ACCEPT

OPEN_ACCEPT specifies the acceptance of a connection request.

See 7.12 for details on connection requests.

7.2.5.11 OPEN_REJECT

OPEN_REJECT specifies that a connection request has been rejected and specifies the reason for the rejection. The result of some OPEN_REJECTs is to abandon (i.e., not retry) the connection request and the result of other OPEN_REJECTs is to retry the connection request.

All of the OPEN_REJECT versions defined in table 81 shall result in the originating port abandoning the connection request.

<div align="center">

Table 81 — OPEN_REJECT abandon primitives

</div>

Primitive	Originator	Description
OPEN_REJECT (BAD DESTINATION)	Expander phy	A connection request arrives through an expander phy using the direct routing or table routing method and the expander device determines the connection request would have to be routed to the same expander port as the expander port through which the connection request arrived (e.g., the destination SAS address equals the source SAS address), and the expander device has not chosen to return OPEN_REJECT (NO DESTINATION) (see 7.12.4.3 and 7.12.4.4).
OPEN_REJECT (CONNECTION RATE NOT SUPPORTED)	Any phy	The requested connection rate is not supported on some physical link on the pathway between the source phy and destination phy. When a SAS initiator phy is directly attached to a SAS target phy, the requested connection rate is not supported by the destination phy. The connection request may be modified and reattempted as described in 7.12.2.2.
OPEN_REJECT (PROTOCOL NOT SUPPORTED)	Destination phy	Phy with destination SAS address exists but the destination phy does not support the requested initiator/target role, protocol, initiator connection tag, or features (i.e., the values in the INITIATOR PORT bit, the PROTOCOL field, the INITIATOR CONNECTION TAG field, and/or the FEATURES field in the OPEN address frame are not supported).
OPEN_REJECT (RESERVED ABANDON 0)	Unknown	Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 1)	Unknown	Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 2)	Unknown	Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (RESERVED ABANDON 3)	Unknown	Reserved. Process the same as OPEN_REJECT (WRONG DESTINATION).
OPEN_REJECT (STP RESOURCES BUSY)	Destination phy	STP target port with destination SAS address exists but the STP target port has an affiliation with another STP initiator port or all of the available task file registers have been allocated to other STP initiator ports (see 7.17.5). Process the same as OPEN_REJECT (WRONG DESTINATION) for non-STP connection requests.
OPEN_REJECT (WRONG DESTINATION)	Destination phy	The destination SAS address does not match the SAS address of the SAS port to which the connection request was delivered.

All of the OPEN_REJECT versions defined in table 82 shall result in the originating port retrying the connection request.

Table 82 — OPEN_REJECT retry primitives

Primitive	Originator	Description
OPEN_REJECT (NO DESTINATION) [a]	Expander phy	Either: a) No such destination phy; b) the expander device determines the connection request would have to be routed to the same expander port as the expander port through which the connection request arrived (e.g., the destination SAS address equals the source SAS address) and the expander device has not chosen to return OPEN_REJECT (BAD DESTINATION) (see 7.12.4.3 and 7.12.4.4); or c) the SAS address is valid for an STP target port in an STP/SATA bridge, but the initial Register - Device to Host FIS has not been successfully received (see 10.4.3.7).
OPEN_REJECT (PATHWAY BLOCKED) [b]	Expander phy	An expander device determined the pathway was blocked by higher priority connection requests.
OPEN_REJECT (RESERVED CONTINUE 0) [c]	Unknown	Reserved. Process the same as OPEN_REJECT (RETRY).
OPEN_REJECT (RESERVED CONTINUE 1) [c]	Unknown	Reserved. Process the same as OPEN_REJECT (RETRY).
OPEN_REJECT (RESERVED INITIALIZE 0) [a]	Unknown	Reserved. Process the same as OPEN_REJECT (NO DESTINATION).
OPEN_REJECT (RESERVED INITIALIZE 1) [a]	Unknown	Reserved. Process the same as OPEN_REJECT (NO DESTINATION).
OPEN_REJECT (RESERVED STOP 0) [b]	Unknown	Reserved. Process the same as OPEN_REJECT (PATHWAY BLOCKED).
OPEN_REJECT (RESERVED STOP 1) [b]	Unknown	Reserved. Process the same as OPEN_REJECT (PATHWAY BLOCKED).
OPEN_REJECT (RETRY) [c]	Destination phy	Phy with destination SAS address exists but is not able to accept connections.

[a] If the I_T Nexus Loss timer is already running, it continues running; if it is not already running, it is initialized and started. Stop retrying the connection request if the I_T Nexus Loss timer expires.
[b] If the I_T Nexus Loss timer is already running, it continues running. Stop retrying the connection request if the I_T Nexus Loss timer expires.
[c] If the I_T Nexus Loss timer (see 8.2.2) is already running, it is stopped.

NOTE 18 - Some SAS phys also transmit OPEN_REJECT (RETRY) if they receive an OPEN address frame while their SL_CC state machines are in the SL_CC5:BreakWait state (see 7.14.4.7).

When a SAS phy detects more than one reason to transmit an OPEN_REJECT, the SL_CC state machine determines the priority in the SL_CC2:Selected state (see 7.14.4.4).

When an expander phy detects more than one reason to transmit an OPEN_REJECT, the ECM determines the priority (see 7.12.4).

See 7.12 for details on connection requests.

7.2.5.12 SOAF (Start of address frame)

SOAF specifies the start of an address frame.

See 7.8 for details on address frames.

7.2.6 Primitives used only inside SSP and SMP connections

7.2.6.1 ACK (Acknowledge)

ACK specifies the positive acknowledgement of an SSP frame.

See 7.16.3 for details on SSP frame transmission.

7.2.6.2 CREDIT_BLOCKED

CREDIT_BLOCKED specifies that no more credit is going to be sent during this connection.

See 7.16.4 for details on SSP flow control.

7.2.6.3 DONE

DONE is used to start closing an SSP connection and specify a reason for doing so. This primitive may be originated by an SSP initiator port or an SSP target port. DONE is not used to close an SMP or STP connection.

The versions of DONE representing different reasons are defined in table 83. The SSP state machine describes when these are used (see 7.16.8).

Table 83 — DONE primitives

Primitive	Description
DONE (ACK/NAK TIMEOUT)	The SSP state machine (see 7.16.8) timed out waiting for an ACK or NAK, and the phy is going to transmit BREAK in 1 ms unless DONE is received within 1 ms of transmitting the DONE (ACK/NAK TIMEOUT).
DONE (RESERVED TIMEOUT 0)	Reserved. Processed the same as DONE (ACK/NAK TIMEOUT).
DONE (RESERVED TIMEOUT 1)	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)	The SSP state machine (see 7.16.8) timed out waiting for an RRDY or received a CREDIT BLOCKED, and the phy is going to transmit BREAK if it provides transmit frame credit for 1 ms without receiving a frame or a DONE.

See 7.16.7 for details on closing SSP connections.

7.2.6.4 EOF (End of frame)

EOF specifies the end of an SSP or SMP frame.

See 7.16.3 for details on SSP frame transmission and 7.18.1 for details on SMP frame transmission.

7.2.6.5 NAK (Negative acknowledgement)

NAK specifies the negative acknowledgement of an SSP frame and the reason for doing so.

The versions of NAK representing different reasons are defined in table 84.

Table 84 — NAK primitives

Primitive	Description
NAK (CRC ERROR)	The frame had a bad CRC, or an invalid dword or an ERROR was received during frame reception.
NAK (RESERVED 0)	Reserved. Processed the same as NAK (CRC ERROR).
NAK (RESERVED 1)	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.

7.2.6.6 RRDY (Receiver ready)

RRDY is used to increase SSP frame credit.

The versions of RRDY representing different reasons are defined in table 85.

Table 85 — 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).

A phy shall not transmit RRDY after transmitting CREDIT_BLOCKED in a connection. See 7.16.4 for details on SSP flow control.

7.2.6.7 SOF (Start of frame)

SOF specifies the start of an SSP or SMP frame.

See 7.16.3 for details on SSP frame transmission and 7.18.1 for details on SMP frame transmission.

7.2.7 Primitives used only inside STP connections and on SATA physical links

7.2.7.1 SATA_ERROR

SATA_ERROR should be sent by an expander device when it is forwarding dwords from a SAS physical link to a SATA physical link and it receives an invalid dword or an ERROR.

> NOTE 19 - Since an 8b10b coding error in one dword is sometimes not detected until the next dword (see table 56 in 6.3.5), expander devices should avoid deleting invalid dwords or ERRORs unless necessary (e.g., if the elasticity buffer is full) to avoid hiding evidence that an error has occurred.

See 6.9 for details on error handling by expander devices.

Although included in this subclause, SATA_ERROR is not a primitive (see 3.1.144) since it starts with K28.6. It does not appear inside STP connections. It is an invalid dword.

7.2.7.2 SATA_PMACK, SATA_PMNAK, SATA_PMREQ_P, and SATA_PMREQ_S (Power management acknowledgements and requests)

SATA_PMREQ_P and SATA_PMREQ_S request entry into the interface power management partial and slumber states. SATA_PMACK is used to accept a power management request. SATA_PMNAK is used to reject a power management request.

See 7.10 for rules on handling the power management primitives.

7.2.7.3 SATA_HOLD and SATA_HOLDA (Hold and hold acknowledge)

See 7.17.3 for rules on STP flow control, which uses SATA_HOLD and SATA_HOLDA.

7.2.7.4 SATA_R_RDY and SATA_X_RDY (Receiver ready and transmitter ready)

When a SATA port has a frame to transmit, it transmits SATA_X_RDY and waits for SATA_R_RDY before transmitting the frame. Expander devices shall not transmit SATA_R_RDY or SATA_X_RDY on the SATA physical link until the STP connection is established.

7.2.7.5 Other primitives used inside STP connections and on SATA physical links

Other primitives used in STP connections and on SATA physical links are defined in SATA.

7.3 Clock skew management

The internal clock for a device is typically based on a PLL with its own clock generator and is used when transmitting dwords on the physical link. When receiving, however, dwords need to be latched based on a clock derived from the input bit stream itself. Although the input clock is nominally a fixed frequency, it may differ slightly from the internal clock frequency up to the physical link rate tolerance defined in table 41 (see 5.3.3). Over time, if the input clock is faster than the internal clock, the device may receive a dword and not be able to forward it to an internal buffer; this is called an overrun. If the input clock is slower than the internal clock, the device may not have a dword when needed in an internal buffer; this is called an underrun.

To solve this problem, transmitting devices insert ALIGNs or NOTIFYs in the dword stream. Receivers may pass ALIGNs and NOTIFYs through to their internal buffers, or may strip them out when an overrun occurs. Receivers add ALIGNs or NOTIFYs when an underrun occurs. The internal logic shall ignore all ALIGNs and NOTIFYs that arrive in the internal buffers.

Elasticity buffer circuitry, as shown in figure 132, is required to absorb the slight differences in frequencies between the SAS initiator phy, SAS target phy, and expander phys. The frequency tolerance for a phy is specified in 5.3.3. The depth of the elasticity buffer is vendor-specific but shall accommodate the clock skew management ALIGN insertion requirements in table 86.
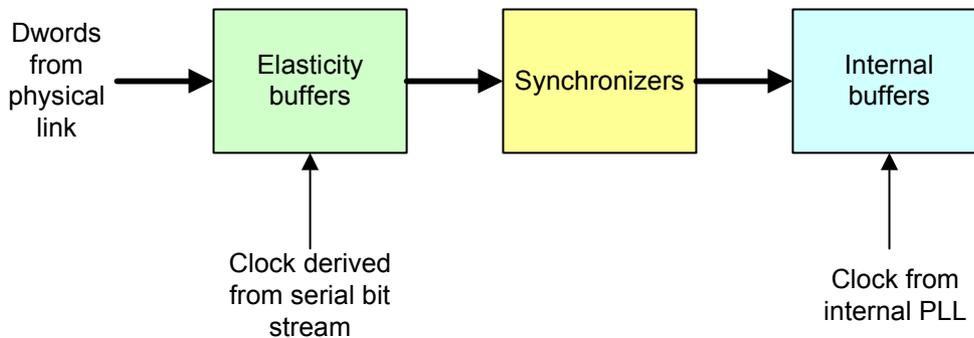
Figure 132 — Elasticity buffers

A phy that is the original source for the dword stream (i.e., a phy that is not an expander phy forwarding dwords from another expander phy) shall insert one ALIGN or NOTIFY for clock skew management as described in table 86.

Table 86 — Clock skew management ALIGN insertion requirement

Physical link rate	Requirement
1,5 Gbit/s	One ALIGN or NOTIFY within every 2 048 dwords
3,0 Gbit/s	Two ALIGNs or NOTIFYs within every 4 096 dwords

ALIGNs and NOTIFYs inserted for clock skew management are in addition to ALIGNs and NOTIFYs inserted for rate matching (see 7.13) and STP initiator phy throttling (see 7.17.2). See Annex H for a summary of their combined requirements.

See 7.2.5.2 for details on rotating through ALIGN (0), ALIGN (1), ALIGN (2), and ALIGN (3). NOTIFYs may also be used in place of ALIGNs (see 7.2.5.9) on SAS physical links.

An expander device that is forwarding dwords (i.e., is not the original source) is allowed to insert or delete as many ALIGNs and/or NOTIFYs as required to match the transmit and receive connection rates. It is not required to transmit the number of ALIGNs and/or NOTIFYs for clock skew management described in table 86 when forwarding to a SAS physical link. It may increase or reduce that number based on clock frequency differences between the phy transmitting the dwords to the expander device and the expander device's receiving phy.

> NOTE 20 - One possible implementation for expander devices forwarding dwords is for the expander device to delete all ALIGNs and NOTIFYs received and to insert ALIGNs and/or NOTIFYs at the transmit port whenever its elasticity buffer is empty.

The STP target port of an STP/SATA bridge is allowed to insert or delete as many ALIGNs and/or NOTIFYs as required to match the transmit and receive connection rates. It is not required to transmit any particular number of ALIGNs and/or NOTIFYs for clock skew management when forwarding to a SAS physical link and is not required to ensure that any ALIGNs and/or NOTIFYs it transmits are in pairs.

> NOTE 21 - Due to clock skew ALIGN and NOTIFY removal, the STP target port may not receive a pair of ALIGNs and/or NOTIFYs every 256 dwords, even though the STP initiator port transmitted at least one pair. However, the rate of the dword stream allows for ALIGN or NOTIFY insertion by the STP/SATA bridge. One possible implementation is for the STP/SATA bridge to delete all ALIGNs and NOTIFYs received by the STP target port and to insert two consecutive ALIGNs at the SATA host port when its elasticity buffer is empty or when 254 non-ALIGN dwords have been transmitted. It may need to buffer up to 2 dwords concurrently being received by the STP target port while it does so.

7.4 Idle physical links

Idle dwords are vendor-specific data dwords which are scrambled (see 7.6).

Phys shall transmit idle dwords if there are no other dwords to transmit and:

   a)   no connection is open; or
   b)   an SSP or SMP connection is open.

SATA_SYNC is a continued primitive sequence which may contain vendor-specific data dwords (see 7.2.4.4) which are scrambled (see 7.6) during an STP connection.

7.5 CRC

7.5.1 CRC overview

All frames include cyclic redundancy check (CRC) values to help detect transmission errors.

Frames transmitted in an STP connection shall include a CRC as defined by SATA (see ATA/ATAPI-7 V3). Address frames, SSP frames, and SMP frames shall include a CRC as defined by this standard.

Annex D contains information on CRC generation/checker implementation.

Table 87 defines the CRC polynomials.

Table 87 — CRC polynomials

Function	Definition
F(x)	A polynomial representing the frame contents which are 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 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 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 quotient polynomial calculated during CRC generation by the transmitter. 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)	A polynomial representing the transmitted frame followed by the transmitted CRC.
M'(x)	A polynomial representing the received frame followed by the received CRC.
C(x)	A unique polynomial remainder produced by the receiver upon reception of an error free sequence. This polynomial has the value: $$C(x) = x^{32} \times \frac{L(x)}{G(x)}$$ $C(x) = x^{31} + x^{30} + x^{26} + x^{25} + x^{24} + x^{18} + x^{15} + x^{14} + x^{12} + x^{11} + x^{10} + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$ (i.e., C(x) = C704DD7Bh)

7.5.2 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 22 - Adding L(x) (all ones) to R(x) produces the one's complement of R(x). This means that R(x) is inverted before it is transmitted.

The transmitter shall calculate the CRC by inverting the first 32 bytes of F(x) and dividing by G(x) to obtain the remainder R(x):

$$\frac{(x^{32} \times F(x)) + (x^K \times L(x))}{G(x)} = Q(x) + \frac{R(x)}{G(x)}$$

The CRC value is R(x) inverted. The CRC value is appended to the end of F(x) for transmission:

$$M(x) = (x^{32} \times F(x)) + L(x) + R(x)$$

The bit order of F(x) presented to the CRC function is the same order as the bit transmission order (i.e., the bits within each byte encoded into a data dword are transposed to match the implicit transposition in the 8b10b encoding process). This order is shown in figure 133.
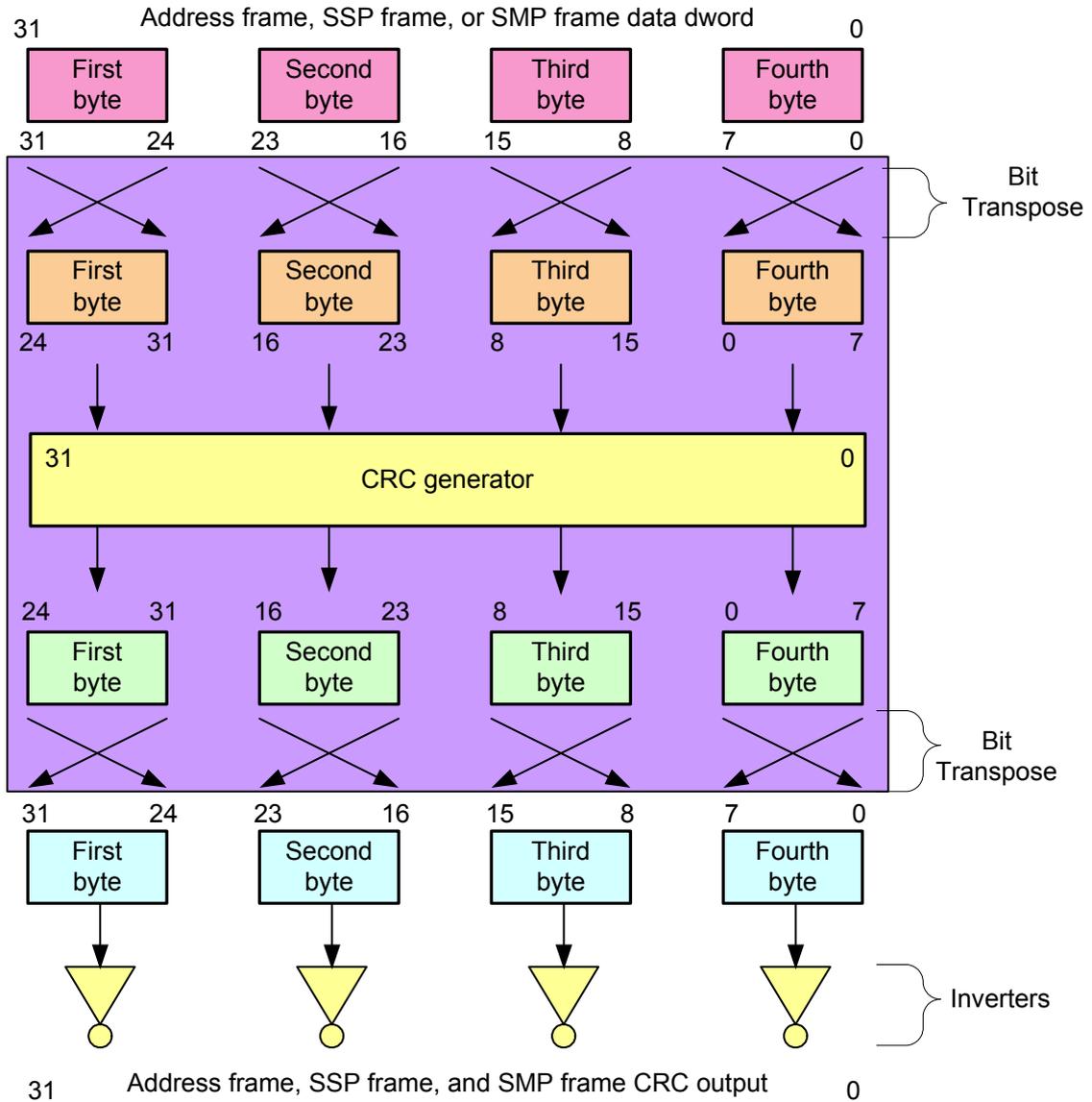
Figure 133 — Address frame, SSP frame, and SMP frame CRC bit ordering

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 134 shows the STP CRC bit ordering.
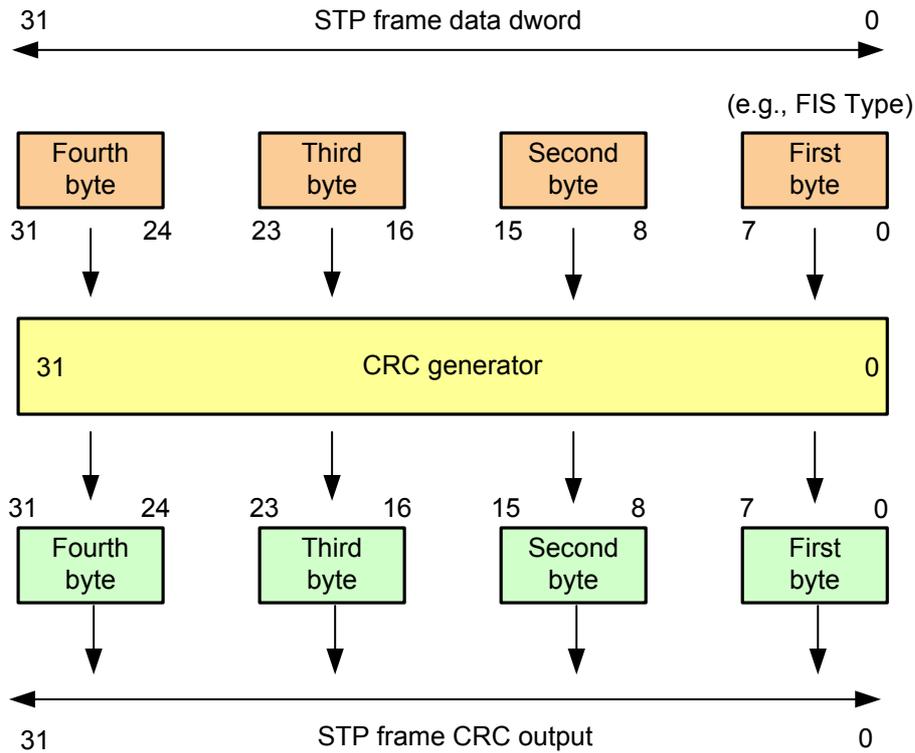
Figure 134 — STP frame CRC bit ordering

Since STP is little-endian, the first byte of a dword is in bits 7:0 rather than 31:24 as in SSP and SMP. Thus, the first byte contains the least-significant bit. In SSP and SMP, the first byte contains the most-significant bit.

See 7.7 for details on how the CRC generator fits into the dword flow along with the scrambler.

7.5.3 CRC checking

M'(x) (i.e., the polynomial representing the received frame followed by the received CRC) may differ from M(x) (i.e., the polynomial representing the transmitted frame followed by the transmitted CRC) 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. The received sequence checking is shown by the following equation:

$$x^{32} \times \frac{M'(x) + (x^K \times L(x))}{G(x)} = Q'(x) + \frac{P(x)}{G(x)}$$

In the absence of errors in both the frame and the CRC, the unique remainder is the remainder of the division (i.e., C(x)) as shown by the following equation:

$$\frac{P(x)}{G(x)} = x^{32} \times \frac{L(x)}{G(x)} = C(x)$$

The bit order of F(x) presented to the CRC checking function is the same order as the CRC generation bit order (see figure 133). See 7.7 for details on where the CRC checker fits into the dword flow along with the descrambler.

7.6 Scrambling

Scrambling is used to reduce the probability of long strings of repeated patterns appearing on the physical link.

All data dwords are scrambled. Table 88 lists the scrambling for different types of data dwords.

Table 88 — Scrambling for different data dword types

Connection state	Data dword type	Description of scrambling
Outside connections	SAS idle dword	When a connection is not open and there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
	Address frame	After an SOAF, all data dwords shall be scrambled until the EOAF.
Inside SSP connection	SSP frame	After an SOF, all data dwords shall be scrambled until the EOF.
	SSP idle dword	When there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
Inside SMP connection	SMP frame	After an SOF, all data dwords shall be scrambled until the EOF.
	SMP idle dword	When there are no other dwords to transmit, vendor-specific scrambled data dwords shall be transmitted.
Inside STP connection	STP frame	After a SATA_SOF, all data dwords shall be scrambled until the SATA_EOF.
	Continued primitive	After a SATA_CONT, vendor-specific scrambled data dwords shall be sent until a primitive other than ALIGN or NOTIFY is transmitted.

Data dwords being transmitted shall be XORed with a defined pattern to produce a scrambled value encoded and transmitted on the physical link. Received data dwords shall be XORed with the same pattern after decoding to produce the original data dword value, provided there are no transmission errors.

The pattern that is XORed with the data dwords is defined by the output of a linear feedback shift register implemented with the following polynomial:

$$G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1.$$

The output of the pattern generator is 16 bits wide. For each data dword the output of the generator is applied to the lower 16 bits (i.e., bits 15 through 0) of the 32-bit data dword being transmitted or received; the next output of the generator is applied to the upper 16 bits (i.e., bits 31 through 16).

NOTE 23 - Scrambling is not based on data feedback, so the sequence of values XORed with the data being transmitted is constant.

The value of the linear feedback shift register shall be initialized at each SOF and SOAF to FFFFh.

For detailed requirements about scrambling of data dwords following SATA_SOF and SOF_CONT, see ATA/ATAPI-7 V3.

NOTE 24 - STP scrambling uses two linear feedback shift registers, since continued primitive sequences may occur inside STP frames and the STP frame and the continued primitive sequence have independent scrambling patterns.

Annex F contains information on scrambling implementations.

7.7 Bit order of CRC and scrambler

Figure 135 shows how data dwords and primitives are routed to the bit transmission logic in figure 109 (see 6.5). Data dwords go through the CRC generator and scrambler.
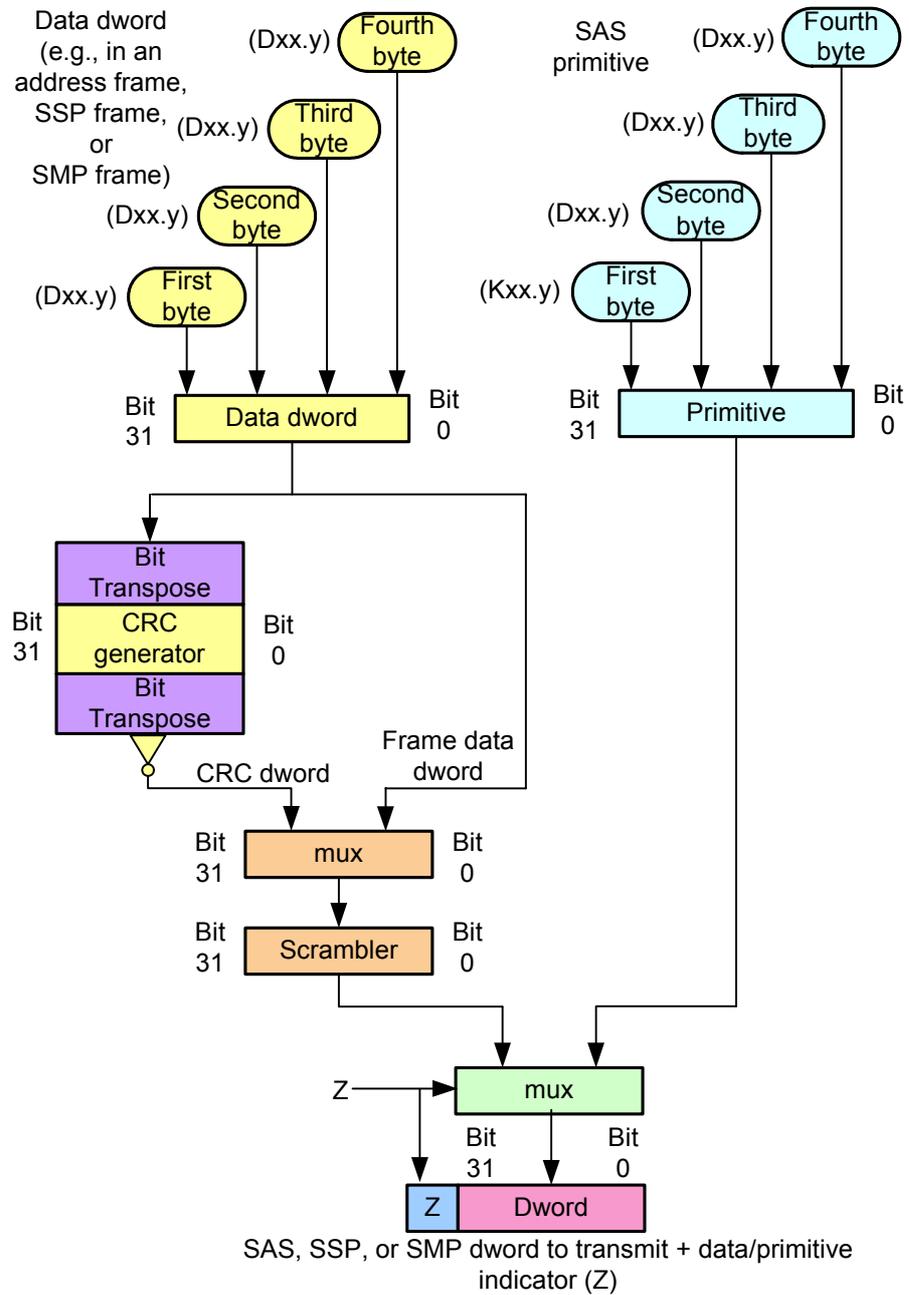
Figure 135 — Transmit path bit ordering

Figure 136 shows the routing of dwords received from the bit reception logic in figure 110 (see 6.5).
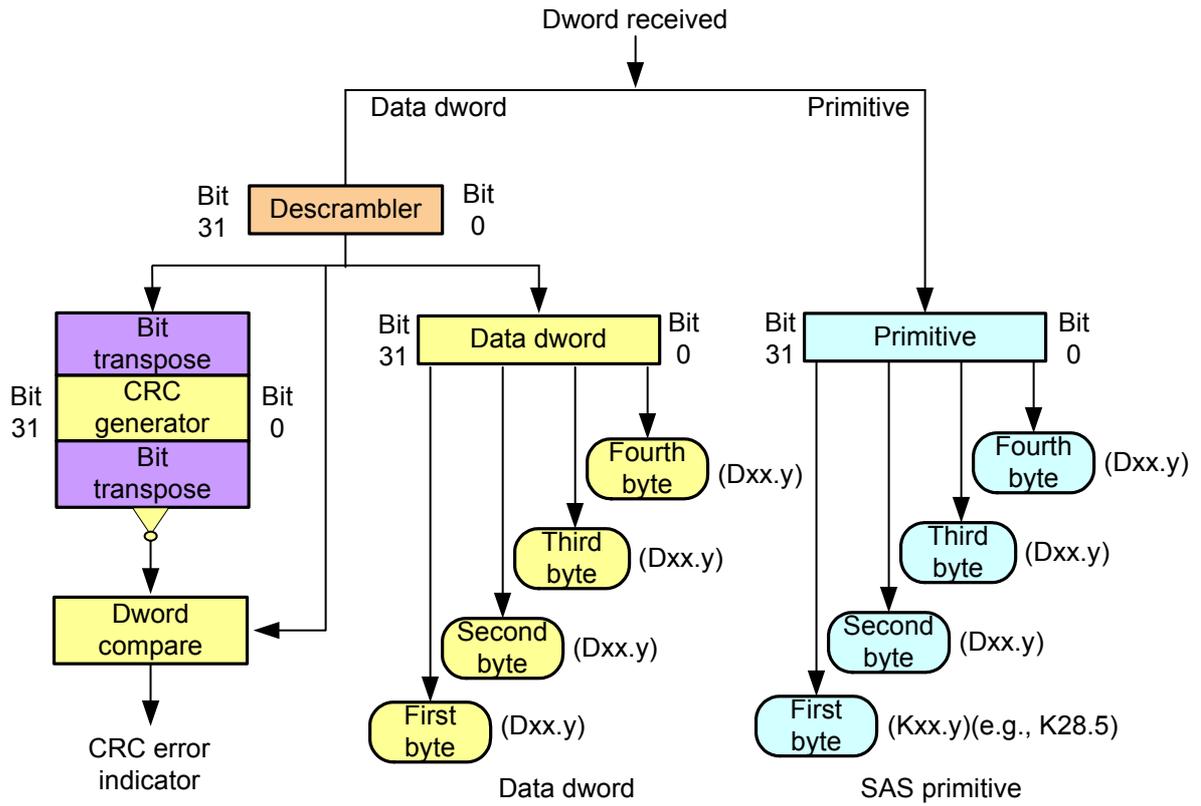
Figure 136 — Receive path bit ordering

Figure 137 shows the STP transmit path bit ordering.
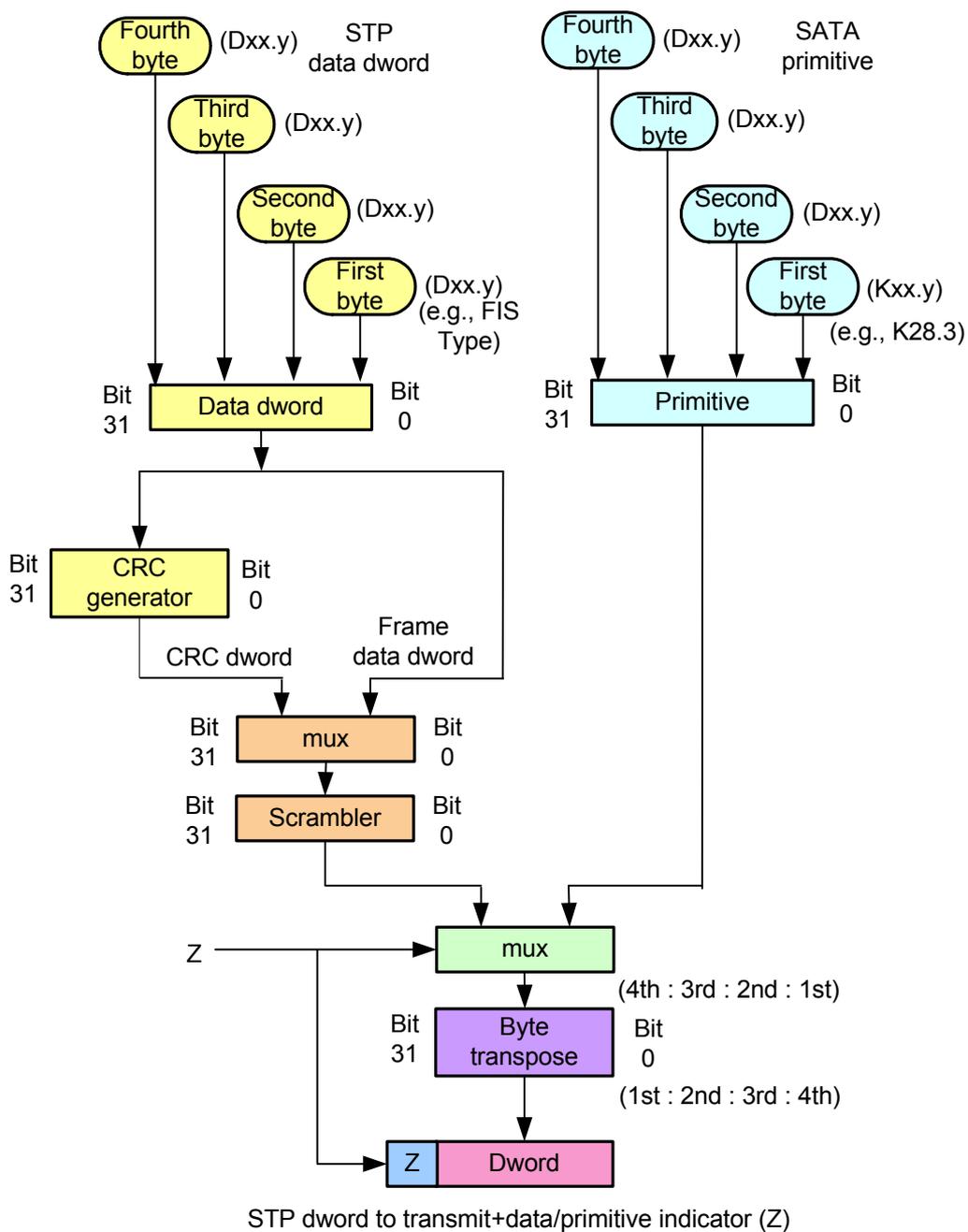
Figure 137 — STP transmit path bit ordering

Figure 138 shows the STP receive path bit ordering.
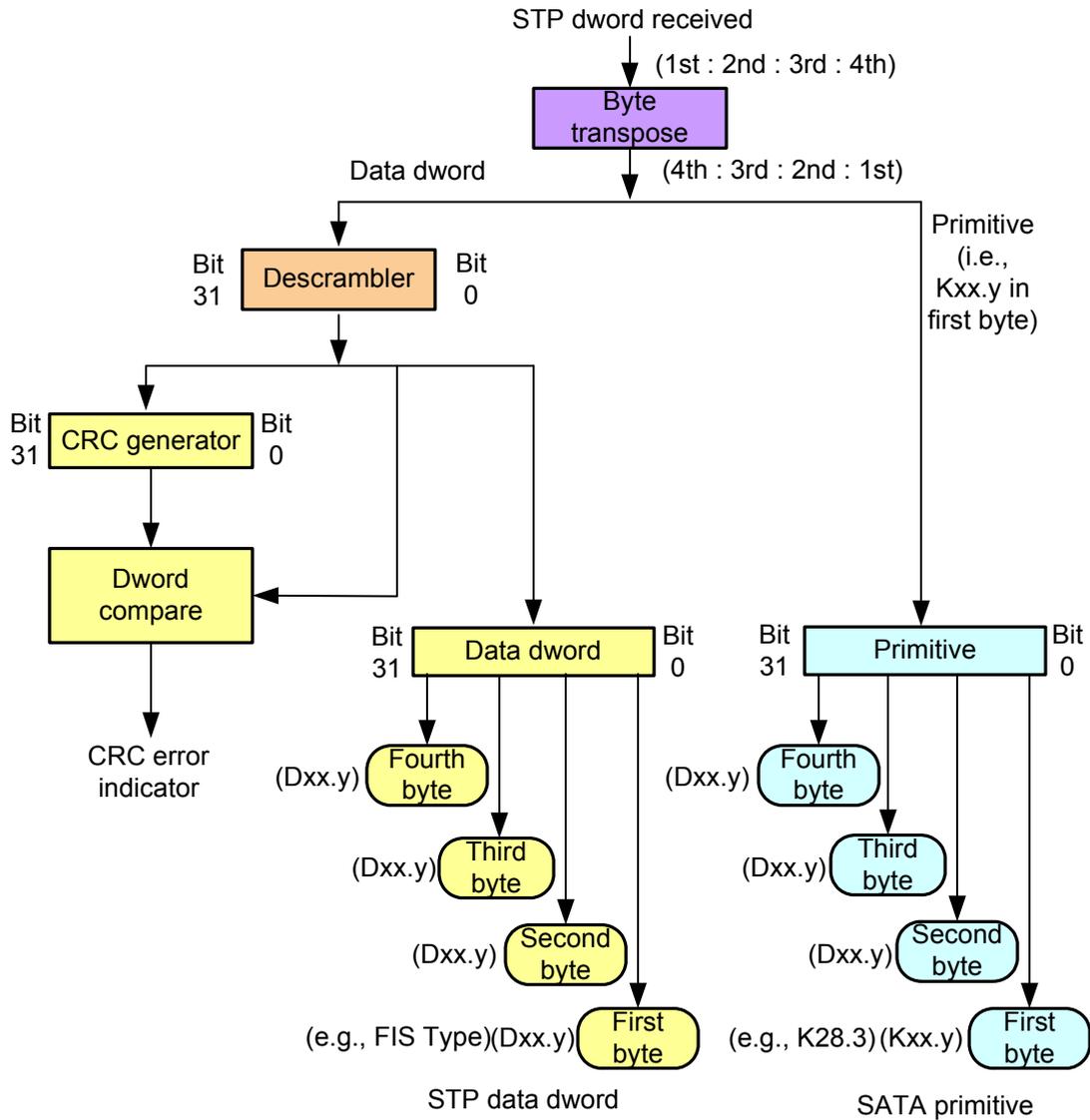
Figure 138 — STP receive path bit ordering

7.8 Address frames

7.8.1 Address frames overview

Address frames are used for the identification sequence and for connection requests.

Address frames are preceded by SOAF and followed by EOAF as shows in figure 139.
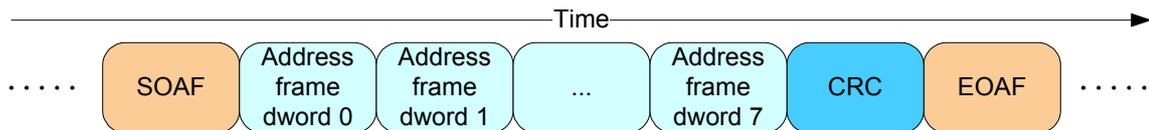
Figure 139 — Address frame transmission

Address frames shall only be sent outside connections. Address frames shall not be terminated early. All data dwords in an address frame shall be scrambled.

Table 89 defines the address frame format.

Table 89 — Address frame format

Byte\Bit	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 specifies the type of address frame and is defined in table 90. This field determines the definition of the frame type dependent bytes.

Table 90 — ADDRESS FRAME TYPE **field**

Code	Frame type	Description
0h	Identify	Identification sequence
1h	Open	Connection request
All others	Reserved	

The CRC field contains a CRC value (see 7.5) that is computed over the entire address frame prior to the CRC field.

Address frames with unknown address frame types, incorrect lengths, or CRC errors shall be ignored by the recipient.

7.8.2 IDENTIFY address frame

Table 91 defines the IDENTIFY address frame format used for the identification sequence. The IDENTIFY address frame is sent after the phy reset sequence completes if the physical link is a SAS physical link.

Table 91 — IDENTIFY address frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	Restricted (for OPEN address frame)	DEVICE TYPE			ADDRESS FRAME TYPE (0h)			
1	Restricted (for OPEN address frame)							
2	Reserved				SSP INITIATOR PORT	STP INITIATOR PORT	SMP INITIATOR PORT	Restricted (for OPEN address frame)
3	Reserved				SSP TARGET PORT	STP TARGET PORT	SMP TARGET PORT	Restricted (for OPEN address frame)
4 ... 11	Restricted (for OPEN address frame)							
12 ... 19	SAS ADDRESS							
20	PHY IDENTIFIER							
21 ... 27	Reserved							
28	(MSB)							
31	CRC							(LSB)

The DEVICE TYPE field specifies the type of device containing the phy, and is defined in table 92.

Table 92 — DEVICE TYPE field

Code	Description
001b	End device
010b	Edge expander device
011b	Fanout expander device
All others	Reserved

The ADDRESS FRAME TYPE field shall be set to 0h.

An SSP INITIATOR PORT bit set to one specifies that an SSP initiator port is present. An SSP INITIATOR PORT bit set to zero specifies that an SSP initiator port is not present. Expander devices shall set the SSP INITIATOR PORT bit to zero.

An STP INITIATOR PORT bit set to one specifies that an STP initiator port is present. An STP INITIATOR PORT bit set to zero specifies that an STP initiator port is not present. Expander devices shall set the STP INITIATOR PORT bit to zero.

An SMP INITIATOR PORT bit set to one specifies that an SMP initiator port is present. An SMP INITIATOR PORT bit set to zero specifies that an SMP initiator port is not present. Expander devices may set the SMP INITIATOR PORT bit to one.

An SSP TARGET PORT bit set to one specifies that an SSP target port is present. An SSP TARGET PORT bit set to zero specifies that an SSP target port is not present. Expander devices shall set the SSP TARGET PORT bit to zero.

An STP TARGET PORT bit set to one specifies that an STP target port is present. An STP TARGET PORT bit set to zero specifies that an STP target port is not present. Expander devices shall set the STP TARGET PORT bit to zero.

An SMP TARGET PORT bit set to one specifies that an SMP target port is present. An SMP TARGET PORT bit set to zero specifies that an SMP target port is not present. Expander devices shall set the SMP TARGET PORT bit to one.

For SAS ports, the SAS ADDRESS field specifies the port identifier (see 4.2.6) of the SAS port transmitting the IDENTIFY address frame. For expander ports, the SAS ADDRESS field specifies the device name (see 4.2.4) of the expander device transmitting the IDENTIFY address frame.

The PHY IDENTIFIER field specifies the phy identifier of the phy transmitting the IDENTIFY address frame.

See 4.1.3 for additional requirements concerning the DEVICE TYPE field, SSP INITIATOR PORT bit, STP INITIATOR PORT bit, SMP INITIATOR PORT bit, SSP TARGET PORT bit, STP TARGET PORT bit, SMP TARGET PORT bit, and SAS ADDRESS field.

The CRC field is defined in 7.8.1.

7.8.3 OPEN address frame

Table 93 defines the OPEN address frame format used for connection requests.

Table 93 — OPEN address frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	INITIATOR PORT	PROTOCOL			ADDRESS FRAME TYPE (1h)			
1	FEATURES				CONNECTION RATE			
2	(MSB)	INITIATOR CONNECTION TAG						
3								(LSB)
4		DESTINATION SAS ADDRESS						
11								
12		SOURCE SAS ADDRESS						
19								
20	COMPATIBLE FEATURES							
21	PATHWAY BLOCKED COUNT							
22	(MSB)	ARBITRATION WAIT TIME						
23								(LSB)
24		MORE COMPATIBLE FEATURES						
27								
28	(MSB)	CRC						
31								(LSB)

An INITIATOR PORT bit set to one specifies that the source port is acting as a SAS initiator port. An INITIATOR PORT bit set to zero specifies that the source port is acting as a SAS target port. If a SAS target/initiator port sets the INITIATOR PORT bit to one, it shall operate only in its initiator role during the connection. If a target/initiator port sets the INITIATOR PORT bit to zero, it shall operate only in its target role during the connection.

If a SAS target/initiator port accepts an OPEN address frame with the INITIATOR PORT bit set to one, it shall operate only in its target role during the connection. If a SAS target/initiator port accepts an OPEN address frame with the INITIATOR PORT bit set to zero, it shall operate only in its initiator role during the connection.

The PROTOCOL field specifies the protocol for the connection being requested and is defined in table 94.

Table 94 — PROTOCOL field

Code	Description
000b	SMP
001b	SSP
010b	STP
All others	Reserved

The ADDRESS FRAME TYPE field shall be set to 1h.

The FEATURES field shall be set to zero.

The CONNECTION RATE field specifies the connection rate (see 4.1.10) being requested between the source and destination, and is defined in table 95.

Table 95 — CONNECTION RATE **field**

Code	Description
8h	1,5 Gbit/s
9h	3,0 Gbit/s
All others	Reserved

When requesting a connection to a SAS target port, a SAS initiator port shall set the CONNECTION RATE field to a value supported by at least one potential pathway.

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 SAS target port should send frames in a connection to the SAS initiator port regardless of whether the saved connection rate for that command matches the current connection rate; the SAS 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 STP initiator port unless the STP target port has received an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED).

The INITIATOR CONNECTION TAG field is used for SSP and STP connection requests to provide a SAS initiator port an alternative to using the SAS target port's SAS address for context lookup when the SAS target port originates a connection request. SSP or STP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh if they do not require the field be provided by the SAS target port. If they do require the field to be provided, an SSP or STP initiator port should set the INITIATOR CONNECTION TAG field to a unique value per SAS target port. When requesting a connection to a SAS initiator port, a SAS target port shall set the INITIATOR CONNECTION TAG field to the most recent value received or the value received in one of the connection requests for one of the outstanding commands or task management functions from the SAS initiator port. A SAS initiator port shall use the same INITIATOR CONNECTION TAG field value for all connection requests to the same SAS target port, and shall only change the INITIATOR CONNECTION TAG field value when it has no commands or task management functions outstanding to that SAS target port. SAS target ports are not required to check consistency of the INITIATOR CONNECTION TAG field in different connection requests from the same SAS initiator port. SMP initiator ports shall set the INITIATOR CONNECTION TAG field to FFFFh for SMP connection requests.

The DESTINATION SAS ADDRESS field specifies the port identifier (see 4.2.6) of the SAS port to which a connection is being requested.

The SOURCE SAS ADDRESS field specifies the port identifier (see 4.2.6) of the SAS port that originated the OPEN address frame.

The COMPATIBLE FEATURES field shall be set to zero. The destination device shall ignore the COMPATIBLE FEATURES field.

The PATHWAY BLOCKED COUNT field specifies 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 ARBITRATION WAIT TIME field specifies how long the port transmitting the OPEN address frame has been waiting for a connection request to be accepted or rejected. This time is maintained by the port layer in an Arbitration Wait Time timer (see 8.2.2). For values from 0000h to 7FFFh, the Arbitration Wait Time timer increments in one microsecond steps. For values from 8000h to FFFFh, the Arbitration Wait Time timer increments in one millisecond

steps. The maximum value represents 32 767 ms + 32 768 µs. Table 96 describes several values of the ARBITRATION WAIT TIME field. See 7.12.3 for details on arbitration fairness.

Table 96 — ARBITRATION WAIT TIME **field**

Code	Description
0000h	0 µs
0001h	1 µs
...	...
7FFFh	32 767 µs
8000h	0 ms + 32 768 µs
8001h	1 ms + 32 768 µs
...	...
FFFFh	32 767 ms + 32 768 µs

The MORE COMPATIBLE FEATURES field shall be set to zero. The destination device shall ignore the MORE COMPATIBLE FEATURES field.

The CRC field is defined in 7.8.1.

7.9 Identification and hard reset sequence

7.9.1 Identification and hard reset sequence overview

After the phy reset sequence has been completed indicating the physical link is using SAS rather than SATA, each phy transmits either:

a) an IDENTIFY address frame (see 7.8.2); or
b) a HARD_RESET primitive sequence.

Each phy receives an IDENTIFY address frame or a HARD_RESET primitive sequence 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 (see 4.4.1).

If a phy receives a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, the SAS address in the outgoing IDENTIFY address frame and the SAS address in the incoming IDENTIFY address frame determine the port to which a phy belongs (see 4.1.3). The phy ignores subsequent IDENTIFY address frames and HARD_RESET primitives until another phy reset sequence occurs.

If a phy receives a HARD_RESET primitive sequence within 1 ms of phy reset sequence completion, it shall be considered a reset event and cause a hard reset (see 4.4.2) of the port containing that phy.

If a phy does not receive a HARD_RESET primitive sequence or a valid IDENTIFY address frame within 1 ms of phy reset sequence completion, it shall restart the phy reset sequence.

7.9.2 SAS initiator device rules

After a link reset sequence, or after receiving a BROADCAST (CHANGE), a management application client behind an SMP initiator port should perform a discover process (see 4.7).

When a discover process is performed after a link reset sequence, the management application client discovers all the devices in the SAS domain. When a discover process is performed after a BROADCAST (CHANGE), the management application client determines which devices have been added to or removed from the SAS domain.

The discover information may be used to select connection rates for connection requests (see 7.8.3).

7.9.3 Fanout expander device rules

After completing the identification sequence on a phy and completing internal initialization, the ECM within a fanout expander device shall be capable of routing connection requests through that phy. The expander device may return OPEN_REJECT (NO DESTINATION) until it is ready to process connection requests.

After a link reset sequence, or after receiving a BROADCAST (CHANGE), the management application client behind an SMP initiator port in a fanout expander device that does not have a configurable expander route table shall follow the SAS initiator device rules (see 7.9.2) to perform a discover process.

The ECM of a fanout expander device that has a configurable expander route table is dependent on the completion of the discover process (see 4.7) for routing connection requests using the table routing method.

7.9.4 Edge expander device rules

After completing the identification sequence on a phy and completing internal initialization, the ECM within an edge expander device shall be capable of routing connection requests through that phy. The expander device may return OPEN_REJECT (NO DESTINATION) until it is ready to process connection requests.

The ECM of an edge expander device that has a configurable expander route table is dependent on the completion of the discover process (see 4.7) for routing connection requests using the table routing method.

7.9.5 SL_IR (link layer identification and hard reset) state machines

7.9.5.1 SL_IR state machines overview

The SL_IR (link layer identification and hard reset) state machines control the flow of dwords on the physical link that are associated with the identification and hard reset sequences. The state machines are as follows:

    a) SL_IR_TIR (transmit IDENTIFY or HARD_RESET) state machine (see 7.9.5.3);
    b) SL_IR_RIF (receive IDENTIFY address frame) state machine (see 7.9.5.4); and
    c) SL_IR_IRC (identification and hard reset control) state machine (see 7.9.5.5).

The SL_IR state machines send the following messages to the SL state machines (see 7.14) in SAS devices or the XL (see 7.15) state machine in expander devices:

    a) Enable Disable SAS Link (Enable); and
    b) Enable Disable SAS Link (Disable).

The SL_IR_IRC state machine shall maintain the timers listed in table 97.

Table 97 — SL_IR_IRC timers

Timer	Initial value
Receive Identify Timeout timer	1 ms

Figure 140 shows the SL_IR state machines.
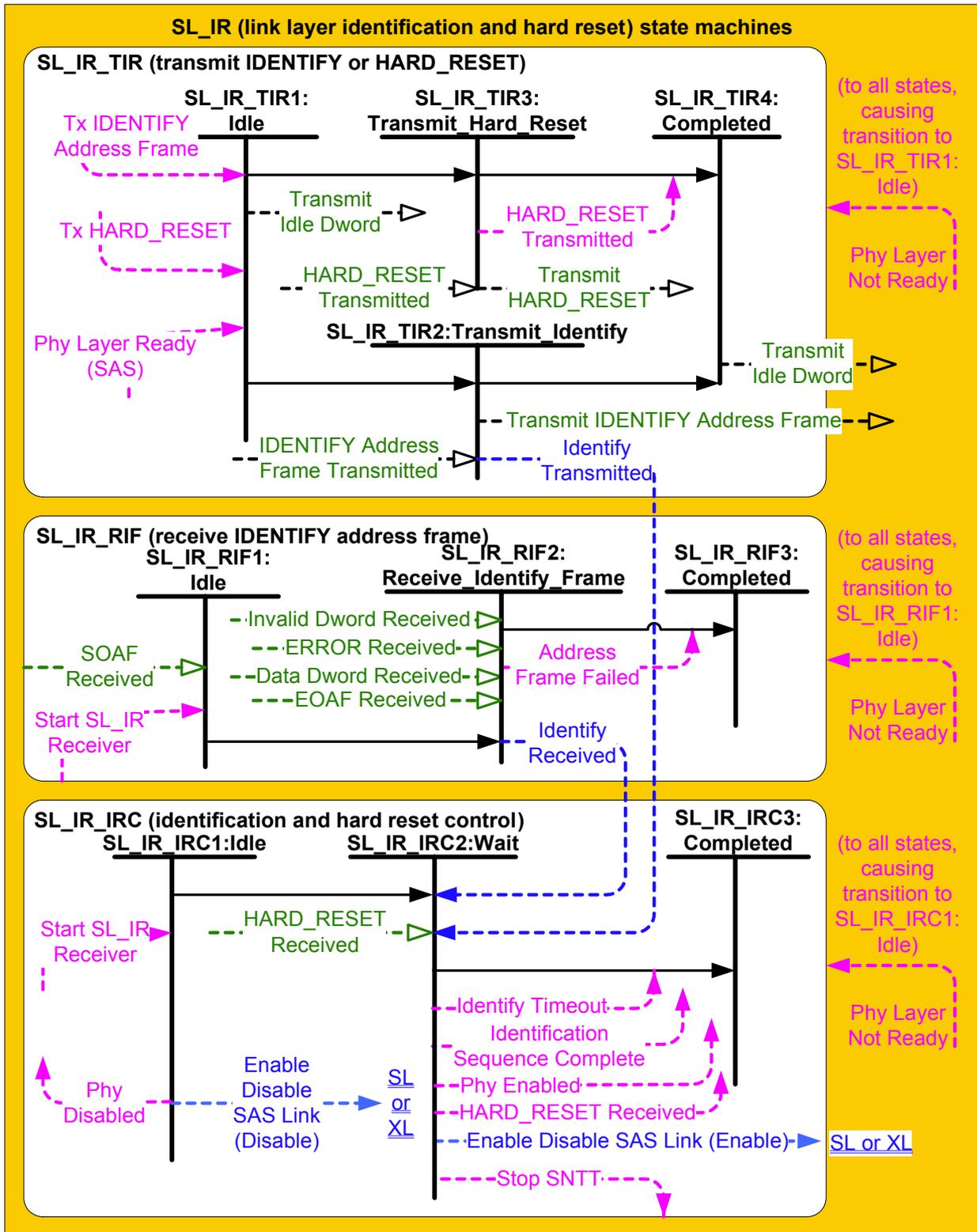
Figure 140 — SL_IR (link layer identification and hard reset) state machines

7.9.5.2 SL_IR transmitter and receiver

The SL_IR transmitter receives the following messages from the SL_IR state machines indicating primitive sequences, frames, and dwords to transmit:

    a)   Transmit IDENTIFY Address Frame;
    b)   Transmit HARD_RESET; and

c) Transmit Idle Dword.

The SL_IR transmitter sends the following messages to the SL_IR state machines:

a) HARD_RESET Transmitted; and
b) IDENTIFY Address Frame Transmitted.

The SL_IR receiver sends the following messages to the SL_IR state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

a) SOAF Received;
b) Data Dword Received;
c) EOAF Received;
d) ERROR Received;
e) Invalid Dword Received; and
f) HARD_RESET Received.

The SL_IR receiver shall ignore all other dwords.

7.9.5.3 SL_IR_TIR (transmit IDENTIFY or HARD_RESET) state machine

7.9.5.3.1 SL_IR_TIR state machine overview

The SL_IR_TIR state machine's function is to transmit a single IDENTIFY address frame or HARD_RESET primitive after the phy layer enables the link layer. This state machine consists of the following states:

a) SL_IR_TIR1:Idle (see 7.9.5.3.2)(initial state);
b) SL_IR_TIR2:Transmit_Identify (see 7.9.5.3.3);
c) SL_IR_TIR3:Transmit_Hard_Reset (see 7.9.5.3.4); and
d) SL_IR_TIR4:Completed (see 7.9.5.3.5).

This state machine shall start in the SL_IR_TIR1:Idle state. This state machine shall transition to the SL_IR_TIR1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.5.3.2 SL_IR_TIR1:Idle state

7.9.5.3.2.1 State description

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL_IR transmitter.

7.9.5.3.2.2 Transition SL_IR_TIR1:Idle to SL_IR_TIR2:Transmit_Identify

This transition shall occur after both:

a) a Phy Layer Ready (SAS) confirmation is received; and
b) a Tx IDENTIFY Address Frame request is received.

7.9.5.3.2.3 Transition SL_IR_TIR1:Idle to SL_IR_TIR3:Transmit_Hard_Reset

This transition shall occur after both:

a) a Phy Layer Ready (SAS) confirmation is received; and
b) a Tx HARD_RESET request is received.

7.9.5.3.3 SL_IR_TIR2:Transmit_Identify state

7.9.5.3.3.1 State description

Upon entry into this state, this state shall send a Transmit IDENTIFY Address Frame message to the SL_IR transmitter.

After this state receives an IDENTIFY Address Frame Transmitted message, this state shall send an Identify Transmitted message to the SL_IR_IRC state machine.

7.9.5.3.3.2 Transition SL_IR_TIR2:Transmit_Identify to SL_IR_TIR4:Completed

This transition shall occur after sending an Identify Transmitted message to the SL_IR_IRC state machine.

7.9.5.3.4 SL_IR_TIR3:Transmit_Hard_Reset state

7.9.5.3.4.1 State description

Upon entry into this state, this state shall send a Transmit HARD_RESET message to the SL_IR transmitter.

After this state receives a HARD_RESET Transmitted message, this state shall send a HARD_RESET Transmitted confirmation to the management application layer.

7.9.5.3.4.2 Transition SL_IR_TIR3:Transmit_Hard_Reset to SL_IR_TIR4:Completed

This transition shall occur after sending a HARD_RESET Transmitted confirmation to the management application layer.

7.9.5.3.5 SL_IR_TIR4:Completed state

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL_IR transmitter.

7.9.5.4 SL_IR_RIF (receive IDENTIFY address frame) state machine

7.9.5.4.1 SL_IR_RIF state machine overview

The SL_IR_RIF state machine receives an IDENTIFY address frame and checks the IDENTIFY address frame to determine if the frame should be accepted or discarded by the link layer.

This state machine consists of the following states:

    a) SL_IR_RIF1:Idle (see 7.9.5.4.2)(initial state);
    b) SL_IR_RIF2:Receive_Identify_Frame (see 7.9.5.4.3); and
    c) SL_IR_RIF3:Completed (see 7.9.5.4.4).

This state machine shall start in the SL_IR_RIF1:Idle state. This state machine shall transition to the SL_IR_RIF1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.5.4.2 SL_IR_RIF1:Idle state

7.9.5.4.2.1 State description

This state waits for an SOAF to be received from the physical link, indicating an address frame is arriving.

7.9.5.4.2.2 Transition SL_IR_RIF1:Idle to SL_IR_RIF2:Receive_Identify_Frame

This transition shall occur after both:

    a) a Start SL_IR Receiver confirmation is received; and
    b) an SOAF Received message is received.

7.9.5.4.3 SL_IR_RIF2:Receive_Identify_Frame state

7.9.5.4.3.1 State description

This state receives the dwords of an address frame and the EOAF.

If this state receives an SOAF Received message, then this state shall discard the address frame (i.e., the subsequent Data Dword Received and EOAF Received messages) and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

If this state receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state shall discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOAF Received message and before an EOAF Received message, then this state shall:

    a) ignore the invalid dword or ERROR; or

    b)   discard the address frame and send an Address Frame Failed confirmation to the management application layer to indicate that an invalid IDENTIFY address frame was received.

After receiving an EOAF Received message, this state shall check if it the received frame is a valid IDENTIFY address frame.

This state shall accept an IDENTIFY address frame and send an Identify Received message to the SL_IR_IRC state machine if:

    a)   the ADDRESS FRAME TYPE field is set to 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 invalid IDENTIFY address frame was received.

7.9.5.4.3.2 Transition SL_IR_RIF2:Receive_Identify_Frame to SL_IR_RIF3:Completed

This transition shall occur after sending an Identify Received message or Address Frame Failed confirmation.

7.9.5.4.4 SL_IR_RIF3:Completed state

This state waits for a Phy Layer Not Ready confirmation.

7.9.5.5 SL_IR_IRC (identification and hard reset control) state machine

7.9.5.5.1 SL_IR_IRC state machine overview

The SL_IR_IRC state machine ensures that IDENTIFY address frames have been both received and transmitted before enabling the rest of the link layer, and notifies the link layer if a HARD_RESET primitive sequence is received before an IDENTIFY address frame has been received.

This state machine consists of the following states:

    a)   SL_IR_IRC1:Idle (see 7.9.5.5.2)(initial state);
    b)   SL_IR_IRC2:Wait (see 7.9.5.5.3); and
    c)   SL_IR_IRC3:Completed (see 7.9.5.5.4).

This state machine shall start in the SL_IR_IRC1:Idle state. This state machine shall transition to the SL_IR_IRC1:Idle state from any other state after receiving a Phy Layer Not Ready confirmation.

7.9.5.5.2 SL_IR_IRC1:Idle state

7.9.5.5.2.1 State description

This state waits for the link layer to be enabled. Upon entry into this state, this state shall:

    a)   send an Enable Disable SAS Link (Disable) message to SL state machines (see 7.14) or XL state machine (see 7.15) halting any link layer activity; and
    b)   send a Phy Disabled confirmation to the port layer and the management application layer indicating that the phy is not ready for use.

7.9.5.5.2.2 Transition SL_IR_IRC1:Idle to SL_IR_IRC2:Wait

This transition shall occur after a Start SL_IR Receiver confirmation is received.

7.9.5.5.3 SL_IR_IRC2:Wait state

7.9.5.5.3.1 State description

This state ensures that an IDENTIFY address frame has been received by the SL_IR_RIF state machine and that a IDENTIFY address frame has been transmitted by the SL_IR_TIR state machine before enabling the rest of the link layer. The IDENTIFY address frames may be transmitted and received on the physical link in any order.

After this state receives an Identify Received message, it shall send a Stop SNTT request to the phy layer.

After this state receives an Identify Transmitted message, it shall initialize and start the Receive Identify Timeout timer. If an Identify Received message is received before the Receive Identify Timeout timer expires, this state shall:

a) send an Identification Sequence Complete confirmation to the management application layer, with arguments carrying the contents of the incoming IDENTIFY address frame;

b) send an Enable Disable SAS Link (Enable) message to the SL state machines (see 7.14) in a SAS phy or the XL state machine (see 7.15) in an expander phy indicating that the rest of the link layer may start operation; and

c) send a Phy Enabled confirmation to the port layer and the management application layer indicating that the phy is ready for use.

If the Receive Identify Timeout timer expires before an Identify Received message is received, this state shall send an Identify Timeout confirmation to the management application layer to indicate that an identify timeout occurred.

If this state receives a HARD_RESET Received message before an Identify Received message is received, this state shall send a HARD_RESET Received confirmation to the port layer and a Stop SNTT request to the phy layer.

If this state receives a HARD_RESET Received message after an Identify Received message is received, the HARD_RESET Received message shall be ignored.

7.9.5.5.3.2 Transition SL_IR_IRC2:Wait to SL_IR_IRC3:Completed

This transition shall occur after sending a HARD_RESET Received confirmation, Identify Timeout confirmation, or an Identification Sequence Complete and an Phy Enabled confirmation.

7.9.5.5.4 SL_IR_IRC3:Completed state

This state waits for a Phy Layer Not Ready confirmation.

7.10 Power management

SATA interface power management is not supported in STP.

STP initiator ports shall not generate SATA_PMREQ_P, SATA_PMREQ_S, or SATA_PMACK. If an STP initiator port receives SATA_PMREQ_P or SATA_PMREQ_S, it shall reply with SATA_PMNAK.

If an expander device receives SATA_PMREQ_P or SATA_PMREQ_S from a SATA device while an STP connection is not open, it shall not forward it to any STP initiator port and shall reply with SATA_PMNAK. If one of these primitives arrives while an STP connection is open, it may forward the primitive to the STP initiator port.

SCSI idle and standby power conditions, implemented with the START STOP UNIT command (see SBC-2) and the Power Condition mode page (see SPC-3), may be supported by SSP initiator ports and SSP target ports as described in 10.2.10.

ATA idle and standby power modes, implemented with the IDLE, IDLE IMMEDIATE, STANDBY, STANDBY IMMEDIATE, and CHECK POWER MODE commands (see ATA/ATAPI-7 V1), may be supported by STP initiator ports. The ATA sleep power mode, implemented with the SLEEP command, shall not be used.

7.11 SAS domain changes

After power on or receiving BROADCAST (CHANGE), the management application client in each SAS initiator port should scan the SAS domain using the discover process (see 4.7) to search for SAS initiator devices, SAS target devices, and expander devices.

The expander device shall 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's SP state machine transitions from the SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready state to the SP0:OOB_COMINIT state (see 6.8);

NOTE 25 - This occurs when the expander phy is reset or disabled with the SMP PHY CONTROL function DISABLE, LINK RESET, HARD RESET, or TRANSMIT SATA PORT SELECTION SIGNAL phy operations (see 10.4.3.10) as well as when dword synchronization is unexpectedly lost;

b) after a virtual phy has been disabled with the SMP PHY CONTROL function DISABLE phy operation or started processing a reset requested by the LINK RESET or HARD RESET phy operations (see 10.4.3.10);

c) after an expander phy's SP state machine reaches the SATA spinup hold state (see 6.8.7and 6.10);

d) after an expander phy's SP state machine receives a COMWAKE Detected message in states SP0:OOB_COMINIT, SP1:OOB_AwaitCOMX, SP3:OOB_AwaitCOMINIT_Sent, or SP4:OOB_COMSAS if the value of the ATTACHED SATA PORT SELECTOR bit in the DISCOVER response is zero prior to receiving the COMWAKE detected message (see 6.8.3 and table 196 in 10.4.3.5);

e) after an expander phy's SP state machine transitions from the SP1:OOB_AwaitCOMX state to the SP0:OOB_COMINIT state if the value of the ATTACHED SATA PORT SELECTOR bit was one in the DISCOVER response upon entry to SP1:OOB_AwaitCOMX, and if no COMWAKE detected message was received while in SP1:OOB_AwaitCOMX before the transition to SP0:OOB_COMINIT (see 6.8.3.3.2);

f) after the link reset sequence completes (see 7.9);

g) after a virtual phy has been enabled or completed processing a reset requested by the SMP PHY CONTROL function LINK RESET or HARD RESET phy operations (see 10.4.3.10);

h) after a self-configuring expander device has completed configuration and has changed its CONFIGURING bit from one to zero in the SMP REPORT GENERAL function (see 10.4.3.3);

i) after an STP/SATA bridge receives an initial Register - Device to host FIS (see 9.3.1); and

j) after an expander phy receives BROADCAST (CHANGE).

For a virtual phy, if there is any time after a reset is originated during which connection requests to the attached SAS address result in connection responses of OPEN_REJECT (NO DESTINATION), the expander device shall transmit the BROADCAST (CHANGE) twice, once at the start of the reset (i.e., when the SAS address becomes unavailable) and once at its completion (i.e., when the SAS address becomes available). If there is no such time window, the expander device shall transmit the BROADCAST (CHANGE) once.

BROADCAST (CHANGE) may be sent by SAS initiator ports to force other SAS initiator ports and expander ports to re-run the discover process, but should not be sent by SAS target ports.

A SAS initiator port that detects BROADCAST (CHANGE) shall follow the SAS initiator device rules (see 7.9.2) to discover the topology.

A fanout expander device that detects BROADCAST (CHANGE) shall follow the fanout device rules (see 7.9.3) to discover the topology.

An edge expander device that detects BROADCAST (CHANGE) shall follow the edge device rules (see 7.9.4).

See 10.4.3.3 for details on counting BROADCAST (CHANGE) generation in an expander device.

7.12 Connections

7.12.1 Connections overview

A connection is opened between a SAS initiator port and a SAS target port before communication begins. A connection is established between one SAS initiator phy in the SAS initiator port and one SAS target phy in the SAS target port.

SSP initiator ports open SSP connections to transmit SCSI commands, task management functions, or transfer read data. SSP target ports open SSP connections to transfer write data or transmit status.

SMP initiator ports open SMP connections to transmit SMP requests and receive SMP responses.

STP initiator ports and STP target ports open STP connections to transmit SATA frames. An STP target port in an expander device opens STP connections on behalf of SATA devices.

The OPEN address frame is used to request that a connection be opened (see 7.12.2.1). AIP, OPEN_ACCEPT and OPEN_REJECT are the responses to an OPEN address frame (see 7.12.2.2). BREAK is used to abort connection requests (see 7.12.5) and to unilaterally break a connection (see 7.12.7). CLOSE is used for orderly closing a connection (see 7.12.6).

Connections use a single pathway from the SAS initiator phy to the SAS target phy. While a connection is open, only one pathway shall be used for that connection.

For STP connections, connections may be between the STP initiator port and an STP target port of an STP/SATA bridge in an expander device. The SATA device behind the STP/SATA bridge is not aware of SAS connection management.

A wide port may have separate connections on each of its phys.

7.12.2 Opening a connection

7.12.2.1 Connection request

The OPEN address frame (see 7.8.3) is used to open a connection from a source port to a destination port using one source phy and one destination phy.

To make a connection request, the source port shall transmit an OPEN address frame through an available phy. The source phy shall transmit idle dwords after the OPEN address frame until it receives a response or aborts the connection request with BREAK.

After transmitting an OPEN address frame, the source phy shall initialize and start a 1 ms Open Timeout timer. Whenever an AIP is received, the source phy shall reinitialize and restart the Open Timeout timer. Source phys are not required to enforce a limit on the number of AIPs received before aborting the connection request. When any connection response is received, the source phy shall reinitialize the Open Timeout timer. If the Open Timeout timer expires before a connection response is received, the source phy shall transmit BREAK to abort the connection request (see 7.12.5).

The OPEN address frame flows through expander devices onto intermediate physical links. If an expander device on the pathway is unable to forward the connection request, it returns OPEN_REJECT (see 7.12.4). If the OPEN address frame reaches the destination, it returns either OPEN_ACCEPT or OPEN_REJECT unless the OPEN address frame passed an OPEN address frame from the destination with higher arbitration priority (see 7.12.3). Rate matching shall be used on any physical links in the pathway with negotiated physical link rates that are faster than the requested connection rate (see 7.13).

7.12.2.2 Results of a connection request

After a phy transmits an OPEN address frame, it shall expect one or more of the results listed in table 98.

Table 98 — Connection Results of a connection request

Result	Description
Receive AIP	Arbitration in progress. When an expander device is trying to open a connection to the selected destination port, it returns an AIP to the source phy. The source phy shall reinitialize and restart its Open Timeout timer each time it receives an AIP. AIP is sent by an expander device while it is internally arbitrating for access to an expander port.
Receive OPEN_ACCEPT	Connection request accepted. OPEN_ACCEPT is transmitted by the destination phy.
Receive OPEN_REJECT	Connection request rejected. OPEN_REJECT is transmitted by the destination phy or by an expander device in the partial pathway. The different versions are described in 7.2.5.11. See 4.5 for I_T nexus loss handling.
Receive 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. Arbitration fairness determines which one wins (see 7.12.3).
Receive BREAK	The destination phy or an expander device in the partial pathway may reply with BREAK indicating the connection is not being established.
Open Timeout timer expires	The source phy shall abort the connection request by transmitting BREAK (see 7.12.5). See 4.5 for I_T nexus loss handling.

After an OPEN_REJECT (CONNECTION RATE NOT SUPPORTED) has been received by a SAS target port, the SAS target device shall set the connection rate for future requests for that I_T_L_Q nexus to:

    a) the last value received in a connection request from the SAS initiator port;
    b) 1,5 Gbit/s; or

c) the connection rate in effect when the command was received.

7.12.3 Arbitration fairness

SAS supports least-recently used arbitration fairness for connection requests.

Each SAS port and expander port shall include an Arbitration Wait Time timer which counts the time from the moment when the port makes a connection request until the request is accepted or rejected. The Arbitration Wait Time timer is in the port layer state machine (see 8.2.2). The Arbitration Wait Time timer shall count in microseconds from 0 μs to 32 767 μs and in milliseconds from 32 768 μs to 32 767 ms + 32 768 μs. The Arbitration Wait Time timer shall stop incrementing when its value reaches 32 767 ms + 32 768 μs.

A SAS port (i.e., SAS initiator ports and SAS target ports) shall start the Arbitration Wait Time timer when it transmits the first OPEN address frame (see 7.8.3) for the connection request. When the SAS port retransmits the OPEN address frame (e.g., after losing arbitration and handling an inbound OPEN address frame), it shall set the ARBITRATION WAIT TIME field to the current value of the Arbitration Wait Time timer.

A SAS port should set the Arbitration Wait Time timer to zero when it transmits the first OPEN address frame for the connection request. A SAS initiator port or SAS target port may be unfair by setting the ARBITRATION WAIT TIME field in the OPEN address frame (see 7.8.3) to a higher value than its Arbitration Wait Time timer indicates. However, an unfair SAS port shall not set the ARBITRATION WAIT TIME field to a value greater than or equal to 8000h; this limits the amount of unfairness and helps prevent livelocks.

The expander port that receives an OPEN address frame shall set the Arbitration Wait Time timer to the value of the incoming ARBITRATION WAIT TIME field and start the Arbitration Wait Time timer as it arbitrates for internal access to the outgoing expander port. When the expander device transmits the OPEN address frame out another expander port, it shall set the outgoing ARBITRATION WAIT TIME field to the current value of the Arbitration Wait Time timer maintained by the incoming expander port.

A SAS port shall stop the Arbitration Wait Time timer and set it to zero when it has no more frames to send.

A SAS port shall stop the Arbitration Wait Time timer and set it to zero when it receives one of the following connection responses:

  a) OPEN_ACCEPT;
  b) OPEN REJECT (PROTOCOL NOT SUPPORTED);
  c) OPEN_REJECT (STP RESOURCES BUSY);
  d) OPEN_REJECT (WRONG DESTINATION); or
  e) OPEN_REJECT (RETRY).

  NOTE 26 - Connection responses that are conclusively from the destination phy (see table 81 and table 82 in 7.2.5.11) are included in the list. Connection responses that are only from or may be from expander phys are not included.

A SAS port should not stop the Arbitration Wait Time timer and set it to zero when it receives an incoming OPEN address frame that has priority over the outgoing OPEN address frame according to table 99, regardless of whether it replies with an OPEN_ACCEPT or an OPEN_REJECT.

When arbitrating for access to an outgoing expander port, the expander device shall select the connection request based on the rules described in 7.12.4.

If two connection requests pass on a physical link, the phy shall determine the winner by comparing OPEN address frame field contents using the arbitration priority described in table 99.

Table 99 — Arbitration priority for OPEN address frames passing on a physical link

Bits 79-64 (79 is MSB)	Bits 63-0 (0 is LSB)
ARBITRATION WAIT TIME field value	SOURCE SAS ADDRESS field value

See 7.8.3 for details on the OPEN address frame and the ARBITRATION WAIT TIME field.

7.12.4 Arbitration and resource management in an expander device

7.12.4.1 Arbitration and resource management in an expander device overview

The ECM shall arbitrate and assign or deny path resources for Request Path requests from each expander phy.

Arbitration includes adherence to the SAS arbitration fairness algorithm and path recovery. Path recovery is used to avoid potential deadlock scenarios within the SAS topology by deterministically choosing which partial pathway(s) to tear down to allow at least one connection to complete.

The ECM responds to connection requests by returning an Arb Won, Arb Lost, or Arb Reject confirmation to the requesting expander phy.

Several of the Request Path arguments are used for arbitration. The Arbitration Wait Time, Source SAS Address, and Connection Rate arguments are filled in from the received OPEN address frame and are used to by the ECM to compare Request Path requests. The Retry Priority Status argument is used to prevent the Arbitration Wait Time argument from being considered during an arbitration which occurs after a Backoff Retry response is sent by an expander phy (see 7.15.4).

An expander phy shall set the Retry Priority Status argument to IGNORE AWT when it requests a path after:

  a) it has forwarded an OPEN address frame to the physical link;
  b) an OPEN address frame is received with higher arbitration priority (see 7.12.3); and
  c) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame (see 7.15.4 and 7.15.9).

Otherwise, the expander phy shall set the Retry Priority Status argument to NORMAL.

If two or more Request Path requests contend and all of the Request Path requests include a Retry Priority Status argument set to NORMAL, the ECM shall select the winner by comparing the OPEN address frame contents described in table 100.

Table 100 — Arbitration priority for contending Request Path requests in the ECM when all requests have Retry Priority Status arguments of NORMAL

Bits 83-68 (83 is MSB)	Bits 67-4	Bits 3-0 (0 is LSB)
ARBITRATION WAIT TIME field value	SOURCE SAS ADDRESS field value	CONNECTION RATE field value

If two or more Request Path requests contend and one or more of the Request Path requests include a Retry Priority Status argument set to IGNORE AWT, the ECM shall select the winner from the set of Request Path requests with Retry Priority Status arguments of IGNORE AWT by comparing the OPEN address frame contents described in table 101.

Table 101 — Arbitration priority for contending Request Path requests in the ECM among requests with Retry Priority Status arguments of IGNORE AWT

Bits 67-4 (67 is MSB)	Bits 3-0 (0 is LSB)
SOURCE SAS ADDRESS field value	CONNECTION RATE field value

The ECM shall generate the Arb Reject confirmation when any of the following conditions are met and all the Arb Won conditions are not met:

  1) Arb Reject (No Destination) or Arb Reject (Bad Destination) if the connection request does not map to an expander phy that is not part of the same expander port as the requesting expander phy (i.e., there is no direct routing or table routing match and there is no subtractive phy)(see 7.12.4.3 and 7.12.4.4);
  2) Arb Reject (Bad Connection Rate) if the connection request does not map to any expander phy that supports the connection rate (i.e., none of the prospective physical links support the requested connection rate); or
  3) Arb Reject (Pathway Blocked) if the connection request maps to expander phys that all contain blocked partial pathways (i.e., are all returning Phy Status (Blocked Partial Pathway)) and pathway recovery rules require this connection request to release path resources (see 7.12.4.6).

The ECM shall generate the Arb Lost confirmation when all of the following conditions are met:

   a) the connection request maps to an expander phy that:
       A) supports the connection rate; and
       B) is not reporting a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response unless that expander phy is arbitrating for the expander phy making this connection request;
   b) there are sufficient routing resources to complete the connection request; and
   c) the destination expander phy of this connection request has received a higher priority OPEN address frame with this expander phy as its destination (i.e., when two expander phys both receive an OPEN address frame destined for each other, the ECM shall provide the Arb Lost confirmation to the expander phy that received the lowest priority OPEN address frame).

The ECM shall generate the Arb Won confirmation when all of the following conditions are met:

   a) the connection request maps to an expander phy that:
       A) supports the connection rate; and
       B) is not reporting a Phy Status (Partial Pathway), Phy Status (Blocked Partial Pathway), or Phy Status (Connection) response, unless that expander phy is arbitrating for the expander phy making this connection request;
   b) there are sufficient routing resources to complete the connection request;
   c) no higher priority connection requests are present with this expander phy as the destination; and
   d) the connection request is chosen as the highest priority connection request in the expander device mapping to the specified destination expander phy.

7.12.4.2 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 recovery.

The arbitration status of an expander phy is set to the last type of AIP received.

Before an expander device transmits AIP, it may have transmitted an OPEN address frame on the same physical link. Arbitration fairness dictates which OPEN address frame wins (see 7.12.3).

After an expander device transmits an AIP, it shall not transmit an OPEN address frame unless it has higher arbitration priority than the incoming connection request.

Expander devices shall transmit no more than three consecutive AIPs without transmitting an idle dword. Expander devices may transmit three consecutive AIPs to provide better tolerance of errors. Expander devices shall transmit at least one AIP every 128 dwords while transmitting AIP (NORMAL), AIP (WAITING ON PARTIAL), or AIP (WAITING ON CONNECTION).

   NOTE 27 - Future versions of this standard may require that expander devices transmit three consecutive AIPs.

Expander devices shall transmit an AIP (e.g., an AIP (NORMAL)) within 128 dwords of receiving an OPEN address frame.

7.12.4.3 Edge expander devices

When the ECM in an edge expander device receives a connection request:

   1) if the destination SAS address is that of the expander device itself, the ECM shall arbitrate for access to its SMP target port and forward the connection request;
   2) if the destination SAS address matches the SAS address of a device to which one of the expander phys is attached, the ECM shall arbitrate for access to one of the matching phys and forward the connection request;
   3) if the destination SAS address matches an enabled SAS address in the expander route table for one or more expander phys which has a table routing attribute (see 4.6.7.1) and is attached to an edge expander device, the ECM shall arbitrate for access to one of the matching phys and forward the connection request; and
   4) if at least one phy has the subtractive routing attribute and is attached to an expander device (i.e., it is attached to an edge expander device or a fanout expander device and the phy is using the subtractive

routing method), and the request did not come from that expander device, the ECM shall arbitrate for access to one of the subtractive routing phys and forward the connection request.

If it does not find a match and no phy using the subtractive routing method is available, the ECM shall reply with Arb Reject (No Destination).

If the destination phy is in the same expander port as the source phy and is using the subtractive routing method, the ECM shall reply with Arb Reject (No Destination).

If the destination phy is in the same expander port as the source phy and is using the direct routing method or the table routing method, the ECM shall reply with either Arb Reject (No Destination) or Arb Reject (Bad Destination); it should reply with Arb Reject (No Destination).

> NOTE 28 - ECMs in edge expander devices compliant with previous versions of this standard were required to reply with Arb Reject (Bad Destination).

7.12.4.4 Fanout expander devices

When the ECM in a fanout expander device receives a connection request:

1) if the destination SAS address is that of the expander device itself, the ECM shall arbitrate for access to its SMP target port and forward the connection request;
2) if the destination SAS address matches the SAS address of a device to which one of the expander phys is attached, the ECM shall arbitrate for access to one of the matching phys and forward the connection request; and
3) if the destination SAS address matches an enabled SAS address in the expander route table for one or more expander phys which has a table routing attribute (see 4.6.7.1) and is attached to an edge expander device, the ECM shall arbitrate for access to one of the matching phys and forward the connection request.

If it does not find a match, the ECM shall reply with Arb Reject (No Destination).

If the destination phy is in the same expander port as the source phy, the ECM shall reply with either Arb Reject (No Destination) or Arb Reject (Bad Destination); it should reply with Arb Reject (No Destination).

> NOTE 29 - ECMs in fanout expander devices compliant with previous versions of this standard were required to reply with Arb Reject (Bad Destination).

7.12.4.5 Partial Pathway Timeout timer

Each expander phy shall maintain a Partial Pathway Timeout timer. This timer is used to identify potential deadlock conditions and to request resolution by the ECM. An expander phy shall initialize the Partial Pathway Timeout timer to the partial pathway timeout value it reports in the SMP DISCOVER function (see 10.4.3.5) and run the Partial Pathway Timeout timer whenever the ECM provides confirmation to the expander phy that all expander phys within the requested destination port are blocked waiting on partial pathways.

> NOTE 30 - The partial pathway timeout value allows flexibility in specifying how long an expander device waits before attempting pathway recovery. The recommended default value (see 10.4.3.5) was chosen to cover a wide range of topologies. Selecting small partial pathway timeout value values within a large topology may compromise performance because of the time a device waits after receiving OPEN_REJECT (PATHWAY BLOCKED) before it retries the connection request. Similarly, selecting large partial pathway timeout value values within a small topology may compromise performance due to waiting longer than necessary to detect pathway blockage.

When the Partial Pathway Timeout timer is not running, an expander phy shall initialize and start the Partial Pathway Timeout timer when all expander phys within the requested destination port contain a blocked partial pathway (i.e., are returning Phy Status (Blocked Partial Pathway)).

> NOTE 31 - The Partial Pathway Timeout timer is not initialized and started if one or more of the expander phys within a requested destination port are being used for a connection.

When one of the conditions above is not met, the expander phy shall stop the Partial Pathway Timeout timer. If the timer expires, pathway recovery shall occur (see 7.12.4.6).

7.12.4.6 Pathway recovery

Pathway recovery provides a means to abort connection requests in order to prevent deadlock using pathway recovery priority comparisons. Pathway recovery priority comparisons compare the PATHWAY BLOCKED COUNT fields

and SOURCE SAS ADDRESS fields of the OPEN address frames of the blocked connection requests as described in table 102.

Table 102 — Pathway recovery priority

Bits 71-64 (71 is MSB)	Bits 63-0 (0 is LSB)
PATHWAY BLOCKED COUNT field value	SOURCE SAS ADDRESS field value

When the Partial Pathway Timeout timer for an arbitrating expander phy expires (i.e., reaches a value of zero), the ECM shall determine whether to continue the connection request or to abort the connection request.

The ECM shall reply to a connection request with Arb Reject (Pathway Blocked) when:

a) the Partial Pathway Timeout timer expires; and
b) the pathway recovery priority of the arbitrating expander phy (i.e., the expander phy requesting the connection) is less than or equal to the pathway recovery priority of any of the expander phys within the destination port that are sending Phy Status (Blocked Partial Pathway) responses to the ECM.

The pathway blocked count and source SAS address values used to form the pathway recovery priority of a destination phy are those of the Request Path request if the phy sent a Request Path request to the ECM or those of the Forward Open indication if the phy received a Forward Open indication from the ECR.

7.12.5 Aborting a connection request

BREAK may be used to abort a connection request. The source phy shall transmit a BREAK after the Open Timeout timer expires or if it chooses to abort its request for any other reason before a connection is established.

After transmitting BREAK, the source phy shall initialize a Break Timeout timer to 1 ms and start the Break Timeout timer.

After a phy transmits a BREAK to abort a connection request, it shall expect one of the results listed in table 103.

Table 103 — Results of aborting a connection request

Result	Description
Receive BREAK	This confirms that the connection request has been aborted.
Break Timeout timer expires	The originating phy shall assume the connection request has been aborted.

When a phy sourcing a BREAK is attached to an expander device, the BREAK response to the source phy is generated by the expander phy to which the source phy is attached, not the other SAS phy in the connection. If the expander device has transmitted a connection request to the destination, it shall also transmit BREAK to the destination. If the expander device has not transmitted a connection request to the destination, it shall not transmit BREAK to the destination. After transmitting BREAK back to the originating phy, the expander device shall ensure that an open response does not occur (i.e., the expander device shall not forward dwords from the destination any more). Figure 141 shows an example of BREAK usage.

Case 1: OPEN address frame has not propagated through the expander device:

OPEN address frame

| Source phy | idle | Expander device | idle | Destination phy |

idle
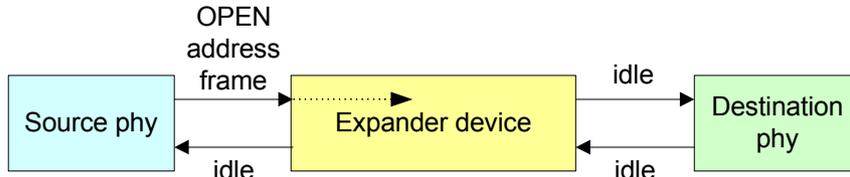
Case 1 result: BREAK only on source device physical link

BREAK

| Source phy | | Expander device | idle | Destination phy |

BREAK   idle

Case 2: OPEN address frame has propagated through the expander device:

OPEN address frame   OPEN address frame

| Source phy | | Expander device | | Destination phy |

idle   idle
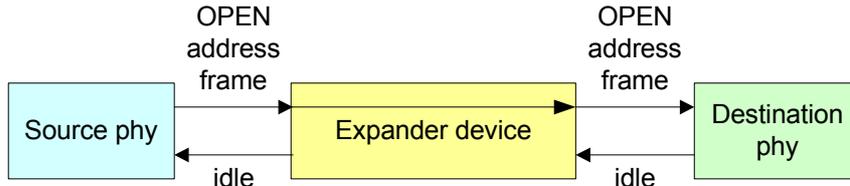
Case 2 result: BREAK on source device's physical link, then on destination device's physical link

BREAK   BREAK

| Source phy | | Expander device | | Destination phy |

BREAK   BREAK

Figure 141 — Aborting a connection request with BREAK

Figure 142 shows the sequence for a connection request where the Open Timeout timer expires.
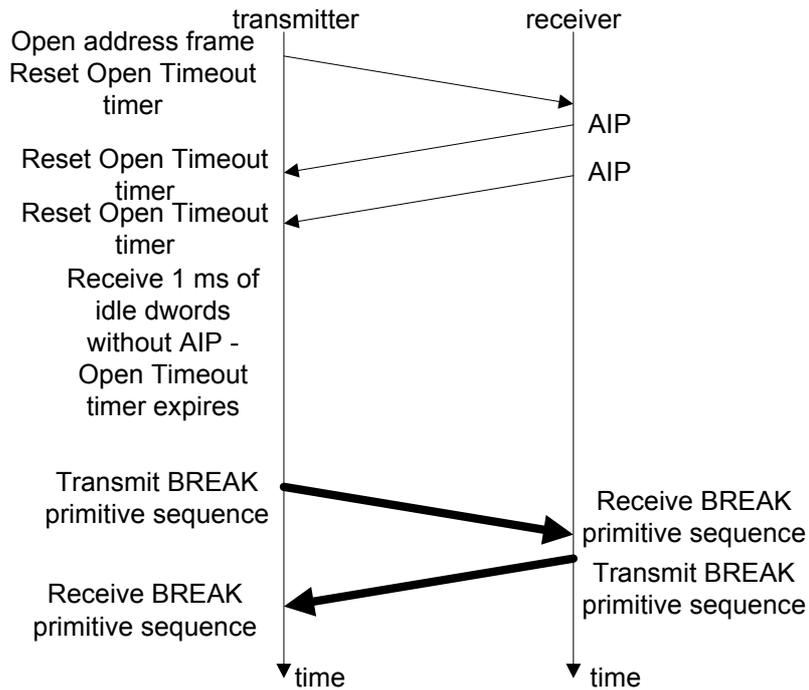
Figure 142 — Connection request timeout example

7.12.6 Closing a connection

CLOSE is used to close a connection of any protocol. See 7.16.7 for details on closing SSP connections, 7.17.7 for details on closing STP connections, and 7.18.3 for details on closing SMP connections.

After transmitting CLOSE, the source phy shall initialize a Close Timeout timer to 1 ms and start the Close Timeout timer.

After a phy transmits CLOSE to close a connection, it shall expect one of the results listed in table 104.

Table 104 — Results of closing a connection

Result	Description
Receive CLOSE	This confirms that the connection has been closed.
Close Timeout timer expires	The originating phy shall attempt to break the connection (see 7.12.7).

No additional dwords for the connection shall follow the CLOSE. Expander devices shall close the full-duplex connection upon detecting a CLOSE in each direction.

When a phy has both transmitted and received CLOSE, it shall consider the connection closed.

Figure 143 shows example sequences for closing a connection.

Example 1: CLOSEs sent one at a time

transmitter   receiver

CLOSE primitive
sequence

CLOSE primitive
sequence

CLOSE primitive
sequence

time   time

Example 2: CLOSEs sent simultaneously

transmitter   receiver

CLOSE primitive   CLOSE primitive
sequence   sequence

time   time
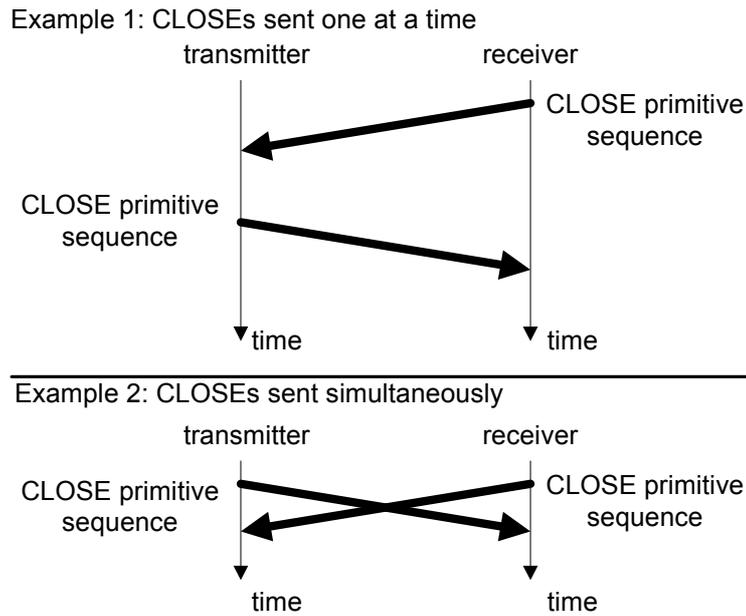
Figure 143 — Closing a connection example

7.12.7 Breaking a connection

In addition to aborting a connection request, BREAK may also be used to break a connection, in cases where CLOSE is not available. After transmitting BREAK, the originating phy shall ignore all incoming dwords except for BREAKs.

After transmitting BREAK, the source phy shall initialize a Break Timeout timer to 1 ms and start the Break Timeout timer.

After a phy transmits a BREAK to break a connection, it shall expect one of the results listed in table 105.

Table 105 — Results of breaking a connection

Result	Description
Receive BREAK	This confirms that the connection has been broken.
Break Timeout timer expires	The originating phy shall assume the connection has been broken. The originating phy may perform a link reset sequence.

In addition to a BREAK, a connection is considered broken if a link reset sequence starts (i.e., the SP state machine transitions from SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)).

See 7.16.6 for additional rules on breaking an SSP connection.

7.13 Rate matching

Each successful connection request contains the connection rate (see 4.1.10) of the pathway.

Each phy in the pathway shall insert ALIGNs and/or NOTIFYs between dwords if its physical link rate is faster than the connection rate as described in table 106.

Table 106 — Rate matching ALIGN and/or NOTIFY insertion requirements

Physical link rate	Connection rate	Requirement
1,5 Gbit/s	1,5 Gbit/s	None
3,0 Gbit/s	1,5 Gbit/s	One ALIGN or NOTIFY within every 2 dwords that are not clock skew management ALIGNs or NOTIFYs (i.e., every overlapping window of 2 dwords)(e.g., a repeating pattern of an ALIGN or NOTIFY followed by a dword or a repeating pattern of a dword followed by an ALIGN or NOTIFY)
	3,0 Gbit/s	None

ALIGNs and NOTIFYs inserted for rate matching are in addition to ALIGNs and NOTIFYs inserted for clock skew management (see 7.3) and STP initiator phy throttling (see 7.17.2). See Annex H for a summary of their combined requirements.

Figure 144 shows an example of rate matching between a 3,0 Gbit/s source phy and a 3,0 Gbit/s destination phy, with an intermediate 1,5 Gbit/s physical link in between them.
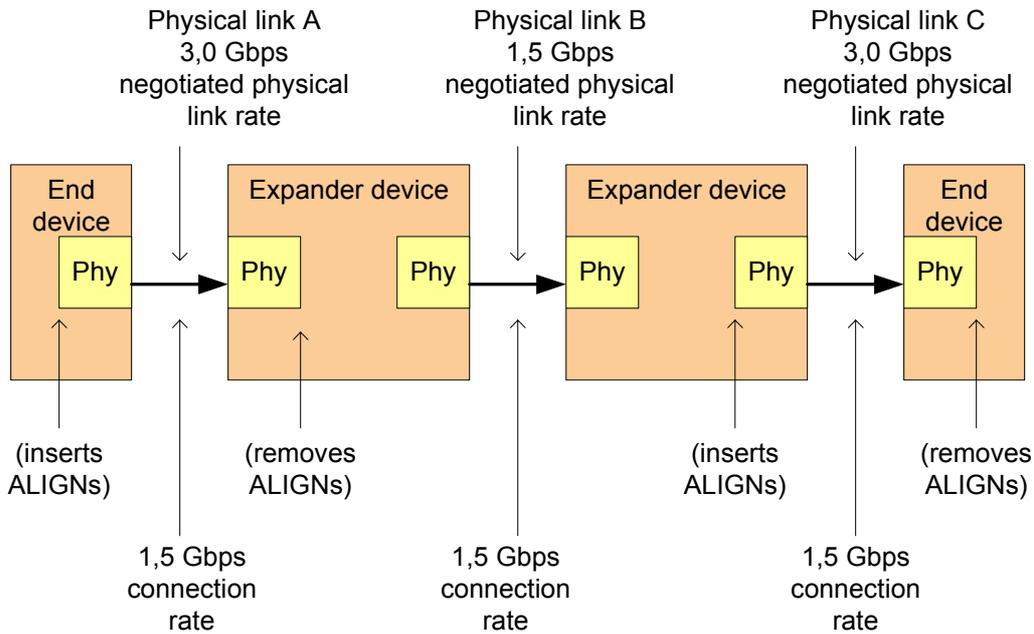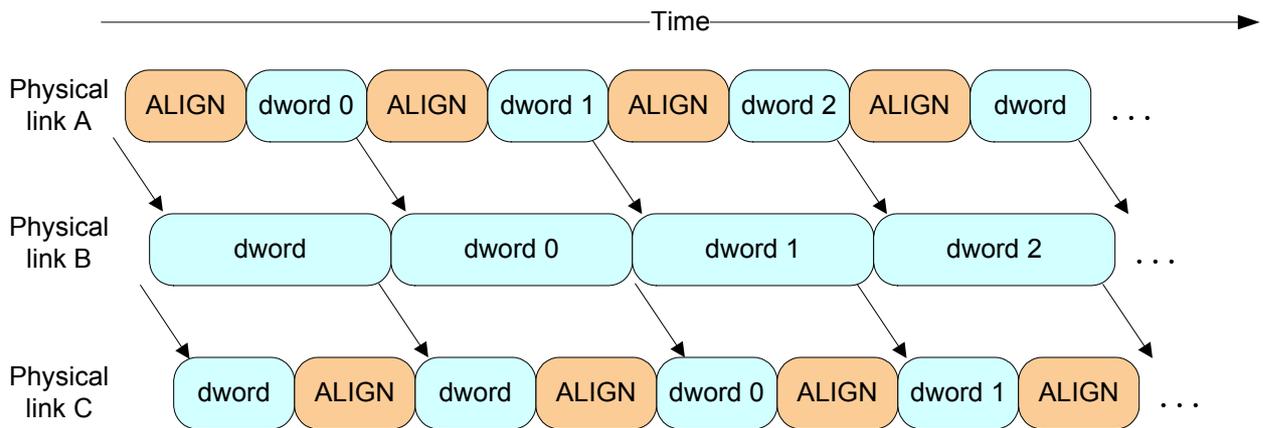
Figure 144 — Rate matching example

A phy shall start rate matching at the selected connection rate starting with the first dword that is not an ALIGN or NOTIFY inserted for clock skew management following:

    a) transmitting the EOAF for an OPEN address frame; or
    b) transmitting an OPEN_ACCEPT.

The source phy transmits idle dwords including ALIGNs and NOTIFYs at the selected connection rate while waiting for the connection response. This enables each expander device to start forwarding dwords from the source phy to the destination phy after forwarding an OPEN_ACCEPT.

A phy shall stop inserting ALIGNs and/or NOTIFYs for rate matching after:

    a) transmitting the first dword in a CLOSE;
    b) transmitting the first dword in a BREAK;
    c) receiving an OPEN_REJECT for a connection request; or
    d) losing arbitration to a received OPEN address frame.

If an expander phy attached to a SATA phy is using a physical link rate greater than the maximum connection rate supported by the pathway from an STP initiator port, a management application client should use the SMP PHY CONTROL function (see 10.4.3.10) to set the PROGRAMMED MAXIMUM PHYSICAL LINK RATE field of the expander phy to the maximum connection rate supported by the pathway from that STP initiator port.

7.14 SL (link layer for SAS phys) state machines

7.14.1 SL state machines overview

The SL (link layer for SAS phys) state machines controls connections, handling both connection requests (OPEN address frames), CLOSEs, and BREAKs. The SL state machines are as follows:

    a) SL_RA (receive OPEN address frame) state machine (see 7.14.3); and
    b) SL_CC (connection control) state machine (see 7.14.4).

All the SL state machines shall begin after receiving an Enable Disable SAS Link (Enable) message from the SL_IR state machines.

If a state machine consists of multiple states the initial state is as indicated in the state machine description.

Figure 145 shows the SL state machines.
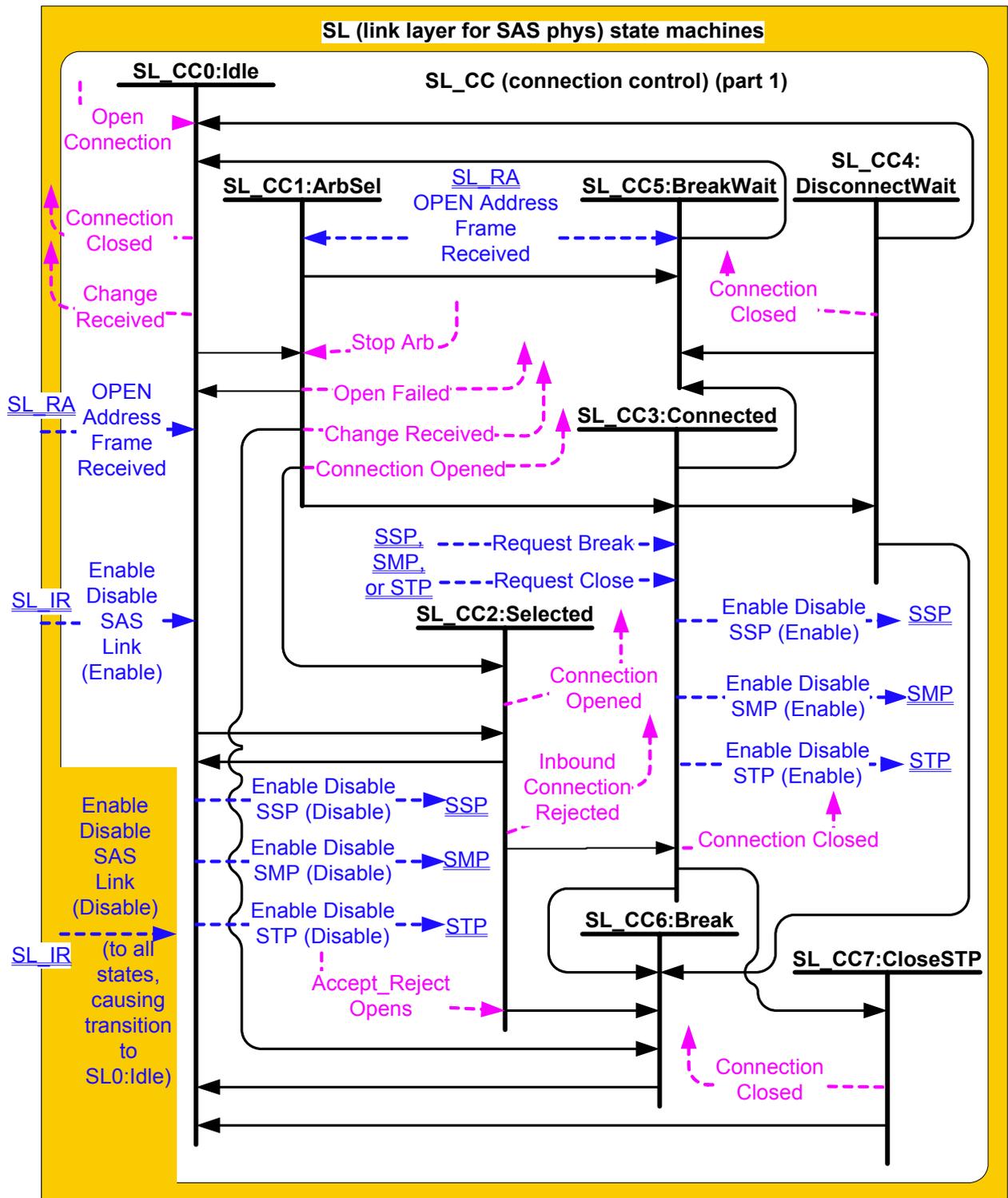
Figure 145 — SL (link layer for SAS phys) state machines (part 1)

Figure 146 shows the messages sent to the SL transmitter and received from the SL receiver.
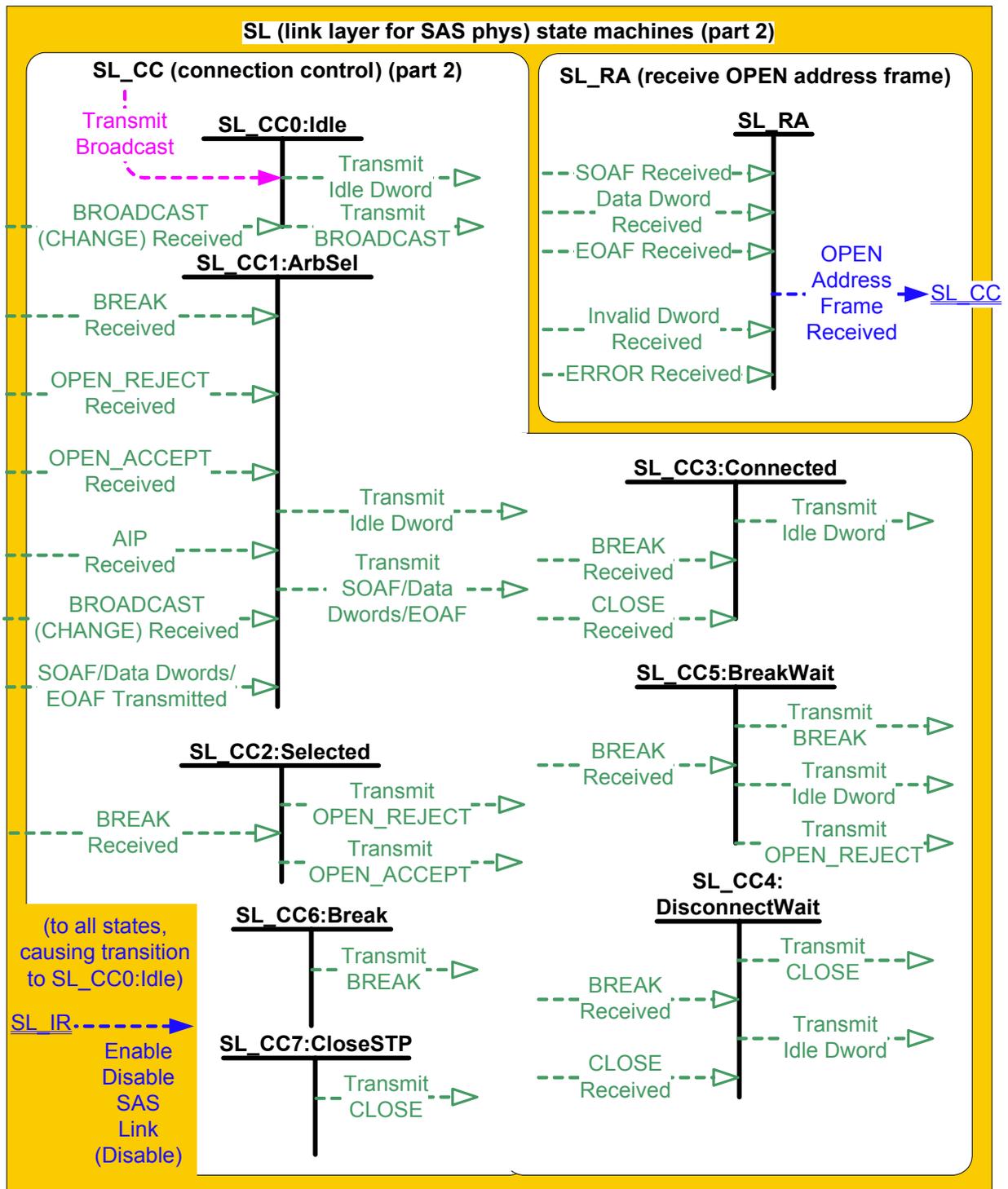
Figure 146 — SL (link layer for SAS phys) state machines (part 2)

7.14.2 SL transmitter and receiver

The SL transmitter receives the following messages from the SL state machines specifying primitive sequences, frames, and dwords to transmit:

   a) Transmit Idle Dword;
   b) Transmit SOAF/Data Dwords/EOAF;
   c) Transmit OPEN_ACCEPT;
   d) Transmit OPEN_REJECT with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (Retry));

e) Transmit BREAK;
f) Transmit BROADCAST; and
g) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal)).

When the SL transmitter is requested to transmit a dword from any state within any of the SL state machines, it shall transmit that dword. If there are multiple requests to transmit, the following priority should be followed when selecting the dword to transmit:

1) BREAK;
2) CLOSE;
3) OPEN_ACCEPT or OPEN_REJECT;
4) SOAF or data dword or EOAF; then
5) idle dword.

When there is no outstanding message specifying a dword to transmit, the SL transmitter shall transmit idle dwords.

The SL transmitter sends the following messages to the SL state machines based on dwords that have been transmitted:

a) SOAF/Data Dwords/EOAF Transmitted.

The SL receiver sends the following messages to the SL state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

a) SOAF Received;
b) Data Dword Received;
c) EOAF Received;
d) BROADCAST Received with an argument indicating the specific type (e.g., BROADCAST Received (Change));
e) BREAK Received;
f) OPEN_ACCEPT Received;
g) OPEN_REJECT Received with an argument indicating the specific type (e.g., OPEN_REJECT Received (No Destination));
h) AIP Received;
i) CLOSE Received with an argument indicating the specific type (e.g., CLOSE Received (Normal));
j) ERROR Received; and
k) Invalid Dword Received.

The SL receiver shall ignore all other dwords.

7.14.3 SL_RA (receive OPEN address frame) state machine

The SL_RA state machine's function is to receive address frames and determine if the received address frame is an OPEN address frame and whether or not it was received successfully. This state machine consists of one state.

This state machine receives SOAFs, dwords of an OPEN address frames, and EOAFs.

This state machine shall ignore all messages except SOAF Received, Data Dword Received, and EOAF Received.

If this state machine receives a subsequent SOAF Received message after receiving an SOAF Received message but before receiving an EOAF Received message, then this state machine shall discard the Data Dword Received messages received before the subsequent SOAF Received message.

If this state machine receives more than eight Data Dword Received messages after an SOAF Received message and before an EOAF Received message, then this state machine shall discard the address frame.

If this state machine receives an Invalid Dword Received message or an ERROR Received message after an

SOAF Received message and before an EOAF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the address frame.

After receiving an EOAF Received message, this state machine shall check if the address frame is a valid OPEN address frame.

This state machine shall accept an address frame if:

a) the ADDRESS FRAME TYPE field is set to Open;
b) the number of data dwords between the SOAF and EOAF is 8; and
c) the CRC field contains a valid CRC.

Otherwise, this state machine shall discard the address frame. If the frame is not discarded then this state machine shall send a OPEN Address Frame Received message to the SL_CC0:Idle state and the SL_CC1:ArbSel state with an argument that contains all the data dwords received in the OPEN address frame.

7.14.4 SL_CC (connection control) state machine

7.14.4.1 SL_CC state machine overview

The state machine consists of the following states:

a) SL_CC0:Idle (see 7.14.4.2)(initial state);
b) SL_CC1:ArbSel (see 7.14.4.3);
c) SL_CC2:Selected (see 7.14.4.4);
d) SL_CC3:Connected (see 7.14.4.5);
e) SL_CC4:DisconnectWait (see 7.14.4.6);
f) SL_CC5:BreakWait (see 7.14.4.7);
g) SL_CC6:Break (see 7.14.4.8); and
h) SL_CC7:CloseSTP (see 7.14.4.9).

The state machine shall start in the SL_CC0:Idle state. The state machine shall transition to the SL_CC0:Idle state from any other state after receiving an Enable Disable SAS Link (Disable) message from the SL_IR state machines (see 7.9.5).

The SL_CC state machine receives the following messages from the SSP link layer state machine (see 7.16.8), the STP link layer state machine, and SMP link layer state machine (see 7.18.4):

a) Request Break; and
b) Request Close.

The SL_CC state machine sends the following messages to the SSP link layer state machine, the STP link layer state machine, and SMP link layer state machine:

a) Enable Disable SSP (Enable);
b) Enable Disable SSP (Disable);
c) Enable Disable STP (Enable);
d) Enable Disable STP (Disable);
e) Enable Disable SMP (Enable); and
f) Enable Disable SMP (Disable).

The SL_CC state machine receives the following messages from the SL_IR state machines (see 7.9.5):

a) Enable Disable SAS Link (Enable); and
b) Enable Disable SAS Link (Disable).

Any message received by a state that is not referred to in the description of that state shall be ignored.

Any detection of an internal error shall cause the SL_CC state machine to transition to the SL_CC5:BreakWait state.

The SL_CC state machine shall maintain the timers listed in table 107.

Table 107 — SL_CC timers

Timer	Initial value
Open Timeout timer	1 ms
Close Timeout timer	1 ms
Break Timeout timer	1 ms

7.14.4.2 SL_CC0:Idle state

7.14.4.2.1 State description

This state is the initial state and is the state that is used when there is no connection pending or established.

Upon entry into this state, this state shall send:

  a) an Enable Disable SSP (Disable) message to the SSP link layer state machines;
  b) an Enable Disable SMP (Disable) message to the SMP link layer state machines;
  c) an Enable Disable STP (Disable) message to the STP link layer state machines; and
  d) a Connection Closed (Transition to Idle) confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter (see 7.4).

If a BROADCAST Received (Change) message is received, this state shall send a Change Received confirmation to the management layer.

If a Transmit Broadcast request is received with any argument, this state shall send a Transmit BROADCAST message with the same argument to the SL transmitter.

7.14.4.2.2 Transition SL_CC0:Idle to SL_CC1:ArbSel

This transition shall occur after receiving both an Enable Disable SAS Link (Enable) confirmation and an Open Connection request. The Open Connection request includes these arguments:

  a) initiator port bit;
  b) protocol;
  c) connection rate;
  d) initiator connection tag.
  e) destination SAS address;
  f) source SAS address;
  g) pathway blocked count; and
  h) arbitration wait time.

7.14.4.2.3 Transition SL_CC0:Idle to SL_CC2:Selected

This transition shall occur after receiving both an Enable Disable SAS Link (Enable) confirmation and an OPEN Address Frame Received message.

7.14.4.3 SL_CC1:ArbSel state

7.14.4.3.1 State description

This state is used to make a connection request.

Upon entry into this state, this state shall:

  1) request an OPEN address frame be transmitted by sending a Transmit SOAF/Data Dwords/EOAF message to the SL transmitter with the dwords containing the OPEN address frame with its fields set to the arguments received with the Open Connection request;
  2) initialize and start the Open Timeout timer; and
  3) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter.

This state shall ignore OPEN_REJECT Received and OPEN_ACCEPT Received messages from the time a Transmit SOAF/Data Dwords/EOAF message is sent to the SL transmitter until an SOAF/Data Dwords/EOAF Transmitted message is received from the SL transmitter.

If a BROADCAST Received (Change) message is received this state shall send a Change Received confirmation to the management layer.

If an AIP Received message is received after requesting the OPEN address frame be transmitted, this state shall reinitialize and restart the Open Timeout timer. The state machine shall not enforce a limit on the number of AIPs received.

If this state receives an OPEN_REJECT Received (No Destination) message after transmitting the OPEN address frame, this state shall send an Open Failed (No Destination) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Bad Destination) message after transmitting the OPEN address frame, this state shall send an Open Failed (Bad Destination) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Wrong Destination) message after transmitting the OPEN address frame, this state shall send an Open Failed (Wrong Destination) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (STP Resources Busy) message after transmitting the OPEN address frame, this state shall send an Open Failed (STP Resources Busy) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Connection Rate Not Supported) message after transmitting the OPEN address frame, this state shall send an Open Failed (Connection Rate Not Supported) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Protocol Not Supported) message after transmitting the OPEN address frame, this state shall send an Open Failed (Protocol Not Supported) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Retry) message after transmitting the OPEN address frame, this state shall send an Open Failed (Retry) confirmation to the port layer.

If this state receives an OPEN_REJECT Received (Pathway Blocked) message after transmitting the OPEN address frame, this state shall send an Open Failed (Pathway Blocked) confirmation to the port layer.

7.14.4.3.2 Transition SL_CC1:ArbSel to SL_CC0:Idle

This transition shall occur after sending an Open Failed confirmation.

7.14.4.3.3 Transition SL_CC1:ArbSel to SL_CC2:Selected

This transition shall occur after transmitting the OPEN address frame if:

   a) one or more AIP Received messages have been received before an OPEN Address Frame Received message is received (i.e., the incoming OPEN address frame overrides the outgoing OPEN address frame); or
   b) no AIP Received messages have been received before an OPEN Address Frame Received message is received, and the arbitration fairness rules (see 7.12.3) indicate the received OPEN address frame overrides the outgoing OPEN address frame.

The arbitration fairness comparison shall compare:

   a) the value of the arbitration wait time argument in the Open Connection request for the outgoing OPEN address frame; and
   b) the value of the ARBITRATION WAIT TIME field received in the incoming OPEN address frame.

7.14.4.3.4 Transition SL_CC1:ArbSel to SL_CC3:Connected

This transition shall occur if this state receives an OPEN_ACCEPT Received message after transmitting the OPEN address frame.

If the PROTOCOL field in the transmitted OPEN address frame was set to STP, then this state shall send a Connection Opened (STP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open STP Connection argument. At this point an STP connection has been opened between the source phy and the destination phy.

If the PROTOCOL field in the transmitted OPEN address frame was set to SSP, then this state shall send a Connection Opened (SSP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open SSP Connection argument. At this point an SSP connection has been opened between the source phy and the destination phy.

If the PROTOCOL field in the transmitted OPEN address frame was set to SMP, then this state shall send a Connection Opened (SMP, Source Opened) confirmation to the port layer before the transition. This transition shall include an Open SMP Connection argument. At this point an SMP connection has been opened between the source phy and the destination phy.

7.14.4.3.5 Transition SL_CC1:ArbSel to SL_CC5:BreakWait

This transition shall occur if a BREAK Received message has not been received and after:

  a) a Stop Arb request is received and after sending an Open Failed (Port Layer Request) confirmation to the port layer; or
  b) there is no response to the OPEN address frame before the Open Timeout timer expires and after sending an Open Failed (Open Timeout Occurred) confirmation to the port layer.

7.14.4.3.6 Transition SL_CC1:ArbSel to SL_CC6:Break

This transition shall occur after:

  a) receiving a BREAK Received message; and
  b) sending an Open Failed (Break Received) confirmation to the port layer.

7.14.4.4 SL_CC2:Selected state

7.14.4.4.1 State description

This state completes the establishment of an SSP, SMP, or STP connection when an incoming connection request has won arbitration by sending a Transmit OPEN_ACCEPT message, or rejects opening a connection by sending a Transmit OPEN_REJECT message to the SL transmitter.

This state shall respond to an incoming OPEN address frame using the following rules:

  1) If the OPEN address frame DESTINATION SAS ADDRESS field does not match the SAS address of this port, this state shall send a Transmit OPEN_REJECT (Wrong Destination) message to the SL transmitter (see 7.14.4.4.2);
  2) If the OPEN address frame INITIATOR PORT bit, PROTOCOL field, FEATURES field, and/or INITIATOR CONNECTION TAG field are set to values that are not supported (e.g., a connection request from an SMP target port), this state shall send a Transmit OPEN_REJECT (Protocol Not Supported) message to the SL transmitter (see 7.14.4.4.2);
  3) If the OPEN address frame CONNECTION RATE field is set to a connection rate that is not supported, this state shall send a Transmit OPEN_REJECT (Connection Rate Not Supported) message to the SL transmitter (see 7.14.4.4.2);
  4) If the OPEN address frame PROTOCOL field is set to STP, the source SAS address is not that of the STP initiator port with an affiliation established or the source SAS address is not that of an STP initiator port with task file register set resources (see 7.17.5), this state shall send a Transmit OPEN_REJECT (STP Resources Busy) message to the SL transmitter (see 7.14.4.4.2);
  5) If an Accept_Reject Opens (Reject SSP) request, Accept_Reject Opens (Reject SMP) request, or Accept_Reject Opens (Reject STP) request is received and the requested protocol is the corresponding protocol, this state shall send a Transmit OPEN_REJECT (Retry) message to the SL transmitter (see 7.14.4.4.2);
  6) If the requested protocol is SSP and this state has not received an Accept_Reject Opens (Reject SSP) request then this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (SSP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3);
  7) If the requested protocol is SMP and this state has not received an Accept_Reject Opens (Reject SMP) request then this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (SMP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3); or
  8) If the requested protocol is STP and this state has not received an Accept_Reject Opens (Reject STP) request then this state shall send a Transmit OPEN_ACCEPT message to the SL transmitter and send a Connection Opened (STP, Destination Opened) confirmation to the port layer (see 7.14.4.4.3).

If this state sends a Transmit OPEN_REJECT message to the SL transmitter, it shall also send an Inbound Connection Rejected confirmation to the port layer.

> NOTE 32 - Possible livelock scenarios can occur when a SAS phy transmits BREAK to abort a connection request (e.g., if its Open Timeout timer expires). SAS phys should respond to OPEN Address frames faster than 1 ms to reduce susceptibility to this problem.

7.14.4.4.2 Transition SL_CC2:Selected to SL_CC0:Idle

This transition shall occur after this state sends a Transmit OPEN_REJECT message to the SL transmitter.

7.14.4.4.3 Transition SL_CC2:Selected to SL_CC3:Connected

This transition shall occur after sending a Connection Opened confirmation.

This transition shall include an Open SSP Connection, Open STP Connection, or Open SMP Connection argument based on the requested protocol.

7.14.4.4.4 Transition SL_CC2:Selected to SL_CC6:Break

This transition shall occur after a BREAK Received message is received.

7.14.4.5 SL_CC3:Connected state

7.14.4.5.1 State description

This state enables the SSP, STP, or SMP link layer state machine to transmit dwords during a connection. See 7.13 for details on rate matching during the connection.

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open SMP Connection then this state shall send an Enable Disable SMP (Enable) message to the SMP link layer state machines (see 7.18.4).

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open SSP Connection then this state shall send an Enable Disable SSP (Enable) message to the SSP link layer state machines (see 7.16.8).

If this state is entered from SL_CC1:ArbSel state or the SL_CC2:Selected state with an argument of Open STP Connection then this state shall send an Enable Disable STP (Enable) message to the STP link layer state machines (see 7.17.9).

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SL transmitter until the SSP, SMP, or STP link layer state machine starts transmitting.

A CLOSE Received message may be received at any time while in this state, but shall be ignored during SSP and SMP connections. If a CLOSE Received (Clear Affiliation) is received during an STP connection, this state shall clear any affiliation (see 7.17.5).

7.14.4.5.2 Transition SL_CC3:Connected to SL_CC4:DisconnectWait

This transition shall occur if a Request Close message is received.

7.14.4.5.3 Transition SL_CC3:Connected to SL_CC5:BreakWait

This transition shall occur after sending a Connection Closed (Break Requested) confirmation to the port layer if:

   a)   a Request Break message is received; and
   b)   a BREAK Received message has not been received.

7.14.4.5.4 Transition SL_CC3:Connected to SL_CC6:Break

This transition shall occur if a BREAK Received message is received and after sending a Connection Closed (Break Received) confirmation to the port layer.

7.14.4.5.5 Transition SL_CC3:Connected to SL_CC7:CloseSTP

This transition shall occur if a CLOSE Received message is received during an STP connection.

7.14.4.6 SL_CC4:DisconnectWait state

7.14.4.6.1 State description

This state closes the connection and releases all resources associated with the connection.

Upon entry into this state, this state shall:

   1)   send a Transmit CLOSE (Normal) message or Transmit CLOSE (Clear Affiliation) message to the SL transmitter (see 7.17.7); and
   2)   initialize and start the Close Timeout timer.

A CLOSE Received message may be received at any time while in this state. If a CLOSE Received (Clear Affiliation) is received during an STP connection, this state shall clear any affiliation (see 7.17.5).

> NOTE 33 - Possible livelock scenarios can occur when a SAS phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). SAS phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

7.14.4.6.2 Transition SL_CC4:DisconnectWait to SL_CC0:Idle

This transition shall occur after:

- a) sending a Transmit CLOSE message to the SL transmitter;
- b) receiving a CLOSE Received message; and
- c) sending a Connection Closed (Normal) confirmation to the port layer.

7.14.4.6.3 Transition SL_CC4:DisconnectWait to SL_CC5:BreakWait

This transition shall occur if:

- a) a BREAK Received message has not been received;
- b) no CLOSE Received message is received in response to a Transmit CLOSE message before the Close Timeout timer expires; and
- c) after sending a Connection Closed (Close Timeout) confirmation to the port layer.

7.14.4.6.4 Transition SL_CC4:DisconnectWait to SL_CC6:Break

This transition shall occur after receiving a BREAK Received message and after sending a Connection Closed (Break Received) confirmation to the port layer.

7.14.4.7 SL_CC5:BreakWait state

7.14.4.7.1 State description

This state closes the connection if one is established and releases all resources associated with the connection.

Upon entry into this state, this state shall:

1) send a Transmit BREAK message to the SL transmitter; and
2) initialize and start the Break Timeout timer.

> NOTE 34 - Some SAS phys send a Transmit OPEN_REJECT (Retry) message to the SL transmitter in response to each OPEN Address Frame Received message received while in this state.

7.14.4.7.2 Transition SL_CC5:BreakWait to SL_CC0:Idle

This transition shall occur after:

- a) receiving a BREAK Received message; or
- b) the Break Timeout timer expires.

7.14.4.8 SL_CC6:Break state

7.14.4.8.1 State description

This state closes any connection and releases all resources associated with this connection.

Upon entry into this state, this state shall send a Transmit BREAK message to the SL transmitter (see 7.14.4.8.2).

7.14.4.8.2 Transition SL_CC6:Break to SL_CC0:Idle

This transition shall occur after sending a Transmit BREAK message to the SL transmitter.

7.14.4.9 SL_CC7:CloseSTP state

7.14.4.9.1 State description

This state closes an STP connection and releases all resources associated with the connection.

Upon entry into this state, this state shall:

1) send a Transmit CLOSE (Normal) message or Transmit CLOSE (Clear Affiliation) message to the SL transmitter (see 7.17.7); and

2) send a Connection Closed (Normal) confirmation to the port layer (see 7.14.4.9.2).

> NOTE 35 - Possible livelock scenarios can occur when a SAS phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). SAS phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

7.14.4.9.2 Transition SL_CC7:CloseSTP to SL_CC0:Idle

This transition shall occur after sending a Connection Closed (Normal) confirmation to the port layer.

7.15 XL (link layer for expander phys) state machine

7.15.1 XL state machine overview

The 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 (e.g., the ECM and ECR).

This state machine consists of the following states:

a) XL0:Idle (see 7.15.3)(initial state);
b) XL1:Request_Path (see 7.15.4);
c) XL2:Request_Open (see 7.15.5);
d) XL3:Open_Confirm_Wait (see 7.15.6);
e) XL4:Open_Reject (see 7.15.7);
f) XL5:Forward_Open (see 7.15.8);
g) XL6:Open_Response_Wait (see 7.15.9);
h) XL7:Connected (see 7.15.10);
i) XL8:Close_Wait (see 7.15.11);
j) XL9:Break (see 7.15.12); and
k) XL10:Break_Wait (see 7.15.13).

The XL state machine shall start in the XL0:Idle state. The XL state machine shall transition to the XL0:Idle state from any other state after receiving an Enable Disable SAS Link (Disable) message from the SL_IR state machines (see 7.9.5).

The XL state machine receives the following messages from the SL_IR state machine:

a) Enable Disable SAS Link (Enable); and
b) Enable Disable SAS Link (Disable).

Any message received by a state that is not referred to in the description of that state shall be ignored.

The XL state machine shall maintain the timers listed in table 108.

Table 108 — XL timers

Timer	Initial value
Partial Pathway Timeout timer	Partial pathway timeout value (see 7.12.4.5)
Break Timeout timer	1 ms

Figure 147 shows several states in the XL state machine.
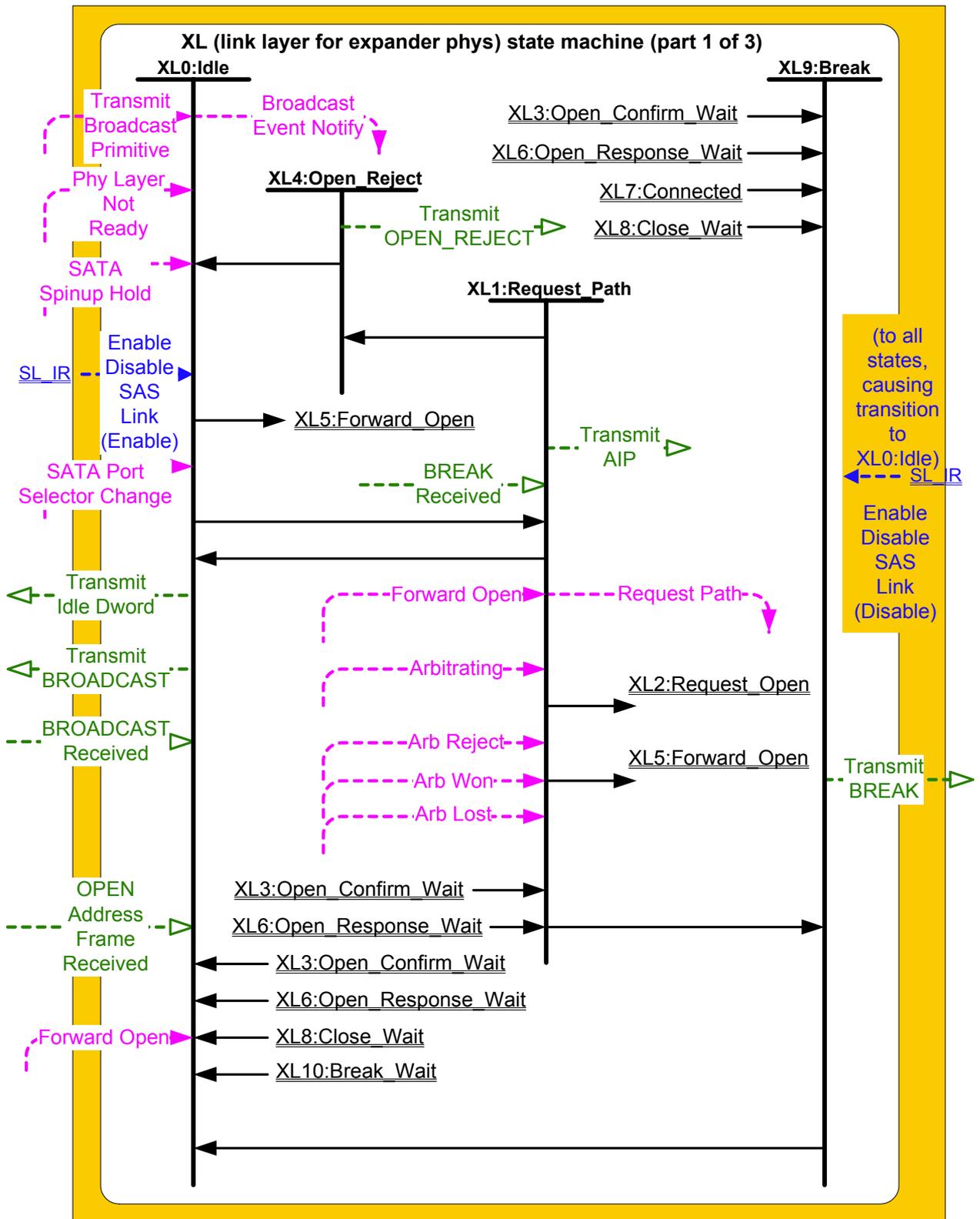
Figure 147 — XL (link layer for expander phys) state machine (part 1)

Figure 148 shows additional states in the XL state machine.
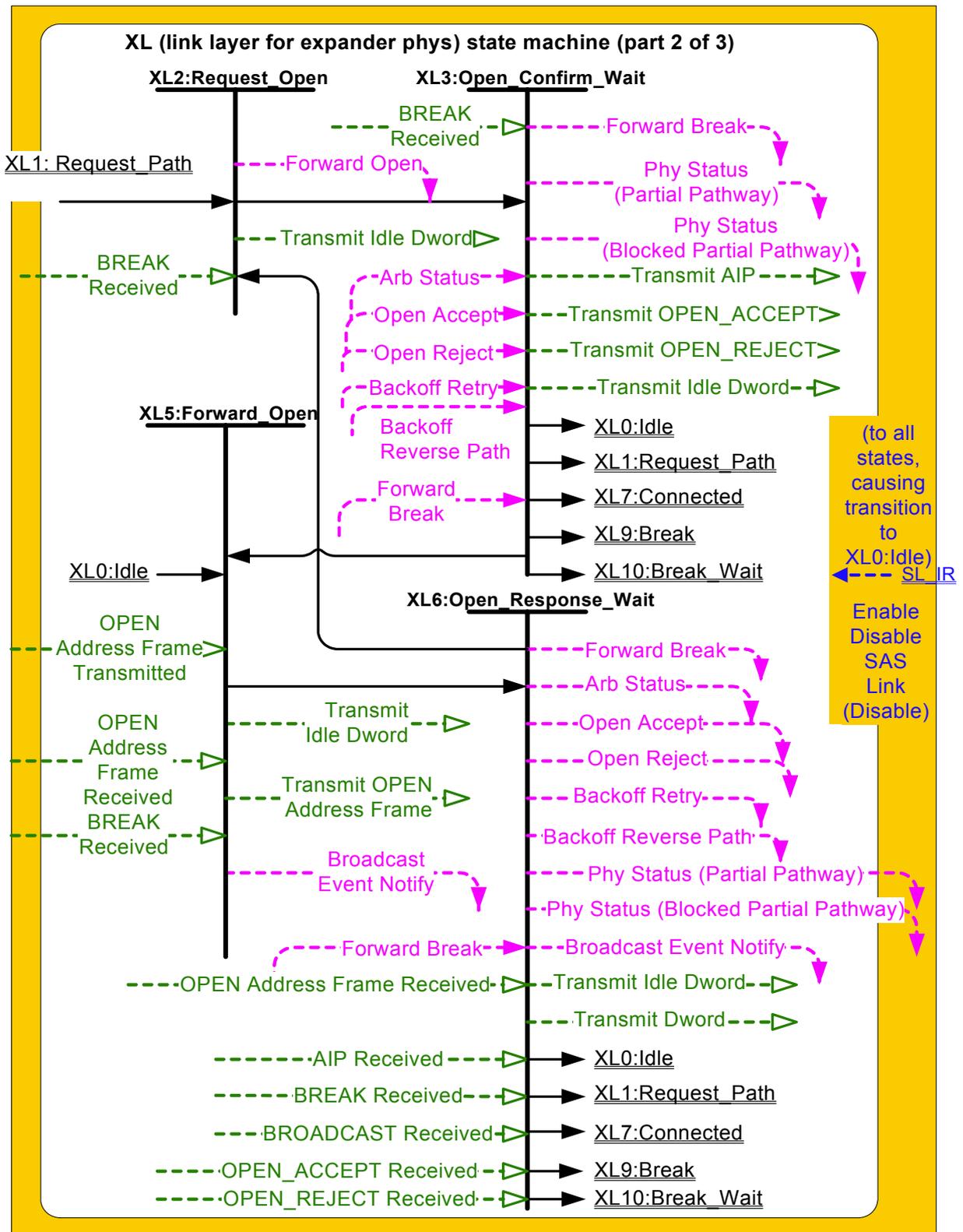
Figure 148 — XL (link layer for expander phys) state machine (part 2)

Figure 149 shows additional states in the XL state machine.
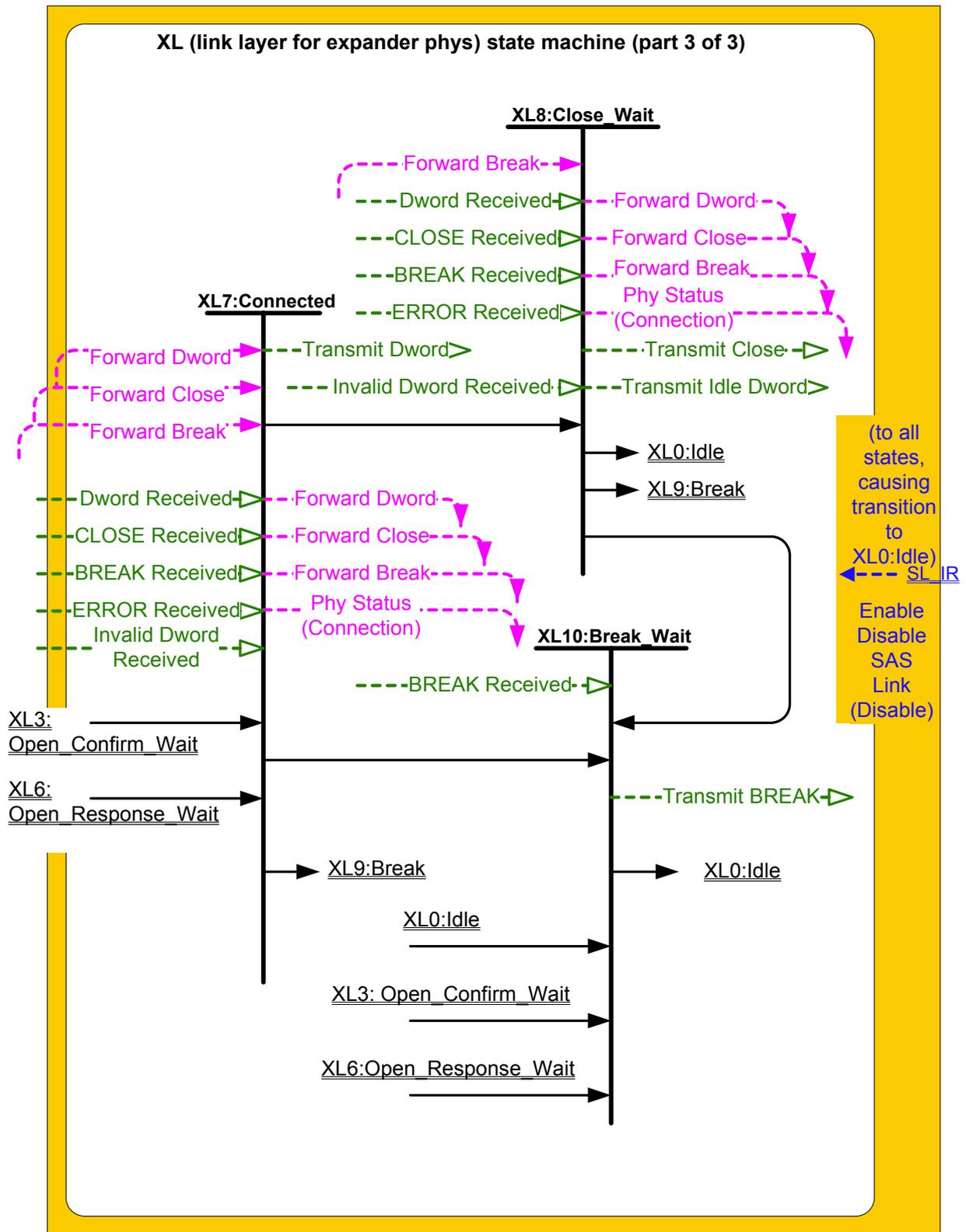
Figure 149 — XL (link layer for expander phys) state machine (part 3)

7.15.2 XL transmitter and receiver

The XL transmitter receives the following messages from the XL state machine specifying primitive sequences, frames, and dwords to transmit:

    a)   Transmit Idle Dword;

b) Transmit AIP with an argument indicating the specific type (e.g., Transmit AIP (Normal));
c) Transmit BREAK;
d) Transmit BROADCAST with an argument indicating the specific type (e.g., Transmit BROADCAST (Change));
e) Transmit CLOSE with an argument indicating the specific type (e.g., Transmit CLOSE (Normal));
f) Transmit OPEN_ACCEPT;
g) Transmit OPEN_REJECT, with an argument indicating the specific type (e.g., Transmit OPEN_REJECT (No Destination));
h) Transmit OPEN Address Frame; and
i) Transmit Dword.

The XL transmitter sends the following messages to the XL state machine based on dwords that have been transmitted:

a) OPEN Address Frame Transmitted.

The XL transmitter shall ensure clock skew management requirements are met (see 7.3) while originating dwords.

The XL transmitter shall ensure clock skew management requirements are met (see 7.3) during and after switching from forwarding dwords to originating dwords, including, for example:

a) when transmitting BREAK;
b) when transmitting CLOSE;
c) when transmitting an idle dword after closing a connection (i.e., after receiving BREAK or CLOSE);
d) while transmitting a SATA frame to a SAS physical link, when transmitting the first SATA_HOLDA in response to detection of SATA_HOLD; and
e) while receiving dwords of a SATA frame from a SAS physical link, when transmitting SATA_HOLD.

> NOTE 36 - The XL transmitter may always insert an ALIGN or NOTIFY before transmitting a BREAK, CLOSE, or SATA_HOLDA to meet clock skew management requirements.

The XL transmitter shall insert an ALIGN or NOTIFY before switching from originating dwords to forwarding dwords, including, for example:

a) when transmitting OPEN_ACCEPT;
b) when transmitting the last idle dword before a connection is established (i.e., after receiving OPEN_ACCEPT);
c) while transmitting a SATA frame to a SAS physical link, when transmitting the last dword from the SATA flow control buffer in response to release of SATA_HOLD;
d) while transmitting a SATA frame to a SAS physical link, when transmitting the last SATA_HOLDA in response to release of SATA_HOLD (e.g., if the SATA flow control buffer is empty); and
e) while receiving dwords of a SATA frame from a SAS physical link, when transmitting the last SATA_HOLD.

> NOTE 37 - This ensures that clock skew management requirements are met, even if the forwarded dword stream does not include an ALIGN or NOTIFY until the last possible dword.

The XL transmitter shall ensure rate matching requirements are met during a connection (see 7.13).

The XL transmitter shall ensure STP initiator phy throttling requirements are met (see 7.17.2) when:

a) transmitting dwords in the direction of an STP target port while originating dwords (e.g., while transmitting SATA_HOLD, SATA_HOLDA, or unloading the SATA flow control buffer);
b) switching from forwarding dwords to originating dwords; and
c) switching from originating dwords to forwarding dwords.

When there is no outstanding message specifying a dword to transmit, the XL transmitter shall transmit idle dwords.

The XL receiver sends the following messages to the XL state machine indicating primitive sequences, frames, and dwords received from the SP_DWS receiver (see 6.9.2):

a) AIP Received with an argument indicating the specific type (e.g., AIP Received (Normal));
b) BREAK Received;
c) BROADCAST Received;
d) CLOSE Received;
e) OPEN_ACCEPT Received;

    f)   OPEN_REJECT Received;

    g)   OPEN Address Frame Received;

    h)   Dword Received with an argument indicating the data dword or primitive received; and

    i)   Invalid Dword Received.

The XL receiver shall ignore all other dwords.

While receiving an address frame, if the XL receiver receives an invalid dword or ERROR, then the XL receiver shall:

    a)   ignore the invalid dword or ERROR; or

    b)   discard the address frame.

7.15.3 XL0:Idle state

7.15.3.1 State description

This state is the initial state and is the state that is used when there is no connection pending or established.

If a Phy Layer Not Ready confirmation is received, this state shall send a Broadcast Event Notify (Phy Not Ready) request to the BPP.

If a SATA Spinup Hold confirmation is received, this state shall send a Broadcast Event Notify (SATA Spinup Hold) request to the BPP.

If an Enable Disable SAS Link (Enable) message is received, this state shall send a Broadcast Event Notify (Identification Sequence Complete) request to the BPP.

If a SATA Port Selector Change confirmation is received, this state shall send a Broadcast Event Notify (SATA Port Selector Change) request to the BPP.

If a BROADCAST Received message is received, this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., CHANGE Received).

If a Transmit Broadcast indication is received, this state shall send a Transmit BROADCAST message to the XL transmitter with an argument specifying the specific type from the Transmit Broadcast indication. Otherwise, this state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

7.15.3.2 Transition XL0:Idle to XL1:Request_Path

This transition shall occur if:

    a)   an Enable Disable SAS Link (Enable) message has been received;

    b)   a Forward Open indication is not being received; and

    c)   an OPEN Address Frame Received message is received.

This state shall include an OPEN Address Frame Received argument with the transition.

7.15.3.3 Transition XL0:Idle to XL5:Forward_Open

This transition shall occur if:

    a)   an Enable Disable SAS Link (Enable) message has been received; and

    b)   a Forward Open indication is received.

This transition shall include a set of arguments containing the arguments received in the Forward Open indication.

If an OPEN Address Frame Received message is received, this state shall include an OPEN Address Frame Received argument with the transition.

7.15.4 XL1:Request_Path state

7.15.4.1 State description

This state is used to arbitrate for connection resources and to specify the destination of the connection.

If an Arbitrating (Normal) confirmation is received, this state shall repeatedly send Transmit AIP (Normal) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.2).

If an Arbitrating (Waiting On Partial) or Arbitrating (Blocked On Partial) confirmation is received, this state shall repeatedly send Transmit AIP (Waiting On Partial) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.2).

If an Arbitrating (Waiting On Partial) confirmation is received, this state shall repeatedly send a Phy Status (Partial Pathway) message to the ECM.

If an Arbitrating (Blocked On Partial) confirmation is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) message to the ECM.

If an Arbitrating (Waiting On Connection) confirmation is received, this state shall repeatedly send Transmit AIP (Waiting On Connection) and Transmit Idle Dword messages to the XL transmitter in accordance with AIP transmission rules (see 7.12.4.2).

If an Arbitrating (Waiting On Connection) confirmation is received, this state shall repeatedly send a Phy Status (Connection) message to the ECM.

If this state is entered from the XL6:Open_Response_Wait state, the Retry Priority Status argument shall be set to IGNORE AWT. If this state is entered from any other state, the Retry Priority Status argument shall be set to NORMAL.

Upon entry into this state, this state shall send a Request Path request to the ECM with the following arguments:

   a) initiator port bit;
   b) protocol;
   c) connection rate;
   d) initiator connection tag;
   e) destination SAS address;
   f) source SAS address;
   g) pathway blocked count;
   h) arbitration wait time; and
   i) retry priority status.

This state maintains the Partial Pathway Timeout timer.

If the Partial Pathway Timeout timer is not already running, the Partial Pathway Timeout timer shall be initialized and started when an Arbitrating (Blocked On Partial) confirmation is received.

If the Partial Pathway Timeout timer is already running, the Partial Pathway Timeout timer shall continue to run if an Arbitrating (Blocked On Partial) confirmation is received.

The Partial Pathway Timeout timer shall be stopped when one of the following confirmations is received:

   a) Arbitrating (Waiting On Partial); or
   b) Arbitrating (Waiting On Connection);

If the Partial Pathway Timeout timer expires, this state shall send a Partial Pathway Timeout Timer Expired request to the ECM.

7.15.4.2 Transition XL1:Request_Path to XL0:Idle

This transition shall occur if:

   a) a BREAK Received message has not been received; and
   b) an Arb Lost confirmation is received.

7.15.4.3 Transition XL1:Request_Path to XL2:Request_Open

This transition shall occur if:

   a) a BREAK Received message has not been received; and
   b) an Arb Won confirmation is received.

7.15.4.4 Transition XL1:Request_Path to XL4:Open_Reject

This transition shall occur if:

   a) a BREAK Received message has not been received; and
   b) an Arb Reject confirmation is received.

This transition shall include an Arb Reject argument corresponding to the Arb Reject confirmation.

7.15.4.5 Transition XL1:Request_Path to XL5:Forward_Open

This transition shall occur if a Forward Open indication is received and none of the following confirmations have been received:

   a)   Arbitrating (Normal);
   b)   Arbitrating (Waiting On Partial);
   c)   Arbitrating (Blocked On Partial);
   d)   Arbitrating (Waiting On Connection);
   e)   Arb Won;
   f)   Arb Lost;
   g)   Arb Reject (No Destination);
   h)   Arb Reject (Bad Destination);
   i)   Arb Reject (Bad Connection Rate); or
   j)   Arb Reject (Pathway Blocked).

This transition shall include:

   a)   an OPEN Address Frame Received argument containing the arguments received in the Forward Open indication; and
   b)   a BREAK Received argument if a BREAK Received message was received.

7.15.4.6 Transition XL1:Request_Path to XL9:Break

This transition shall occur after receiving a BREAK Received message if a Forward Open indication has not been received.

7.15.5 XL2:Request_Open state

7.15.5.1 State description

This state is used to forward an OPEN address frame through the ECR to a destination phy.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

Upon entry into this state, this state shall send a Forward Open request to the ECR, received by the destination phy as a Forward Open indication (see 7.15.5.2). The arguments to the Forward Open request are:

   a)   initiator port bit;
   b)   protocol;
   c)   features;
   d)   connection rate;
   e)   initiator connection tag;
   f)   destination SAS address;
   g)   source SAS address;
   h)   compatible features;
   i)   pathway blocked count;
   j)   arbitration wait time; and
   k)   more compatible features.

7.15.5.2 Transition XL2:Request_Open to XL3:Open_Confirm_Wait

This transition shall occur after sending a Forward Open request to the ECR.

If a BREAK Received message is received, this state shall include a BREAK Received argument with the transition.

7.15.6 XL3:Open_Confirm_Wait state

7.15.6.1 State description

This state waits for confirmation to an OPEN address frame sent on a destination phy.

This state shall send the following messages to the XL transmitter:

    a) Transmit AIP (Normal) when an Arb Status (Normal) confirmation is received;

    b) Transmit AIP (Waiting On Partial) when an Arb Status (Waiting On Partial) confirmation is received;

    c) Transmit AIP (Waiting On Connection) when an Arb Status (Waiting On Connection) confirmation is received;

    d) Transmit AIP (Waiting On Device) when an Arb Status (Waiting On Device) confirmation is received;

    e) Transmit OPEN_ACCEPT when an Open Accept confirmation is received (see 7.15.6.5);

    f) Transmit OPEN_REJECT when an Open Reject confirmation is received with the argument from the Open Reject confirmation, after releasing path resources (see 7.15.6.2); or

    g) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages when none of the previous conditions are present.

If a Backoff Retry confirmation is received, this state shall release path resources.

If a BREAK Received message is received or a BREAK Received argument is included in the transition into this state, this state shall send a Forward Break request to the ECR (see 7.15.6.6).

This state shall repeatedly send a Phy Status (Partial Pathway) response to the ECM. After an Arb Status (Waiting on Partial) confirmation is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) response to the ECM.

7.15.6.2 Transition XL3:Open_Confirm_Wait to XL0:Idle

This transition shall occur after sending a Transmit OPEN_REJECT message to the XL transmitter if:

    a) a BREAK Received message has not been received; and

    b) a BREAK Received argument was not included in the transition into this state.

7.15.6.3 Transition XL3:Open_Confirm_Wait to XL1:Request_Path

This transition shall occur after receiving a Backoff Retry confirmation, after releasing path resources if:

    a) a BREAK Received message has not been received; and

    b) a BREAK Received argument was not included in the transition into this state.

7.15.6.4 Transition XL3:Open_Confirm_Wait to XL5:Forward_Open

This transition shall occur after receiving a Backoff Reverse Path confirmation if:

    a) a BREAK Received message has not been received; and

    b) a BREAK Received argument was not included in the transition into this state.

The transition shall include the Backoff Reverse Path arguments (i.e., the OPEN address frame).

7.15.6.5 Transition XL3:Open_Confirm_Wait to XL7:Connected

This transition shall occur after sending a Transmit OPEN_ACCEPT message to the XL transmitter if:

    a) a BREAK Received message has not been received; and

    b) a BREAK Received argument was not included in the transition into this state.

7.15.6.6 Transition XL3:Open_Confirm_Wait to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.6.7 Transition XL3:Open_Confirm_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if:

    a) a BREAK Received message has not been received; and

    b) a BREAK Received argument was not included in the transition into this state.

7.15.7 XL4:Open_Reject state

7.15.7.1 State description

This state is used to reject a connection request.

This state shall send one of the following messages to the XL transmitter (see 7.15.7.2):

   a) a Transmit OPEN_REJECT (No Destination) message when an Arb Reject (No Destination) argument is received with the transition into this state;
   b) a Transmit OPEN_REJECT (Bad Destination) message when an Arb Reject (Bad Destination) argument is received with the transition into this state;
   c) a Transmit OPEN_REJECT (Connection Rate Not Supported) message when an Arb Reject (Bad Connection Rate) argument is received with the transition into this state; or
   d) a Transmit OPEN_REJECT (Pathway Blocked) message when an Arb Reject (Pathway Blocked) argument is received with the transition into this state.

7.15.7.2 Transition XL4:Open_Reject to XL0:Idle

This transition shall occur after sending a Transmit OPEN_REJECT message to the XL transmitter.

7.15.8 XL5:Forward_Open state

7.15.8.1 State description

This state is used to transmit an OPEN address frame passed with the transition into this state.

If a BROADCAST Received message is received, this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., CHANGE Received).

Upon entry into this state, this state shall send a Transmit OPEN Address Frame message to the XL transmitter with the fields set to the values specified with the transition into this state.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

7.15.8.2 Transition XL5:Forward_Open to XL6:Open_Response_Wait

This transition shall occur after receiving an OPEN Address Frame Transmitted message.

If an OPEN Address Frame Received message or argument is received, this state shall include an OPEN Address Frame Received argument with the transition.

If a BREAK Received message or argument is received, this state shall include a BREAK Received argument with the transition.

7.15.9 XL6:Open_Response_Wait state

7.15.9.1 State description

This state waits for a response to a transmitted OPEN address frame and determines the appropriate action to take based on the response.

This state shall either:

   a) request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter, honoring ALIGN insertion rules for rate matching and clock skew management; or
   b) send Transmit Dword messages to the XL transmitter to transmit all dwords received with Forward Dword indications.

If a BROADCAST Received message is received before an AIP Received message is received this state shall send a Broadcast Event Notify request to the BPP with the argument indicating the specific BROADCAST primitive received (e.g., CHANGE Received).

This state shall send the following responses to the ECR, which are received by the source phy as confirmations:

   a) an Open Accept response when an OPEN_ACCEPT Received message is received (see 7.15.9.5);
   b) an Open Reject response when an OPEN_REJECT Received message is received, after releasing any path resources (see 7.15.9.2);
   c) a Backoff Retry response, after releasing path resources (see 7.15.9.3), when:
      A) an AIP Received message has not been received;

       B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state containing a higher priority OPEN address frame according to the arbitration fairness comparison (see 7.12.3); and

       C) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame;

   d) a Backoff Retry response, after releasing path resources (see 7.15.9.3), when:

       A) an AIP Received message has been received;

       B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state; and

       C) the destination SAS address and connection rate of the received OPEN address frame are not equal to the source SAS address and connection rate of the transmitted OPEN address frame;

   e) a Backoff Reverse Path response (see 7.15.9.4) when:

       A) an AIP Received message has not been received,

       B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state containing a higher priority OPEN address frame according to the arbitration fairness comparison (see 7.12.3); and

       C) the destination SAS address and connection rate of the received OPEN address frame are equal to the source SAS address and connection rate of the transmitted OPEN address frame;

     and

   f) a Backoff Reverse Path response (see 7.15.9.4) when:

       A) an AIP Received message has been received;

       B) an OPEN Address Frame Received message is received or an OPEN Address Frame Received argument is included in the transition into this state; and

       C) the destination SAS address and connection rate of the received OPEN address frame are equal to the source SAS address and connection rate of the transmitted OPEN address frame.

A Backoff Reverse Path response shall include the contents of the OPEN Address Frame Received message or argument.

This state shall send the following responses to the ECR, which are received by the source phy as confirmations:

   a) an Arb Status (Waiting On Device) response upon entry into this state;

   b) an Arb Status (Normal) response when an AIP Received (Normal) message is received;

   c) an Arb Status (Waiting On Partial) response when an AIP Received (Waiting On Partial) message is received;

   d) an Arb Status (Waiting On Connection) response when an AIP Received (Waiting On Connection) message is received; and

   e) an Arb Status (Waiting On Device) response when an AIP Received (Waiting On Device) message is received.

If a BREAK Received message is received or a BREAK Received argument is included in the transition into this state, this state shall send a Forward Break request to the ECR (see 7.15.9.6).

This state shall repeatedly send a Phy Status (Partial Pathway) response to the ECM until an AIP Received (Waiting On Partial) message is received. After an AIP Received (Waiting On Partial) message is received, this state shall repeatedly send a Phy Status (Blocked Partial Pathway) response to the ECM.

7.15.9.2 Transition XL6:Open_Response_Wait to XL0:Idle

This transition shall occur after sending an Open Reject response to the ECR.

7.15.9.3 Transition XL6:Open_Response_Wait to XL1:Request_Path

This transition shall occur after sending a Backoff Retry response to the ECR.

7.15.9.4 Transition XL6:Open_Response_Wait to XL2:Request_Open

This transition shall occur after sending a Backoff Reverse Path response to the ECR.

7.15.9.5 Transition XL6:Open_Response_Wait to XL7:Connected

This transition shall occur after sending an Open Accept response to the ECR.

7.15.9.6 Transition XL6:Open_Response_Wait to XL9:Break

This transition shall occur after sending a Forward Break response to the ECR.

7.15.9.7 Transition XL6:Open_Response_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if:

   a) a BREAK Received message has not been received; and
   b) a BREAK Received argument was not included in the transition into this state.

7.15.10 XL7:Connected state

7.15.10.1 State description

This state provides a full-duplex circuit between two phys within an expander device.

This state shall send Transmit Dword messages to the XL transmitter to transmit all dwords received with Forward Dword indications.

This state shall send Forward Dword requests to the ECR containing each valid dword except BREAK and CLOSE primitives received with Dword Received messages.

If:

   a) an Invalid Dword Received message is received; and
   b) the expander phy is forwarding to an expander phy attached to a SAS physical link,

the expander phy shall:

   a) send an ERROR primitive with the Forward Dword request instead of the invalid dword; or
   b) delete the invalid dword.

If:

   a) an ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
   b) the expander phy is forwarding to an expander phy attached to a SATA phy,

the expander phy shall:

   a) send a SATA_ERROR with the Forward Dword request instead of the invalid dword or ERROR primitive; or
   b) delete the ERROR primitive or invalid dword.

If a CLOSE Received message is received, this state shall send a Forward Close request to the ECR with the argument from the CLOSE Received message.

If a BREAK Received message is received, this state shall send a Forward Break request to the ECR (see 7.15.10.3).

This state shall repeatedly send a Phy Status (Connection) response to the ECM.

7.15.10.2 Transition XL7:Connected to XL8:Close_Wait

This transition shall occur after receiving a Forward Close indication if a BREAK Received message has not been received.

7.15.10.3 Transition XL7:Connected to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.10.4 Transition XL7:Connected to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if a BREAK Received message has not been received.

7.15.11 XL8:Close_Wait state

7.15.11.1 State description

This state closes a connection and releases path resources.

Upon entry into this state, this state shall send a Transmit CLOSE message to the XL transmitter with the argument from the Forward Close indication, then shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the XL transmitter.

> NOTE 38 - Possible livelock scenarios can occur when a SAS phy transmits BREAK to break a connection (e.g., if its Close Timeout timer expires). SAS phys should respond to CLOSE faster than 1 ms to reduce susceptibility to this problem.

If a Dword Received message is received containing a valid dword except a BREAK or CLOSE primitive, this state shall send Forward Dword requests to the ECR containing that dword.

If:

   a) an Invalid Dword Received message is received; and
   b) the expander phy is forwarding to an expander phy attached to a SAS physical link,

the expander phy shall:

   a) send an ERROR primitive with the Forward Dword request instead of the invalid dword; or
   b) delete the invalid dword.

If:

   a) an ERROR primitive is received with the Dword Received message or an Invalid Dword Received message is received; and
   b) the expander phy is forwarding to an expander phy attached to a SATA phy,

the expander phy shall:

   a) send a SATA_ERROR with the Forward Dword request instead of the invalid dword or ERROR primitive; or
   b) delete the ERROR primitive or invalid dword.

If a CLOSE Received message is received, this state shall release path resources and send a Forward Close request to the ECR with the argument from the CLOSE Received message (see 7.15.11.2).

If a BREAK Received message is received, this state shall send a Forward Break request to the ECR (see 7.15.11.3).

This state shall repeatedly send a Phy Status (Connection) response to the ECM.

7.15.11.2 Transition XL8:Close_Wait to XL0:Idle

This transition shall occur after sending a Forward Close request to the ECR.

7.15.11.3 Transition XL8:Close_Wait to XL9:Break

This transition shall occur after sending a Forward Break request to the ECR.

7.15.11.4 Transition XL8:Close_Wait to XL10:Break_Wait

This transition shall occur after receiving a Forward Break indication if a BREAK Received message has not been received.

7.15.12 XL9:Break state

7.15.12.1 State description

This state closes the connection if there is one and releases all path resources associated with the connection.

This state shall send a Transmit BREAK message to the XL transmitter (see 7.15.12.2).

7.15.12.2 Transition XL9:Break to XL0:Idle

This transition shall occur after sending a Transmit BREAK message to the XL transmitter.

7.15.13 XL10:Break_Wait state

7.15.13.1 State description

This state closes the connection if there is one and releases path resources associated with the connection.

Upon entry into this state, this state shall:

1) send a Transmit BREAK message to the XL transmitter; and
2) initialize and start the Break Timeout timer.

7.15.13.2 Transition XL10:Break_Wait to XL0:Idle

This transition shall occur after:

a) a BREAK Received message is received; or
b) the Break Timeout timer expires.

7.16 SSP link layer

7.16.1 Opening an SSP connection

An SSP phy that accepts an OPEN address frame shall transmit at least one RRDY in that connection within 1 ms of transmitting an OPEN_ACCEPT. If the SSP phy is not able to grant credit, it shall respond with OPEN_REJECT (RETRY) and not accept the connection request.

7.16.2 Full duplex

SSP is a full duplex protocol. An SSP phy may receive an SSP frame or primitive in a connection while it is transmitting an SSP frame or primitive in the same connection. A wide SSP port may send and/or receive SSP frames or primitives concurrently on different connections (i.e., on different phys).

When a connection is open and an SSP phy has no more SSP frames to transmit on that connection, it transmits a DONE to start closing the connection (see 8.2.2.3.5). The other direction may still be active, so the DONE may be followed by one or more of the following primitives: CREDIT_BLOCKED, RRDY, ACK, or NAK.

7.16.3 SSP frame transmission and reception

During an SSP connection, SSP frames are preceded by SOF and followed by EOF as shown in figure 150.
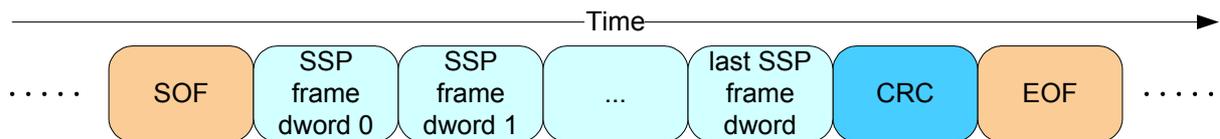
Figure 150 — SSP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5). The SSP link layer state machine checks that the frame is not too short and that the CRC is valid (see 7.16.8.7).

Receiving SSP phys shall acknowledge SSP frames within 1 ms, if not discarded as described in 7.16.8.7, with either:

a) ACK (i.e., positive acknowledgement) if the SSP frame was received into a frame buffer without errors; or
b) NAK (CRC ERROR) (i.e., negative acknowledgement) if the SSP frame was received with a CRC error, an invalid dword, or an ERROR primitive.

   NOTE 39 - It is not required that frame recipients generate NAK (CRC ERROR) from invalid dwords and ERRORs (see 7.16.8.2).

Either the transport layer (see 9.2.4) retries sending SSP frames that encounter a link layer error (e.g., are NAKed or create an ACK/NAK timeout), or the application layer aborts the SCSI command associated with the SSP frame that encountered a link layer error.

7.16.4 SSP flow control

An SSP phy uses RRDY to grant credit for permission for the other SSP phy in the connection to transmit frames. Each RRDY increments credit by one frame. Frame transmission decrements credit by one frame. Credit of zero frames is established at the beginning of each connection.

SSP phys shall not increment credit past 255 frames.

To prevent deadlocks where an SSP initiator port and SSP target port are both waiting on each other to provide credit, an SSP initiator port shall 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 (e.g., temporary buffer full conditions).

An SSP target port may refuse to provide credit for any reason, including because it needs to transmit a frame itself.

If credit is zero, SSP phys that are going to be unable to provide credit for 1 ms may send CREDIT_BLOCKED. The other phy may use this to avoid waiting 1 ms to transmit DONE (CREDIT TIMEOUT) (see 7.16.8).

If credit is nonzero, SSP phys that are going to be unable to provide additional credit for 1 ms, even if they receive frames per the existing credit, may transmit CREDIT_BLOCKED.

After sending CREDIT_BLOCKED, an SSP phy shall not transmit any additional RRDYs in the connection.

7.16.5 Interlocked frames

Table 109 shows which SSP frames shall be interlocked and which are non-interlocked.

Table 109 — SSP frame interlock requirements

SSP frame type	Interlock requirement
COMMAND	Interlocked
TASK	Interlocked
XFER_RDY	Interlocked
DATA	Non-interlocked
RESPONSE	Interlocked
See 9.2 for SSP frame type definitions.	

Before transmitting an interlocked frame, an SSP phy shall wait for all SSP frames to be acknowledged with ACK or NAK, even if credit is available. After transmitting an interlocked frame, an SSP phy shall not transmit another SSP frame until it has been acknowledged with ACK or NAK, even if credit is available.

Before transmitting a non-interlocked frame, an SSP phy shall wait for:

    a)   all non-interlocked frames with different tags; and
    b)   all interlocked frames;

to be acknowledged with ACK or NAK, even if credit is available.

After transmitting a non-interlocked frame, an SSP phy may transmit another non-interlocked frame with the same tag if credit is available. The phy shall not transmit:

    a)   a non-interlocked frame with a different tag; or
    b)   an interlocked frame;

until all SSP frames have been acknowledged with ACK or NAK, even if credit is available.

Interlocking does not prevent transmitting and receiving interlocked frames simultaneously (e.g., an SSP initiator phy could be transmitting a COMMAND frame while receiving XFER_RDY, DATA, or RESPONSE frames for a different command).

An SSP phy may transmit primitives responding to traffic it is receiving (e.g., an ACK or NAK to acknowledge an SSP frame, an RRDY to grant more receive credit, or a CREDIT_BLOCKED to specify that no more RRDYs are going to be transmitted in the connection) while waiting for an interlocked frame it transmitted to be acknowledged. These primitives may also be interspersed within an SSP frame.

Figure 151 shows an example of interlocked frame transmission.
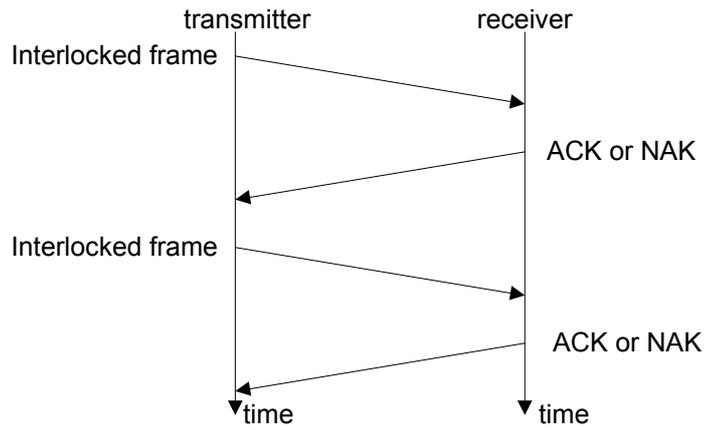
Figure 151 — Interlocked frames

Figure 152 shows an example of non-interlocked frame transmission with the same tags.
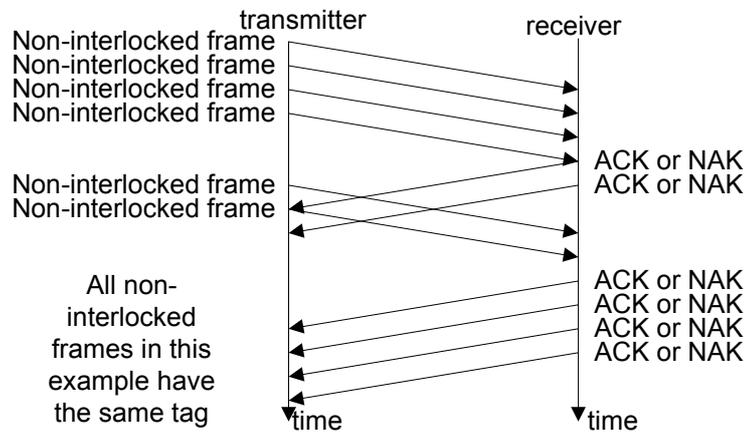
Figure 152 — Non-interlocked frames with the same tag

Figure 153 shows an example of non-interlocked frame transmission with different tags.
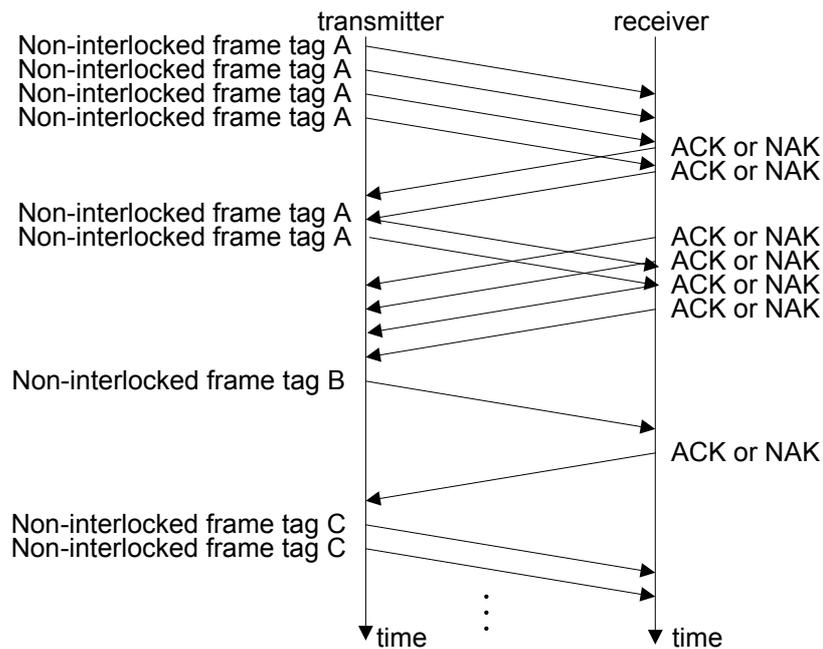
Figure 153 — Non-interlocked frames with different tags

7.16.6 Breaking an SSP connection

In addition to the actions described in 7.12.7, the following shall be the responses by an SSP phy 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.

7.16.7 Closing an SSP connection

DONE shall be exchanged prior to closing an SSP connection (see 8.2.2.3.5). There are several versions of the DONE primitive indicating additional information about why the SSP connection is being closed:

   a) DONE (NORMAL) specifies normal completion; the transmitter has no more SSP frames to transmit;
   b) DONE (CREDIT TIMEOUT) specifies that the transmitter still has SSP frames to transmit but did not receive an RRDY granting frame credit within 1 ms, or the transmitter has received a CREDIT_BLOCKED and has consumed all RRDYs received; and
   c) DONE (ACK/NAK TIMEOUT) specifies that the transmitter transmitted an SSP frame but did not receive the corresponding ACK or NAK within 1 ms. As a result, the ACK/NAK count is not balanced and the transmitter is going to transmit a BREAK in 1 ms unless the recipient replies with DONE and the connection is closed.

If the transmitter has no more SSP frames to transmit and receives a CREDIT_BLOCKED, it may transmit either DONE (NORMAL) or DONE (CREDIT TIMEOUT).

After transmitting DONE, the transmitting phy initializes and starts a 1 ms DONE Timeout timer (see 7.16.8.5).

After transmitting DONE, the transmitting phy shall not transmit any more SSP frames during this connection. However, the phy may transmit ACK, NAK, RRDY, and CREDIT_BLOCKED as needed after transmitting DONE if the other phy is still transmitting SSP frames in the reverse direction. Once an SSP phy has both transmitted and received DONE, it shall close the connection by transmitting CLOSE (NORMAL) (see 7.12.6).

Figure 154 shows the sequence for a closing an SSP connection.
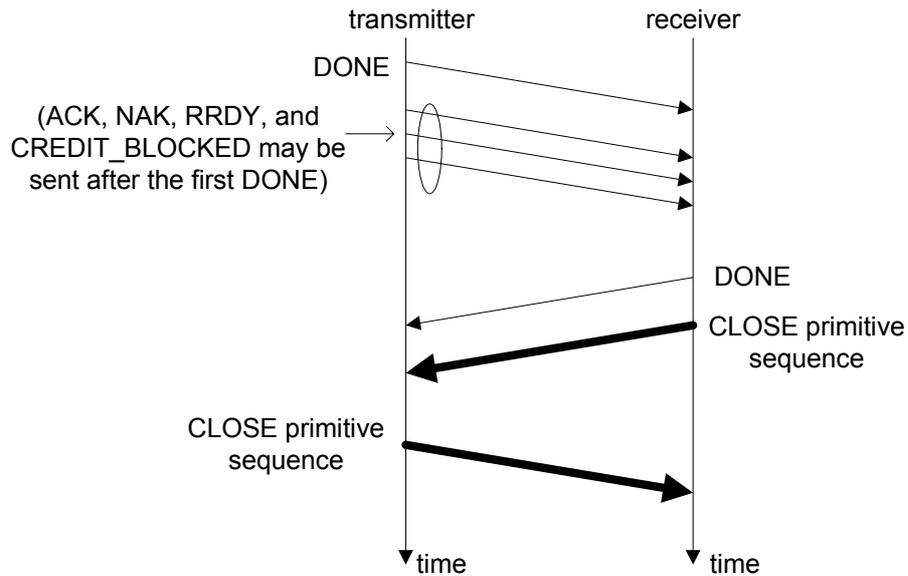
Figure 154 — Closing an SSP connection example

7.16.8 SSP (link layer for SSP phys) state machines

7.16.8.1 SSP state machines overview

The SSP link layer contains several state machines that run in parallel to control the flow of dwords on the physical link during an SSP connection. The SSP state machines are as follows:

   a) SSP_TIM (transmit interlocked frame monitor) state machine (see 7.16.8.3);
   b) SSP_TCM (transmit frame credit monitor) state machine (see 7.16.8.4);
   c) SSP_D (DONE control) state machine (see 7.16.8.5);
   d) SSP_TF (transmit frame control) state machine (see 7.16.8.6);
   e) SSP_RF (receive frame control) state machine (see 7.16.8.7);
   f) SSP_RCM (receive frame credit monitor) state machine (see 7.16.8.8);
   g) SSP_RIM (receive interlocked frame monitor) state machine (see 7.16.8.9);
   h) SSP_TC (transmit credit control) state machine (see 7.16.8.10); and
   i) SSP_TAN (transmit ACK/NAK control) state machine (see 7.16.8.11).

All the SSP state machines shall start after receiving an Enable Disable SSP (Enable) message from the SL state machines (see 7.14).

All the SSP state machines shall terminate after:

   a) receiving an Enable Disable SSP (Disable) message from the SL state machines;
   b) receiving a Request Close message from the SSP_D state machine indicating that the connection has been closed; or
   c) receiving a Request Break message from the SSP_D state machine indicating that a BREAK has been 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 state machines shall maintain the timers listed in table 110.

Table 110 — SSP link layer timers

Timer	Initial value	State machine	Reference
ACK/NAK Timeout timer	1 ms	SSP_TIM	7.16.8.3
DONE Timeout timer	1 ms	SSP_D	7.16.8.5
Credit Timeout timer	1 ms	SSP_TF	7.16.8.6

Figure 155 shows the SSP state machines and states related to frame transmission.
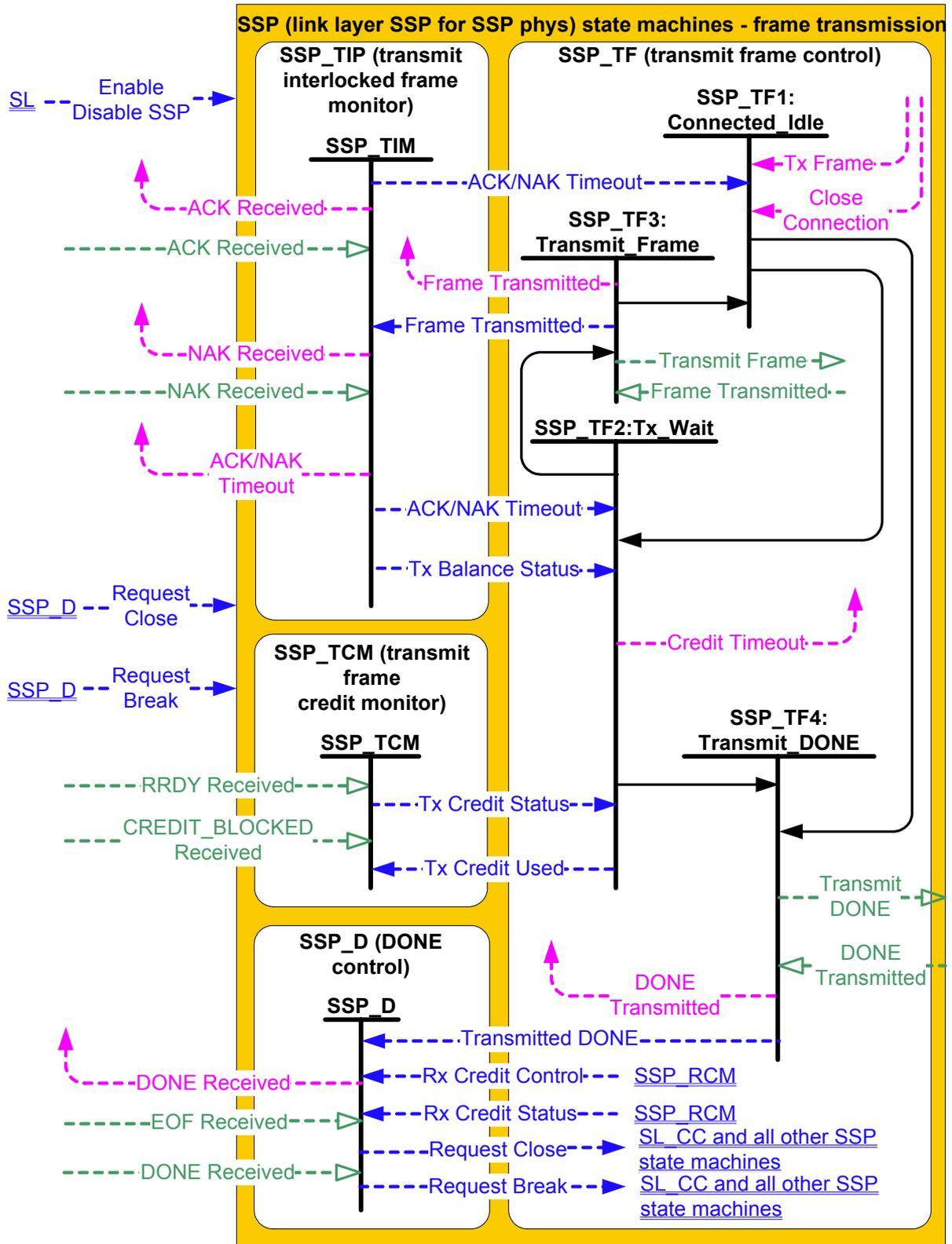
Figure 155 — SSP (link layer for SSP phys) state machines (part 1 - frame transmission)

Figure 156 shows the SSP state machines and states related to frame reception.
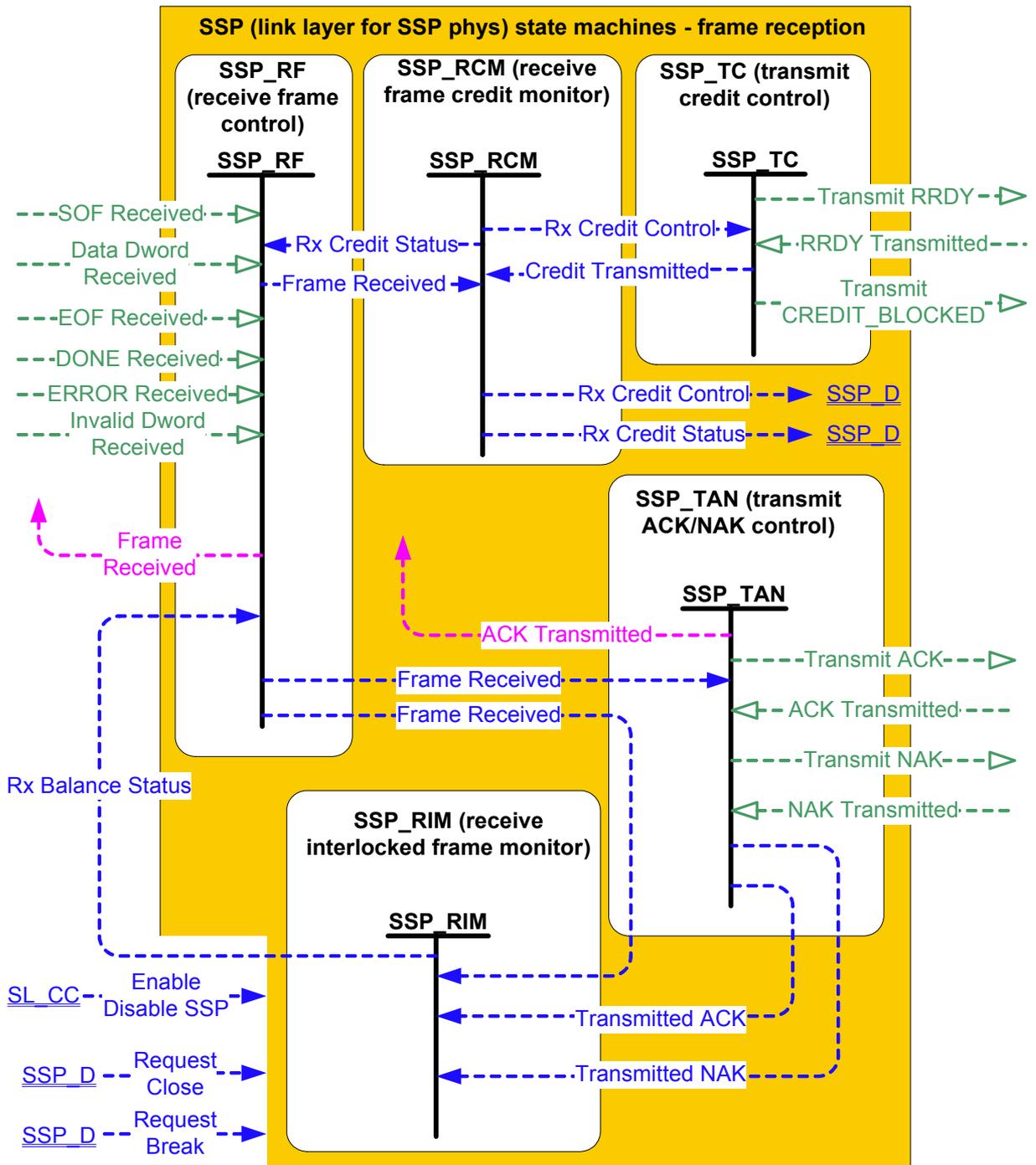
Figure 156 — SSP (link layer for SSP phys) state machines (part 2 - frame reception)

7.16.8.2 SSP transmitter and receiver

The SSP transmitter receives the following messages from the SSP state machines specifying primitive sequences and frames to transmit:

    a)   Transmit RRDY with an argument indicating the specific type (e.g., Transmit RRDY (Normal));
    b)   Transmit CREDIT_BLOCKED;
    c)   Transmit ACK;
    d)   Transmit NAK with an argument indicating the specific type (e.g., Transmit NAK (CRC Error));
    e)   Transmit Frame (i.e., SOF/data dwords/EOF); and
    f)   Transmit DONE with an argument indicating the specific type (e.g., Transmit DONE (Normal)).

The SSP transmitter sends the following messages to the SSP state machines based on dwords that have been transmitted:

   a) DONE Transmitted;
   b) RRDY Transmitted;
   c) CREDIT_BLOCKED Transmitted;
   d) ACK Transmitted;
   e) NAK Transmitted; and
   f) Frame Transmitted.

When there is no outstanding message specifying a dword to transmit, the SSP transmitter shall transmit idle dwords.

The SSP receiver sends the following messages to the SSP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

   a) ACK Received;
   b) NAK Received;
   c) RRDY Received;
   d) CREDIT_BLOCKED Received;
   e) DONE Received with an argument indicating the specific type (e.g., DONE Received (Normal));
   f) SOF Received;
   g) Data Dword Received;
   h) EOF Received;
   i) ERROR Received; and
   j) Invalid Dword Received.

The SSP receiver shall ignore all other dwords.

7.16.8.3 SSP_TIM (transmit interlocked frame monitor) state machine

The SSP_TIM state machine's function is to ensure that ACKs or NAKs are received for each transmitted frame before the ACK/NAK timeout. This state machine consists of one state.

This state machine monitors the number of frames transmitted with a Number Of Frames Transmitted counter and monitors the number of ACKs and NAKs received with a Number Of ACKs/NAKs Received counter. This state machine ensures that an ACK or NAK is received for each frame transmitted and reports an ACK/NAK timeout if they are not.

When the Number Of Frames Transmitted counter equals the Number Of ACKs/NAKs Received counter, the ACK/ NAK count is balanced and this state machine shall send the Tx Balance Status (Balanced) message to the SSP_TF2:Tx_Wait state. When the Number Of Frames Transmitted counter does not equal the Number Of ACKs/ NAKs Received counter, the ACK/NAK count is not balanced and this state machine shall send the Tx Balance Status (Not Balanced) message to the SSP_TF2:Tx_Wait state.

Each time a Frame Transmitted message is received, this state machine shall increment the Number Of Frames Transmitted counter.

If the ACK/NAK count is not balanced, each time an ACK Received message is received, this state machine shall:

   a) increment the Number Of ACKs/NAKs Received counter; and
   b) send an ACK Received confirmation to the port layer.

If the ACK/NAK count is not balanced, each time a NAK Received message is received, this state machine shall:

   a) increment the Number Of ACKs/NAKs Received counter; and
   b) send an NAK Received confirmation to the port layer.

If the ACK/NAK count is balanced, the ACK Received message and NAK Received message shall be ignored and the ACK/NAK Timeout timer shall be stopped.

Each time the ACK/NAK count is not balanced, the ACK/NAK Timeout timer shall be initialized and started. The ACK/NAK Timeout timer shall be re-initialized each time the Number Of ACKs/NAKs Received counter is incremented. If the ACK/NAK Timeout timer expires, this state machine shall send the ACK/NAK Timeout confirmation to the port layer and to the following states:

   a) SSP_TF1:Connected_Idle; and
   b) SSP_TF2:Tx_Wait state.

When this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the Number Of Frames Transmitted counter shall be set to zero and the Number Of ACKs/NAKs Received counter shall be set to zero.

7.16.8.4 SSP_TCM (transmit frame credit monitor) state machine

The SSP_TCM state machine's function is to ensure that transmit frame credit is available before a frame is transmitted. This state machine consists of one state.

This state machine shall keep track of the number of transmit frame credits available. This state machine shall add one transmit frame credit for each RRDY Received message received and subtract one transmit frame credit for each Tx Credit Used message received.

The CREDIT_BLOCKED Received message indicates that transmit frame credit is blocked. After receiving a CREDIT_BLOCKED Received message, this state machine may ignore additional RRDY Received messages until it receives a Request Close message or a Request Break message.

When transmit frame credit is available, this state machine shall send the Tx Credit Status (Available) message to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and transmit frame credit is not blocked, this state machine shall send the Tx Credit Status (Not Available) message to the SSP_TF2:Tx_Wait state.

When transmit frame credit is not available and transmit frame credit is blocked, this state machine shall send the Tx Credit Status (Blocked) message to the SSP_TF2:Tx_Wait state.

When this state machine receives an Enable Disable SSP (Enable) message, a Request Close message, or a Request Break message, this state shall set transmit frame credit to not available and transmit frame credit shall be set to not blocked.

7.16.8.5 SSP_D (DONE control) state machine

The SSP_D state machine's function is to ensure a DONE has been received and transmitted before the SL_CC state machine disables the SSP state machines. This state machine consists of one state.

This state machine ensures that a DONE is received and transmitted before the connection is closed. The DONE may be transmitted and received in any order.

If the DONE Received message has been received before the Transmitted DONE message is received, this state machine shall send the Request Close message to the SL_CC state machine (see 7.14) and all the SSP state machines after receiving the Transmitted DONE message.

If a DONE Received message, the Transmitted DONE (Normal) message, or the Transmitted DONE (Credit Timeout) message has not been received and the Rx Credit Status (Extended) message or the Rx Credit Control (Blocked) message has been received, then this state shall initialize and start the DONE Timeout timer after receiving the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message.

If the DONE Received message has not been received and the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, this state machine shall initialize and start the DONE Timeout timer each time:

a) the Rx Credit Status (Extended) message is received; or
b) the Rx Credit Control (Blocked) message is received.

If the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, the DONE Timeout timer shall be reinitialized each time the EOF Received message is received.

If the Transmitted DONE (Normal) message or the Transmitted DONE (Credit Timeout) message has been received, the DONE Timeout timer shall be stopped after:

a) the Rx Credit Status (Exhausted) message is received; and
b) the Rx Credit Control (Blocked) message has not been received.

> NOTE 40 - Stopping the timer ensures that, if credit remains exhausted long enough that the Credit Timeout timer of the other phy in the connection expires, the other phy is able to transmit a DONE (CREDIT TIMEOUT).

If the DONE Received message has not been received and the Transmitted DONE (ACK/NAK Timeout) message has been received:

a)   this state machine shall initialize and start the DONE Timeout timer; and
b)   this state shall not reinitialize the DONE Timeout timer if an EOF Received message is received.

If the DONE Received message is received before the DONE Timeout timer expires, this state machine shall send the Request Close message to the SL_CC state machine and all the SSP state machines.

If the DONE Received message is not received before the DONE Timeout timer expires, this state machine shall send a Request Break message to the SL_CC state machine and all the SSP state machines.

Any time a DONE Received message is received this state machine shall send a DONE Received confirmation to the port layer. A DONE Received (ACK/NAK Timeout) confirmation informs the port layer that the SSP transmitter is going to close the connection within 1 ms; other DONE Received confirmations (e.g., DONE Received (Close Connection) and DONE Received (Credit Timeout)) may be used by the application layer to decide when to reuse tags.

> NOTE 41 - The DONE Timeout timer in one phy (e.g., phy A) may expire concurrently with the ACK/NAK Timeout timer in the other phy (e.g., phy B) in a connection.
>
> For example, if phy A receives DONE (NORMAL) indicating phy B has no more frames to transmit, and phy A then transmits a series of non-interlocked frames where one or more of the SOFs is corrupted, then phy A waits to receive all the ACKs and/or NAKs after transmitting the series of non-interlocked frames. However, since phy B did not receive the full number of SOFs, it does not transmit as many ACKs and/or NAKs as phy A is expecting. The ACK/NAK Timeout timer in phy A expires and phy A transmits DONE (ACK/NAK TIMEOUT). Meanwhile, despite having transmitted DONE, phy B stops receiving frames while phy A is waiting for the final ACKs and/or NAKs. Since phy B does not receive DONE or any more frames, its DONE Timeout timer expires and phy B transmits BREAK.
>
> Since the timers may expire at slightly different times (e.g., due to timer resolution differences), the DONE (ACK/NAK TIMEOUT) may be transmitted before, concurrently with, or after the BREAK. Nevertheless, the phys handle the link layer error (i.e., the ACK/NAK timeout or the DONE timeout) the same way (see 9.2.4.5).

7.16.8.6 SSP_TF (transmit frame control) state machine

7.16.8.6.1 SSP_TF state machine overview

The SSP_TF state machine's function is to control when the SSP transmitter transmits SOF, frame dwords, EOF, and DONE. This state machine consists of the following states:

a)   SSP_TF1:Connected_Idle (see 7.16.8.6.2)(initial state);
b)   SSP_TF2:Tx_Wait (see 7.16.8.6.3);
c)   SSP_TF3:Transmit_Frame (see 7.16.8.6.4); and
d)   SSP_TF4:Transmit_DONE (see 7.16.8.6.5).

This state machine shall start in the SSP_TF1:Connected_Idle state.

7.16.8.6.2 SSP_TF1:Connected_Idle state

7.16.8.6.2.1 State description

This state waits for a request to transmit a frame or to close the connection.

7.16.8.6.2.2 Transition SSP_TF1:Connected_Idle to SSP_TF2:Tx_Wait

This transition shall occur after a Tx Frame request is received or a Close Connection request is received.

If a Tx Frame (Balance Required) request was received this transition shall include a Transmit Frame Balance Required argument.

If a Tx Frame (Balance Not Required) request was received this transition shall include a Transmit Frame Balance Not Required argument.

If a Close Connection request was received this transition shall include a Close Connection argument.

7.16.8.6.2.3 Transition SSP_TF1:Connected_Idle to SSP_TF4:Transmit_DONE

This transition shall occur if an ACK/NAK Timeout message is received. This transition shall include an ACK/NAK Timeout argument.

7.16.8.6.3 SSP_TF2:Tx_Wait state

7.16.8.6.3.1 State description

This state monitors the Tx Balance Status message and the Tx Credit Status message to ensure that frames are transmitted and connections are closed at the proper time.

If this state is entered from the SSP_TF1:Connected_Idle state with a Transmit Frame Balance Required argument or a Transmit Frame Balance Not Required argument, and:

a) if the last Tx Credit Status message received had an argument of Not Available, this state shall initialize and start the Credit Timeout timer; or
b) if the last Tx Credit Status message had an argument other than Not Available, this state shall stop the Credit Timeout timer.

7.16.8.6.3.2 Transition SSP_TF2:Tx_Wait to SSP_TF3:Transmit_Frame

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Frame Balance Required if:

a) the last Tx Balance Status message received had an argument of Balanced; and
b) the last Tx Credit Status message received had an argument of Available.

This transition shall occur if this state was entered from the SSP_TF1:Connected_Idle state with an argument of Transmit Frame Balance Not Required and if the last Tx Credit Status message received had an argument of Available.

This transition shall occur after sending a Tx Credit Used message to the SSP_TCM state machine.

7.16.8.6.3.3 Transition SSP_TF2:Tx_Wait to SSP_TF4:Transmit_DONE

This transition shall occur and include an ACK/NAK Timeout argument if an ACK/NAK Timeout message is received.

This transition shall occur and include a Close Connection argument if:

a) this state was entered from the SSP_TF1:Connected_Idle state with an argument of Close Connection; and
b) the last Tx Balance Status message received had an argument of Balanced.

This transition shall occur after sending a Credit Timeout confirmation and include a Credit Timeout argument if:

a) this state was entered from the SSP_TF1:Connected_Idle state with a Transmit Frame Balance Required argument or a Transmit Frame Balance Not Required argument;
b) the Credit Timeout timer expired before a Tx Credit Status message was received with an argument of Available, or the last Tx Credit Status message received had an argument of Blocked;
c) a Tx Balance Status message was received with an argument of Balanced (i.e., the Credit Timeout argument shall not be included in this transition for this reason unless the ACK/NAK count is balanced); and
d) an ACK/NAK Timeout message was not received.

7.16.8.6.4 SSP_TF3:Transmit_Frame state

7.16.8.6.4.1 State description

This state shall request a frame transmission by sending a Transmit Frame message to the SSP transmitter. Each time a Transmit Frame message is sent to the SSP transmitter, one SSP frame (i.e., SOF, frame contents, and EOF) is transmitted.

In this state receiving a Frame Transmitted message indicates that the frame has been transmitted.

7.16.8.6.4.2 Transition SSP_TF3:Transmit_Frame to SSP_TF1:Connected_Idle

This transition shall occur after:

a) receiving a Frame Transmitted message;
b) sending an Frame Transmitted message to the SSP_TIM state machine; and
c) sending a Frame Transmitted confirmation to the port layer.

7.16.8.6.5 SSP_TF4:Transmit_DONE state

This state shall send one of the following messages to an SSP transmitter:

a) a Transmit DONE (Normal) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Close Connection;
b) a Transmit DONE (ACK/NAK Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state or the SSP_TF1:Connected_Idle state with an argument of ACK/NAK Timeout; or
c) a Transmit DONE (Credit Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Credit Timeout.

> NOTE 42 - Possible livelock scenarios can occur when a SAS phy transmits BREAK to break a connection (e.g., if its Done Timeout timer expires). SAS phys should respond to DONE faster than 1 ms to reduce susceptibility to this problem.

After a DONE Transmitted message is received this state shall send the DONE Transmitted confirmation to the port layer and send one of the following messages to the SSP_D state machine:

a) a Transmitted DONE (Normal) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Close Connection;
b) a Transmitted DONE (ACK/NAK Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state or the SSP_TF1:Connected_Idle state with an argument of ACK/NAK Timeout; or
c) a Transmitted DONE (Credit Timeout) message if this state was entered from the SSP_TF2:Tx_Wait state with an argument of Credit Timeout.

7.16.8.7 SSP_RF (receive frame control) state machine

The SSP_RF state machine's function is to receive frames and determine whether or not those frames were received successfully. This state machine consists of one state.

This state machine:

a) checks the frame to determine if the frame should be accepted or discarded;
b) checks the frame to determine if an ACK or NAK should be transmitted; and
c) sends a Frame Received confirmation to the port layer.

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame if:

a) this state receives more than 263 Data Dword Received messages after an SOF Received message and before an EOF Received message;
b) this state receives fewer than 7 Data Dword Received messages after an SOF Received message and before an EOF Received message,
c) this state receives an Rx Credit Status (Credit Exhausted) message; or
d) this state receives a DONE Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after receiving an SOF Received message and before receiving an EOF Received message, then this state machine shall:

a) ignore the invalid dword or ERROR; or
b) discard the frame, send a Frame Received message to the SSP_RCM state machine, send a Frame Received message to the SSP_RIM state machine, and send a Frame Received (Unsuccessful) message to the SSP_TAN state machine.

If the frame is not discarded and the frame CRC is bad, this state machine shall:

a) send a Frame Received message to the SSP_RCM state machine;
b) send a Frame Received message to the SSP_RIM state machine; and
c) send a Frame Received (Unsuccessful) message to the SSP_TAN state machine.

If the frame is not discarded and the frame CRC is good, this state machine shall send a Frame Received (Successful) message to the SSP_TAN state machine and:

a) send a Frame Received message to the SSP_RCM state machine;
b) send a Frame Received message to the SSP_RIM state machine; and
c) send a Frame Received (Successful) message to the SSP_TAN state machine, and:
    A) if the last Rx Balance Status message received had an argument of Balanced, send a Frame Received (ACK/NAK Balanced) confirmation to the port layer; or
    B) if the last Rx Balance Status message received had an argument of Not Balanced, send a Frame Received (ACK/NAK Not Balanced) confirmation to the port layer.

7.16.8.8 SSP_RCM (receive frame credit monitor) state machine

The SSP_RCM state machine's function is to ensure that there was credit given to the originator for every frame that is received. This state machine consists of one state.

This state machine monitors the receiver's resources and keeps track of the number of RRDYs transmitted versus the number of frames received.

Any time resources are released or become available, if this state machine has not sent the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine, this state machine shall send the Rx Credit Control (Available) message to the SSP_TC state machine. This state machine shall only send the Rx Credit Control (Available) message to the SSP_TC state machine after frame receive resources become available. The specifications for when or how resources become available is outside the scope of this standard.

This state machine may send the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine when no further receive frame credit is going to become available within a credit timeout (i.e., less than 1 ms), even if frames are received per the existing receive frame credit. After sending the Rx Credit Control (Blocked) message to the SSP_TC state machine and the SSP_D state machine, this state machine shall not send the Rx Credit Control (Available) message to the SSP_TC state machine or the SSP_D state machine for the duration of the current connection.

This state machine shall indicate through the Rx Credit Control message only the amount of resources available to handle received frames (e.g., if this state machine has resources for five frames the maximum number of Rx Credit Control requests with the Available argument outstanding is five).

This state machine shall use the Credit Transmitted message to keep track of the number of RRDYs transmitted. This state machine shall use the Frame Received message to keep a track of the number of frames received.

Any time the number of Credit Transmitted messages received exceeds the number of Frame Received messages received this state machine shall send a Rx Credit Status (Extended) message to the SSP_RF state machine and the SSP_D state machine.

Any time the number of Credit Transmitted messages received equals the number of Frame Received messages received this state machine shall send a Rx Credit Status (Exhausted) message to the SSP_RF state machine and the SSP_D state machine.

If this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the frame receive resources shall be initialized to the no credit value for the current connection.

7.16.8.9 SSP_RIM (receive interlocked frame monitor) state machine

The SSP_RIM state machine's function is to inform the SSP_RF state machine when the number of ACKs and NAKs transmitted equals the number of the EOFs received. This state machine consists of one state.

This state machine monitors the number of frames received with a Number Of Frames Received counter and monitors the number of ACKs and NAKs transmitted with a Number Of ACKs/NAKs Transmitted counter.

Each time a Frame Received message is received, this state machine shall increment the Number Of Frames Received counter.

Each time an ACK Transmitted message or a NAK Transmitted is received, this state machine shall increment the Number Of ACKs/NAKs Transmitted counter.

When the Number Of Frames Received counter equals the Number Of ACKs/NAKs Transmitted counter, this state machine shall send an Rx Balance Status (Balanced) message to the SSP_RF state machine.

When the Number Of Frames Received counter does not equal the Number Of ACKs/NAKs Transmitted counter, this state machine shall send an Rx Balance Status (Not Balanced) message to the SSP_RF state machine.

When this state machine receives an Enable Disable SSP (Enable) message, Request Close message, or Request Break message, the Number Of Frames Received counter shall be set to zero and the Number Of ACKs/ NAKs Transmitted counter shall be set to zero.

7.16.8.10 SSP_TC (transmit credit control) state machine

The SSP_TC state machine's function is to control the sending of requests to transmit an RRDY or CREDIT_BLOCKED. This state machine consists of one state.

Any time this state machine receives a Rx Credit Control (Available) message it shall send a number of Transmit RRDY (Normal) messages to the SSP transmitter as indicated by the amount of resources available to handle received frames (e.g., if the Available argument indicates five RRDYs are to be transmitted this state machine sends five Transmit RRDY (Normal) messages to the SSP transmitter).

Any time this state machine receives a RRDY Transmitted message it shall send a Credit Transmitted message to the SSP_RCM state machine.

Any time this state machine receives a Rx Credit Control (Blocked) message it shall send a Transmit CREDIT_BLOCKED message to the SSP transmitter.

7.16.8.11 SSP_TAN (transmit ACK/NAK control) state machine

The SSP_TAN state machine's function is to control the sending of requests to transmit an ACK or NAK to the SSP transmitter. This state machine consists of one state.

Any time this state machine receives a Frame Received (Successful) message it shall send a Transmit ACK message to the SSP transmitter.

Any time this state machine receives a Frame Received (Unsuccessful) message it shall send a Transmit NAK (CRC Error) message to the SSP transmitter.

If multiple Frame Received (Unsuccessful) messages and Frame Received (Successful) messages are received, then the order in which the Transmit ACK messages and Transmit NAK messages are sent to the SSP transmitter shall be the same order as the Frame Received (Unsuccessful) messages and Frame Received (Successful) messages were received.

Any time this state machine receives an ACK Transmitted message it shall:

  a) send a Transmitted ACK message to the SSP_RIM state machine; and
  b) send an ACK Transmitted confirmation to the port layer.

Any time this state receives a NAK Transmitted argument it shall send a Transmitted NAK message to the SSP_RIM state machine.

7.17 STP link layer

7.17.1 STP frame transmission and reception

STP frame transmission is defined by SATA (see ATA/ATAPI-7 V3). During an STP connection, frames are preceded by SATA_SOF and followed by SATA_EOF as shown in figure 157.
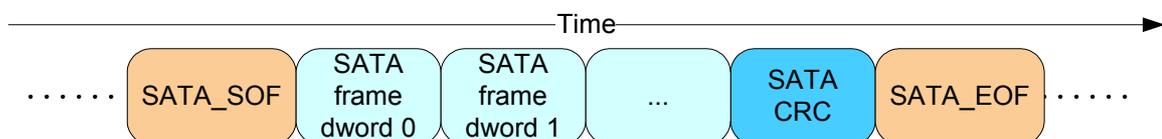
Figure 157 — STP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5).

Other primitives may be interspersed during the connection as defined by SATA.

STP encapsulates SATA with connection management. Table 111 summarizes STP link layer differences from the SATA link layer (see ATA/ATAPI-7 V3) that affect behavior during an STP connection.

Table 111 — STP link layer differences from SATA link layer during an STP connection

Feature	Description	Reference
STP initiator phy throttling	Limit the number of dwords transmitted to make room for more ALIGN primitives that may be deleted by the SATA device. This meets the ALIGN requirements for the physical link from the STP/SATA bridge to the attached SATA device.	7.17.2
STP flow control	Flow control through an STP connection is point-to-point, not end-to-end. Expander devices accept dwords in a temporary holding buffer after transmitting SATA HOLD to avoid losing data en-route before the transmitting phy acknowledges the SATA_HOLD with SATA_HOLDA.	7.17.3
Continued primitive sequence	Sustain the continued primitive sequence if a SATA_CONT appears after the continued primitive sequence has begun.	7.17.4

7.17.2 STP initiator phy throttling

On a SATA physical link, phys are required to transmit two consecutive ALIGN (0) primitives within every 256 dwords. To ensure an STP/SATA bridge is able to meet this requirement, an STP initiator phy has to reduce (i.e., throttle) the rate at which it is sourcing dwords by the same amount.

During an STP connection, an STP initiator phy shall insert two ALIGNs or NOTIFYs within every 256 dwords (i.e., within every overlapping window of 256 dwords) that are not ALIGNs or NOTIFYs for clock skew management or rate matching. They are not required to be inserted consecutively, because a phy in the pathway may delete one of them for clock skew management since STP initiator phy throttling ALIGNs and NOTIFYs are indistinguishable from clock skew management ALIGNs and NOTIFYs.

STP target phys are not required to insert extra ALIGNs and/or NOTIFYs, because SATA hosts are not supported by SAS domains. STP initiator phys, the only recipients of data from STP target phys, do not require extra ALIGNs or NOTIFYs.

ALIGNs and NOTIFYs inserted for STP initiator phy throttling are in addition to ALIGNs and NOTIFYs inserted for clock skew management (see 7.3) and rate matching (see 7.13). See Annex H for a summary of their combined requirements.

A phy shall start inserting ALIGNs and NOTIFYs for STP initiator phy throttling after:

    a)   transmitting an OPEN_ACCEPT; or
    b)   sending the first SATA primitive after receiving an OPEN_ACCEPT.

A phy shall stop inserting ALIGNs and NOTIFYs for STP initiator phy throttling after:

    a)   transmitting the first dword in a CLOSE; or
    b)   transmitting the first dword in a BREAK.

7.17.3 STP flow control

Each STP phy (i.e., STP initiator phy and STP target phy) and expander phy through which the STP connection is routed shall implement the SATA flow control protocol on each physical link in the pathway. The flow control primitives are not forwarded through expander devices like other dwords.

When an STP phy is receiving a frame and its buffer begins to fill up, it shall transmit SATA_HOLD. After transmitting SATA_HOLD, it shall accept the following number of data dwords for the frame:

    a)   24 dwords at 1,5 Gbit/s; or
    b)   28 dwords at 3,0 Gbit/s.

When an STP phy is transmitting a frame and receives SATA_HOLD, it shall transmit no more than 20 data dwords for the frame and respond with SATA_HOLDA.

NOTE 43 - The receive buffer requirements are based on $(20 + (4 \times 2^n))$ where n is 1 for 1,5 Gbit/s and 2 for 3,0 Gbit/s. The 20 portion of this equation is based on the frame transmitter requirements (see ATA/ATAPI-7 V3). The $(4 \times n)$ portion of this equation is based on:
    a) One-way propagation time on a 10 m cable = (5 ns/m propagation delay) $\times$ (10 m cable) = 50 ns;
    b) Round-trip propagation time on a 10 m cable = 100 ns (e.g., time to send SATA_HOLD and receive SATA_HOLDA);
    c) Time to transmit a 1,5 Gbit/s dword = (0,667 ns/bit unit interval) $\times$ (40 bits/dword) = 26,667 ns; and
    d) Number of 1,5 Gbit/s dwords on the wire during round-trip propagation time = (100 ns / 26,667 ns) = 3,75.
Receivers may support longer cables by providing larger buffer sizes.

When a SATA host phy in an STP/SATA bridge is receiving a frame from a SATA physical link, it shall transmit a SATA_HOLD when it is only capable of receiving 21 more dwords.

NOTE 44 - SATA requires that frame transmission cease and SATA_HOLDA be transmitted within 20 dwords of receiving SATA_HOLD. Since the SATA physical link has non-zero propagation time, one dword of margin is included.

When a SATA host phy in an STP/SATA bridge is transmitting a frame to a SATA physical link, it shall transmit no more than 19 data dwords after receiving SATA_HOLD.

NOTE 45 - SATA assumes that once a SATA_HOLD is transmitted, frame transmission ceases and SATA_HOLDA arrives within 20 dwords. Since the SATA physical link has non-zero propagation time, one dword of margin is included.

Figure 158 shows STP flow control between:

    a)   an STP initiator phy receiving a frame;
    b)   an expander device (the first expander device);
    c)   an expander device with an STP/SATA bridge (the second expander device); and
    d)   a SATA device phy transmitting a frame.

Figure 158 — STP flow control

After the STP initiator phy transmits SATA_HOLD, it receives a SATA_HOLDA reply from the first expander device within 24 dwords. The first expander device transmits SATA_HOLD to the second expander device and receives

SATA_HOLDA within 24 dwords, buffering data dwords it is no longer able to forward to the STP initiator phy. The second expander device transmits SATA_HOLD to the SATA device phy and receives SATA_HOLDA within 21 dwords, buffering data dwords it is no longer able to forward to the first expander device. When the SATA device phy stops transmitting data dwords, its previous data dwords are stored in the buffers in both expander devices and the STP initiator phy.

After the STP initiator phy drains its buffer and transmits SATA_R_IP, it receives data dwords from the first expander device's buffer, followed by data dwords from the second expander device's buffer, followed by data dwords from the SATA device phy.

7.17.4 Continued primitive sequence

Primitives that form continued primitive sequences (e.g., SATA_HOLD) shall be transmitted two times, then be followed by SATA_CONT, if needed, then be followed by vendor-specific scrambled data dwords, if needed. ALIGNs and NOTIFYs may be sent inside continued primitive sequences as described in 7.2.4.1.

After the SATA_CONT, during the vendor-specific scrambled data dwords:

a) a SATA_CONT continues the continued primitive sequence; and
b) any other STP primitive, including the primitive that is being continued, ends the continued primitive sequence.

Figure 159 shows an example of transmitting a continued primitive sequence.
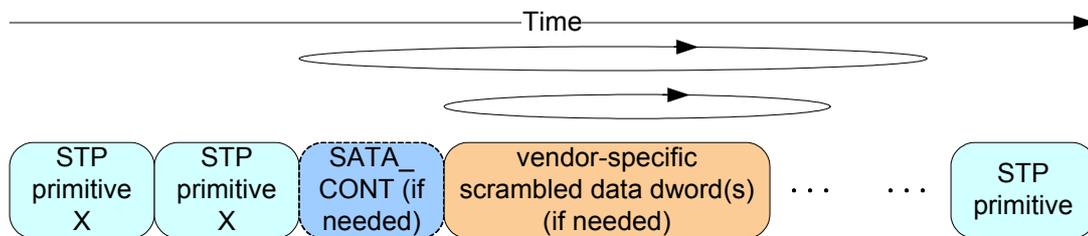
Figure 159 — Transmitting a continued primitive sequence

Receivers shall detect a continued primitive sequence after at least one primitive is received. The primitive may be followed by one or more of the same primitive. The primitive may be followed by one or more SATA_CONTs, each of which may be followed by vendor-specific data dwords. Receivers shall ignore invalid dwords before, during, or after the SATA_CONT(s). Receivers do not count the number of times the continued primitive, the SATA_CONTs, or the vendor-specific data dwords are received (i.e., receivers are simply in the state of receiving the primitive).

Expanders forwarding dwords may or may not detect an incoming sequence of the same primitive and convert it into a continued primitive sequence.

Figure 160 shows an example of receiving a continued primitive sequence.
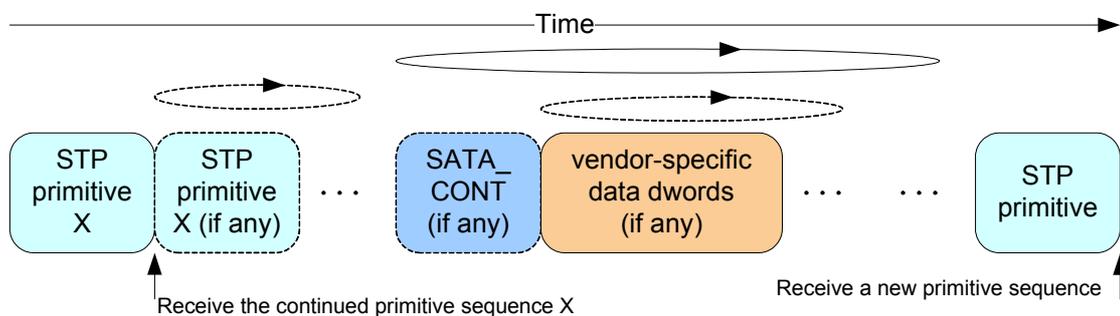
Figure 160 — Receiving a continued primitive sequence

7.17.5 Affiliations

Coherent access to the SATA task file registers shall be provided for each STP initiator port. STP target ports that do not track all commands by the STP initiator ports' SAS addresses shall implement affiliations to provide coherency. STP target ports that track all commands by the STP initiator ports' SAS addresses shall not implement affiliations.

An affiliation is a state entered by an STP target port where it refuses to accept connection requests from STP initiator ports other than the one that has established an affiliation.

An STP target port that supports affiliations shall establish an affiliation whenever it accepts a connection request. When an affiliation is established, the STP target port shall reject all subsequent connection requests from other STP initiator ports with OPEN_REJECT (STP RESOURCES BUSY).

An STP target port shall maintain an affiliation until any of the following occurs:

  a) power on;
  b) the SAS target device receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of HARD RESET (see 10.4.3.10) from any SMP initiator port;
  c) the SAS target device receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of TRANSMIT SATA PORT SELECTION SIGNAL (see 10.4.3.10) from any SMP initiator port;
  d) the SAS target device receives an SMP PHY CONTROL request specifying the phy with the affiliation and specifying a phy operation of CLEAR AFFILIATION (see 10.4.3.10) from the same SAS initiator port that has the affiliation.

     If a connection is already established to the STP target port on one phy while an SMP PHY CONTROL request specifying a phy operation of CLEAR AFFILIATION is processed by an SMP target port on another phy, the affiliation shall be cleared and the STP target port shall respond to new connection attempts with:

     A) AIP (WAITING ON CONNECTION) and/or OPEN_REJECT (RETRY), if the STP target port is in an expander device; or
     B) OPEN_REJECT (RETRY), if the STP target port is in a SAS device;

     rather than OPEN_REJECT (STP RESOURCES BUSY);

  e) a connection to the phy with the affiliation is closed with CLOSE (CLEAR AFFILIATION); or
  f) the STP target port is part of a STP/SATA bridge and a link reset sequence is begun on the SATA physical link that was not requested by an SMP PHY CONTROL request specifying the phy and specifying a phy operation of LINK RESET (see 10.4.3.10).

An affiliation established when a command is transmitted shall be maintained until all frames for the command have been delivered. An STP initiator port implementing command queuing (see SATAII-EXT) shall maintain an affiliation while any commands are outstanding to avoid confusing SATA devices, which only understand one SATA host. STP initiator ports may keep affiliations for longer tenures, but this is discouraged.

An STP target port that implements affiliations shall implement one affiliation per STP target port. Multiple phys on the same STP target port shall use the same affiliation. Support for affiliations is indicated in the SMP REPORT PHY SATA function response (see 10.4.3.7).

7.17.6 Opening an STP connection

If no STP connection exists when the SATA host port in an STP/SATA bridge receives a SATA_X_RDY from the attached SATA device, the STP target port in the STP/SATA bridge shall establish an STP connection to the appropriate STP initiator port before it transmits a SATA_R_RDY to the SATA device.

Wide STP initiator ports shall not request more than one connection at a time to an STP target port. Wide STP target ports shall not request more than one connection at a time to an STP initiator port.

While a wide STP target port is waiting for a response to a connection request or has established a connection to an STP initiator port, it shall:

  a) reject incoming connection requests from that STP initiator port with OPEN_REJECT (RETRY); and
  b) if affiliations are supported, reject incoming connection requests from other STP initiator ports with OPEN_REJECT (STP RESOURCES BUSY).

While a wide STP initiator port is waiting for a response to a connection request to an STP target port, it shall not reject an incoming connection request from that STP target port because of its outgoing connection request. It may reject incoming connection requests for other reasons (see 7.2.5.11).

If a wide STP initiator port receives an incoming connection request from an STP target port while it has a connection established with that STP target port, it shall reject the request with OPEN_REJECT (RETRY).

The first dword that an STP phy sends inside an STP connection after OPEN_ACCEPT that is not an ALIGN or NOTIFY shall be an STP primitive (e.g., SATA_SYNC).

7.17.7 Closing an STP connection

Either STP port (i.e., either the STP initiator port or the STP target port) may originate closing an STP connection. An STP port shall not originate closing an STP connection after sending a SATA_X_RDY or SATA_R_RDY until after both sending and receiving SATA_SYNC. An STP port shall transmit CLOSE after receiving a CLOSE if it has not already transmitted CLOSE.

When an STP initiator port closes an STP connection, it shall transmit a CLOSE (NORMAL) or CLOSE (CLEAR AFFILIATION). When an STP target port closes an STP connection, it shall transmit a CLOSE (NORMAL).

An STP initiator port may issue CLOSE (CLEAR AFFILIATION) in place of a CLOSE (NORMAL) to cause the STP target port to clear the affiliation (see 7.17.5) along with closing the connection. If an STP target port receives CLOSE (CLEAR AFFILIATION), the STP target port shall clear the affiliation for the STP initiator port that sent the CLOSE (CLEAR AFFILIATION).

See 7.12.6 for additional details on closing connections.

An STP/SATA bridge shall break an STP connection if its SATA host phy loses dword synchronization (see 7.12.7).

7.17.8 STP connection management examples

The STP/SATA bridge adds the outgoing OPEN address frames and CLOSEs so the STP initiator port sees an STP target port. The STP/SATA bridge removes incoming OPEN address frame and CLOSEs so the SATA device port sees only a SATA host port. While the connection is open, the STP/SATA bridge passes through all dwords without modification. Both STP initiator port and STP target port use SATA, with SATA flow control (see 7.17.3), while the connection is open.

Figure 161 shows an STP initiator port opening a connection, transmitting a single SATA frame, and closing the connection.
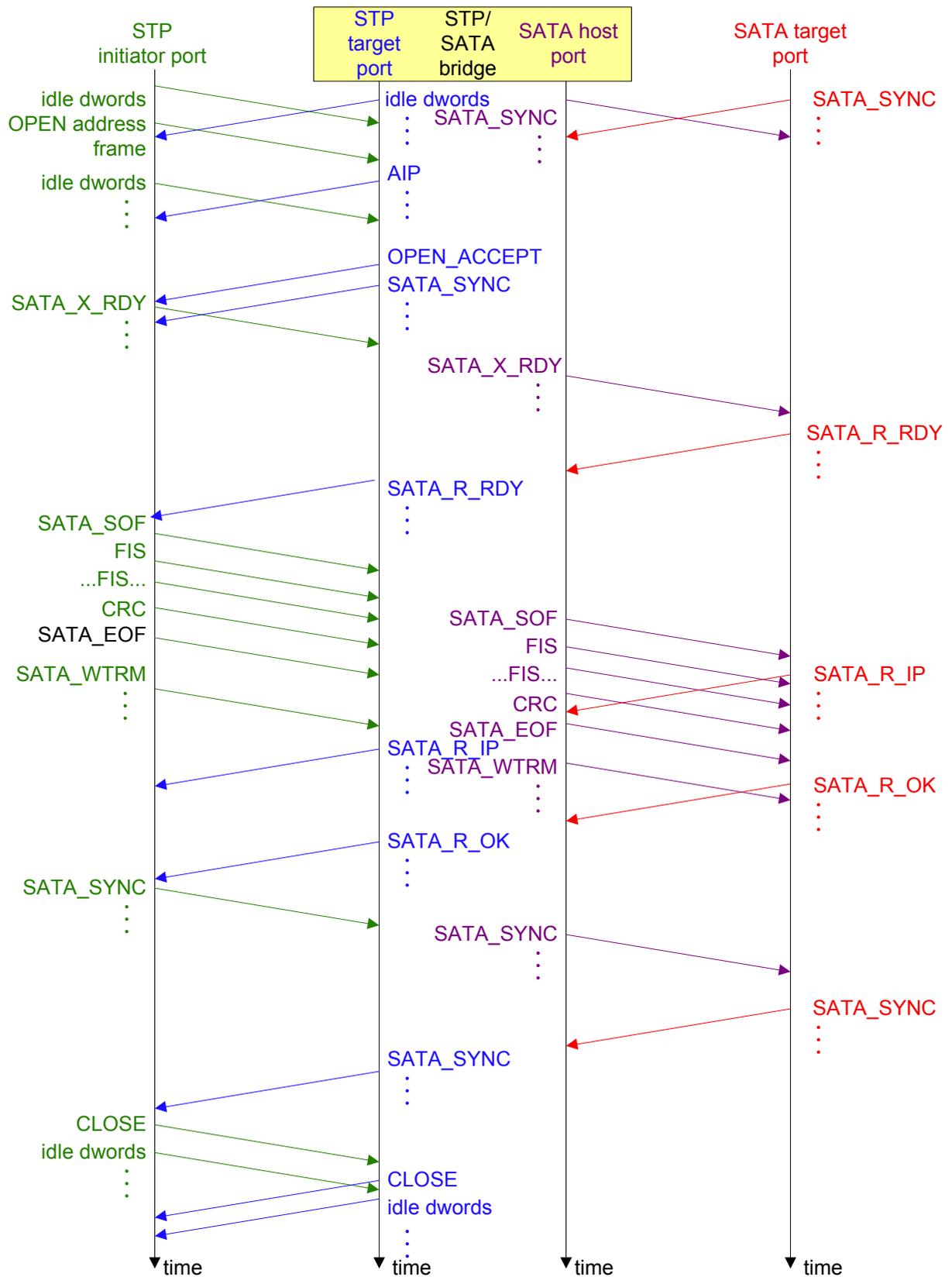
Figure 161 — STP initiator port opening an STP connection

Figure 162 shows a SATA device transmitting a SATA frame. In this example, the STP target port in the STP/SATA bridge opens a connection to an STP initiator port to send just one frame, then closes the connection.
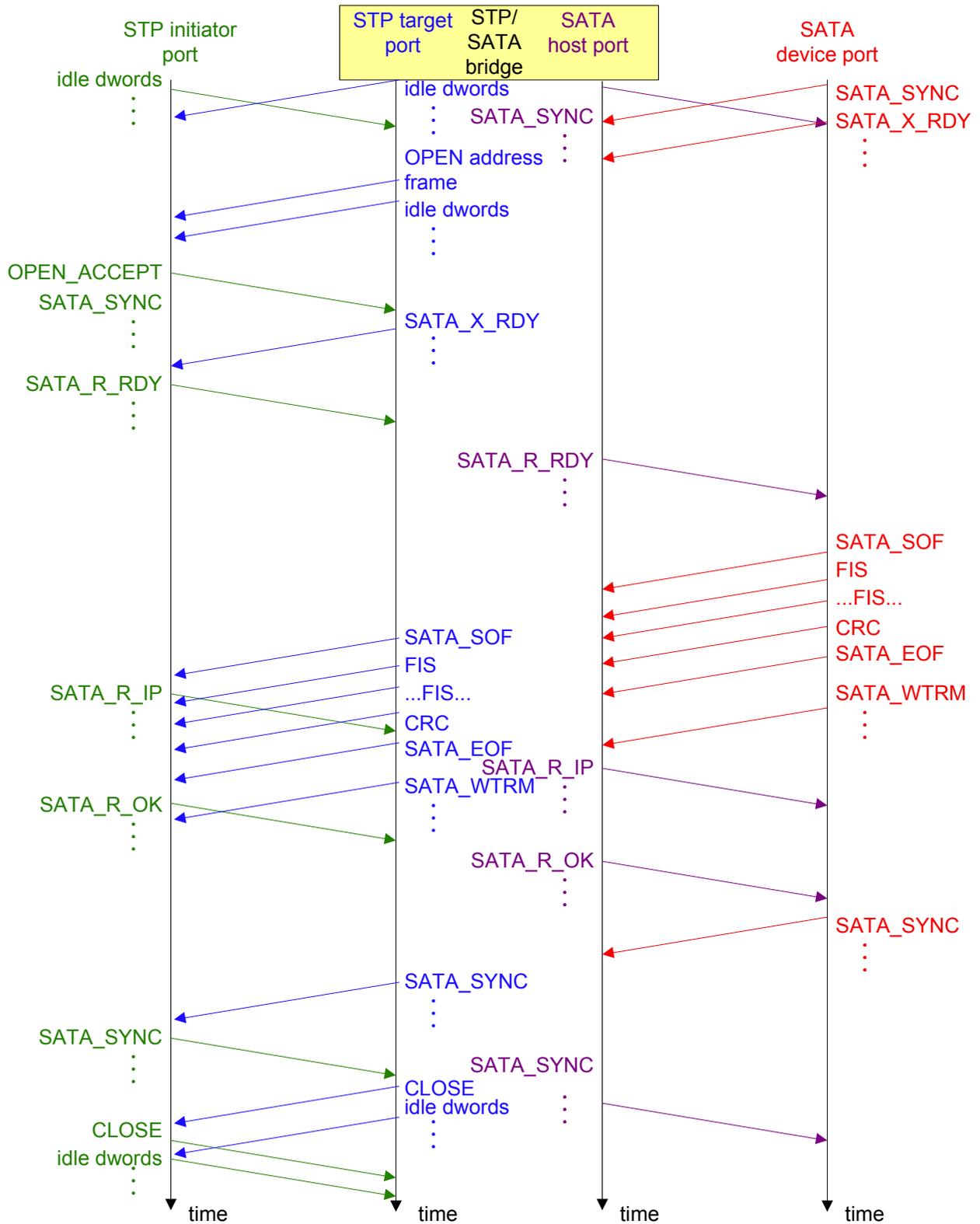
Figure 162 — STP target port opening an STP connection

7.17.9 STP (link layer for STP phys) state machines

The STP link layer uses the SATA link layer state machines (see ATA/ATAPI-7 V3), modified to:

   a) communicate with the port layer rather than directly with the transport layer;

b) interface with the SL state machines for connection management (e.g., to select when to open and close STP connections, and to tolerate idle dwords between an OPEN address frame and the first SATA primitive); and

c) implement affiliations (see 7.17.5).

These modifications are not described in this standard.

7.17.10 SMP target port support

A SAS device that contains an STP target port shall also contain an SMP target port.

7.18 SMP link layer

7.18.1 SMP frame transmission and reception

Inside an SMP connection, the 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 163. See 7.18.4 for error handling details.

> NOTE 46 - Unlike SSP, there is no acknowledgement of SMP frames with ACK and NAK and there is no credit exchange with RRDY.
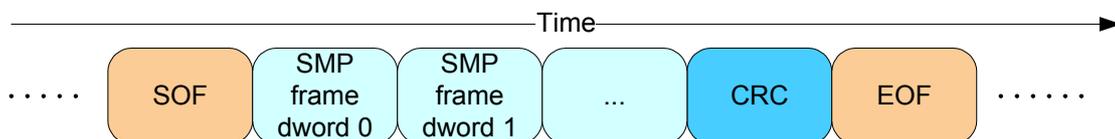
Figure 163 — SMP frame transmission

The last data dword after the SOF prior to the EOF always contains a CRC (see 7.5). The SMP link layer state machine checks that the frame is not too short and that the CRC is valid (see 7.18.4).

7.18.2 SMP flow control

By accepting an SMP connection, the destination device indicates it is ready to receive one SMP_REQUEST frame.

After the source device transmits one SMP_REQUEST frame, it shall be ready to receive one SMP_RESPONSE frame.

7.18.3 Closing an SMP connection

After receiving the SMP_RESPONSE frame, the source device shall transmit a CLOSE (NORMAL) to close the connection.

After transmitting the SMP_RESPONSE frame, the destination device shall reply with a CLOSE (NORMAL).

See 7.12.6 for additional details on closing connections.

7.18.4 SMP (link layer for SMP phys) state machines

7.18.4.1 SMP state machines overview

The SMP state machines control the flow of dwords on the physical link during an SMP connection. The SMP state machines are as follows:

a) SMP_IP (link layer for SMP initiator phys) state machine (see 7.18.4.3); and
b) SMP_TP (link layer for SMP target phys) state machine (see 7.18.4.4).

7.18.4.2 SMP transmitter and receiver

The SMP transmitter receives the following messages from the SMP state machines specifying dwords and frames to transmit:

a) Transmit Idle Dword; and

   b) Transmit Frame.

The SMP transmitter sends the following messages to the SMP state machines based on dwords that have been transmitted:

   a) Frame Transmitted.

When there is no outstanding message specifying a dword to transmit, the SMP transmitter shall transmit idle dwords.

The SMP receiver sends the following messages to the SMP state machines indicating primitive sequences and dwords received from the SP_DWS receiver (see 6.9.2):

   a) SOF Received;
   b) Data Dword Received;
   c) EOF Received;
   d) ERROR Received; and
   e) Invalid Dword Received.

The SMP receiver shall ignore all other dwords.

7.18.4.3 SMP_IP (link layer for SMP initiator phys) state machine

7.18.4.3.1 SMP_IP state machine overview

The SMP_IP state machine's function is to transmit an SMP request frame and then receive the corresponding response frame. This state machine consists of the following states:

   a) SMP_IP1:Idle (see 7.18.4.3.2)(initial state);
   b) SMP_IP2:Transmit_Frame (see 7.18.4.3.3); and
   c) SMP_IP3:Receive_Frame (see 7.18.4.3.4).

This state machine shall start in the SMP_IP1:Idle state on receipt of an Enable Disable SMP (Enable) message from the SL state machines (see 7.14).

The SMP_IP state machine shall terminate after receiving an Enable Disable SMP (Disable) message from the SL state machines.

Figure 164 shows the SMP_IP state machine.
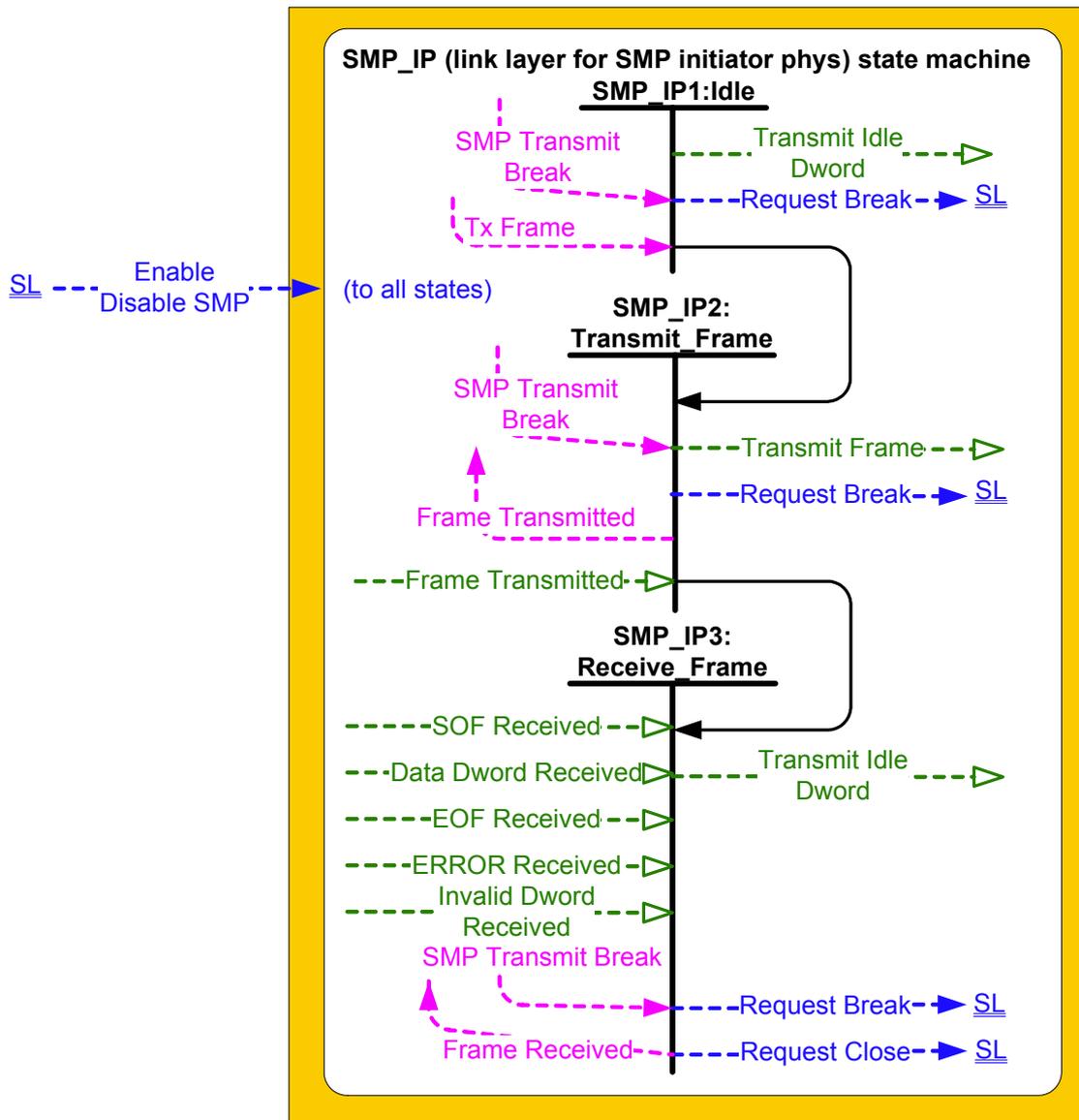
Figure 164 — SMP_IP (link layer for SMP initiator phys) state machine

7.18.4.3.2 SMP_IP1:Idle state

7.18.4.3.2.1 State description

This state is the initial state.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines (see 7.14).

7.18.4.3.2.2 Transition SMP_IP1:Idle to SMP_IP2:Transmit_Frame

This transition shall occur after a Tx Frame request is received.

7.18.4.3.3 SMP_IP2:Transmit_Frame state

7.18.4.3.3.1 State description

This state shall send a Transmit Frame message to the SMP transmitter.

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines (see 7.14) and terminate.

After the Frame Transmitted message is received, this state shall send a Frame Transmitted confirmation to the port layer.

7.18.4.3.3.2 Transition SMP_IP2:Transmit_Frame to SMP_IP3:Receive_Frame

This transition shall occur after sending a Frame Transmitted confirmation to the port layer.

7.18.4.3.4 SMP_IP3:Receive_Frame state

This state checks the SMP response frame and determines if the SMP response frame was successfully received (e.g., no CRC error).

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine if:

   a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
   b) this state receives fewer than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

   a) ignore the invalid dword or ERROR; or
   b) discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine.

If the SMP response frame is received with a CRC error, this state shall discard the frame, send a Frame Received (SMP Failure) confirmation to the port layer, send a Request Break message to the SL state machines, and terminate the state machine.

If the SMP response frame is received with no CRC error and the SMP response frame is valid, this state shall:

   a) send a Frame Received confirmation to the port layer; and
   b) send a Request Close message to the SL state machines (see 7.14).

If an SMP Transmit Break request is received, this state shall send a Request Break message to the SL state machines and this state machine shall terminate.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

7.18.4.4 SMP_TP (link layer for SMP target phys) state machine

7.18.4.4.1 SMP_TP state machine overview

The SMP_TP state machine's function is to receive an SMP request frame and then transmit the corresponding SMP response frame. The SMP_TP state machine consists of the following states:

   a) SMP_TP1:Receive_Frame (see 7.18.4.4.2)(initial state); and
   b) SMP_TP2:Transmit_Frame (see 7.18.4.4.3).

This state machine shall start in the SMP_TP1:Receive_Frame state after receiving an Enable Disable SMP (Enable) message from the SL state machines (see 7.14).

The SMP_TP state machine shall terminate after receiving an Enable Disable SMP (Disable) message from the SL state machines.

Figure 165 shows the SMP_TP state machine.
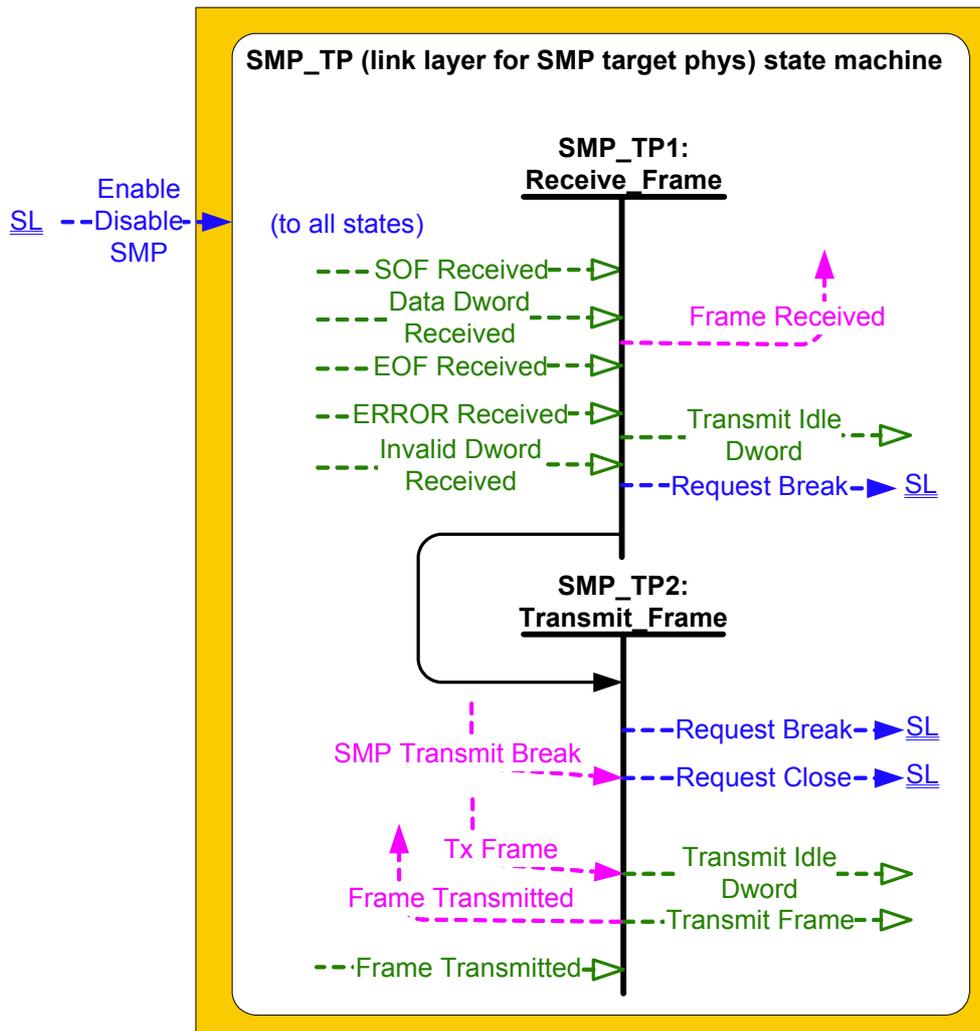
Figure 165 — SMP_TP (link layer for SMP target phys) state machine

7.18.4.4.2 SMP_TP1:Receive_Frame state

7.18.4.4.2.1 State description

This state waits for an SMP frame and determines if the SMP frame was successfully received (e.g., no CRC error).

If this state receives a subsequent SOF Received message after receiving an SOF Received message but before receiving an EOF Received message (i.e., SOF, data dwords, SOF, data dwords, and EOF instead of SOF, data dwords, EOF, SOF, data dwords, and EOF), then this state shall discard the Data Dword Received messages received before the subsequent SOF Received message.

This state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine if:

   a) this state receives more than 258 Data Dword Received messages after an SOF Received message and before an EOF Received message; or
   b) this state receives fewer than 2 Data Dword Received messages after an SOF Received message and before an EOF Received message.

If this state receives an Invalid Dword Received message or an ERROR Received message after an SOF Received message and before an EOF Received message, then this state machine shall:

   a) ignore the invalid dword or ERROR; or
   b) discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

If the SMP request frame is received with a CRC error, this state shall discard the frame, send a Request Break message to the SL state machines (see 7.14) and shall terminate the state machine.

Otherwise, this state shall send a Frame Received confirmation to the port layer.

This state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

7.18.4.4.2.2 Transition SMP_TP1:Receive_Frame to SMP_TP2:Transmit_Frame

This transition shall occur after sending a Frame Received confirmation to the port layer.

7.18.4.4.3 SMP_TP2:Transmit_Frame state

If this state receives an SMP Transmit Break request, this state shall send a Request Break message to the SL state machines and terminate.

If this state receives a Tx Frame request, this state shall send a Transmit Frame message to the SMP transmitter, then wait for a Frame Transmitted message. After receiving a Frame Transmitted message, this state shall send a Frame Transmitted confirmation to the port layer, send a Request Close message to the SL state machines (see 7.14) and terminate.

After sending Transmit Frame message to the SMP transmitter, this state shall request idle dwords be transmitted by repeatedly sending Transmit Idle Dword messages to the SMP transmitter.

8 Port layer

8.1 Port layer overview

The port layer (PL) state machines interface with one or more SAS link layer state machines and one or more SSP, SMP, and STP transport layer state machines to establish port connections and disconnections. The port layer state machines also interpret or pass transmit data, receive data, commands, and confirmations between the link and transport layers.

8.2 PL (port layer) state machines

8.2.1 PL state machines overview

The PL (port layer) consists of state machines that run in parallel and perform the following functions:

    a) receive requests from the SSP, SMP, and STP transport layer state machines for connection management (e.g., requests to open or close connections) and frame transmission;
    b) send requests to the SAS link layer state machines for connection management and frame transmission;
    c) receive confirmation from the SAS link layer state machines; and
    d) send confirmations to the SSP, SMP, and STP transport layer state machines.

The port layer state machines are as follows:

    a) PL_OC (port layer overall control) state machines (see 8.2.2); and
    b) PL_PM (port layer phy manager) state machines (see 8.2.3).

There is one PL_OC state machine per port (see 4.1.3). There is one PL_PM state machine for each phy contained in the port. Phys are assigned to ports by the management application layer. More than one port in a SAS device may have the same SAS address.

Figure 166 shows examples of the port layer state machines and their interaction with the transport and link layers.
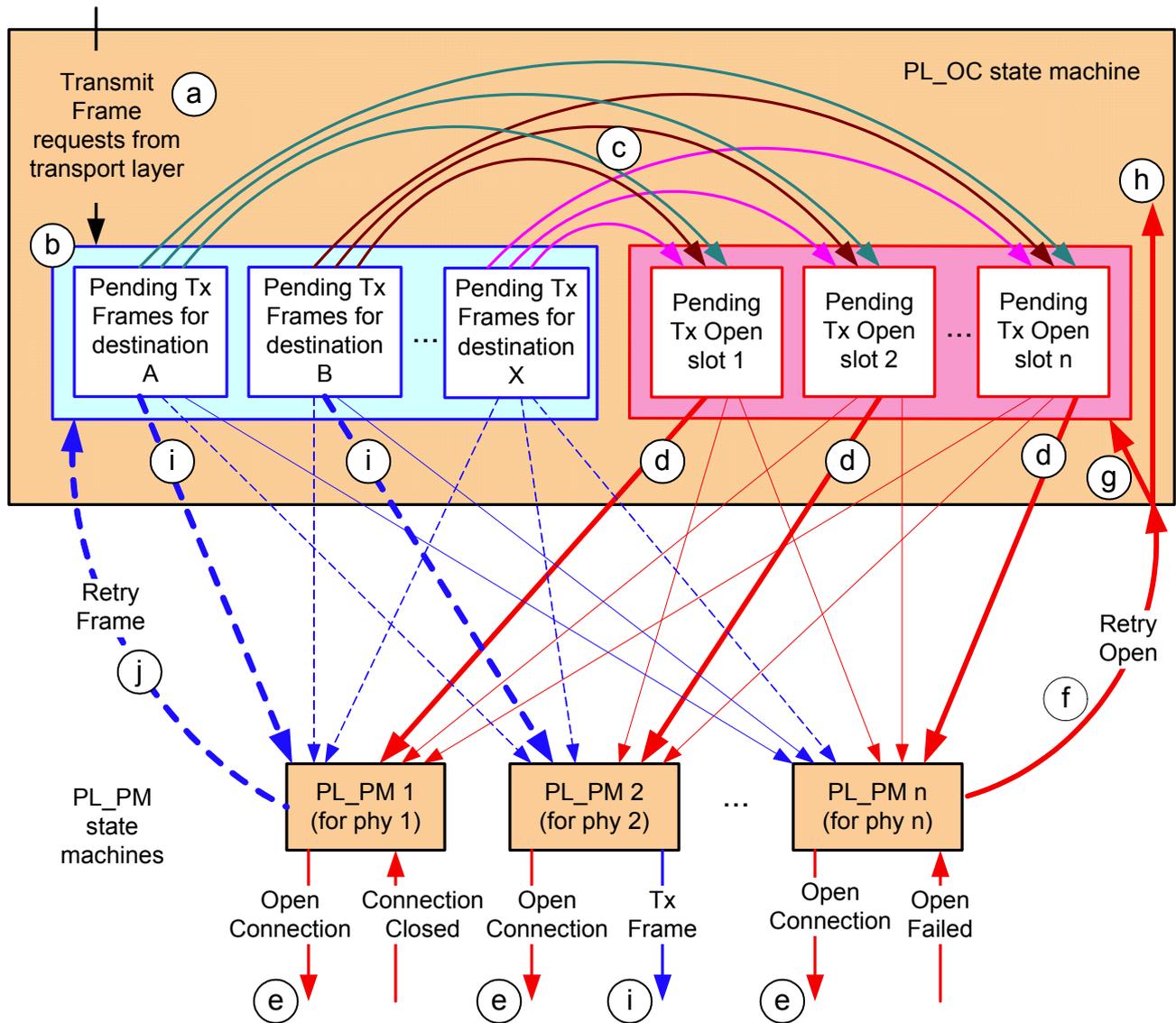
Figure 166 — Port layer examples

The following is a description of the example processes in figure 166. These example processes do not describe all of the possible condition or actions.

a)   Transmit Frame requests are received by the PL_OC state machine;

b)   the PL_OC state machine converts Transmit Frame requests into pending Tx Frame messages associated with the destination SAS address;

c)   the PL_OC state machine generates a pending Tx Open message for a pending Tx Frame message when there is a pending Tx Open slot available (i.e., the number of pending Tx Open messages is less than or equal to the number of phys);

d)   the PL_OC state machine sends a pending Tx Open message as a Tx Open message to a PL_PM state machine when a PL_PM machine is available; a slot is then available for a new pending Tx Open message;

e)   when a PL_PM state machine receives a Tx Open message, the PL_PM state machine attempts to establish a connection with the destination SAS address through the link layer;

f)   if a PL_PM state machine is unable to establish a connection with the destination SAS address, then the PL_PM state machine sends a Retry Open message to the PL_OC state machine;

g)   the PL_OC state machine converts a Retry Open message to a pending Tx Open message if there is a pending Tx Open slot available; if the PL_OC state machine converts a Retry Open message into a pending Tx Open message, then the pathway blocked count and arbitration wait time context from the Retry Open message are applied to the pending Tx Open message;

h) if the PL_OC state machine does not convert a Retry Open to a pending Tx Open frame, then the PL_OC discards the Retry Open message. The PL_OC state machine may create a new Tx Open message for the same pending Tx Frame at a later time. If the PL_OC state machine discards a Retry Open message, then the pathway blocked count and arbitration wait time context from the Retry Open message are also discarded;

i) after a PL_PM state machine establishes a connection with a destination SAS address, the PL_OC state machine sends pending Tx Frame messages for the destination to the PL_PM state machine as Tx Frame messages;

j) if a PL_PM state machine is unable to send a Tx Frame message to the link layer as a Tx Frame request (e.g., due to a credit timeout), then the PL_PM state machine sends a Retry Frame message to the PL_OC state machine, and the PL_OC state machine converts the Retry Frame message into a pending Tx Frame message; and

k) if the PL_PM state machine is able to send a Tx Frame message as a Tx Frame request to the link layer, then the PL_PM state machine sends a Transmission Status confirmation to the transport layer.

The Transmission Status confirmation from either the PL_OC state machine or a PL_PM state machine shall include the following as arguments:

a) tag;
b) destination SAS address; and
c) source SAS address.

8.2.2 PL_OC (port layer overall control) state machine

8.2.2.1 PL_OC state machine overview

A PL_OC state machine:

a) receives requests from the SSP, SMP, and STP transport layers;
b) sends messages to the PL_PM state machine;
c) receives messages from the PL_PM state machine;
d) selects frames to transmit;
e) selects phys on which to transmit frames;
f) receives confirmations from the link layer;
g) sends confirmations to the transport layer;
h) has Arbitration Wait Time timers; and
i) has I_T Nexus Loss timers.

This state machine consists of the following states:

a) PL_OC1:Idle (see 8.2.2.2) (initial state); and
b) PL_OC2:Overall_Control (see 8.2.2.3).

After power on this state machine shall start in the PL_OC1:Idle state.

The PL_OC state machine shall maintain:

a) a pool of pending Tx Frame messages for each destination SAS address; and
b) as many pending Tx Open message slots as there are phys in the port.

The PL_OC state machine shall maintain the timers listed in table 112.

Table 112 — PL_OC state machine timers

Timer	Maximum number of timers	Initial value
I_T Nexus Loss timer	One per destination SAS address	The value in the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page (see 10.2.7.2).
Arbitration Wait Time timer	One per pending Tx Open message	0000h, a vendor-specific value less than 8000h (see 7.12.3), or the value received with a Retry Open message.

Figure 167 shows the PL_OC state machine.
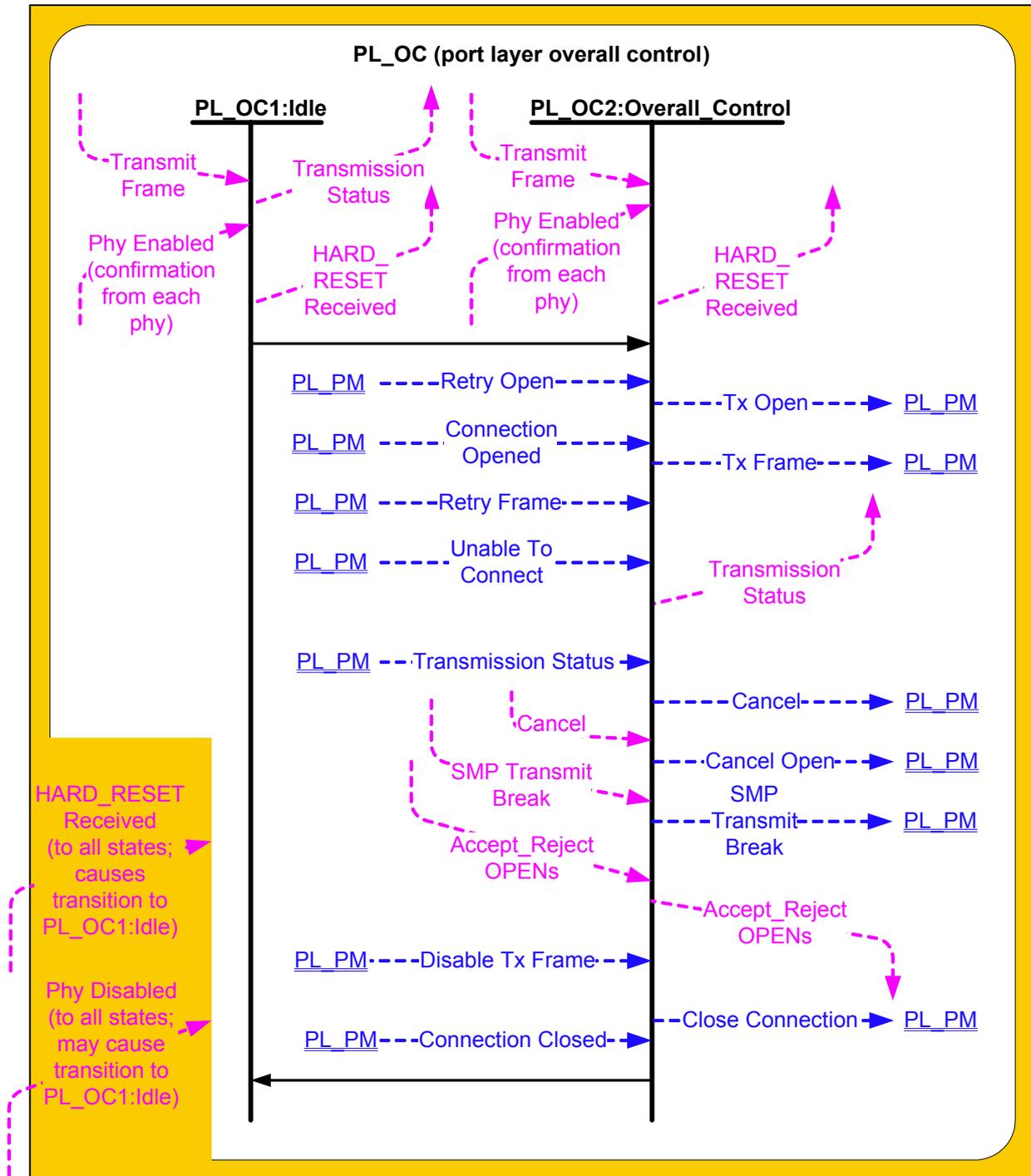
Figure 167 — PL_OC (port layer overall control) state machine

8.2.2.2 PL_OC1:Idle state

8.2.2.2.1 PL_OC1:Idle state description

This state is the initial state of the PL_OC state machine.

If this state receives a HARD_RESET Received confirmation, then this state shall send a HARD_RESET Received confirmation to the transport layer.

If this state receives a Transmit Frame request, then this state shall send a Transmission Status (No Phys In Port) confirmation to the transport layer.

If an I_T Nexus Loss timer expires for a destination SAS address, this state shall perform the following:

   a) delete the I_T Nexus Loss timer for the SAS address;
   b) send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message for the SAS address; and
   c) discard each pending Tx Frame message for the SAS address and any corresponding pending Tx Open messages.

8.2.2.2.2 Transition PL_OC1:Idle to PL_OC2:Overall_Control

This transition shall occur after a Phy Enabled confirmation is received for at least one phy assigned to the port.

8.2.2.3 PL_OC2:Overall_Control state

8.2.2.3.1 PL_OC2:Overall_Control state overview

This state may receive Transmit Frame requests from the transport layers (i.e., SSP and SMP) and Retry frame messages from PL_PM state machines. This state shall create a pending Tx Frame message for each received Transmit Frame request and Retry Frame message. There may be more than one pending Tx Frame message at a time for each SSP transport layer. There shall be only one pending Tx Frame message at a time for each SMP transport layer.

This state selects PL_PM state machines through which connections are established. This state shall only attempt to establish connections through PL_PM state machines whose phys are enabled. In a vendor-specific manner, this state selects PL_PM state machines on which connections are established to transmit frames. This state shall receive a response to a message from a PL_PM state machine before sending another message to that PL_PM state machine.

This state also:

   a) receives connection management requests from the transport layers;
   b) sends connection management messages to PL_PM state machines;
   c) receives connection management messages from PL_PM state machines; and
   d) sends connection management confirmations to the transport layers.

After receiving a Transmit Frame request for a destination SAS address for which there is no connection established and for which no I_T Nexus Loss timer has been created, this state shall create an I_T Nexus Loss timer for that SAS address if:

   a) the protocol is SSP;
   b) this state machine is in an SSP target port;
   c) the Protocol-Specific Port mode page is implemented by the SSP target port; and
   d) the I_T nexus loss time is not 0000h.

When this state creates an I_T Nexus Loss timer it shall:

   a) initialize the I_T Nexus Loss timer; and
   b) not start the I_T Nexus Loss timer.

If this state machine is in an SSP initiator port, then this state may create an I_T Nexus Loss timer for the SAS address. If a state machine in an SSP initiator port and creates an I_T Nexus Loss timer, then the state machine should use the value in the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page for the SSP target port (see 10.2.7.2) as the initial value for its I_T Nexus Loss timer.

Other SAS ports may detect an I_T nexus loss in a vendor-specific manner.

If there are no pending Tx Frame messages for a destination SAS address and an I_T Nexus Loss timer has been created for that destination SAS address, then this state shall delete the I_T Nexus Loss timer for that destination SAS address.

If this state receives a HARD_RESET Received confirmation, then this state shall discard all pending Tx Frame messages and delete all I_T Nexus Loss timers and send a HARD_RESET Received confirmation to the transport layer.

8.2.2.3.2 PL_OC2:Overall_Control state establishing connections

This state receives Phy Enabled confirmations indicating when a phy is available.

This state receives Retry Open messages from a PL_PM state machine.

This state creates pending Tx Open messages based on pending Tx Frame messages and Retry Open messages. Pending Tx Open messages are sent to a PL_PM state machine as Tx Open messages.

If this state receives a Retry Open (Retry) message, then this state shall process the Retry Open message.

If this state receives a Retry Open (No Destination) or a Retry Open (Open Timeout Occurred) message and an I_T Nexus Loss timer has not been created for the destination SAS address (e.g., an SSP target port does not support the I_T NEXUS LOSS TIME field in the Protocol-Specific Port mode page or the field is set to 0000h), then this state shall process the Retry Open message as either a Retry Open message or an Unable To Connect message. This selection is vendor-specific.

If this state receives a Retry Open (Pathway Blocked) message and an I_T Nexus Loss timer has not been created for the destination SAS address, then this state shall process the Retry Open message.

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message, and an I_T Nexus Loss timer has been created for the destination SAS address with an initial value of FFFFh, then this state shall process the Retry Open message (i.e., the Retry Open message is never processed as an Unable to Connect message).

If this state receives a Retry Open (No Destination) or a Retry Open (Open Timeout Occurred) message, an I_T Nexus Loss timer has been created for the destination SAS address, and there is no connection established with the destination SAS address, then this state shall check the I_T Nexus Loss timer, and:

   a) if the I_T Nexus Loss timer is not running and the I_T nexus loss time is not set to FFFFh, then this state shall start the timer;
   b) if the I_T Nexus Loss timer is running, then this state shall not stop the timer; and
   c) if the I_T Nexus Loss timer has expired, then this state shall process the Retry Open message as if it were an Unable To Connect message (see 8.2.2.3.4).

If this state receives a Retry Open (Pathway Blocked) message, an I_T Nexus Loss timer has been created for the destination SAS address, and there is no connection established with the destination SAS address, then this state shall check the I_T Nexus Loss timer, and:

   a) if the I_T Nexus Loss timer is running, then this state shall not stop the timer; and
   b) if the I_T Nexus Loss timer has expired, then this state shall process the Retry Open message as if it were an Unable To Connect message (see 8.2.2.3.4).

If this state receives a Retry Open (Retry) and an I_T Nexus Loss timer is running for the destination SAS address, then this state shall:

   a) stop the I_T Nexus Loss timer (if the timer has been running); and
   b) initialize the I_T Nexus Loss timer.

This state shall create a pending Tx Open message if:

   a) this state has a pending Tx Frame message or has received a Retry Open message;
   b) this state has fewer pending Tx Open messages than the number of PL_PM state machines (i.e., the number of phys in the port);
   c) there is no pending Tx Open message for the destination SAS address; and
   d) there is no connection established with the destination SAS address.

This state may create a pending Tx Open message if:

   a) this state has a pending Tx Frame message, or this state has received a Retry Open message and has not processed the message by sending a confirmation; and
   b) this state has fewer pending Tx Open messages than the number of PL_PM state machines.

This state shall have no more pending Tx Open messages than the number of PL_PM state machines.

If this state receives a Retry Open message and there are pending Tx Frame messages for which pending Tx Open messages have not been created, then this state should create a pending Tx Open message from the Retry Open message.

If this state does not create a pending Tx Open message from a Retry Open message (e.g., the current number of pending Tx Open messages equals the number of phys), then this state shall discard the Retry Open message. This state may create a new pending Tx Open message at a later time for the pending Tx Frame message that resulted in the Retry Open message.

If this state receives a Retry Open (Opened By Destination) message and the initiator port bit and protocol arguments match those in the Tx Open messages that resulted in the Retry Open message, then this state may discard the Retry Open message and use the established connection to send pending Tx Frame messages as Tx Frame messages to the destination SAS address. If this state receives a Retry Open (Opened By Destination) message, then, if this state has a pending Tx Open slot available, this state may create a pending Tx Open message from the Retry Open message.

> NOTE 47 - If a connection is established by another port as indicated by a Retry Open (Opened By Destination) message, credit may not be granted for frame transmission. In this case this state may create a pending Tx Open message from a Retry Open message in order to establish a connection where credit is granted.

This state shall send a pending Tx Open message as a Tx Open message to a PL_PM state machine that has an enabled phy and does not have a connection established. If there is more than one pending Tx Open message, this state should send a Tx Open message for the pending Tx Open message that has been pending for the longest time first.

If this state creates a pending Tx Open message from one of the following messages:

  a) a Retry Open (Opened By Destination);
  b) a Retry Open (Opened By Other);
  c) a Retry Open (Collided); or
  d) a Retry Open (Pathway Blocked),

then this state shall:

  a) create an Arbitration Wait Time timer for the pending Tx Open message;
  b) set the Arbitration Wait Time timer for the pending Tx Open message to the arbitration wait time argument from the Retry Open message; and
  c) start the Arbitration Wait Time timer for the pending Tx Open message.

When a pending Tx Open message is sent to a PL_PM state machine as a Tx Open message, the Tx Open message shall contain the following arguments to be used in an OPEN address frame:

  a) initiator port bit from the Transmit Frame request;
  b) protocol from the Transmit Frame request;
  c) connection rate from the Transmit Frame request;
  d) initiator connection tag from the Transmit Frame request;
  e) destination SAS address from the Transmit Frame request;
  f) source SAS address from the Transmit Frame request;
  g) pathway blocked count; and
  h) arbitration wait time.

If this state creates a pending Tx Open message from one of the following:

  a) a Transmit Frame request;
  b) a Retry Open (No Destination) message;
  c) a Retry Open (Open Timeout Occurred) message; or
  d) a Retry Open (Retry) message,

then this state shall:

  a) set the pathway blocked count argument in the Tx Open message to zero; and
  b) set the arbitration wait time argument in the Tx Open message to zero or a vendor-specific value less than 8000h (see 7.12.3).

If a pending Tx Open message was created as the result this state receiving a Retry Open (Pathway Blocked) message, then this state shall set the pathway blocked count argument in the Tx Open message to the value of the pathway blocked count argument received with the message plus one, unless the pathway blocked count received with the argument is FFh.

If a pending Tx Open message was created as the result of this state receiving one of the following:

  a) a Retry Open (Opened By Destination) message;
  b) a Retry Open (Opened By Other) message;
  c) a Retry Open (Collided) message; or
  d) a Retry Open (Pathway Blocked) message;

then this state shall set the arbitration wait time argument in the Tx Open message to be the value from the Arbitration Wait Time timer created as a result of the Retry Open message.

After this state sends a Tx Open message, this state shall discard the pending Tx Open message from which the Tx Open messages was created. After this state discards a pending Tx Open message, this state may create a new pending Tx Open message.

If this state receives a Connection Opened message and the initiator port bit and protocol arguments match those in any pending Tx Frame messages, then this state may use the established connection to send pending Tx Frame messages as Tx Frame messages to the destination SAS address.

8.2.2.3.3 PL_OC2:Overall_Control state connection established

If this state receives a Connection Opened message or a Retry Open (Opened By Destination) message for a SAS address, and an I_T Nexus Loss timer has been created for the SAS address, then this state shall:

    a)   stop the I_T Nexus Loss timer for the SAS address, if the timer has been running; and
    b)   initialize the I_T Nexus Loss timer.

8.2.2.3.4 PL_OC2:Overall_Control state unable to establish a connection

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message and the I_T Nexus Loss timer for the SAS address has expired, then this state shall perform the following:

    a)   delete the I_T Nexus Loss timer for the SAS address;
    b)   discard the Retry Open message;
    c)   send a Transmission Status (I_T Nexus Loss) confirmation for the pending Tx Frame message from which the Retry Open message resulted;
    d)   discard the pending Tx Frame message from which the Retry Open message resulted;
    e)   if this state has any pending Tx Frame messages with the same destination SAS address and protocol as the Retry Open message, and this state has not sent a Tx Open message to a PL_PM state machine for the messages, then this state shall send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message and discard the pending Tx Frame messages and any corresponding pending Tx Open messages; and
    f)   if this state has any pending Tx Frame messages with the same destination SAS address and protocol as the Retry Open message, and this state has sent a Tx Open message to a PL_PM state machine for a message, then this state shall send a Cancel Open message to each PL_PM state machine to which it has sent a Tx Open message. After receiving an Unable To Connect (Cancel Acknowledge) message from a PL_PM state machine in response to the Cancel Open message, then this state shall send a Transmission Status (I_T Nexus Loss) confirmation for each pending Tx Frame message and discard the pending Tx Frame messages and any corresponding pending Tx Open messages.

If this state receives a Retry Open (No Destination), Retry Open (Open Timeout Occurred), or Retry Open (Pathway Blocked) message and processes it as an Unable To Connect message, or this state receives an Unable To Connect message, then this state shall send a Transmission Status confirmation as defined in table 113.

Table 113 — Confirmations from Unable To Connect or Retry Open messages

Message received	Confirmation to be sent to transport layer
Retry Open (No Destination)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired, or Transmission Status (No Destination) if it has not
Retry Open (Open Timeout Occurred)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired, or Transmission Status (Open Timeout Occurred) if it has not
Retry Open (Pathway Blocked)	Transmission Status (I_T Nexus Loss) if the I_T Nexus Loss timer for the SAS address has expired
Unable to Connect (Bad Destination)	Transmission Status (Bad Destination)
Unable To Connect (Break Received)	Transmission Status (Break Received)
Unable To Connect (Connection Rate Not Supported)	Transmission Status (Connection Rate Not Supported)
Unable To Connect (Port Layer Request)	Transmission Status (Cancel Acknowledge)
Unable To Connect (Protocol Not Supported)	Transmission Status (Protocol Not Supported)
Unable To Connect (STP Resources Busy)	Transmission Status (STP Resources Busy)
Unable To Connect (Wrong Destination)	Transmission Status (Wrong Destination)

If this state receives an Unable To Connect (Connection Rate Not Supported), Unable To Connect (Protocol Not Supported), or Unable To Connect (STP Resources Busy) message and an I_T Nexus Loss timer is running for the SAS address, then this state shall:

 a) stop the I_T Nexus Loss timer, if the timer has been running; and
 b) initialize the I_T Nexus Loss timer.

This state shall discard the pending Tx Frame message for which the Transmission Status confirmation was sent.

8.2.2.3.5 PL_OC2:Overall_Control state connection management

If this state receives an Accept_Reject Opens request, then this state shall send an Accept_Reject Opens message to all phys in the port.

If this state receives an SMP Transmit Break request, then this state shall send an SMP Transmit Break message to the PL_PM state machine associated with the corresponding SMP transport state machine. If there is no PL_PM state machine associated with the request, the PM_OC state shall ignore the request.

If this state receives one of the following:

 a) a Connection Closed (Close Timeout) message;
 b) a Connection Closed (Break Requested) message; or
 c) a Connection Closed (Break Received) message,

then this state shall not send a Tx Open or Tx Frame message to the PL_PM state machine that sent the message until this state receives a Connection Closed (Transition to Idle) message from that PL_PM state machine.

If this state receives a Connection Closed (Normal) message or a Connection Closed (Transition to Idle) message indicating that a connection with a destination SAS address is no longer open and this state has pending Tx Open messages, then this state may send a Tx Open message to the PL_PM state machine that sent the Connection Closed message.

If this state is in a wide SSP port, then this state shall not reject an incoming connection request on one phy because it has an outgoing connection request on another phy.

If this state is in an SSP port, there are no pending Tx Frame messages for a destination SAS address with which a PL_PM state machine has established a connection, and the connection was established by a message from this state, then this state should send a Close Connection message to the PL_PM state machine.

If this state is in an SSP port, has no pending Tx Frame messages for a destination SAS address with which a PL_PM state machine has established a connection, and the connection was established by the destination, then this state may wait a vendor-specific time and then shall send a Close Connection message to the PL_PM state machine.

If this state has received a Disable Tx Frame message from a PL_PM state machine, then this state should send a Close Connection message to the PL_PM state machine.

> NOTE 48 - The PL_PM state machine sends a Close Connection request to the link layer upon receipt of a Close Connection message or on expiration of the Bus Inactivity Time Limit timer (see 8.2.3.4.1).

8.2.2.3.6 PL_OC2:Overall_Control state frame transmission

In order to prevent livelocks, If this state is in a wide SSP port, has multiple connections established, and has a pending Tx Frame message, then this state shall send at least one Tx Frame message to a PL_PM state machine before sending a Close Connection message to the PL_PM state machine.

After this state receives a Connection Opened message from a PL_PM state machine, this state selects pending Tx Frame messages for the destination SAS address with the same initiator port bit and protocol arguments, and, as an option, the same connection rate argument, and sends the messages to the PL_PM state machine as Tx Frame messages.

This state may send a Tx Frame message to any PL_PM state machine that has established a connection with the destination SAS address when the initiator port bit and protocol arguments match those in the Tx Frame message.

After this state sends a Tx Frame message to a PL_PM state machine, it shall not send another Tx Frame message to that PL_PM state machine until it receives a Transmission Status (Frame Transmitted) message.

This state may send a Tx Frame message containing a COMMAND frame for a destination SAS address to a PL_PM state machine while waiting for one of the following messages for Tx Frame messages containing COMMAND frames for the same destination SAS address from different PL_PM state machines:

a) Transmission Status (ACK Received);
b) Transmission Status (NAK Received);
c) Transmission Status (ACK/NAK Timeout); or
d) Transmission Status (Connection Lost Without ACK/NAK).

This state shall not send a Tx Frame message containing a TASK frame for a task that only affects an I_T_L_Q nexus (e.g., an ABORT TASK or QUERY TASK task management function (see SAM-3)) until this state has received one of the following messages for each Tx Frame message with the same I_T_L_Q nexus:

a) Transmission Status (ACK Received);
b) Transmission Status (NAK Received);
c) Transmission Status (ACK/NAK Timeout); or
d) Transmission Status (Connection Lost Without ACK/NAK).

This state shall not send a Tx Frame message containing a TASK frame for a task that only affects an I_T_L nexus (e.g., an ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, or LOGICAL UNIT RESET task management function (see SAM-3)) until this state has received one of the following messages for each Tx Frame message with the same I_T_L nexus:

a) Transmission Status (ACK Received);
b) Transmission Status (NAK Received);
c) Transmission Status (ACK/NAK Timeout); or
d) Transmission Status (Connection Lost Without ACK/NAK).

This state shall not send a Tx Frame message containing a TASK frame for a task that only affects an I_T nexus until this state has received one of the following messages for each Tx Frame message with the same I_T nexus:

a) Transmission Status (ACK Received);

   b) Transmission Status (NAK Received);
   c) Transmission Status (ACK/NAK Timeout); or
   d) Transmission Status (Connection Lost Without ACK/NAK).

Once this state has sent a Tx Frame message containing a DATA frame to a PL_PM state machine, this state shall not send a Tx Frame message containing a DATA frame with the same I_T_L_Q to another PL_PM state machine until this state has received one of the following messages for each Tx Frame message containing a DATA frame for the same I_T_L_Q nexus:

   a) Transmission Status (ACK Received);
   b) Transmission Status (NAK Received);
   c) Transmission Status (ACK/NAK Timeout); or
   d) Transmission Status (Connection Lost Without ACK/NAK).

Read DATA frames and write DATA frames for the same I_T_L_Q nexus may be transmitted and received simultaneously on the same or different phys.

If this state is in an SMP initiator port, then this state shall send the Tx Frame message containing the SMP REQUEST frame to the PL_PM state machine on which the connection was established for the Tx Open message. If this state is in an SMP target port, then this state shall send the Tx Frame message containing the SMP RESPONSE frame to the PL_PM state machine on which the connection was established for the Tx Open message. See 7.18 for additional information about SMP connections.

Characteristics of STP connections are defined by SATA (also see 7.17).

The following arguments shall be included with the Tx Frame message:

   a) the frame to be transmitted; and
   b) Balance Required or Balance Not Required.

A Balance Not Required argument shall only be included if:

   a) the request was a Transmit Frame (Non-Interlocked) request (i.e., the request included a DATA frame); and
   b) the last Tx Frame message sent to this PL_PM state machine while this connection has been established was for a DATA frame having the same logical unit number and tag value as the DATA frame in this Tx Frame message.

If a Balance Not Required argument is not included in the Tx Frame message, then a Balance Required argument shall be included.

If this state receives a Disable Tx Frames message from a PL_PM state machine, then this state should send no more Tx Frame messages to that state machine until a new connection is established.

8.2.2.3.7 PL_OC2:Overall_Control state frame transmission cancellations

Cancel requests cause this state to cancel previous Transmit Frame requests. A Cancel request includes the following arguments:

   a) destination SAS address; and
   b) tag.

If this state receives a Cancel request and has not already sent a Tx Frame message for the Transmit Frame request to a PL_PM state machine for the Transmit Frame request specified by the Cancel request, then this state shall:

   a) discard all Transmit Frame requests for the specified destination SAS address and tag; and
   b) send a Transmission Status (Cancel Acknowledge) confirmation to the transport layer.

If this state receives a Cancel request and has already sent a Tx Frame message to a PL_PM state machine for the Transmit Frame request specified by the Cancel request, then this state shall send a Cancel message to the PL_PM state machine to which the Tx Frame message was sent. The Cancel message shall include the tag.

8.2.2.3.8 Transition PL_OC2:Overall_Control to PL_OC1:Idle

This transition shall occur after:

   a) sending a HARD_RESET Received confirmation to the transport layer; or
   b) a Phy Disabled confirmation is received from all of the link layers in the port.

8.2.3 PL_PM (port layer phy manager) state machine

8.2.3.1 PL_PM state machine overview

A PL_PM state machine:

   a)   receives messages from the PL_OC state machine;
   b)   sends requests to the link layer;
   c)   receives confirmations from the link layer;
   d)   sends confirmations to the transport layer;
   e)   sends messages to PL_OC state machine;
   f)   has an Arbitration Wait Time timer;
   g)   may have a Bus Inactivity Time Limit timer; and
   h)   may have Maximum Connect Time Limit timer.

This state machine consist of the following states:

   a)   PL_PM1:Idle (see 8.2.3.2) (initial state);
   b)   PL_PM2:Req_Wait (see 8.2.3.3);
   c)   PL_PM3:Connected (see 8.2.3.4); and
   d)   PL_PM4:Wait_For_Close (see 8.2.3.5).

After power on this state machine shall start in the PL_PM1:Idle state.

The PL_PM state machine shall maintain the timers listed in table 114.

Table 114 — PL_PM state machine timers

Timer	Initial value
Arbitration Wait Time timer	The arbitration wait time argument from a Retry Open message (see 8.2.2.3.1).
Bus Inactivity Time Limit timer	The value in the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1).
Maximum Connect Time Limit timer	The value in the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1).

Figure 168 shows part 1 of the PL_PM state machine.
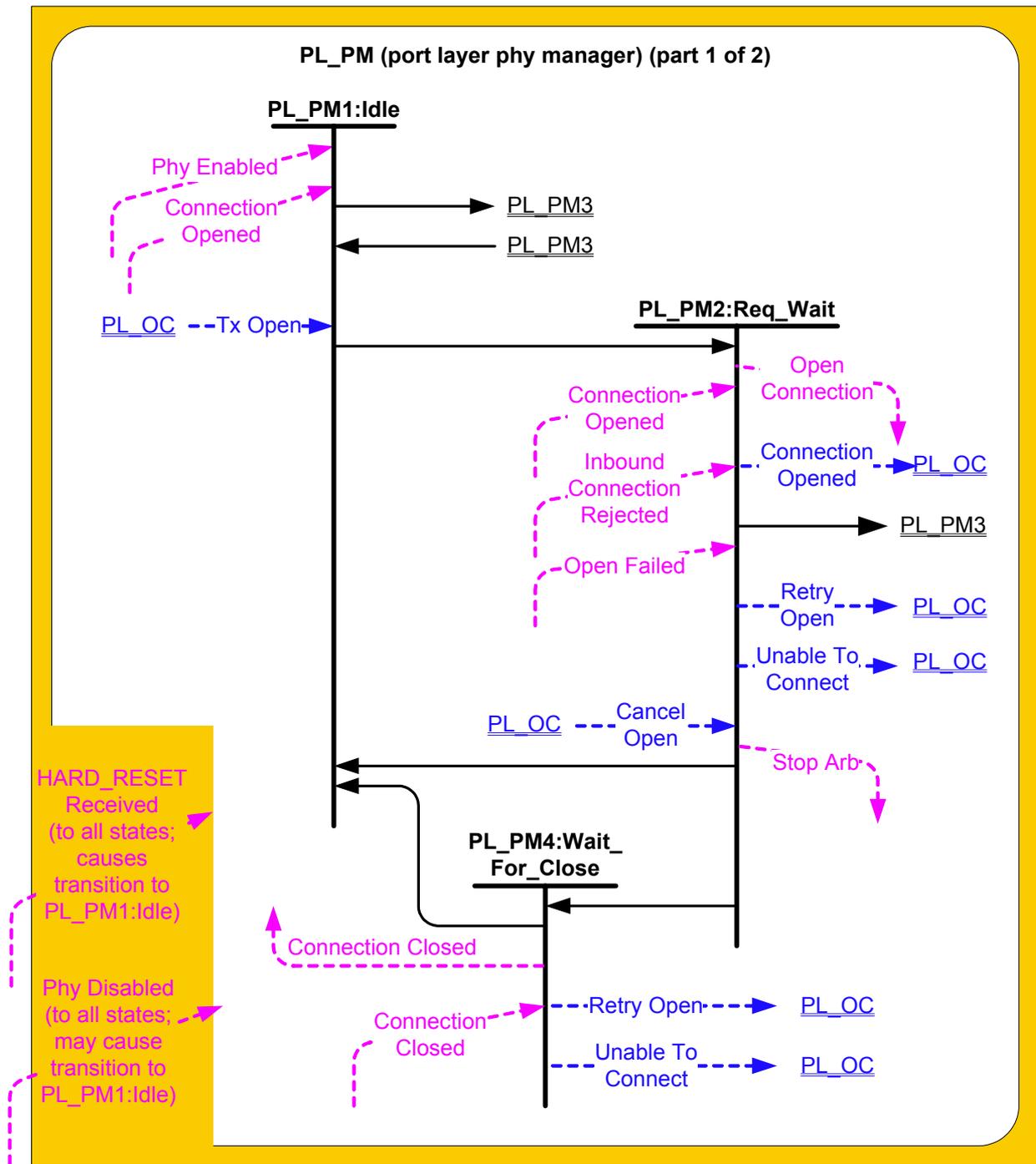
Figure 168 — PL_PM (port layer phy manager) state machine (part 1)

Figure 169 shows part 2 of the PL_PM state machine.
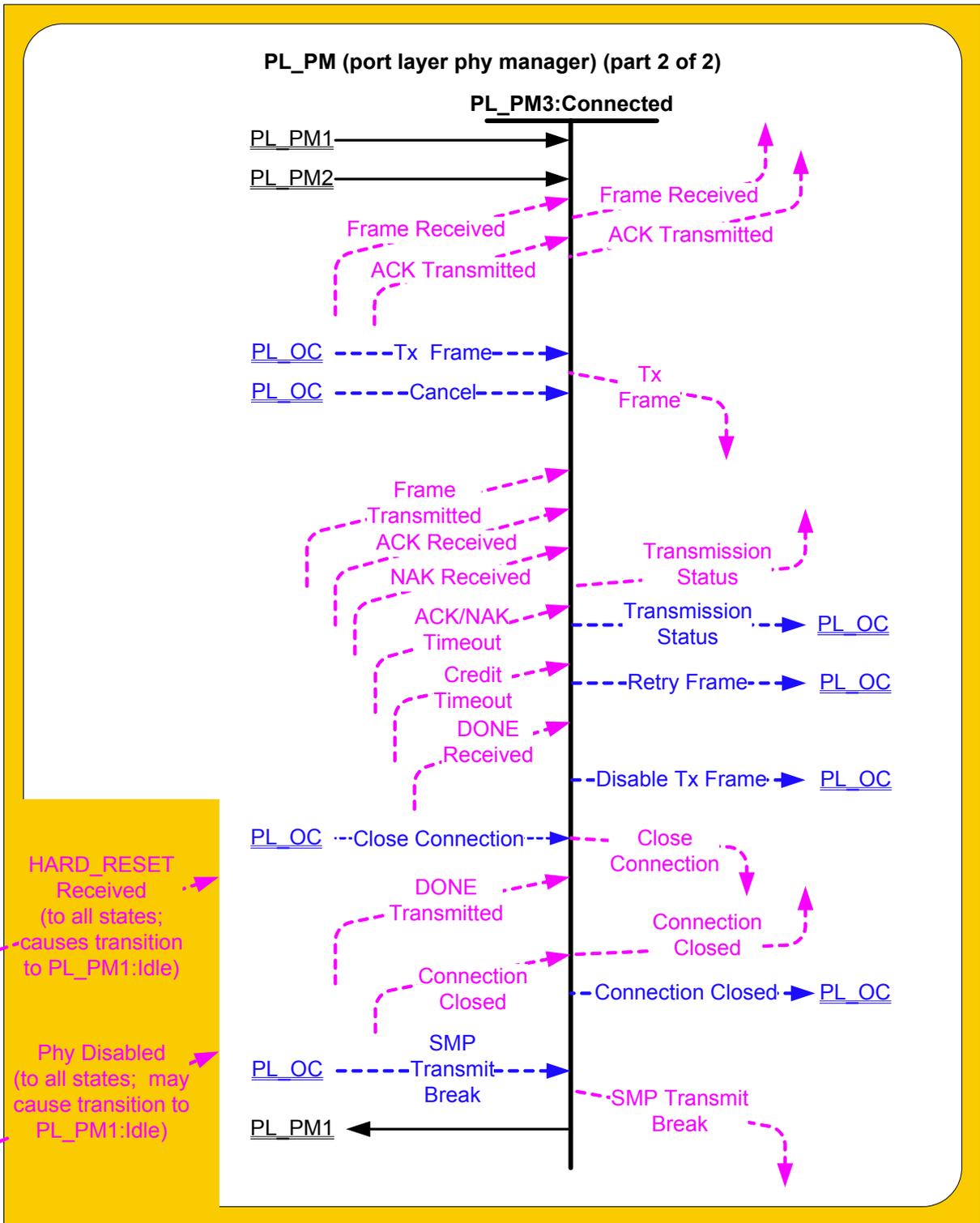
Figure 169 — PL_PM (port layer phy manager) state machine (part 2)

8.2.3.2 PL_PM1:Idle state

8.2.3.2.1 PL_PM1:Idle state description

This is the initial state of the PL_PM state machine.

8.2.3.2.2 Transition PL_PM1:Idle to PL_PM2:Req_Wait

This transition shall occur after:

- a) a Phy Enabled confirmation is received; and
- b) a Tx Open message is received.

8.2.3.2.3 Transition PL_PM1:Idle to PL_PM3:Connected

This transition shall occur after a Connection Opened confirmation is received.

8.2.3.3 PL_PM2:Req_Wait state

8.2.3.3.1 PL_PM2:Req_Wait state overview

This state sends an Open Connection request to the link layer and waits for a confirmation. This state sends and receives connection management messages to and from the PL_OC state machine.

If this state receives a HARD_RESET Received confirmation, then this state shall terminate all operations.

8.2.3.3.2 PL_PM2:Req_Wait establishing a connection

Upon entry into this state, this state shall:

- a) create an Arbitration Wait Time timer;
- b) initialize the Arbitration Wait Time timer to the arbitration wait time argument received with the Tx Open message;
- c) start the Arbitration Wait Time timer; and
- d) send an Open Connection request to the link layer.

The Open Connection request shall contain the following arguments from the Tx Open message to be used in an OPEN address frame:

- a) initiator port bit;
- b) protocol;
- c) connection rate;
- d) initiator connection tag;
- e) destination SAS address;
- f) source SAS address;
- g) pathway blocked count; and
- h) arbitration wait time.

8.2.3.3.3 PL_PM2:Req_Wait connection established

If this state receives a Connection Opened confirmation, then this state shall send a Connection Opened message to the PL_OC state machine.

If this state receives a Connection Opened confirmation and the confirmation was not in response to an Open Connection request from this state (i.e., the connection was established in response to an OPEN address frame from another device), then this state shall discard any Open Connection request and send a Retry Open message to the PL_OC state machine. If the Connection Opened confirmation was from the destination of the Open Connection request, then this state shall send a Retry Open (Opened By Destination) message. If the Connection Opened confirmation was from a destination other than the destination of the Open Connection request, then this state shall send a Retry Open (Opened By Other) message.

A Retry Open (Opened By Destination) or Retry Open (Opened By Other) message shall contain the following arguments:

- a) initiator port bit set to the value received with the Tx Open message;
- b) protocol set to the value received with the Tx Open message;
- c) connection rate set to the value received with the Tx Open message;
- d) initiator connection tag set to the value received with the Tx Open message;
- e) destination SAS address set to the value received with the Tx Open message;
- f) source SAS address set to the value received with the Tx Open message;
- g) pathway blocked count set to the value received with the Tx Open message; and
- h) arbitration wait time set to the value of the Arbitration Wait Time timer.

8.2.3.3.4 PL_PM2:Req_Wait unable to establish a connection

If this state receives one of the Open Failed confirmations listed in table 115, then this state shall send either a Retry Open message or an Unable To Connect message to the PL_OC state machine.

Table 115 defines the message to be sent for each Open Failed confirmation.

Table 115 — Messages from Open Failed confirmations

Confirmation received	Message to be sent to PL_OC
Open Failed (Pathway Blocked)	Retry Open (Pathway Blocked)
Open Failed (Retry)	Retry Open (Retry)
Open Failed (No Destination)	Retry Open (No Destination)
Open Failed (Bad Destination)	Unable To Connect (Bad Destination)
Open Failed (Connection Rate Not Supported)	Unable To Connect (Connection Rate Not Supported)
Open Failed (Protocol Not Supported)	Unable To Connect (Protocol Not Supported)
Open Failed (STP Resources Busy)	Unable To Connect (STP Resources Busy)
Open Failed (Wrong Destination)	Unable To Connect (Wrong Destination)

If this state receives an Inbound Connection Rejected confirmation after sending an Open Connection request, then this state shall discard the Open Connection request and send a Retry Open (Collided) message to the PL_OC state machine.

A Retry Open message shall include the following arguments:

a) initiator port bit set to the value received with the Tx Open message;
b) protocol set to the value received with the Tx Open message;
c) connection rate set to the value received with the Tx Open message;
d) initiator connection tag set to the value received with the Tx Open message;
e) destination SAS address set to the value received with the Tx Open message;
f) source SAS address set to the value received with the Tx Open message;
g) pathway blocked count argument set to the value received with the Tx Open message; and
h) arbitration wait time set to the value of the Arbitration Wait Time timer.

An Unable To Connect message shall include the following arguments:

a) initiator connection tag set to the value received with the Tx Open message;
b) destination SAS address set to the value received with the Tx Open message; and
c) source SAS address set to the value received with the Tx Open message.

8.2.3.3.5 PL_PM2:Req_Wait connection management

If this state receives a Cancel Open message and a Connection Opened confirmation has not been received, then this state shall send a Stop Arb request to the link layer.

8.2.3.3.6 Transition PL_PM2:Req_Wait to PL_PM1:Idle

This transition shall occur after:

a) a Retry Open message is sent to the PL_OC state machine;
b) an Unable To Connect message is sent to the PL_OC state machine;
c) all operations have been terminated after a HARD_RESET Received confirmation is received; or
d) a Phy Disabled confirmation is received.

8.2.3.3.7 Transition PL_PM2:Req_Wait to PL_PM3:Connected

This transition shall occur after a Connection Opened confirmation is received.

8.2.3.3.8 Transition PL_PM2:Req_Wait to PL_PM4:Wait_For_Close

This transition shall occur after one of the following confirmations is received:

    a) an Open Failed (Open Timeout Occurred);
    b) an Open Failed (Break Received); or
    c) an Open Failed (Port Layer Request).

8.2.3.4 PL_PM3:Connected state

8.2.3.4.1 PL_PM3:Connected state description

If the protocol for the connection is SSP, and this state is in an SSP target port, and the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1) is not set to zero, then, upon entry into this state, this state shall:

    a) create a Maximum Connect Time Limit timer;
    b) initialize the Maximum Connect Time Limit timer; and
    c) start the Maximum Connect Time Limit timer.

If the protocol for the connection is SSP, and this state is in an SSP target port, and the MAXIMUM CONNECT TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1) is set to zero, then this state shall not create a Maximum Connect Time Limit timer (i.e., there is no maximum connect time limit).

Other SAS ports may implement a Maximum Connect Time Limit timer in a vendor-specific manner.

If the protocol for the connection is SSP, and this state is in an SSP target port, and the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1) is set to a non-zero value, then, upon entry into this state, this state shall:

    a) create a Bus Inactivity Time Limit timer;
    b) initialize the Bus Inactivity Time Limit timer; and
    c) not start the Bus Inactivity Time Limit timer.

If the protocol for the connection is SSP, and this state is in an SSP target port, and the BUS INACTIVITY TIME LIMIT field in the Disconnect-Reconnect mode page (see 10.2.7.1) is set to zero, then this state shall not create a Bus Inactivity Time Limit timer (i.e., there is no maximum bus inactivity time limit).

Other SAS ports may implement a Bus Inactivity Time Limit timer in a vendor-specific manner.

If a Bus Inactivity Time Limit timer has been created and this state receives a Tx Frame message, then this state shall:

    a) stop the Bus Inactivity Time Limit timer, if it is running; and
    b) initialize the Bus Inactivity Time Limit timer.

If this state receives a Tx Frame message, this state shall send a Tx Frame request to the link layer. The following arguments from the Tx Frame message shall be included with the Tx Frame request:

    a) the frame to be transmitted; and
    b) if this state is in an SSP port, Balance Required or Balance Not Required.

For STP connections, this state connects the STP transport layer to the STP link layer.

If a Bus Inactivity Time Limit timer has been created and this state receives an ACK Received or NAK Received confirmation, then this state shall start the Bus Inactivity Time Limit timer. If the Bus Inactivity Time Limit timer expires before this state receives a Tx Frame message, then this state shall send a Close Connection request to the link layer.

If a Maximum Connect Time Limit timer has been created and this state receives an ACK Received or NAK Received confirmation, then this state shall check the Maximum Connect Time Limit timer. If the Maximum Connect Time Limit timer has expired, then this state shall send a Close Connection request to the link layer.

If this state receives a Tx Frame message after sending a Close Connection request but before receiving a Connection Closed confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

If this state receives a Frame Received confirmation, then this state shall send a Frame Received confirmation to the transport layer. The confirmation shall include the arguments received with the confirmation (e.g., the frame).

If this state receives an ACK Transmitted confirmation, then this state shall send an ACK Transmitted confirmation to the transport layer including the tag of the frame that was ACKed.

If this state receives a Frame Transmitted confirmation, then this state shall send a Transmission Status (Frame Transmitted) confirmation to the transport layer.

If this state receives an ACK Received confirmation, then this state shall send a Transmission Status (ACK Received) confirmation to the transport layer.

If this state receives a NAK Received confirmation, then this state shall send a Transmission Status (NAK Received) confirmation to the transport layer.

If this state receives an ACK/NAK Timeout confirmation, then this state shall send a Transmission Status (ACK/ NAK Timeout) confirmation to the transport layer.

If this state receives a Cancel message, then this state shall:

   a)   discard all Tx Frame requests for the specified tag;
   b)   send a Transmission Status (Cancel Acknowledge) confirmation to the transport layer including the destination SAS address and the tag as arguments; and
   c)   discard any subsequent confirmations for previous Tx Frame requests sent for the tag.

If this state receives a Close Connection message from the PL_OC state machine, then this state shall send a Close Connection request to the link layer.

If this state receives one of the following:

   a)   a Connection Closed (Normal) confirmation;
   b)   a Connection Closed (Close Timeout) confirmation;
   c)   a Connection Closed (Break Requested) confirmation;
   d)   a Connection Closed (Break Received) confirmation; or
   e)   a Connection Closed (Transition to Idle) confirmation,

then this state shall send a Connection Closed message to the PL_OC state machine including the argument received with the confirmation.

If this state receives a Connection Closed (Transition to Idle) confirmation after receiving:

   a)   a Connection Closed (Break Received) confirmation; or
   b)   a Connection Closed (Break Requested) confirmation,

then this state shall send a Transmission Status (Break Received) confirmation to the transport layer.

If this state receives a Connection Closed (Normal) confirmation, a Connection Closed (Transition to Idle) confirmation, or a Phy Disabled confirmation after sending a Transmission Status (Frame Transmitted) confirmation, but before this state receives an ACK Received or NAK Received confirmation, then this state shall send a Transmission Status (Connection Lost Without ACK/NAK) confirmation to the transport layer.

If this state receives a Connection Closed (Normal) confirmation, a Connection Closed (Transition to Idle) confirmation, or a Phy Disabled confirmation after sending a Tx Frame request but before receiving a Frame Transmitted confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

If this state receives a Connection Closed confirmation during an SMP connection, this state shall send a Connection Closed confirmation to the transport layer.

If this state receives a Credit Timeout confirmation, then this state shall send a Retry Frame message to the PL_OC state machine.

A Retry Frame message shall include the following arguments from the Tx Frame message:

   a)   initiator port bit;
   b)   protocol;
   c)   connection rate;
   d)   initiator connection tag;
   e)   destination SAS address;
   f)   source SAS address; and
   g)   frame.

After this state receives a DONE Received (Normal) or DONE Received (Credit Blocked) confirmation, if it does not receive a Tx Frame message within 1 ms, then this state shall send a Disable Tx Frames message to the PL_OC state machine.

If this state receives a DONE Received (ACK/NAK Timeout) or DONE Transmitted confirmation, then this state shall send a Disable Tx Frames message to the PL_OC state machine.

If this state receives an SMP Transmit Break message, then this state shall send an SMP Transmit Break request to the link layer.

If this state receives a HARD_RESET Received confirmation, then this state machine shall terminate all operations.

8.2.3.4.2 Transition PL_PM3:Connected to PL_PM1:Idle

This transition shall occur after:

   a)   a Connection Closed (Transition to Idle) message is sent to the PL_OC state machine; or
   b)   all operations are terminated after a HARD_RESET Received confirmation is received.

8.2.3.5 PL_PM4:Wait_For_Close state

8.2.3.5.1 PL_PM4:Wait_For_Close state description

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered as the result of the PL_PM2:Req_Wait state receiving an Open Failed (Open Timeout Occurred) confirmation, then this state shall send a Retry Open (Open Timeout Occurred) message to the PL_OC state machine. The Retry Open message shall include the following arguments:

   a)   initiator port bit set to the value received with the Tx Open message;
   b)   protocol set to the value received with the Tx Open message;
   c)   connection rate set to the value received with the Tx Open message;
   d)   initiator connection tag set to the value received with the Tx Open message;
   e)   destination SAS address set to the value received with the Tx Open message;
   f)   source SAS address set to the value received with the Tx Open message;
   g)   pathway blocked count argument set to the value received with the Tx Open message; and
   h)   arbitration wait time set to the value of the Arbitration Wait Time timer.

If this state receives a Connection Closed confirmation and the connection request was for an SMP connection, this state shall send a Connection Closed confirmation to the transport layer.

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered after the PL_PM2:Req_Wait state received an Open Failed (Port Layer Request) confirmation (i.e., as the result of the PL_PM2:Req_Wait state sending a Stop Arb request), then this state shall send an Unable to Connect (Port Layer Request) message to the PL_OC state machine.

After receiving a Connection Closed (Transition to Idle) confirmation, if this state was entered as the result of the PL_PM2:Req_Wait state receiving an Open Failed (Break Received) confirmation, then this state shall send an Unable to Connect (Break Received) message to the PL_OC state machine.

The Unable To Connect message shall include the following arguments:

   a)   initiator connection tag set to the value received with the Tx Open message;
   b)   destination SAS address set to the value received with the Tx Open message; and
   c)   source SAS address set to the value received with the Tx Open message.

If this state receives a HARD_RESET Received confirmation, then this state shall terminate all operations.

8.2.3.5.2 Transition PL_PM4:Wait_For_Close to PL_PM1:Idle

This transition shall occur after:

   a)   a Retry Open or Unable To Connect message is sent to the PL_OC state machine; or
   b)   all operations are terminated after a HARD_RESET Received confirmation is received.

9 Transport layer

9.1 Transport layer overview

The transport layer defines frame formats. Transport layer state machines interface to the application layer and port layer and construct and parses frame contents. For SSP, the transport layer only receives frames from the port layer for which an ACK is going to be transmitted by the link layer.

9.2 SSP transport layer

9.2.1 SSP frame format

Table 116 defines the SSP frame format.

Table 116 — SSP frame format

Byte\Bit	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					RETRY DATA FRAMES	RETRANSMIT	CHANGING DATA POINTER
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)			DATA OFFSET				
23								(LSB)
24	INFORMATION UNIT (e.g., see table 118, table 120, table 122, table 123, or table 124)							
m								
	Fill bytes, if needed							
n - 3	(MSB)			CRC				
n								(LSB)

Table 117 defines the FRAME TYPE field, which defines the format of the INFORMATION UNIT field.

Table 117 — FRAME TYPE field

Code	Name of frame	Type of information unit	Originator	Information unit size (bytes)	Reference
01h	DATA frame (i.e., write DATA frame or read DATA frame)	Data information unit (i.e., write Data information unit or read Data information unit)	SSP initiator port or SSP target port	1 to 1 024	9.2.2.4
05h	XFER_RDY frame	Transfer Ready information unit	SSP target port	12	9.2.2.3
06h	COMMAND frame	Command information unit	SSP initiator port	28 to 284	9.2.2.1
07h	RESPONSE frame	Response information unit	SSP target port	24 to 1 024	9.2.2.5
16h	TASK frame	Task Management Function information unit	SSP initiator port	28	9.2.2.2
F0h - FFh	Vendor specific				
All others	Reserved				

The HASHED DESTINATION SAS ADDRESS field contains the hashed value of the destination SAS address (see 4.2.3). See 9.2.6.2.2 and 9.2.6.3.2 for transport layer requirements on checking this field.

The HASHED SOURCE SAS ADDRESS field contains the hashed value of the source SAS address (see 4.2.3). See 9.2.6.2.2 and 9.2.6.3.2 for transport layer requirements on checking this field.

The RETRY DATA FRAMES bit is set to one for XFER_RDY frames under the conditions defined in 9.2.4 and shall be set to zero for all other frame types. When set to one this bit specifies that the SSP initiator port may retry write DATA frames that fail.

The RETRANSMIT bit is set to one for TASK frames, RESPONSE frames, and XFER_RDY frames under the conditions defined in 9.2.4 and shall be set to zero for all other frame types. This bit specifies that the frame is a retransmission after the SSP port failed in its previous attempt to transmit the frame.

The CHANGING DATA POINTER bit is set to one for DATA frames under the conditions defined in 9.2.4 and shall be set to zero for all other frame types. When set to one this bit specifies that the frame is a retransmission after the SSP target port failed in its previous attempt to transmit the frame or a subsequent frame and the DATA OFFSET field of the frame may not be sequentially increased from that of the previous frame.

The NUMBER OF FILL BYTES field specifies the number of fill bytes between the INFORMATION UNIT field and the CRC field. The NUMBER OF FILL BYTES field shall be set to zero for all frame types except DATA frames as specified in 9.2.2.4 and RESPONSE frames as specified in 9.2.2.5 (i.e., all other frame types are already four-byte aligned).

The TAG field contains a value that allows the SSP initiator port to establish a context for commands and task management functions.

For COMMAND frames and TASK frames, the SSP initiator port shall set the TAG field to a value that is unique for the I_T nexus established by the connection (see 7.12). An SSP initiator port shall not reuse the same tag when transmitting COMMAND frames or TASK frames to different LUNs in the same SSP target port. An SSP initiator port may reuse a tag when transmitting frames to different SSP target ports. The TAG field in a COMMAND frame contains the task tag defined in SAM-3. The TAG field in a TASK frame does not correspond to a SAM-3 task tag, but corresponds to an SAM-3 association (see 10.2.1). The tag space used in the TAG fields is shared across COMMAND frames and TASK frames (e.g., if a tag is used for a COMMAND frame, it is not also used for a concurrent TASK frame).

For DATA, XFER_RDY, and RESPONSE frames, the SSP target port shall set the TAG field to the tag of the command or task management function to which the frame pertains.

The TARGET PORT TRANSFER TAG field provides an optional method for an SSP target port to establish the write data context when receiving a write DATA frame (i.e., determine the command to which the write data corresponds). Unlike the TAG field, which was assigned by the SSP initiator port, the TARGET PORT TRANSFER TAG field in a write DATA frame contains a value assigned by the SSP target port that was delivered to the SSP initiator port in the XFER_RDY frame requesting the write data.

> NOTE 49 - The TARGET PORT TRANSFER TAG field may be useful when the SSP target port has more than one XFER_RDY frame outstanding (i.e., the SSP target port has transmitted an XFER_RDY frame for each of two or more commands and has not yet received all the write data for them).

SSP target ports may set the TARGET PORT TRANSFER TAG field to any value when transmitting any SSP frame. SSP target ports that use this field should set the TARGET PORT TRANSFER TAG field in every XFER_RDY frame to a value that is unique for the L_Q portion of the I_T_L_Q nexus (i.e., that is unique for every XFER_RDY that is outstanding from the SSP target port).

SSP initiator ports shall set the TARGET PORT TRANSFER TAG field as follows:

   a) For each write DATA frame that is sent in response to an XFER_RDY frame, the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to the value that was in the corresponding XFER_RDY frame;
   b) For each write DATA frame that is sent containing first burst data (see 9.2.2.4), the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh; and
   c) For frames other than write DATA frames, the SSP initiator port shall set the TARGET PORT TRANSFER TAG field to FFFFh.

For DATA frames, the DATA OFFSET field is described in 9.2.2.4. For all other frame types, the DATA OFFSET field shall be ignored.

The INFORMATION UNIT field contains the information unit, the format of which is defined by the FRAME TYPE field (see table 117). The maximum size of the INFORMATION UNIT field is 1 024 bytes, making the maximum size of the frame 1 052 bytes (1 024 bytes of data + 24 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the INFORMATION UNIT field so the CRC field is aligned on a four byte boundary. The number of fill bytes are specified by the NUMBER OF FILL BYTES field. The contents of the fill bytes are vendor specific.

The CRC field contains a CRC value (see 7.5) that is computed over the entire SSP frame prior to the CRC field including the fill bytes (i.e., all data dwords between the SOF and EOF). The CRC field is checked by the link layer (see 7.16), not the transport layer.

9.2.2 Information units

9.2.2.1 COMMAND frame - Command information unit

The COMMAND frame is sent by an SSP initiator port to request that a command be processed by the device server in a logical unit (see 9.2.3.3, 9.2.3.4, 9.2.3.5, and 9.2.3.6).

Table 118 defines the Command information unit used in the COMMAND frame.

Table 118 — COMMAND frame - Command information unit

Byte\Bit	7	6	5	4	3	2	1	0
0								
7				LOGICAL UNIT NUMBER				
8				Reserved				
9	ENABLE FIRST BURST			TASK PRIORITY			TASK ATTRIBUTE	
10				Reserved				
11	ADDITIONAL CDB LENGTH (n dwords)						Reserved	
12								
27				CDB				
28								
27+n×4				ADDITIONAL CDB BYTES				

The LOGICAL UNIT NUMBER field contains the address of the logical unit. The structure of the LOGICAL UNIT NUMBER field shall be as defined in SAM-3. If the addressed logical unit does not exist, the task manager shall follow the rules for selection of incorrect logical units defined in SAM-3.

An ENABLE FIRST BURST bit set to one specifies that the SSP target port shall expect first burst data for the command as defined by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1). An ENABLE FIRST BURST bit set to zero specifies that the SSP target port shall not expect first burst data for the command (i.e., that the FIRST BURST SIZE field in the Disconnect-Reconnect mode page shall be ignored). Application clients shall only set the ENABLE FIRST BURST bit to one if:

   a) the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is non-zero or changeable; and
   b) the logical unit and target port comply with this standard (e.g., as reported in the standard INQUIRY data version descriptors (see SPC-3)).

The TASK PRIORITY field specifies the relative scheduling of the task containing this command in relation to other tasks already in the task set, if the tasks have SIMPLE task attributes (see SAM-3).

The TASK ATTRIBUTE field is defined in table 119.

Table 119 — TASK ATTRIBUTE field

Code	Task attribute	Description
000b	SIMPLE	Specifies that the task be managed according to the rules for a simple task attribute (see SAM-3).
001b	HEAD OF QUEUE	Specifies that the task be managed according to the rules for a head of queue task attribute (see SAM-3).
010b	ORDERED	Specifies that the task be managed according to the rules for an ordered task attribute (see SAM-3).
011b	Reserved	
100b	ACA	Specifies 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 shall be ignored (e.g., a six-byte CDB occupies the first six bytes of the CDB field, the remaining ten bytes of the CDB field are ignored, and the ADDITIONAL CDB BYTES field is not present).

The contents of the CDB are defined in the SCSI command standards (e.g., SPC-3).

9.2.2.2 TASK frame - Task Management Function information unit

The TASK frame is sent by an SSP initiator port to request that a task management function be processed by the task manager in a logical unit (see 9.2.3.2).

Table 120 defines the Task Management Function information unit used in the TASK frame.

Table 120 — TASK frame - Task Management Function information unit

Byte\Bit	7	6	5	4	3	2	1	0
0								
7				LOGICAL UNIT NUMBER				
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 contains 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 return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and its RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER.

Table 121 defines the TASK MANAGEMENT FUNCTION field.

Table 121 — TASK MANAGEMENT FUNCTION **field**

Code	Task management function	Uses LOGICAL UNIT NUMBER field	Uses TAG OF TASK TO BE MANAGED field	Description
01h	ABORT TASK	yes	yes	The task manager shall perform the ABORT TASK task management function with L set to the value of the LOGICAL UNIT NUMBER field and Q set to the value of the TAG OF TASK TO BE MANAGED field (see SAM-3).
02h	ABORT TASK SET	yes	no	The task manager shall perform the ABORT TASK SET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-3).
04h	CLEAR TASK SET	yes	no	The task manager shall perform the CLEAR TASK SET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-3).
08h	LOGICAL UNIT RESET	yes	no	The task manager shall perform the LOGICAL UNIT RESET task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-3).
20h	Reserved			
40h	CLEAR ACA	yes	no	The task manager shall perform the CLEAR ACA task management function with L set to the value of the LOGICAL UNIT NUMBER field (see SAM-3).
80h	QUERY TASK	yes	yes	The task manager shall perform the QUERY TASK task management function with L set to the value of the LOGICAL UNIT NUMBER field and Q set to the value of the TAG OF TASK TO BE MANAGED field (see SAM-3).
All others	Reserved			

If the TASK MANAGEMENT FUNCTION field contains a reserved or unsupported value, the task manager shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and its RESPONSE CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED.

If the TASK MANAGEMENT FUNCTION field is set to ABORT TASK or QUERY TASK, the TAG OF TASK TO BE MANAGED field specifies the TAG value from the COMMAND frame that contained the task to be aborted or checked. For all other task management functions, the TAG OF TASK TO BE MANAGED field shall be ignored.

9.2.2.3 XFER_RDY frame - Transfer Ready information unit

The XFER_RDY frame is sent by an SSP target port to request write data from the SSP initiator port during a write command or a bidirectional command (see 9.2.3.4 and 9.2.3.6).

Table 122 defines the Transfer Ready information unit used in the XFER_RDY frame.

Table 122 — XFER_RDY frame - Transfer Ready information unit

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
				REQUESTED OFFSET				
3								(LSB)
4	(MSB)							
				WRITE DATA LENGTH				
7								(LSB)
8								
				Reserved				
11								

The REQUESTED OFFSET field contains the application client buffer offset of the segment of write data the SSP initiator port may transmit to the logical unit using write DATA frames. The requested offset shall be a multiple of four (i.e., each write DATA frame shall begin transferring data on a dword boundary).

The REQUESTED OFFSET field shall be zero for the first XFER_RDY frame of a command unless:

   a) the ENABLE FIRST BURST field in the COMMAND frame (see 9.2.2.1) was set to one; and
   b) the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1.5) is not set to zero.

If the ENABLE FIRST BURST field in the COMMAND frame (see 9.2.2.1) was set to one, then in the initial XFER_RDY frame for the command, the SSP target port shall set the REQUESTED OFFSET field to the application client buffer offset of the segment of write data following the first burst data defined by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1.5).

If any additional XFER_RDY frames are required for the command and transport-layer retries are not being used, the REQUESTED OFFSET field shall be set to the sum of the requested offset and write data length of the previous XFER_RDY frame.

The WRITE DATA LENGTH field contains the number of bytes of write data the SSP initiator port may transmit to the logical unit using write DATA frames from the application client buffer starting at the requested offset. The SSP target port shall set the WRITE DATA LENGTH field to a value greater than or equal to 00000001h. If the value in the MAXIMUM BURST SIZE field in the Disconnect-Reconnect mode page is not zero, the SSP target port shall set the WRITE DATA LENGTH field to a value less than or equal to the value in the MAXIMUM BURST SIZE field (see 10.2.7.1.4).

If an SSP target port transmits an XFER_RDY frame containing a WRITE DATA LENGTH field that is not divisible by four, the SSP target port shall not transmit any subsequent XFER_RDY frames for that command (i.e., only the last XFER_RDY for a command may request a non-dword multiple write data length).

9.2.2.4 DATA frame - Data information unit

During a write command or a bidirectional command (see 9.2.3.4 and 9.2.3.6), one or more write DATA frames are sent by an SSP initiator port to deliver write data.

During a read command or a bidirectional command (see 9.2.3.5 and 9.2.3.6), one or more read DATA frames are sent by an SSP target port to deliver read data.

Table 123 defines the Data information unit used in the DATA frame.

Table 123 — DATA frame - Data information unit

Byte\Bit	7	6	5	4	3	2	1	0
0								
				DATA				
n								

The DATA field contains the read data (i.e., data to the application client's data-in buffer) or write data (i.e., data from the application client's data-out buffer). The size of the DATA field (i.e., the data length) is determined by the NUMBER OF FILL BYTES field in the frame header (see 9.2.1) and the link layer detection of EOF (see 7.16.3).

The maximum size of the Data information unit (i.e., the DATA field) is the maximum size of any information unit in an SSP frame (see 9.2.1). The minimum size of the Data information unit is one byte.

An SSP initiator port shall only transmit a write DATA frame:

a) in response to an XFER_RDY frame; or

b) after transmitting a COMMAND frame if the ENABLE FIRST BURST field in the COMMAND frame was set to one (see 9.2.2.1) and the FIRST BURST SIZE field in the Disconnect-Reconnect mode page is not zero (see 10.2.7.1.5).

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 an SSP target port per I_T_L_Q nexus is limited by the value in the MAXIMUM BURST SIZE field (see 10.2.7.1.4).

A write DATA frame shall only contain write data for a single XFER_RDY frame.

An SSP initiator port shall set the NUMBER OF FILL BYTES field to zero in the frame header (see 9.2.1) in all write DATA frames that it transmits in response to an XFER_RDY frame except the last write DATA frame for that XFER_RDY frame. An SSP initiator port may set the NUMBER OF FILL BYTES field to a non-zero value in the last DATA frame that it transmits in response to an XFER_RDY.

> NOTE 50 - Combined with the restrictions on WRITE DATA LENGTH in the XFER_RDY frame (see 9.2.2.3), this ensures that only the last write DATA frame for a command may have data with a length that is not a multiple of four).

An SSP target port shall set the NUMBER OF FILL BYTES field to zero in the frame header (see 9.2.1) in all read DATA frames for a command except the last read DATA frame for that command. The SSP target port may set the NUMBER OF FILL BYTES field to a non-zero value in the last read DATA frame for a command (i.e., only the last read DATA frame for a command may contain data with a length that is not a multiple of four).

An SSP initiator port shall not transmit a write DATA frame for a given I_T_L_Q nexus after it has sent a TASK frame that terminates that task (e.g., an ABORT TASK).

The DATA OFFSET field in the frame header (see 9.2.1) contains the application client buffer offset as described by SAM-3. The data offset shall be a multiple of four (i.e., each DATA frame shall transfer data beginning on a dword boundary).

The DATA OFFSET field shall be set to zero in the initial read DATA frame for a command. If any additional read DATA frames are required for the command and transport-layer retries are not being used, the DATA OFFSET field shall be set to the sum of the data offset and data length of the previous read DATA frame.

The DATA OFFSET field shall be set to zero in the initial write DATA frame for a command. If any additional write DATA frames are required for the command and transport-layer retries are not being used, the DATA OFFSET field shall be set to the sum of the data offset and data length of the previous write DATA frame.

9.2.2.5 RESPONSE frame - Response information unit

9.2.2.5.1 RESPONSE frame - Response information unit overview

The RESPONSE frame is sent by an SSP target port to deliver:

a) service response, SCSI status (e.g., GOOD or CHECK CONDITION) and sense data, if any, for a command (see 9.2.3.3, 9.2.3.4, 9.2.3.5, and 9.2.3.6);

b) service response for a task management function (see 9.2.3.2); or

c) SSP-specific response (e.g., illegal frame format).

Table 124 defines the Response information unit used in the RESPONSE frame.

Table 124 — RESPONSE frame - Response information unit

Byte\Bit	7	6	5	4	3	2	1	0
0								
9				Reserved				
10			Reserved					DATAPRES
11				STATUS				
12								
15				Reserved				
16	(MSB)							
19			SENSE DATA LENGTH (n bytes)					(LSB)
20	(MSB)							
23			RESPONSE DATA LENGTH (m bytes)					(LSB)
24								
23+m			RESPONSE DATA (see table 126 in 9.2.2.5.3)(if any)					
24+m								
23+m+n			SENSE DATA (if any)					

Table 125 defines the DATAPRES field, which specifies the format and content of the STATUS field, SENSE DATA LENGTH field, RESPONSE DATA LENGTH field, RESPONSE DATA field, and SENSE DATA field.

Table 125 — DATAPRES field

Code	Name	Description	Reference
00b	NO_DATA	No response data or sense data present	9.2.2.5.2
01b	RESPONSE_DATA	Response data present	9.2.2.5.3
10b	SENSE_DATA	Sense data present	9.2.2.5.4
11b	Reserved		

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to NO_DATA if a command completes without response data or sense data to return.

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA in response to every TASK frame and in response to errors that occur while the transport layer is processing a COMMAND frame (see 9.2.5.3).

The SSP target port shall return a RESPONSE frame with the DATAPRES field set to SENSE_DATA if a command completes with sense data to return (e.g., CHECK CONDITION status).

If the DATAPRES field is set to a reserved value, then the SSP initiator port shall discard the RESPONSE frame.

9.2.2.5.2 Response information unit - NO_DATA format

If the DATAPRES field is set to NO_DATA, then:

a) the SSP target port shall set the STATUS field to the status code for a command that has ended (see SAM-3 for a list of status codes);

b) the SSP target port shall set the SENSE DATA LENGTH field to zero and the RESPONSE DATA LENGTH field to zero;

c) the SSP initiator port shall ignore the SENSE DATA LENGTH field and the RESPONSE DATA LENGTH field; and

d) the SSP target port shall not include the SENSE DATA field and the RESPONSE DATA field.

9.2.2.5.3 Response information unit - RESPONSE_DATA format

If the DATAPRES field is set to RESPONSE_DATA, then:

a) the SSP target port shall set the STATUS field to zero and the SENSE DATA LENGTH field to zero;

b) the SSP initiator port shall ignore the STATUS field and the SENSE DATA LENGTH field;

c) the SSP target port shall not include the SENSE DATA field;

d) the SSP target port shall set the RESPONSE DATA LENGTH field to 00000004h; and

e) the SSP target port shall include the RESPONSE DATA field.

Table 126 defines the RESPONSE DATA field. The RESPONSE DATA field shall be present if the SSP target port detects any of the conditions described by a non-zero value in the RESPONSE CODE field and shall be present for a RESPONSE frame sent in response to a TASK frame.

Table 126 — RESPONSE DATA **field**

Byte\Bit	7	6	5	4	3	2	1	0
0				Reserved				
2								
3				RESPONSE CODE				

Table 127 defines the RESPONSE CODE field, which specifies the error condition or the completion status of a task management function. See 10.2.1.5 and 10.2.1.15 for the mapping of these response codes to SCSI service responses.

Table 127 — RESPONSE CODE **field**

Code	Description
00h	TASK MANAGEMENT FUNCTION COMPLETE [a]
02h	INVALID FRAME
04h	TASK MANAGEMENT FUNCTION NOT SUPPORTED [a]
05h	TASK MANAGEMENT FUNCTION FAILED [a]
08h	TASK MANAGEMENT FUNCTION SUCCEEDED [a]
09h	INCORRECT LOGICAL UNIT NUMBER [a]
0Ah	OVERLAPPED TAG ATTEMPTED [b]
All others	Reserved
[a] Only valid when responding to a TASK frame [b] Returned in case of command/task management function or task management function/task management function tag conflicts.	

9.2.2.5.4 Response information unit - SENSE_DATA format

If the DATAPRES field is set to SENSE_DATA, then:

a) the SSP target port shall set the STATUS field to the status code for a command that has ended (see SAM-3 for a list of status codes);

b) the SSP target port shall set the RESPONSE DATA LENGTH field to zero;

c) the SSP initiator port shall ignore the RESPONSE DATA LENGTH field;

d) the SSP target port shall not include the RESPONSE DATA field;

e) the SSP target port shall set the SENSE DATA LENGTH field to a non-zero value indicating the number of bytes in the SENSE DATA field. The value in the SENSE DATA LENGTH field shall not be greater than 1 000 (see table 117 in 9.2.1); and

f) the SSP target port shall set the SENSE DATA field to the sense data (see SAM-3).

The value in the SENSE DATA LENGTH field is not required to be a multiple of four. If it is not, the value in the NUMBER OF FILL BYTES field in the SSP frame header is non-zero and fill bytes are present.

9.2.3 Sequences of SSP frames

9.2.3.1 Sequences of SSP frames overview

Table 128 lists the sequences of SSP frames supporting the SCSI transport protocol services described in 10.2.1.

Table 128 — Sequences of SSP frames

Sequence	Reference
Task management function	9.2.3.2
Non-data command	9.2.3.3
Write command	9.2.3.4
Read command	9.2.3.5
Bidirectional command	9.2.3.6

When multiple commands and/or task management functions are outstanding, frames from each of the individual sequences may be interleaved in any order. RESPONSE frames may be returned in any order (i.e., the order in which TASK frames and COMMAND frames are sent has no effect on the order that RESPONSE frames are returned).

Frames in a sequence may be transmitted during one or more connections (see 7.12)(e.g., for a write command using a single XFER_RDY frame, the COMMAND frame could be transmitted in a connection originated by the SSP initiator port, the XFER_RDY frame in a connection originated by the SSP target port, the DATA frames in one or more connections originated by the SSP initiator port, and the RESPONSE frame in a connection originated by the SSP target port. Or, they could all be transmitted in one connection.).

9.2.3.2 Task management function sequence of SSP frames

Figure 170 shows the sequence of SSP frames for a task management function (e.g., ABORT TASK (see SAM-3)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.
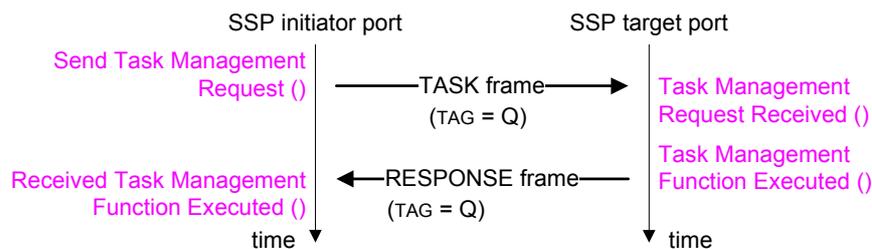
Figure 170 — Task management function sequence of SSP frames

9.2.3.3 Non-data command sequence of SSP frames

Figure 171 shows the sequence of SSP frames for a non-data command (e.g., TEST UNIT READY (see SPC-3)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.
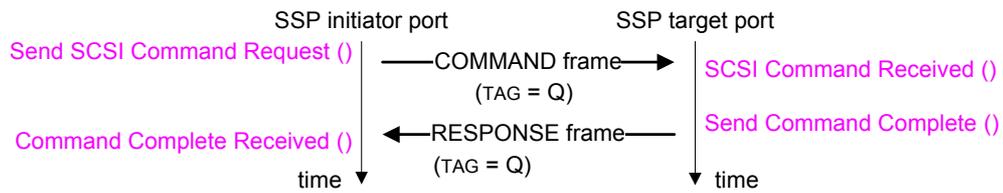
Figure 171 — Non-data command sequence of SSP frames

9.2.3.4 Write command sequence of SSP frames

Figure 172 shows the sequence of SSP frames for a write command (e.g., MODE SELECT (see SPC-3)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.
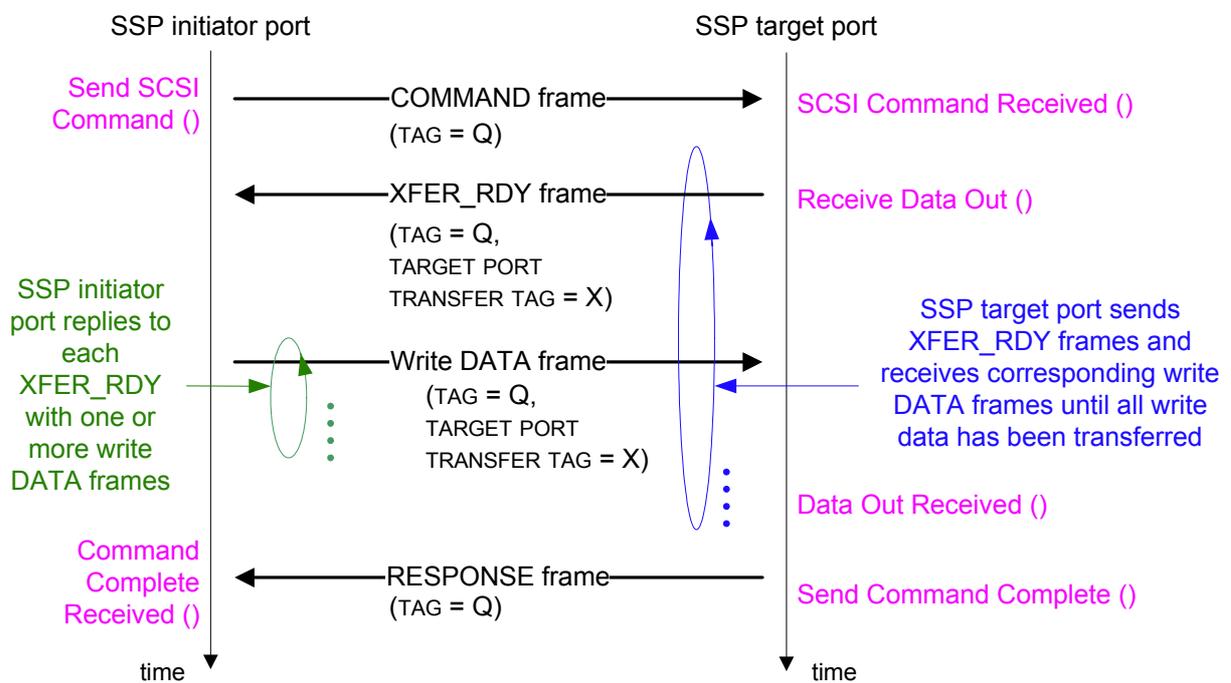
Figure 172 — Write command sequence of SSP frames

9.2.3.5 Read command sequence of SSP frames

Figure 173 shows the sequence of SSP frames for a read command (e.g., INQUIRY, REPORT LUNS, or MODE SENSE (see SPC-3)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.
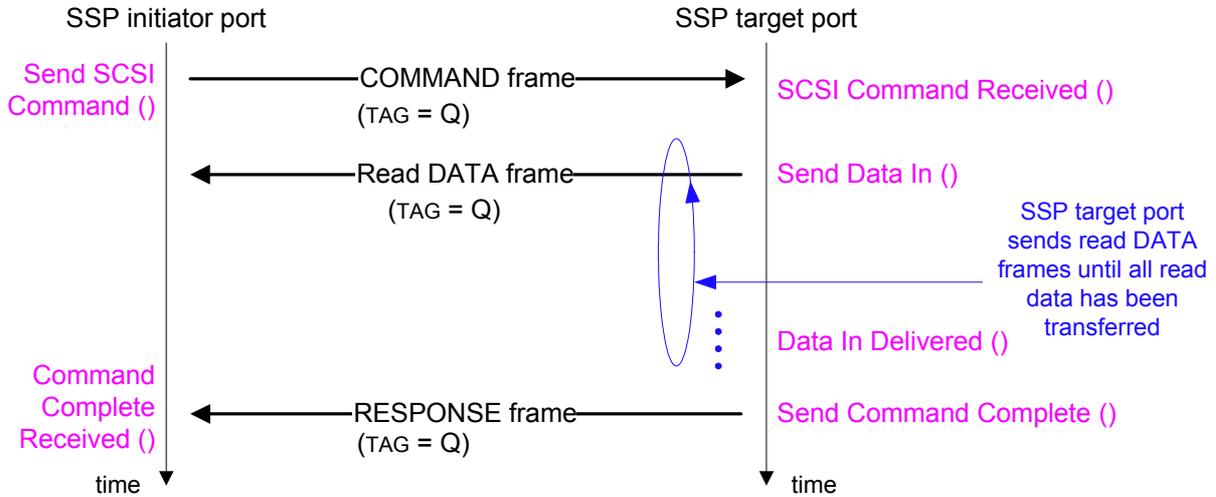
Figure 173 — Read command sequence of SSP frames

9.2.3.6 Bidirectional command sequence of SSP frames

Figure 174 shows the sequence of SSP frames for a bidirectional command (e.g., XDWRITEREAD (see SBC-2)), including the transport protocol services (see 10.2.1) invoked by the SCSI application layer.
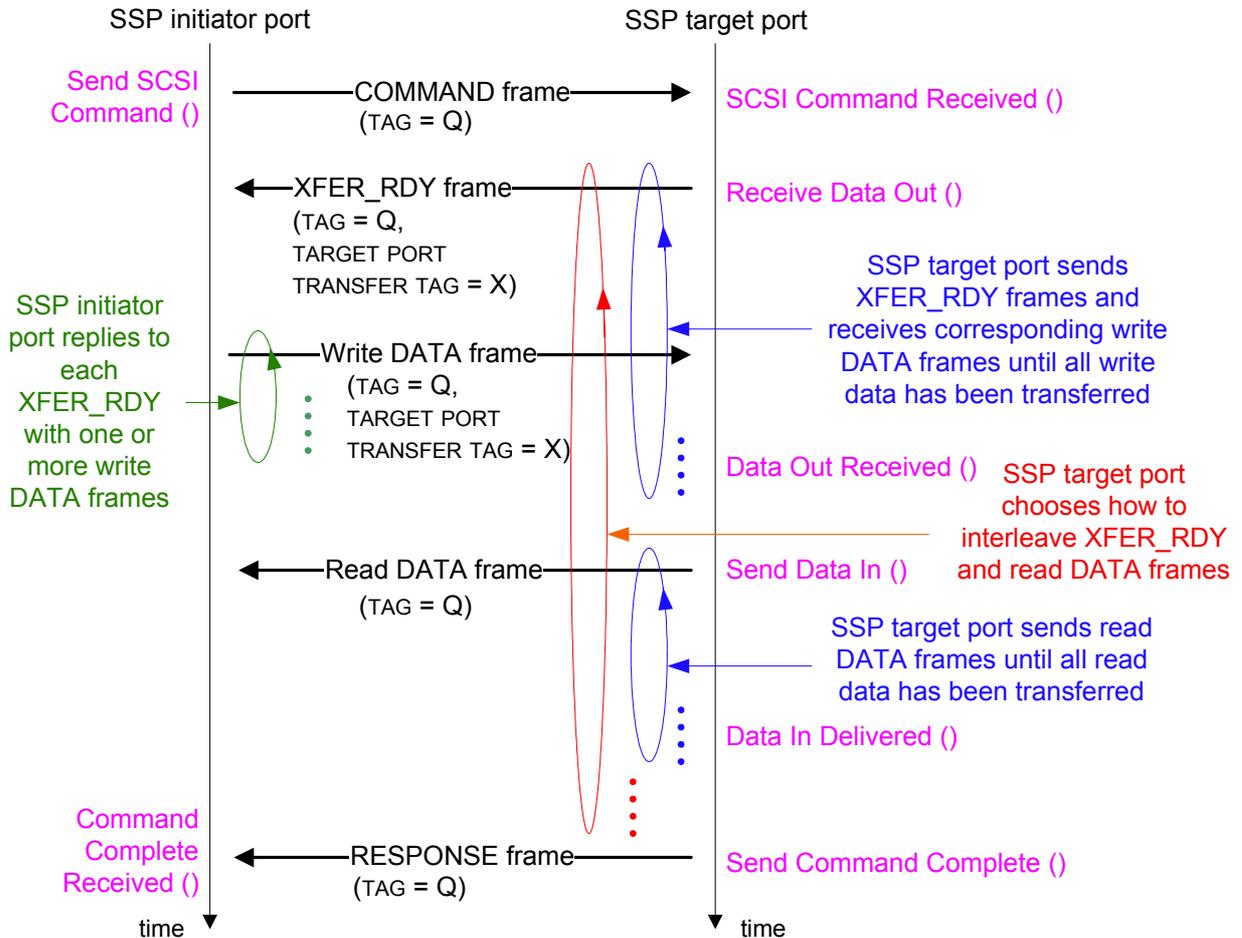
Figure 174 — Bidirectional command sequence of SSP frames

The SSP target port may transmit read DATA frames for a bidirectional command at the same time it is receiving write DATA frames for the same bidirectional command.

9.2.4 SSP transport layer handling of link layer errors

9.2.4.1 SSP transport layer handling of link layer errors overview

The transport layer, sometimes assisted by the application layer, handles some link layer errors (e.g., NAKs and ACK/NAK timeouts). See 9.2.5 for transport layer handling of transport layer errors (e.g., invalid frame contents).

Link layer errors that occur when transmitting XFER_RDY and DATA frames are handled differently based on the TRANSPORT LAYER RETRIES bit in the Protocol-Specific Logical Unit mode page (see 10.2.7.3) of the logical unit that is the source of the frame.

If the TRANSPORT LAYER RETRIES bit is set to zero, a logical unit:

a) disables transport layer retries;
b) sets the RETRY DATA FRAMES bit to zero in each XFER_RDY frame;
c) may or may not select a different value for the TARGET PORT TRANSFER TAG field in each XFER_RDY frame than that used in the previous XFER_RDY frame for that I_T_L_Q nexus;
d) processes XFER_RDY frame link layer errors as described in 9.2.4.4.3; and
e) processes DATA frame link layer errors as described in 9.2.4.5.3.

If the TRANSPORT LAYER RETRIES bit is set to one, the logical unit:

a) enables transport layer retries;
b) supports the QUERY TASK task management function (see SAM-3);
c) sets the RETRY DATA FRAMES bit to one in each XFER_RDY frame;
d) selects a different value for the TARGET PORT TRANSFER TAG field in each XFER_RDY frame than that used in the previous XFER_RDY frame for that I_T_L_Q nexus;
e) processes XFER_RDY frame link layer errors as described in 9.2.4.4.2; and
f) processes DATA frame link layer errors as described in 9.2.4.5.2.

9.2.4.2 COMMAND frame - handling of link layer errors

If an SSP initiator port transmits a COMMAND frame and receives a NAK for that frame, then the COMMAND frame was not received. The SSP initiator port should retransmit, in the same or in a new connection, the COMMAND frame at least one time (see 9.2.6.2.3.3). The SSP initiator port may reuse the tag.

If an SSP initiator port transmits a COMMAND frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
2) to determine whether the command was received, the application client calls Send Task Management Function Request () (see 10.2.2) with:
   A) Nexus set to the I_T_L_Q nexus of the COMMAND frame; and
   B) Function Identifier set to QUERY TASK;

   and

3) the SSP initiator port transmits the TASK frame in a new connection to the SSP target port.

If the SSP initiator port receives an XFER_RDY frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received and is being processed by the target port, and the XFER_RDY frame is valid.

If the SSP initiator port receives a read DATA frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received and is being processed by the target port, and the read DATA frame is valid.

If the SSP initiator port receives a RESPONSE frame for the I_T_L_Q nexus of the command before the RESPONSE frame for the QUERY TASK, then the COMMAND frame was received by the target port, the RESPONSE frame is valid, and the command processing is complete. The SSP initiator port may reuse the tag of the COMMAND frame.

If the SSP initiator port receives a RESPONSE frame for the QUERY TASK with a response code of TASK MANAGEMENT FUNCTION SUCCEEDED, then the COMMAND frame was received by the SSP target port (i.e., ACKed) and the command is being processed.

If the SSP initiator port receives a RESPONSE frame for the QUERY TASK with a response code of TASK MANAGEMENT FUNCTION COMPLETE, then the COMMAND frame is not being processed. If neither an XFER_RDY frame, a read DATA frame, nor a RESPONSE frame has been received for the I_T_L_Q nexus of the command, then the COMMAND frame was not received. The SSP initiator port should retransmit the COMMAND frame at least one time. The SSP initiator port may reuse the tag of the COMMAND frame.

9.2.4.3 TASK frame - handling of link layer errors

If an SSP initiator port transmits a TASK frame and receives a NAK for that frame, then the TASK frame was not received. The SSP initiator port should retransmit, in the same or in a new connection, the TASK frame at least one time with the RETRANSMIT bit set to one (see 9.2.6.2.3.3). The SSP initiator port may reuse the tag.

If an SSP initiator port transmits a TASK frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the application client calls Send Task Management Request () using the same tag (see 10.2.2);
3) the SSP initiator port transmits the TASK frame with the RETRANSMIT bit set to one in a new connection to the SSP target port (see 9.2.6.2.2.2).

If the SSP initiator port receives a RESPONSE frame for the TASK frame that arrives before the ACK or NAK for the TASK frame, then the TASK frame was received by the SSP target port (i.e., ACKed), the RESPONSE frame is valid, and the task management function is complete (see 9.2.6.2.2.3). The initiator port may reuse the tag of the TASK frame.

9.2.4.4 XFER_RDY frame - handling of link layer errors

9.2.4.4.1 XFER_RDY frame overview

If the TRANSPORT LAYER RETRIES bit is set to one in the Protocol-Specific Logical Unit mode page (see 10.2.7.3), then the SSP target port processes link layer errors that occur while transmitting XFER_RDY frames as described in 9.2.4.4.2.

If the TRANSPORT LAYER RETRIES bit is set to zero, then the SSP target port processes link layer errors that occur while transmitting XFER_RDY frames as described in 9.2.4.4.3.

9.2.4.4.2 XFER_RDY frame with transport layer retries enabled

If an SSP target port transmits an XFER_RDY frame and receives a NAK for that frame, the SSP target port retransmits, in the same or a new connection, the XFER_RDY frame with a different value in the TARGET PORT TRANSFER TAG field and with the RETRANSMIT bit set to one (see 9.2.6.3.3.5).

If an SSP target port transmits an XFER_RDY frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the SSP target port retransmits, in a new connection, the XFER_RDY frame with a different value in the TARGET PORT TRANSFER TAG field and with the RETRANSMIT bit set to one (see 9.2.6.3.3.5).

If an SSP initiator port receives a new XFER_RDY frame with the RETRANSMIT bit set to one while processing the previous XFER_RDY frame for that I_T_L_Q nexus, the ST_ITS state machine stops processing the previous XFER_RDY frame (i.e., stops transmitting write DATA frames) and starts servicing the new XFER_RDY frame (see 9.2.6.2.3). The ST_ITS state machine does not transmit any write DATA frames for the previous XFER_RDY frame after transmitting a write DATA frame for the new XFER_RDY frame.

The SSP target port may reuse the value in the TARGET PORT TRANSFER TAG field from the previous XFER_RDY frame after it receives a write DATA frame for the new XFER_RDY frame.

An SSP target port retransmits each XFER_RDY frame that does not receive an ACK at least one time.

9.2.4.4.3 XFER_RDY frame with transport layer retries disabled

If an SSP target port transmits an XFER_RDY frame and receives a NAK for that frame:

1) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to NAK RECEIVED (see 10.2.3); and
2) the SSP target port transmits the RESPONSE frame in the same or a new connection.

If an SSP target port transmits an XFER_RDY frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
2) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to ACK/NAK TIMEOUT (see 10.2.3); and
3) the SSP target port transmits the RESPONSE frame in a new connection.

9.2.4.5 DATA frame - handling of link layer errors

9.2.4.5.1 DATA frame overview

If an SSP target port transmits a read DATA frame for a logical unit that has its TRANSPORT LAYER RETRIES bit set to one in the Protocol-Specific Logical Unit mode page (see 10.2.7.3), then the SSP target port processes link layer errors that occur while transmitting read DATA frames as described in 9.2.4.5.2. If the logical unit has its TRANSPORT LAYER RETRIES bit set to zero, then the SSP target port processes link layer errors that occur while transmitting read DATA frames as described in 9.2.4.5.3.

An SSP initiator port processes link layer errors that occur while transmitting write DATA frames transmitted in response to an XFER_RDY frame that has its RETRY DATA FRAMES bit set to one as described in 9.2.4.5.2. An SSP initiator port processes link layer errors that occur while transmitting write DATA frames in response to an XFER_RDY frame that has its RETRY DATA FRAMES bit set to zero as described in 9.2.4.5.3.

9.2.4.5.2 DATA frame with transport layer retries enabled

If an SSP target port transmits a read DATA frame and receives a NAK for that frame, then the read DATA frame was not received. The SSP target port retransmits, in the same or in a new connection, all the read DATA frames since a previous time when ACK/NAK balance occurred (see 9.2.6.3.3.4).

If an SSP target port transmits a read DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the ST_TTS state machine retransmits, in a new connection, all the read DATA frames since a previous time when ACK/NAK balance occurred (see 9.2.6.3.3.4).

If an SSP initiator port transmits a write DATA frame and receives a NAK for that frame, then the write DATA frame was not received. The SSP_ITS state machine retransmits, in the same or in a new connection, all the write DATA frames for the previous XFER_RDY frame (see 9.2.6.2.3.3.4).

If an SSP initiator port transmits a write DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the ST_ITS state machine retransmits, in a new connection, all the write DATA frames for the previous XFER_RDY frame (see 9.2.6.2.3.3.4).

If that SSP initiator port receives a new XFER_RDY frame or a RESPONSE frame for the command while retransmitting or preparing to retransmit the write DATA frames, the ST_IFR state machine and ST_ITS state machine process the XFER_RDY frame or RESPONSE frame and stop retransmitting the write DATA frames (see 9.2.6.2.2 and 9.2.6.2.3). The ST_ITS state machine does not transmit a write DATA frame for the previous XFER_RDY frame after transmitting a write DATA frame in response to the new XFER_RDY frame.

For both reads and writes, the CHANGING DATA POINTER bit is set to one in the first retransmitted DATA frame and CHANGING DATA POINTER bit is set to zero in subsequent DATA frames.

The ST_ITS state machine and ST_TTS state machine retransmit each DATA frame that does not receive an ACK at least one time (see 9.2.6.2.3 and 9.2.6.3.3). The number of times they retransmit each DATA frame is vendor-specific.

9.2.4.5.3 DATA frame with transport layer retries disabled

If an SSP target port transmits a read DATA frame and receives a NAK for that frame:

1) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to NAK RECEIVED (see 10.2.3); and
2) the SSP target port transmits the RESPONSE frame in the same or a new connection.

If an SSP target port transmits a read DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5);
2) the device server calls Send Command Complete () to return CHECK CONDITION status for that command with the sense key set to ABORTED COMMAND and the additional sense code set to ACK/NAK TIMEOUT (see 10.2.3); and
3) the SSP target port transmits the RESPONSE frame in a new connection.

If an SSP initiator port transmits a write DATA frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the application client aborts the command (see 10.2.2).

If an SSP initiator port transmits a write DATA frame and receives a NAK for that frame, the application client aborts the command (see 10.2.2).

9.2.4.6 RESPONSE frame - handling of link layer errors

If an SSP target port transmits a RESPONSE frame and receives a NAK for that frame, the SSP target port retransmits, in the same or a new connection, the RESPONSE frame at least one time with the RETRANSMIT bit set to one (see 9.2.6.3.3.7.1).

If an SSP target port transmits a RESPONSE frame and does not receive an ACK or NAK for that frame (e.g., times out, or the connection is broken):

1) the SSP_TF state machine closes the connection with DONE (ACK/NAK TIMEOUT) (see 7.16.8.6.5); and
2) the SSP target port retransmits, in a new connection, the RESPONSE frame with the RETRANSMIT bit set to one (see 9.2.6.3.3.7.1).

The ST_TTS state machine retransmits each RESPONSE frame that does not receive an ACK at least one time (see 9.2.6.3.3). The number of times it retransmits each RESPONSE frame is vendor-specific.

If an SSP initiator port receives a RESPONSE frame with a RETRANSMIT bit set to one, and it has previously received a RESPONSE frame for the same I_T_L_Q nexus, the ST_IFR state machine discards the extra RESPONSE frame (see 9.2.6.3.2). If the ST_IFR state machine has not previously received a RESPONSE frame for the I_T_L_Q nexus, then it considers the RESPONSE frame to be the valid RESPONSE frame.

9.2.5 SSP transport layer error handling summary

9.2.5.1 SSP transport layer error handling summary introduction

This subclause contains a summary of how SSP ports process transport layer errors. This summary does not include every error case. See 9.2.4 for transport layer handling of link layer errors (e.g., using transport layer retries).

9.2.5.2 SSP initiator port transport layer error handling summary

If an SSP initiator port receives a COMMAND or TASK frame or an unsupported frame type, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3).

If an SSP initiator port receives an XFER_RDY, read DATA, or RESPONSE frame with an unknown TAG field value, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3). The application client may then abort the command with that tag.

If an SSP initiator port receives an XFER_RDY frame that is not 12 bytes long, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3). The application client may then abort the command.

If an SSP initiator port receives an XFER_RDY frame in response to a command with no write data, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives an XFER_RDY frame requesting more write data than expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives an XFER_RDY frame requesting zero bytes, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If transport layer retries are disabled and an SSP initiator port receives an XFER_RDY frame with a requested offset that was not expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives a read DATA frame in response to a command with no read data, the ST_IFR state machine discards the frame (see 9.2.6.2.2.3) and the application client aborts the command (see 10.2.2).

If an SSP initiator port receives a read DATA frame with more read data than expected, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If an SSP initiator port receives a read DATA frame with zero bytes, the ST_ITS state machine discards the frame (see 9.2.6.2.3.3) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If transport layer retries are disabled and an SSP initiator port receives a read DATA frame with a data offset that was not expected, the ST_ITS state machine discards that frame and any subsequent read DATA frames received for that command (see 9.2.6.2.3.7) and the application client aborts the command (see 10.2.2). The SSP initiator port may receive a RESPONSE for the command before being able to abort the command.

If an SSP initiator port receives a RESPONSE frame that is not the correct length, the ST_IFR state machine considers the command or task management function completed with an error and discards the frame (see 9.2.6.2.2.3).

9.2.5.3 SSP target port transport layer error handling summary

If an SSP target port receives an XFER_RDY or RESPONSE frame or another unsupported frame type, the ST_TFR state machine discards the frame (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame and:

  a) the frame is too short to contain a LUN field;
  b) the frame is too short to contain a CDB; or
  c) the ADDITIONAL CDB LENGTH field specifies that the frame should be a different length,

then the ST_TTS state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port receives a TASK frame that is too short, the ST_TTS state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame with a tag that is already in use for another command, the device server may return CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to OVERLAPPED COMMANDS ATTEMPTED (see 10.2.3).

If an SSP target port receives:

  a) a COMMAND frame with a tag that is already in use for a task management function; or
  b) a TASK frame with a tag that is already in use for a command or another task management function,

then the task router and task manager(s) return a RESPONSE frame with the RESPONSE CODE field set to OVERLAPPED TAG ATTEMPTED (see 10.2.4).

If an SSP target port receives a write DATA frame with an unknown tag, the ST_TFR state machine discards the frame (see 9.2.6.3.2).

If an SSP target port receives a write DATA frame that does not contain first burst data and for which there is no XFER_RDY frame outstanding (i.e., it has received all requested write data), the ST_TFR state machine discards the frame (see 9.2.6.3.2.2).

If an SSP target port receives a TASK frame with an unknown logical unit number, the ST_TFR state machine returns a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER (see 9.2.6.3.2.2).

If an SSP target port receives a COMMAND frame or TASK frame with a target port transfer tag set to a value other than FFFFh, the ST_TFR state machine may return a RESPONSE frame with the DATAPRES field set to RESPONSE_DATA and the RESPONSE CODE field set to INVALID FRAME (see 9.2.6.3.2.2).

If an SSP target port is using target port transfer tags and receives a write DATA frame with an unknown target port transfer tag, the ST_TFR state machine discards the frame (see 9.2.6.3.3).

If transport layer retries are disabled and an SSP target port receives a write DATA frame with a data offset that was not expected, the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to DATA OFFSET ERROR (see 10.2.3).

If an SSP target port receives a write DATA frame with more write data than expected (i.e., the write DATA frame contains data in excess of that requested by an XFER_RDY frame or, for first burst data, indicated by the FIRST BURST LENGTH field in the Disconnect-Reconnect mode page), the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to TOO MUCH WRITE DATA (see 10.2.3).

If an SSP target port receives a write DATA frame with zero bytes, the ST_TTS state machine discards the frame (see 9.2.6.3.3.6.1) and the device server terminates the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to INFORMATION UNIT TOO SHORT (see 10.2.3).

9.2.6 ST (transport layer for SSP ports) state machines

9.2.6.1 ST state machines overview

The ST state machines perform the following functions:

    a) receive and process transport protocol service requests and transport protocol service responses from the SCSI application layer;
    b) receive and process other SAS connection management requests from the application layer;
    c) send transport protocol service indications and transport protocol service confirmations to the SCSI application layer;
    d) send requests to the port layer to transmit frames and manage SAS connections; and
    e) receive confirmations from the port layer.

The following confirmations between the ST state machines and the port layer:

    a) Transmission Status; and
    b) Frame Received;

include the following as arguments:

    a) tag;
    b) destination SAS address; and
    c) source SAS address;

and are used to route the confirmations to the correct ST state machines.

> NOTE 51 - Although allowed by this standard, the ST state machines do not handle bidirectional commands that result in concurrent write DATA frames and read DATA frames.

9.2.6.2 ST_I (transport layer for SSP initiator ports) state machines

9.2.6.2.1 ST_I state machines overview

The ST_I state machines are as follows:

    a)   ST_IFR (initiator frame router) state machine (see 9.2.6.2.2); and
    b)   ST_ITS (initiator transport server) state machine (see 9.2.6.2.3).

The SAS initiator port includes:

    a)   one ST_IFR state machine; and
    b)   one ST_ITS state machine for each possible task and task management function (i.e., for each tag).

Figure 175 shows the ST_I state machines.
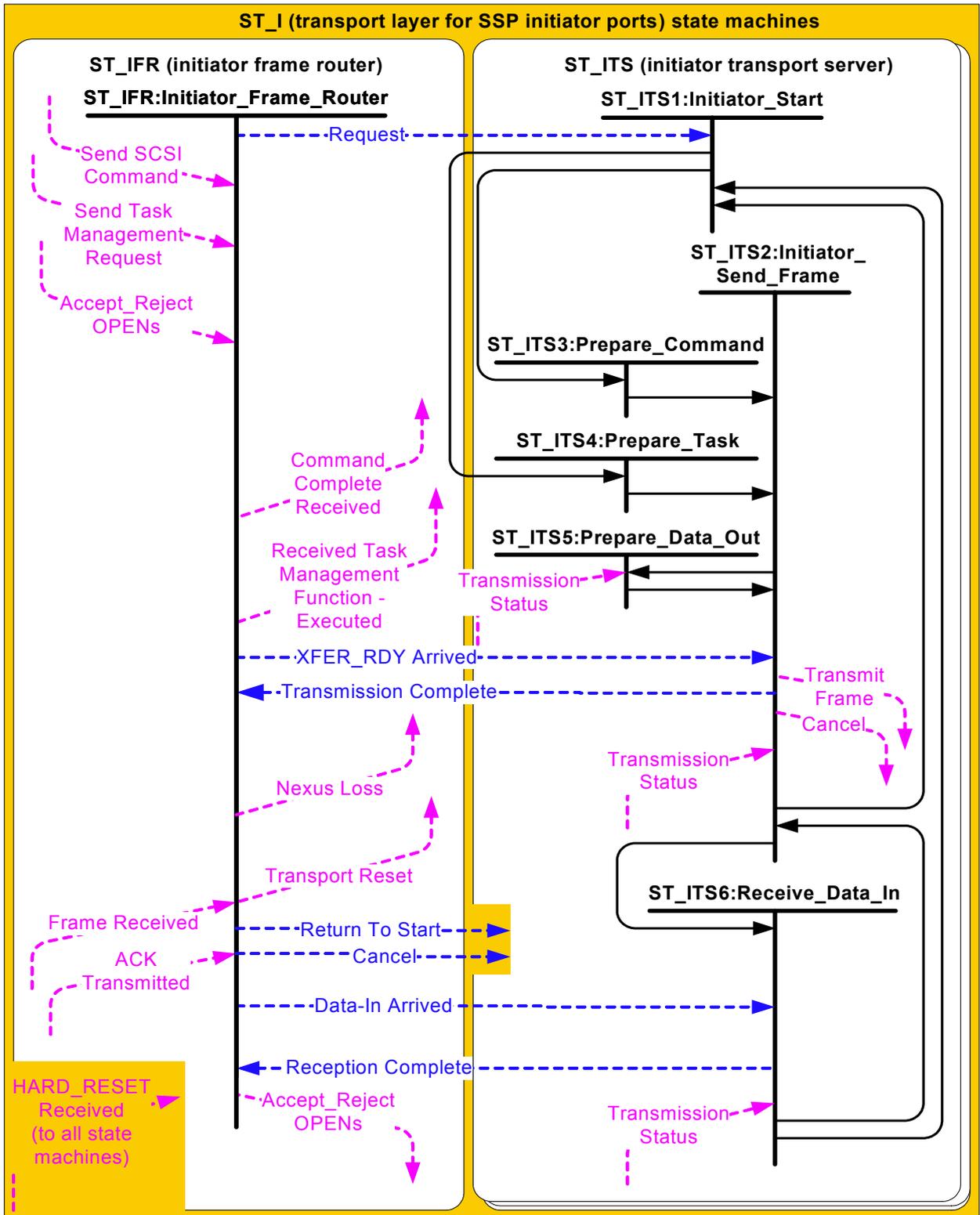
Figure 175 — ST_I (transport layer for SSP initiator ports) state machines

9.2.6.2.2 ST_IFR (initiator frame router) state machine

9.2.6.2.2.1 ST_IFR state machine overview

The ST_IFR state machine performs the following functions:

a) receives Send SCSI Command and Send Task Management transport protocol service requests from the SCSI application layer;
b) sends messages to the ST_ITS state machine;
c) receives messages from the ST_ITS state machine;
d) receives confirmations from the port layer;
e) sends transport protocol service confirmations to the SCSI application layer;
f) receives vendor-specific requests from the SCSI application layer;
g) sends vendor-specific confirmations to the SCSI application layer;
h) receives Accept_Reject OPENs requests from the SCSI application layer;
i) sends Accept_Reject OPENs requests to the port layer;
j) sends I_T Nexus Loss event notifications to the SCSI application layer; and
k) sends Transport Reset event notifications to the SCSI application layer.

This state machine consists of one state.

This state machine shall be started after power on.

9.2.6.2.2.2 Processing transport protocol service requests

If this state machine receives a Send SCSI Command transport protocol service request then this state machine shall send a Request (Send Command) message with Command arguments and Buffer arguments to the ST_ITS state machine for the specified tag.

The following is the list of Command arguments:

a) connection rate;
b) initiator connection tag;
c) destination SAS address;
d) source SAS address set to the SAS address of the SSP initiator port;
e) tag;
f) logical unit number;
g) task priority;
h) task attribute;
i) additional CDB length;
j) CDB; and
k) additional CDB bytes, if any.

The following is the list of Buffer arguments:

a) data-in buffer size;
b) data-out buffer; and
c) data-out buffer size.

If the command is performing a write operations and the Send SCSI Command transport service request contains a First Burst Enabled argument, then the Request (Send Command) message shall also include the Enable First Burst argument and the number of bytes for the First Burst Size argument.

If this state machine receives a Send Task Management Request transport protocol service request, then this state machine shall send a Request (Send Task) message with the Task arguments to the ST_ITS state machine for the specified tag.

The following is the list of Task arguments:

a) connection rate;
b) initiator connection tag;
c) source SAS address set to the SAS address of the SSP initiator port;
d) destination SAS address;
e) retransmit bit;
f) tag;
g) logical unit number;

h) task management function; and
i) tag of task to be managed.

If the ST_ITS state machine for the tag specified in the Send Task Management Request is currently in use, then this state machine shall set the retransmit bit argument to one. If the ST_ITS state machine for the tag specified in the Send Task Management Request is not currently in use, then this state machine shall set the Retransmit Bit argument to zero.

9.2.6.2.2.3 Processing Frame Received confirmations

If this state machine receives a Frame Received (ACK/NAK Balanced) confirmation or Frame Received (ACK/NAK Not Balanced) confirmation, then this state machine shall compare the frame type of the frame received with the received confirmation (see table 117 in 9.2.1). If the confirmation was Frame Received (ACK/NAK Balanced) and the frame type is not XFER_RDY, RESPONSE, or DATA, then this state machine shall discard the frame. If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then this state machine shall discard the frame.

If the frame type is correct relative to the Frame Received confirmation, then this state machine may check that the hashed source SAS address matches the SAS address of the SAS port that transmitted the frame and that the hashed destination SAS address matches the SAS address of the SAS port that received the frame based on the connection information. If this state machine checks these SAS addresses, and they do not match, then this state machine:

a) shall discard the frame; and
b) may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is XFER_RDY then this state machine shall check the length of the information unit. If the length of the information unit is not correct, then this state machine:

a) shall discard the frame; and
b) may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is XFER_RDY and the tag is for a task with no write data, then this state machine shall:

a) discard the frame;
b) send a Command Complete Received transport protocol service confirmation with the Delivery Result argument set to Service Delivery or Target Failure - DATA Not Expected to the SCSI application layer; and
c) if there is an ST_ITS state machine for the tag, send a Return To Start message to that state machine.

If the frame type is DATA and the tag is for a task with no read data, then this state machine shall:

a) discard the frame;
b) send a Command Complete Received transport protocol service confirmation with the Delivery Result argument set to Service Delivery or Target Failure - DATA Not Expected to the SCSI application layer; and
c) if there is an ST_ITS state machine for the tag, send a Return To Start message to that state machine.

If the frame type is RESPONSE then this state machine shall check the length of the information unit. If the length of the information unit is not correct and the RESPONSE frame was for a command, then this state shall discard the frame and send a Command Complete Received confirmation to the SCSI application layer with the Service Response argument set to Service Delivery or Target Failure. If the length of the information unit is not correct and the RESPONSE frame was for a task management function, then this state shall discard the frame and send a Received Task Management Function – Executed confirmation to the SCSI application layer with the Service Response argument set to Service Delivery or Target Failure.

If the frame type is correct relative to the Frame Received confirmation, then this state machine shall check the tag. If the tag does not specify a valid ST_ITS state machine, then this state machine shall discard the frame and may send a vendor-specific confirmation to the SCSI application layer to cause the command using that tag to be aborted.

If the frame type is RESPONSE, and this state machine has received a RESPONSE frame for the I_T_L_Q nexus, then this state machine shall discard the frame.

If the frame type is RESPONSE, the fields checked in the frame are correct, and this state machine has not received a RESPONSE frame for this I_T_L_Q nexus, then this state machine shall send a Return To Start message to the ST_ITS state machine for the specified tag and:

   a) if the RESPONSE frame was for a command, then this state machine shall send a Command Complete Received protocol service confirmation to the SCSI application layer with the arguments set as specified in table 152 (see 10.2.1.5); or
   b) if the RESPONSE frame was for a task management request, then this state machine shall send a Received Task Management Function Executed protocol service confirmation to the SCSI application layer with the arguments set as specified in table 152 (see 10.2.1.5).

If the frame type is XFER_RDY and the fields checked in the frame are correct, then this state machine shall wait to receive an ACK Transmitted confirmation.

If this state machine receives an ACK Transmitted confirmation for an XFER_RDY frame, then this state machine shall send an XFER_RDY Arrived message to ST_ITS state machine specified by the tag. The message shall include the following Xfer_Rdy arguments:

   a) retry data frames;
   b) retransmit bit;
   c) target port transfer tag;
   d) requested offset; and
   e) write data length.

If the frame type is DATA and the fields checked in the frame are correct, then this state machine shall send a Data-In Arrived message to the ST_ITS state machine specified by the tag. The message shall include the following Read Data arguments:

   a) changing data pointer;
   b) number of fill bytes;
   c) data offset; and
   d) data.

9.2.6.2.2.4 Processing Transmission Complete and Reception Complete messages

If this state receives a Transmission Complete (I_T Nexus Loss), then this state machine shall send a Nexus Loss event notification to the SCSI application layer.

Table 129 defines the transport protocol service confirmation and its argument generated as a result of receiving a Transmission Complete message or a Reception Complete message that indicate an error occurred during the transmission or reception of a frame.

Table 129 — Confirmations sent to the SCSI application layer if a frame transmission or reception error occurs

Message received from ST_ITS state machine	Protocol service confirmation and Delivery Result argument sent to the SCSI application layer
Transmission Complete (Command Failed, ACK/NAK Timeout)	Command Complete Received (Service Delivery or Target Failure - ACK/NAK Timeout)
Transmission Complete (Command Failed, NAK Received)	Command Complete Received (Service Delivery or Target Failure - NAK Received)
Transmission Complete (Task Failed, ACK/NAK Timeout)	Received Task Management Function - Executed (Service Delivery or Target Failure - ACK/NAK Timeout)
Transmission Complete (Task Failed, NAK Received)	Received Task Management Function - Executed (Service Delivery or Target Failure - NAK Received)
Transmission Complete (XFER_RDY Incorrect Write Data Length)	Command Complete Received (Service Delivery or Target Failure - XFER_RDY Incorrect Write Data Length)
Transmission Complete (XFER_RDY Requested Offset Error)	Command Complete Received (Service Delivery or Target Failure - XFER_RDY Requested Offset Error)
Transmission Complete (Cancel Acknowledged)	Command Complete Received (Service Delivery or Target Failure - Cancel Acknowledged)
Reception Complete (Data Offset Error)	Command Complete Received (Service Delivery or Target Failure - DATA Offset Error)
Reception Complete (Too Much Read Data)	Command Complete Received (Service Delivery or Target Failure - DATA Too Much Read Data)
Reception Complete (Information Unit Too Short)	Command Complete Received (Service Delivery or Target Failure - DATA Information Unit Too Short)
Reception Complete (Command Failed, ACK/NAK Timeout)	Command Complete Received (Service Delivery or Target Failure - ACK/NAK Timeout)

The protocol service confirmation shall include the tag as an argument.

9.2.6.2.2.5 Processing miscellaneous requests

If this state machine receives an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request, then this state machine shall send a corresponding Accept_Reject OPENs request to the port layer.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall send a Transport Reset event notification to the SCSI application layer.

This state machine may receive vendor-specific requests from the SCSI application layer that cause this state machine to send a Cancel message to an ST_ITS state machine.

9.2.6.2.3 ST_ITS (initiator transport server) state machine

9.2.6.2.3.1 ST_ITS state machine overview

The ST_ITS state machine performs the following functions:

    a) receives and processes messages from the ST_IFR state machine;
    b) sends messages to the ST_IFR state machine;
    c) sends request to the port layer regarding frame transmission;
    d) receives confirmations from the port layer regarding frame transmission; and

    e)   receives HARD_RESET Received confirmations from the port layer.

This state machine consists of the following states:

    a)   ST_ITS1:Initiator_Start state (see 9.2.6.2.3.2) (initial state);
    b)   ST_ITS2:Initiator_Send_Frame state (see 9.2.6.2.3.3);
    c)   ST_ITS3:Prepare_Command state (see 9.2.6.2.3.4);
    d)   ST_ITS4:Prepare_Task state (see 9.2.6.2.3.5);
    e)   ST_ITS5:Prepare_Data_Out state (see 9.2.6.2.3.6); and
    f)   ST_ITS6:Receive_Data_In state (see 9.2.6.2.3.7).

This state machine shall start in the ST_ITS1:Initiator_Start state after power on.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall transition to the ST_ITS1:Initiator_Start state.

This state machine shall maintain the state machine variables defined in table 130.

Table 130 — ST_ITS state machine variables

State machine variable	Description
Data-In Buffer Offset	Current offset in the data-in buffer for read data
Data-Out Buffer Offset	Current offset in the data-out buffer for write data
Previous Requested Offset	Data offset from the last XFER_RDY frame received
Previous Write Data Length	Write data length from the last XFER_RDY frame received

This state machine shall maintain the state machine arguments defined in table 131.

Table 131 — ST_ITS state machine arguments

State machine argument	Description
Command	Consists of the Command arguments received in the Request (Send Command) message
Task	Consists of the arguments received in the Request (Send Task) message
Xfer_Rdy	Consists of the arguments received in the XFER_RDY Arrived message
Data-Out Buffer	The location of the write data buffer
Data-Out Buffer Size	The size in bytes of the write data buffer
Data-In Buffer Size	The size in bytes of the read data buffer

9.2.6.2.3.2 ST_ITS1:Initiator_Start state

9.2.6.2.3.2.1 State description

This state is the initial state of the ST_ITS state machine.

Upon entry into this state, this state shall set the Data-In Buffer Offset state machine variable to zero.

Upon entry into this state, this state shall set the Data-Out Buffer Offset state machine variable to zero.

9.2.6.2.3.2.2 Transition ST_ITS1:Initiator_Start to ST_ITS3:Prepare_Command

This transition shall occur after this state receives a Request (Send Command) message.

9.2.6.2.3.2.3 Transition ST_ITS1:Initiator_Start to ST_ITS4:Prepare_Task

This transition shall occur after this state receives a Request (Send Task) message.

9.2.6.2.3.3 ST_ITS2:Initiator_Send_Frame state

If this state is entered from the ST_ITS3:Prepare_Command state for transmission of a COMMAND frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS6:Receive_Data_In state, and the vendor-specific number of retries has not been reached for the COMMAND frame requesting a read operation, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS4:Prepare_Task state for transmission of an TASK frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_ITS5:Prepare_Data_Out state for transmission of a write DATA frame, then this state shall send a Transmit Frame (Non-Interlocked) request to the port layer if:

   a) this state has received an XFER_RDY Arrived message; or
   b) first burst is enabled and this state has received a Transmission Status (Frame Transmitted) confirmation and a Transmission Status (ACK Received) confirmation for the COMMAND frame.

A Transmit Frame request shall include the COMMAND frame from the ST_ITS3:Prepare_Command state or from the ST_ITS6:Receive_Data_In state, the TASK frame from the ST_ITS4:Prepare_Task state, or the write DATA frame from the ST_ITS5:Prepare_Data_Out state and the following arguments to be used for any OPEN address frame:

   a) initiator port bit set to one;
   b) protocol set to SSP;
   c) Connection Rate argument;
   d) Initiator Connection Tag argument;
   e) Destination SAS Address argument; and
   f) Source SAS Address argument.

After sending a Transmit Frame request this state shall wait to receive a Transmission Status confirmation.

If the confirmation is Transmission Status (I_T Nexus Loss), then this state shall send a Transmission Complete (I_T Nexus Loss) message to the ST_IFR state machine. This Transmission Complete message shall include the tag as an argument.

If the confirmation is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss) (see table 113 in 8.2.2.3.4), and the Transmit Frame request was for a COMMAND frame or a DATA frame, then this state shall send a Transmission Complete (Command Failed, Connection Failed) message to the ST_IFR state machine. The message shall include the tag.

If the confirmation is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss) (see table 113 in 8.2.2.3.4), and the Transmit Frame request was for a TASK frame, then this state shall send a Transmission Complete (Task Failed, Connection Failed) message to the ST_IFR state machine. The message shall include the tag.

If the confirmation is Transmission Status (Frame Transmitted), and the Transmit Frame request was for a COMMAND frame not requesting a read operation, a COMMAND frame not requesting a write operation, a TASK frame, or a write DATA frame where the number of data bytes that have been transmitted equal the request byte count, then this state shall wait to receive one of the following confirmations:

   a) Transmission Status (ACK Received);
   b) Transmission Status (NAK Received);
   c) Transmission Status (ACK/NAK Timeout); or
   d) Transmission Status (Connection Lost Without ACK/NAK).

If the confirmation is Transmission Status (Frame Transmitted), and the Transmit Frame request was for a COMMAND frame requesting a write operation, or a write DATA frame where the number of data bytes that have been transmitted is less than the request byte count and the write data length from the previous XFER_RDY frame, then this state shall wait to receive one of the following confirmations:

   a) Transmission Status (ACK Received);
   b) Transmission Status (NAK Received);
   c) Transmission Status (ACK/NAK Timeout);
   d) Transmission Status (Connection Lost Without ACK/NAK); or
   e) XFER_RDY Arrived message.

If a XFER_RDY Arrived message is received, then the ST_ITS shall respond to the XFER_RDY frame as if a Transmission Status (ACK Received) was received.

> NOTE 52 - If the number of data bytes requested to be transmitted for the Send SCSI Command protocol service request are fewer than the number of bytes in the service request, then this state may send additional Transmit Frame requests for write DATA frames for the protocol service request before receiving a Transmission Status (ACK Received), Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK) confirmation for Transmit Frame requests for previous write DATA frames sent for the I_T_L_Q nexus.

After a Transmission Status (Frame Transmitted) is received, if a confirmation of Transmission Status (NAK Received) is received, the Transmit Frame request was for a COMMAND frame, and the vendor-specific number of retries has not been reached, then this state shall send a Transmit Frame (interlocked) request to the port layer (i.e., the last COMMAND frame is retransmitted).

After a Transmission Status (Frame Transmitted) is received, if a confirmation of Transmission Status (NAK Received) is received, the Transmit Frame request was for a TASK frame, and the vendor-specific number of retries has not been reached, then this state shall send a Transmit Frame (interlocked) request to the port layer (i.e., the last TASK frame is retransmitted).

Table 132 defines the messages that this state shall send to the ST_IFR state machine upon receipt of the listed confirmations, based on the conditions under which each confirmation was received.

Table 132 — Messages sent to the ST_IFR state machine

Confirmation received from the port layer	Conditions under which confirmation was received	Message sent to ST_IFR state machine
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)	The Transmit Frame request was for a COMMAND frame.	Transmission Complete (Command Failed, ACK/NAK Timeout)
	The Transmit Frame request was for a TASK frame.	Transmission Complete (Task Failed, ACK/NAK Timeout)
Transmission Status (NAK Received)	The Transmit Frame request was for a COMMAND frame and the vendor-specific number of retries has been reached.	Transmission Complete (Command Failed, NAK Received)
	The Transmit Frame request was for a TASK frame and the vendor-specific number of retries has been reached.	Transmission Complete (Task Failed, NAK Received)
Transmission Status (NAK Received)	The Transmit Frame request was for a write DATA frame and: a) the RETRY DATA FRAMES bit was set to zero in the XFER_RDY frame requesting the data; or b) the RETRY DATA FRAMES bit was set to one in the XFER_RDY frame requesting the data, and the vendor-specific number of retries has been reached.	Transmission Complete (Data-Out Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)		Transmission Complete (Data-Out Failed, ACK/NAK Timeout)

After this state sends a Transmission Complete (Command Failed, ACK/NAK Timeout) this state shall continue processing messages and confirmations.

> NOTE 53 - The application client may determine the command was received and is being processed by the device server and allow the command to complete. The application client may accomplish this by the use of the QUERY TASK task management request.

If this state receives a Return to Start message or a Return to Start argument, and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to

the port layer. This state may also send a Cancel request to the port layer to cancel a previous Transmit Frame request.

If this state receives a Cancel message or a Cancel argument, and this state has received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Transmission Complete (Cancel Acknowledged) message to the ST_IFR state machine.

If this state receives a Cancel message or a Cancel argument, and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to the port layer. This state may also send a Cancel request to the port layer to cancel a previous Transmit Frame request. The Cancel request shall include the following arguments:

    a) destination SAS address; and
    b) tag.

> NOTE 54 - The Cancel message results from a vendor-specific request from the SCSI application layer after the SCSI application layer has used a task management function to determine that the SAS target port did not receive the COMMAND frame.

If this state receives a Transmission Status (Cancel Acknowledged) confirmation, then this state shall send a Transmission Complete (Cancel Acknowledged) message to the ST_IFR state machine.

If this state receives an XFER_RDY Arrived message, then this state shall verify the Xfer_Rdy state machine argument as specified in table 133. If the verification fails, then this state sends the Transmission Complete message specified in table 133 to the ST_IFR state machine.

Table 133 — Transmission Complete messages for XFER_RDY frame verification failures

Message sent to ST_IFR [a]	Condition
Transmission Complete (XFER_RDY Incorrect Write Data Length)	The Write Data Length Xfer_Rdy state machine argument is zero.
	The value in the Requested Offset Xfer_Rdy state machine argument plus the Write Data Length Xfer_Rdy state machine argument is greater than the Data-Out Buffer Size state machine argument.
Transmission Complete (XFER_RDY Requested Offset Error)	First burst is not enabled, this is the first XFER_RDY frame for a command, and the value in the Requested Offset Xfer_Rdy state machine argument is not set to zero.
	First burst is enabled, this is the first XFER_RDY frame for a command, and the value in the Requested Offset Xfer_Rdy state machine argument is not equal to the value indicated by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1.5).
	Transport layer retries are disabled and the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable plus the Previous Write Data Length Field state machine variable.
	Transport layer retries are enabled, the Retransmit Bit Xfer_Rdy state machine argument is set to zero, and the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable plus the Previous Write Data Length state machine variable.
	Transport layer retries are enabled, this is not the first XFER_RDY frame for the command, the Retransmit Bit Xfer_Rdy state machine argument is set to one, and: a) the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable; and b) the Requested Offset Xfer_Rdy state machine argument is not equal to the Previous Requested Offset state machine variable plus the Previous Write Data Length Field state machine variable.
[a] If more than one condition is true, then this state shall send the Transmission Complete (XFER_RDY Incorrect Write Data Length) message to the ST_IFR state machine.	

After this state verifies an XFER_RDY frame, it shall:

- a) set the Data-Out Buffer Offset state machine variable to the Requested Offset Xfer_Rdy state machine argument;
- b) set the Previous Requested Offset state machine variable to the Requested Offset Xfer_Rdy state machine argument; and
- c) set the Previous Write Data Length state machine variable to the Requested Offset Xfer_Rdy state machine argument.

9.2.6.2.3.3.3 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS1:Initiator_Start

This transition shall occur after:

- a) this state has sent one of the following to the ST_IFR state machine:
  - A) a Transmission Complete (Command Failed, NAK Received) message;
  - B) a Transmission Complete (Task Failed, ACK/NAK Timeout) message;
  - C) a Transmission Complete (Task Failed, NAK Received) message;
  - D) a Transmission Complete (Command Failed, ACK/NAK Timeout) message and the command was for a non-data operation;
  - E) a Transmission Complete (Data-Out Failed, NAK Received) message;
  - F) a Transmission Complete (Data-Out Failed, ACK/NAK Timeout) message;
  - G) a Transmission Complete (XFER_RDY Incorrect Write Data Length) message;
  - H) a Transmission Complete (XFER_RDY Requested Offset Error) message; or
  - I) a Transmission Complete (Cancel Acknowledged) message;

  or
- b) this state has received a Return To Start message or Return To Start argument, and has received:
  - A) confirmations for all Transmit Frame requests sent to the port layer; or
  - B) a Transmission Status (Cancel Acknowledged) confirmation.

9.2.6.2.3.3.4 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS5:Prepare_Data_Out

If first burst is enabled, this transition shall occur and include the First Burst argument after this state receives:

- a) a Transmission Status (Frame Transmitted) confirmation followed by a Transmission Status (ACK Received) for a COMMAND frame requesting a write operation; or
- b) a Transmission Status (Frame Transmitted) confirmation for a Transmit Frame (Non-interlocked) request if the Data-Out Buffer Offset state machine variable is less than the first burst size.

This transition shall occur after this state receives:

- a) an XFER_RDY Arrived message; or
- b) a Transmission Status (Frame Transmitted) confirmation for a Transmit Frame (Non-interlocked) request if the Data-Out Buffer Offset state machine variable is less than the Requested Offset Xfer_Rdy state machine argument plus the Write Data Length Xfer_Rdy state machine argument.

  NOTE 55 - This transition occurs even if this state has not received a Transmission Status (ACK Received) for the write DATA frame.

This transition shall include a Retry argument and occur after:

- a) this state receives one of the following confirmations or arguments for a write DATA frame:
  - A) Transmission Status (NAK Received);
  - B) Transmission Status (ACK/NAK Timeout); or
  - C) Transmission Status (Connection Lost without ACK/NAK);
- b) the RETRY DATA FRAMES bit set to one in the XFER_RDY frame for the write operation
- c) the Data-Out Buffer Offset state machine variable is set to the Requested Offset Xfer_Rdy state machine argument; and
- d) the vendor-specific number of retries, if any, for the write DATA frame has not been reached.

9.2.6.2.3.3.5 Transition ST_ITS2:Initiator_Send_Frame to ST_ITS6:Process_Data_In

This transition shall occur after this state receives a Transmission Status (Frame Transmitted) confirmation for a COMMAND frame for a command requesting a read operation.

NOTE 56 - This transition occurs even if this state has not received a Transmission Status (ACK Received) for the COMMAND frame.

9.2.6.2.3.4 ST_ITS3:Prepare_Command state

9.2.6.2.3.4.1 State description

This state shall construct a COMMAND frame using the Command arguments:

a)   FRAME TYPE field set to 06h (i.e., COMMAND frame);
b)   HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Commands argument;
c)   HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
d)   RETRY DATA FRAMES bit set to zero;
e)   RETRANSMIT bit set to zero;
f)   CHANGING DATA POINTER bit set to zero;
g)   NUMBER OF FILL BYTES field set zero.
h)   TAG field set to the Tag Command argument;
i)   TARGET PORT TRANSFER TAG field set to zero;
j)   DATA OFFSET field set to zero;
k)   in the information unit, LOGICAL UNIT NUMBER field set to the Logical Unit Number Command argument;
l)   in the information unit, ENABLE FIRST BURST bit set to the Enable First Burst Command argument;
m)   in the information unit, TASK PRIORITY field set to the Task Priority Command argument;
n)   in the information unit, TASK ATTRIBUTE field set to the Task Attribute Command argument;
o)   in the information unit, ADDITIONAL CDB LENGTH field set to the Additional CDB Length Command argument;
p)   in the information unit, CDB field set to the CDB Command argument;
q)   in the information unit, ADDITIONAL CDB BYTES field set to the Additional CDB Bytes Command argument, if any; and
r)   no fill bytes.

9.2.6.2.3.4.2 Transition ST_ITS3:Prepare_Command to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state:

a)   constructs a COMMAND frame;
b)   receives a Cancel message; or
c)   receives a Return To Start message.

This transition shall include the:

a)   COMMAND frame as an argument;
b)   if a Cancel message was received, then a Cancel argument; or
c)   if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.5 ST_ITS4:Prepare_Task state

9.2.6.2.3.5.1 State description

This state shall construct a TASK frame using the Task arguments:

a)   FRAME TYPE field set to 16h (i.e., TASK frame);
b)   HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Task argument;
c)   HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
d)   RETRY DATA FRAMES bit set to zero;
e)   RETRANSMIT bit set to the Retransmit Bit Task argument;
f)   CHANGING DATA POINTER bit set to zero;
g)   NUMBER OF FILL BYTES field set zero.
h)   TAG field set to the Tag Task argument;
i)   TARGET PORT TRANSFER TAG field set to zero;
j)   DATA OFFSET field set to zero;
k)   in the information unit, LOGICAL UNIT NUMBER field set to the Logical Unit Number Task argument;
l)   in the information unit, TASK MANAGEMENT FUNCTION field set to the Task Management Function Task argument;

m) in the information unit, TAG OF TASK TO BE MANAGED field set to the Tag Task argument of task to be managed and

n) no fill bytes.

9.2.6.2.3.5.2 Transition ST_ITS4:Prepare_Task to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state:

a) constructs a TASK frame;
b) receives a Cancel message; or
c) receives a Return To Start message.

This transition shall include the:

a) TASK frame as an argument;
b) if a Cancel message was received, then a Cancel argument; or
c) if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.6 ST_ITS5:Prepare_Data_Out state

9.2.6.2.3.6.1 State description

This state shall construct a write DATA frame using the following Xfer_Rdy state machine arguments and Command state machine arguments:

a) FRAME TYPE field set to 01h (i.e., DATA frame);
b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Command argument;
c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP initiator port's SAS address;
d) RETRY DATA FRAMES bit set to zero;
e) RETRANSMIT bit set to zero;
f) CHANGING DATA POINTER bit set as specified in this subclause;
g) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the specified data;
h) TAG field set to the Tag Command argument;
i) TARGET PORT TRANSFER TAG field set to FFFFh if this state received a First Burst argument or the Target Port Transfer Tag Xfer_Rdy argument if this state did not receive a First Burst argument;
j) DATA OFFSET field set to the Data-Out Buffer Offset state machine variable;
k) in the information unit, DATA field set to the information that starts at the location in the Data-Out Buffer state machine argument pointed to by the Data-Out Buffer Offset state machine variable and shall contain the amount of data indicated by the Write Data Length Xfer_Rdy argument; and
l) fill bytes, if any.

If this state is entered without a Retry argument, then this state shall set the CHANGING DATA POINTER bit to zero.

If this state is entered with a Retry argument, then this state shall set the CHANGING DATA POINTER bit to one.

After constructing the write DATA frame, this state shall set the Data-Out Buffer Offset state machine variable to the value of the DATA OFFSET field plus the length of the DATA field in the information unit.

9.2.6.2.3.6.2 Transition ST_ITS5:Prepare_Data_Out to ST_ITS2:Intiator_Send_Frame

This transition shall occur after this state:

a) constructs a write DATA frame;
b) receives a Cancel message; or
c) receives a Return To Start message.

This transition shall include the received Transmission Status, if any, as an argument and the:

a) write DATA frame as an argument;
b) if a Cancel message was received, then a Cancel argument; or
c) if a Return To Start message was received, then a Return To Start argument.

9.2.6.2.3.7 ST_ITS6:Receive_Data_In state

9.2.6.2.3.7.1 State description

If this state receives a Data-In Arrived message, then this state shall verify the values in the read DATA frame received with the message as defined in table 134.

If the verification fails, then this state sends the Reception Complete message specified in table 134 to the ST_IFR state machine.

Table 134 — Reception Complete messages for read DATA frame verification failures

Message sent to ST_IFR [a]	Condition
Reception Complete (Data Offset Error)	Transport layer retries are disabled, and the DATA OFFSET field in the read DATA frame is not equal Data-In Buffer Offset state machine variable.
	The DATA OFFSET field in the read DATA frame is greater than the Data-In Buffer Size state machine argument.
Reception Complete (Too Much Read Data)	The number of bytes in the DATA field in the read DATA information unit plus the Data-In Buffer Offset state machine variable is greater than the Data-In Buffer Size state machine argument.
Reception Complete (Incorrect Data Length)	The number of bytes in the DATA field in the read DATA information unit is zero.
[a] If more than one condition is true, then this state shall select which message to send to the ST_IFR state machine using the following order: 1) Reception Complete (Data Offset Error); 2) Reception Complete (Too Much Read Data); or 3) Reception Complete (Incorrect Data Length).	

If:

    a) transport layer retries are enabled;
    b) the CHANGING DATA POINTER bit is set to zero;
    c) the DATA OFFSET field is not set to the Data-In Buffer Offset state machine variable;
    d) the DATA OFFSET field is less than the Data-In Buffer Size state machine argument; and
    e) the DATA OFFSET field plus the length of the Data information unit is less than or equal to the Data-In Buffer Size state machine argument,

then this state should discard all Data-In Arrived messages until a read DATA frame is received in which the CHANGING DATA POINTER bit is set to one. This state shall resume processing additional Data-In Arrived messages when it receives a Data-In Arrived message with the CHANGING DATA POINTER bit set to one.

If the verification succeeds or after this state resumes processing Data-In Arrived messages, then this state shall process the data received in the read DATA frame and set the Data-In Buffer Offset state machine variable to the DATA OFFSET field plus the length of the Data information unit.

If this state receives Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK), then this state shall send a Reception Complete (Command Failed, Connection Failed) to the ST_IFR state machine.

After this state sends a Reception Complete (Command Failed, Connection Failed) message, this state shall continue processing messages and confirmations.

> NOTE 57 - The application client may determine the command was received and is being processed by the device server and allow the command to complete.

If this state receives a Cancel message, then this state shall send a Reception Complete (Cancel Acknowledged) message to the ST_IFR state machine. The Reception Complete message shall include the tag as an argument.

> NOTE 58 - The Cancel message results from a vendor-specific request from the SCSI application layer after the SCSI application layer has used a task management function to determine that the SAS target port did not receive the COMMAND frame.

9.2.6.2.3.7.2 Transition ST_ITS6:Receive_Data_In to ST_ITS1:Initiator_Start

This transition shall occur after this state:

- a) sends one of the following to the ST_IFR state machine:
  - A) a Reception Complete (Data Offset Error) message;
  - B) a Reception Complete (Too Much Read Data) message;
  - C) a Reception Complete (Incorrect Data Length) message; or
  - D) a Transmission Complete (Cancel Acknowledged) message;

  or

- b) receives a Return To Start message.

9.2.6.2.3.7.3 Transition ST_ITS6:Receve_Data_In to ST_ITS2:Initiator_Send_Frame

This transition shall occur after this state receives a Transmission Status (NAK Received) confirmation for a COMMAND frame for a command requesting a read operation.

9.2.6.3 ST_T (transport layer for SSP target ports) state machines

9.2.6.3.1 ST_T state machines overview

The ST_T state machines are as follows:

- a) ST_TFR (target frame router) state machine (see 9.2.6.3.2); and
- b) ST_TTS (target transport server) state machine (see 9.2.6.3.3).

The SAS target port includes:

- a) one ST_TFR state machine; and
- b) one ST_TTS state machine for each possible task and task management function (i.e., for each tag).

The ST_TTS state machine may maintain the timers listed in table 135.

Table 135 — ST_T state machine timers

| Timer	Initial value
Initiator Response Timeout	The value in the INITIATOR RESPONSE TIMEOUT field in the Protocol-Specific Port mode page (see 10.2.7.2).

Figure 176 shows the ST_T state machines.
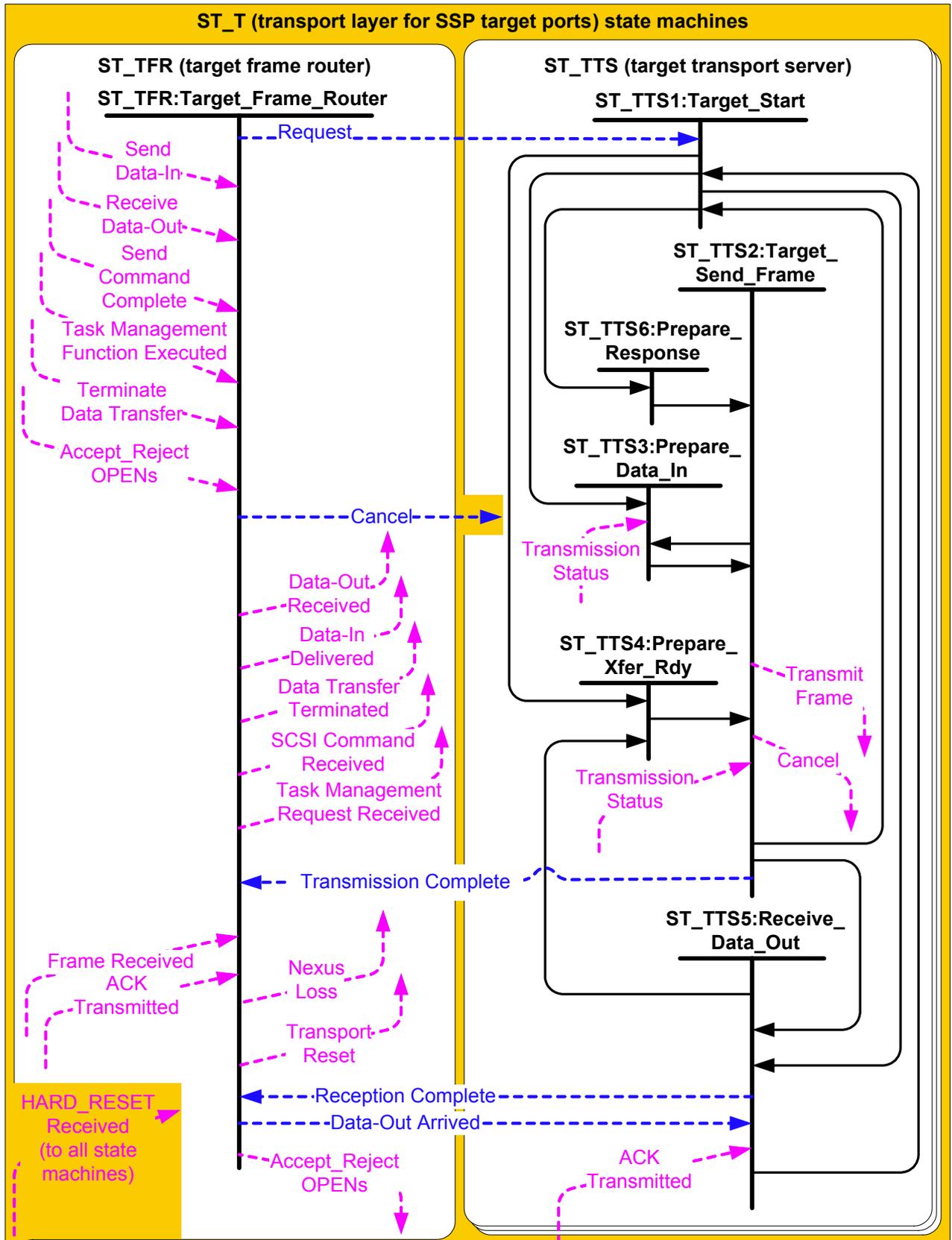
Figure 176 — ST_T (transport layer for SSP target ports) state machines

9.2.6.3.2 ST_TFR (target frame router) state machine

9.2.6.3.2.1 ST_TFR state machine overview

The ST_TFR state machine performs the following functions:

 a) receives confirmations from the port layer;
 b) receives transport protocol service requests from the SCSI application layer;
 c) sends transport protocol service indications to the SCSI application layer;
 d) sends messages to the ST_TTS state machine;
 e) receives messages from the ST_TTS state machine;
 f) receives Accept_Reject OPENs requests from the SCSI application layer;
 g) sends Accept_Reject OPENs requests to the port layer;
 h) sends Nexus Loss event notifications to the SCSI application layer; and
 i) sends Transport Reset event notifications to the SCSI application layer.

This state machine consists of one state.

This state machine shall be started after power on.

9.2.6.3.2.2 Processing Frame Received confirmations

If this state machine receives a Frame Received (ACK/NAK Balanced) or Frame Received (ACK/NAK Not Balanced) confirmation, then this state machine shall check the frame type in the received frame (see table 117 in 9.2.1). If the frame type is not COMMAND, TASK, or DATA, then this state machine shall discard the frame. If the confirmation was Frame Received (ACK/NAK Not Balanced) and the frame type is not DATA, then this state machine shall discard the frame.

This state machine may check that reserved fields in the received frame are zero. If any reserved fields are checked and they are not set to zero, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

 a) a Request (Send Transport Response) message with the Transport Response arguments;
 b) the destination SAS address argument set to the SAS address from which the invalid frame was received; and
 c) the Service Response argument set to Invalid Frame.

The check of reserved fields within the frame shall not apply to the reserved fields within the CDB in a COMMAND frame. Checking of reserved fields in a CDB is described in SPC-3.

The following is the list of Transport Response arguments:

 a) connection rate;
 b) initiator connection tag;
 c) destination SAS address (i.e., the SAS address to which the RESPONSE frame is to be transmitted);
 d) source SAS address set to the SAS address of the SAS port containing the state machine;
 e) tag; and
 f) service response.

If the frame type is correct relative to the Frame Received confirmation, then this state machine may check that the hashed source SAS address matches the SAS address of the SAS port that transmitted the frame and that the hashed destination SAS address matches the SAS address of the SAS port that received the frame based on the connection information. If this state machine checks these SAS addresses, and they do not match, then this state machine shall discard the frame.

If the frame type is COMMAND or TASK then this state machine shall check the length of the information unit. If the length of the information unit is not correct, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

 a) a Request (Send Transport Response) message with the Transport Response arguments;
 b) the destination SAS address set to the SAS address from which the invalid frame was received; and
 c) the Service Response argument set to Invalid Frame.

If the frame type is TASK, this state machine checks tags, the RETRANSMIT bit in the new TASK frame is set to one, and the tag for the new TASK frame is the same as the tag for a previous TASK frame where the task management function for the previous TASK frame is not complete, then this state machine shall discard the new TASK frame and not send a Task Management Request Received confirmation to the port layer.

If the frame type is TASK and this state machine does not check tags, then this state machine shall ignore the RETRANSMIT bit.

If the frame type is COMMAND or TASK, then this state machine may check the target port transfer tag. If this state checks the target port transfer tag and the tag is set to a value other than FFFFh, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

    a) a Request (Send Transport Response) message with the Transport Response arguments;
    b) the destination SAS address set to the SAS address from which the invalid frame was received; and
    c) the Service Response argument set to Invalid Frame.

If the frame type is TASK, then this state machine shall check the logical unit number. If the logical unit number is unknown, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

    a) a Request (Send Transport Response) message with the Transport Response arguments;
    b) the destination SAS address set to the SAS address from which the invalid frame was received; and
    c) the Service Response argument set to Incorrect Logical Unit Number.

If the frame type is DATA and this frame is for first burst data or this state machine did not assign a target port transfer tag for the data transfer, then this state machine may check the target port transfer tag. If target port transfer tag is set to a value other than FFFFh, then this state machine shall send the following to an ST_TTS state machine that does not have an active task and discard the frame:

    a) a Request (Send Transport Response) message with the Transport Response arguments;
    b) the destination SAS address set to the SAS address from which the invalid frame was received; and
    c) the Service Response argument set to Invalid Frame.

If the frame type is COMMAND or TASK and the fields checked in the frame are correct, then this state machine shall wait to receive an ACK Transmitted confirmation.

If the frame type is COMMAND, the fields checked in the frame are correct, and this state machine receives an ACK Transmitted confirmation, then this state machine shall send a SCSI Command Received transport protocol service indication with the following arguments to the SCSI application layer:

    a) source SAS address (i.e., the SAS address that transmitted the COMMAND frame);
    b) tag;
    c) logical unit number;
    d) task attribute;
    e) task priority;
    f) CDB; and
    g) additional CDB bytes, if any.

If the frame type is TASK, the fields checked in the frame are correct, and this state machine receives an ACK Transmitted confirmation, then this state machine shall send a Task Management Request Received transport protocol service indication with the following arguments to the SCSI application layer:

    a) source SAS address (i.e., the SAS address that transmitted the TASK frame);
    b) tag;
    c) logical unit number;
    d) task management function; and
    e) tag of the task to be managed.

If the frame type is DATA, and the tag does not match a tag for an outstanding command performing write operations, then this state machine shall discard the frame.

If the frame type is DATA, and the tag matches a tag for an outstanding command performing write operations when first burst is not enabled or for which no Transmission Complete (Xfer_Rdy Delivered) message has been received from an ST_TTS state machine, then this state machine shall discard the frame.

If the frame type is DATA and a target port transfer tag was received in a Transmission Complete (Xfer_Rdy Delivered) message, then this state machine shall check the target port transfer tag. If the target port transfer tag received in the DATA frame does not match the Target Transport Tag argument in the Transmission Complete (Xfer_Rdy Delivered) message, then this state machine shall discard the frame.

If the frame type is DATA and the fields checked in the frame are correct, and first burst is enabled or this state machine has received a Transmission Complete (Xfer_Rdy Delivered) from the ST_TTS state machine for the

request, then this state machine shall send a Data-Out Arrived message to the ST_TTS state machine specified by the tag in the frame. The message shall include the content of the write DATA frame.

9.2.6.3.2.3 Processing transport protocol service requests and responses

If this state machine receives a Send Data-In transport protocol service request from the SCSI application layer, then this state machine shall send a Request (Send Data-In) message to an ST_TTS state machine that does not have an active task. The message shall include the following Data-In arguments:

   a) connection rate;
   b) initiator connection tag;
   c) destination SAS address (i.e., the SAS address to which the read DATA frame is to be transmitted);
   d) source SAS address set to the SAS address of the SSP target port;
   e) tag;
   f) device server buffer;
   g) request byte count; and
   h) application client buffer offset.

If this state machine receives a Receive Data-Out transport protocol service request from the SCSI application layer, then this state machine shall send a Request (Receive Data-Out) message to an ST_TTS state machine that does not have an active task. The message shall include the following Data-Out arguments:

   a) connection rate;
   b) initiator connection tag;
   c) destination SAS address (i.e., the SAS address to which the XFER_RDY frame is to be transmitted);
   d) source SAS address set to the SAS address of the SSP target port;
   e) tag;
   f) device server buffer;
   g) request byte count;
   h) application client buffer offset; and
   i) target port transfer tag.

If first burst is enabled, then the Request (Receive Data_Out) message shall also include the Enable First Burst argument and First Burst Size argument. The First Burst Size argument shall be set to first burst size from the Disconnect-Reconnect mode page (see 10.2.7.1.5).

If this state machine receives a Send Command Complete transport protocol service response from the SCSI application layer with the Service Response argument set to TASK COMPLETE or LINKED COMMAND COMPLETE, then this state machine shall send a Request (Send Application Response) message to the ST_TTS state machine specified by the tag. The message shall include the following Application Response arguments:

   a) connection rate;
   b) initiator connection tag;
   c) destination SAS address (i.e., the SAS address to which the RESPONSE frame is to be transmitted);
   d) source SAS address set to the SAS address of the SSP target port;
   e) tag;
   f) status; and
   g) sense data, if any.

If this state machine receives a Task Management Function Executed transport protocol service response from the SCSI application layer, then this state machine shall send the following to the ST_TTS state machine specified by the tag:

   a) a Request (Send Transport Response) message with the Transport Response arguments; and
   b) the Service Response argument set as specified in table 136.

Table 136 specifies which argument to send with Request (Send Transport Response) based on the Service Response argument that was received.

Table 136 — Task Management Function Executed Service Response argument mapping to Service Response argument

Task Management Function Executed protocol service response Service Response argument received	Request (Send Transport Response) message Service Response argument
FUNCTION COMPLETE	Task Management Function Complete
FUNCTION SUCCEEDED	Task Management Function Succeeded
FUNCTION REJECTED	Task Management Function Not Supported
INCORRECT LOGICAL UNIT NUMBER	Incorrect Logical Unit Number
SERVICE DELIVERY OR TARGET FAILURE - Overlapped Tag Attempted	Overlapped Tag Attempted

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has not sent a Request message to a ST_TTS state machine for the Send Data-In or Receive Data-Out protocol service request to which the Terminate Data Transfer request applies, then this state machine shall:

1) discard the Terminate Data Transfer request and any corresponding Send Data-In or Receive Data-Out request; and
2) send a Data Transfer Terminated protocol service confirmation to the SCSI application layer.

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has sent a Request message to a ST_TTS state machine for the Send Data-In protocol service request to which the Terminate Data Transfer request applies, then this state machine shall send a Cancel message to the ST_TTS state machine specified by the tag and the Send Data-In protocol service request.

If this state machine receives a Terminate Data Transfer protocol service request from the SCSI application layer and this state machine has sent a Request message to a ST_TTS state machine for the Receive Data-Out protocol service request to which the Terminate Data Transfer request applies, then this state machine shall send a Cancel message to the ST_TTS state machine specified by the tag and the Receive Data-Out protocol service request.

This state machine receives Transmission Complete and Reception Complete messages that may result in this state machine sending a Nexus Loss event notification or a protocol service confirmation to the SCSI application layer. If this state machine receives a Transmission Complete (I_T Nexus Loss) message, then this state machine shall send a Nexus Loss event notification to the SCSI application layer. Table 137 defines messages received

from ST_TTS state machines and the corresponding service confirmations, if any, that shall be sent upon receipt of the message.

Table 137 — Confirmations sent to the SCSI application layer

Message received from ST_TTS state machine	Protocol service confirmation sent to SCSI application layer
Transmission Complete (Xfer_Rdy Delivered)	None
Transmission Complete (Response Delivered)	None
Transmission Complete (Response Failed) [a]	None
Transmission Complete (Data-In Delivered)	Data-In Delivered with the Delivery Result argument set to DELIVERY SUCCESSFUL
Transmission Complete (Xfer_Rdy Failed, NAK Received)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - NAK RECEIVED
Transmission Complete (Xfer_Rdy Failed, ACK/NAK Timeout)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Transmission Complete (Data-In Failed, NAK Received)	Data-In Delivered with the Delivery Result argument set to DELIVERY FAILURE - NAK RECEIVED
Transmission Complete (Data-In Failed, ACK/NAK Timeout)	Data-In Delivered with the Delivery Result argument set to DELIVERY FAILURE - CONNECTION FAILED
Reception Complete (Data-Out Received)	Data-Out Received with the Delivery Result argument set to DELIVERY SUCCESSFUL
Reception Complete (Data Offset Error)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - DATA OFFSET ERROR
Reception Complete (Too Much Write Data)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - TOO MUCH WRITE DATA
Reception Complete (Information Unit Too Short)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - INFORMATION UNIT TOO SHORT.
Reception Complete (Initiator Response Timeout)	Data-Out Received with the Delivery Result argument set to DELIVERY FAILURE - INITIATOR RESPONSE TIMEOUT
Reception Complete (Data Transfer Terminated)	Data Transfer Terminated
[a] SAM-3 does not define a mechanism for the device server to determine the result of its Send Command Complete and Task Management Function Executed transport protocol service response calls.	

Each protocol service confirmation shall include the tag as an argument.

9.2.6.3.2.4 Processing miscellaneous requests and confirmations

If this state machine receives an Accept_Reject OPENs (Accept SSP) or Accept_Reject OPENs (Reject SSP) request, then this state machine shall send a corresponding Accept_Reject OPENs request to the port layer.

If this state machine receives a HARD_RESET Received confirmation, then this state shall send a Transport Reset event notification to the SCSI application layer.

9.2.6.3.3 ST_TTS (target transport server) state machine

9.2.6.3.3.1 ST_TTS state machine overview

The ST_TTS state machine performs the following functions:

a) receives and processes messages from the ST_TFR state machine;
b) sends messages to the ST_TFR state machine;
c) communicates with the port layer using requests and confirmations regarding frame transmission; and
d) receives HARD_RESET Received confirmations from the port layer.

This state machine consists of the following states:

a) ST_TTS1:Target_Start (see 9.2.6.3.3.2) (initial state);
b) ST_TTS2:Target_Send_Frame (see 9.2.6.3.3.3);
c) ST_TTS3:Prepare_Data_In (see 9.2.6.3.3.4);
d) ST_TTS4:Prepare_Xfer_Rdy (see 9.2.6.3.3.5);
e) ST_TTS5:Receive_Data_Out (see 9.2.6.3.3.6); and
f) ST_TTS6:Prepare_Response (see 9.2.6.3.3.7).

This state machine shall start in the ST_TTS1:Target_Start state after power on.

If this state machine receives a HARD_RESET Received confirmation, then this state machine shall transition to the ST_TTS1:Target_Start state.

The state machine shall maintain the state machine variables defined in table 138.

Table 138 — ST_TTS state machine variables

State machine variable	Description
Read Data Offset	Offset into the application client buffer for read data
Balance Point Read Data Offset	Offset into the device server buffer for read data of last DATA frame to have received an ACK Transmitted confirmation
Requested Write Data Offset	Device server requested offset in the application client buffer for write data
Requested Write Data Length	Amount of write data requested by the Device server from the application client buffer

This state machine shall maintain the state machine arguments defined in table 139.

Table 139 — ST_TTS state machine arguments

State machine argument	Description
Data-In	The Data-In arguments received in the Request (Send Data-In) message
Data-Out	The Data-Out arguments received in the Request (Receive Data-Out) message

9.2.6.3.3.2 ST_TTS1:Target_Start state

9.2.6.3.3.2.1 State description

This state is the initial state of the ST_TTS state machine.

Upon entry into this state, this state shall set the Read Data Offset state machine variable to zero.

Upon entry into this state, this state shall set the Balance Point Read Data Offset state machine variable to zero.

Upon entry into this state, this state shall set the Requested Write Data Offset state machine variable to zero.

If this state was entered without an Enable First Burst argument, then the Requested Write Data Length state machine variable shall be set to the Request Byte Count Data-Out state machine argument.

If this state was entered with an Enable First Burst argument, then the Requested Write Data Length state machine variable shall be set to the First Burst Size argument.

9.2.6.3.3.2.2 Transition ST_TTS1:Target_Start to ST_TTS3:Prepare_Data_In

This transition shall occur after this state receives a Request (Send Data-In) message.

9.2.6.3.3.2.3 Transition ST_TTS1:Target_Start to ST_TTS4:Prepare_Xfer_Rdy

If this state was entered without an Enable First Burst argument, then this transition shall occur after a Request (Receive Data-Out) message is received.

9.2.6.3.3.2.4 Transition ST_TTS1:Target_Start to ST_TTS5:Receive_Data_Out

If this state was entered with an Enable First Burst argument, then this transition shall occur after a Request (Receive Data-Out) message is received.

9.2.6.3.3.2.5 Transition ST_TTS1:Target_Start to ST_TTS7:Prepare_Response

This transition shall occur after this state receives a Request (Send Transport Response) message.

The transition shall include the Transport Response arguments.

9.2.6.3.3.3 ST_TTS2:Target_Send_Frame state

If this state is entered from the ST_TTS3:Prepare_Data_In state for transmission of a read DATA frame, then this state shall send a Transmit Frame (Non-Interlocked) request to the port layer.

If this state is entered from the ST_TTS4:Prepare_Xfer_Rdy state for transmission of an XFER_RDY frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

If this state is entered from the ST_TTS6:Prepare_Response state for transmission of a RESPONSE frame, then this state shall send a Transmit Frame (Interlocked) request to the port layer.

All Transmit Frame requests from this state shall include the read DATA frame from the ST_TTS3:Prepare_Data_In state, the XFER_RDY frame from the ST_TTS4:Prepare_Xfer_Rdy state, or the RESPONSE frame from the ST_TTS6:Prepare_Response state and the following arguments to be used for any OPEN address frame:

   a) initiator port bit set to zero;
   b) protocol set to SSP;
   c) Connection Rate argument;
   d) Initiator Connection Tag argument;
   e) Destination SAS Address argument; and
   f) Source SAS Address argument.

After sending a Transmit Frame request this state shall wait to receive a Transmission Status confirmation.

If the confirmation or argument is Transmission Status (I_T Nexus Loss), then this state shall send a Transmission Complete (I_T Nexus Loss) message to the ST_TFR state machine. The Transmission Complete message shall include the tag as an argument.

If the confirmation or argument is not Transmission Status (Frame Transmitted) or Transmission Status (I_T Nexus Loss), then this state shall send the Transmission Complete message defined in table 140 to the ST_TFR state machine. The message shall include the following arguments:

   a) tag; and
   b) arguments received with the Transmission Status confirmation.

If the confirmation is Transmission Status (Frame Transmitted) and the Transmit Frame request was for:

   a) an XFER_RDY frame; or
   b) a RESPONSE frame,

then this state shall wait to receive one of the following confirmations:

   a) Transmission Status (ACK Received);
   b) Transmission Status (NAK Received);
   c) Transmission Status (ACK/NAK Timeout); or
   d) Transmission Status (Connection Lost Without ACK/NAK).

If the confirmation or argument is Transmission Status (Frame Transmitted), the Transmit Frame request was for a read DATA frame, and the Read Data Offset state machine variable is equal to the Request Byte Count Data-In argument, then this state shall wait to receive:

a) Transmission Status (ACK Received) confirmations or arguments for each outstanding read DATA frame; or

b) one of the following:
    A) Transmission Status (NAK Received);
    B) Transmission Status (ACK/NAK Timeout); or
    C) Transmission Status (Connection Lost Without ACK/NAK).

NOTE 59 - If the number of data bytes that have been transmitted for a Request (Send Data-In) message are fewer than the Data-In Request Byte Count argument, then this state transitions to the ST_TTS3:Prepare_Data_In state to construct the additional read DATA frames for the request before receiving a Transmission Status (ACK Received), Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK) confirmation.

When the number of Transmission Status (Frame Transmitted) confirmations for Transmit Frame (Non-Interlocked) requests equals the number of Transmission Status (ACK Received) confirmations and the Transmit Frame request was for a read DATA frame, this state shall set the Balance Point Read Data Offset state machine variable to the current Read Data Offset state machine variable.

If a Transmission Status (ACK Received) confirmation is received, and the Transmit Frame request was for a read DATA frame, then this state shall set the Balance Point Read Data Offset state machine variable to the current balance point read data offset plus the number of read data bytes transmitted in the read DATA frame associated with Transmission Status (ACK Received) confirmation.

If transport layer retries are enabled, the Transmit Frame request was for a RESPONSE frame, the vendor-specific number of retries has not been reached, and this state receives one of the following confirmations:

a) Transmission Status (NAK Received);
b) Transmission Status (ACK/NAK Timeout); or
c) Transmission Status (Connection Lost Without ACK/NAK),

then this state shall:

a) set the RETRANSMIT bit to one; and
b) resend a Transmit Frame (Interlocked) request to the port layer for the failed RESPONSE frame.

If transport layer retries are enabled, the Transmit Frame request was for a XFER_RDY frame, the vendor-specific number of retries has not been reached, and this state receives one of the following confirmations:

a) Transmission Status (NAK Received);
b) Transmission Status (ACK/NAK Timeout); or
c) Transmission Status (Connection Lost Without ACK/NAK),

then this state shall:

a) set the RETRANSMIT bit to one;
b) set the value in the TARGET PORT TRANSFER TAG field to a value that is different than the target port transfer tag in the previous XFER_RDY frame associated with the Data-out arguments and is different than any other target port transfer tag currently in use. If write data is received for a subsequent XFER_RDY frame for a command, then all target port transfer tags used for previous XFER_RDY frames for the command are no longer in use; and
c) resend a Transmit Frame (Interlocked) request to the port layer for the failed XFER_RDY frame.

Table 140 defines the messages that this state shall send to the ST_TFR state machine upon receipt of the listed confirmations, based on the conditions under which each confirmation was received.

Table 140 — Messages sent to the ST_TFR state machine

Confirmation received from the port layer	Conditions under which confirmation was received	Message sent to the ST_TFR state machine
Transmission Status (ACK Received)	The Transmit Frame request was for an XFER_RDY frame.	Transmission Complete (Xfer_Rdy Delivered) with a Target Port Transfer Tag argument
	Transmit Frame request was for a RESPONSE frame	Transmission Complete (Response Delivered)
	The Transmit Frame request was for a read DATA frame and: a) the Read Data Offset state machine variable is equal to the Data-In Request Byte Count argument; and b) the Read Data Offset state machine variable is equal to the Balance Point Read Data Offset state machine variable.	Transmission Complete (Data-In Delivered)
Transmission Status (NAK Received), Transmission Status (ACK/NAK Timeout), or Transmission Status (Connection Lost Without ACK/NAK)	The Transmit Frame request was for a RESPONSE frame and the vendor-specific number of retries has been reached.	Transmission Complete (Response Failed)
Transmission Status (NAK Received)	The Transmit Frame request was for an XFER_RDY frame and: a) if transport layer retries are disabled; or b) if transport layer retries are enabled and the vendor-specific number of retries has been reached.	Transmission Complete (Xfer_Rdy Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)		Transmission Complete (Xfer_Rdy Failed, Connection Failed)
Transmission Status (NAK Received)	The Transmit Frame request was for a read DATA frame and: a) if transport layer retries are disabled; or b) if transport layer retries are enabled and the vendor-specific number of retries has been reached.	Transmission Complete (Data-In Failed, NAK Received)
Transmission Status (ACK/NAK Timeout) or Transmission Status (Connection Lost Without ACK/NAK)		Transmission Complete (Data-In Failed, Connection Failed)

If this state receives a Cancel message or a Cancel argument and this state has received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Transmission Complete (Data Transfer Terminated) message to the ST_TFR state machine.

If this state receives a Cancel message or a Cancel argument and this state has not received confirmations for all Transmit Frame requests sent to the port layer, then this state shall send a Cancel request to the port layer to cancel previous Transmit Frame requests. The Cancel request shall include the following arguments:

    a) destination SAS address; and
    b) tag.

Upon receipt of a Transmission Status (Cancel Acknowledged) confirmation this state shall send a Transmission Complete (Data Transfer Terminated) message to the ST_TFR state machine.

A Transmission Complete message to the ST_TFR state machine shall include the following arguments:

- a)   destination SAS address; and
- b)   tag.

9.2.6.3.3.3.1 Transition ST_TTS2:Target_Send_Frame to ST_TTS1:Target_Start

This transition shall occur after this state sends a Transmission Complete message other than Transmission Complete (Xfer_Rdy Delivered) to the ST_TFR state machine.

9.2.6.3.3.3.2 Transition ST_TTS2:Target_Send_Frame to ST_TTS3:Prepare_Data_In

This transition shall occur after this state receives a Transmission Status (Frame Transmitted) confirmation for a read DATA frame and Read Data Offset state machine variable is less than the Request Byte Count Data-In argument.

If transport layer retries are enabled and the vendor-specific number of retries, if any, for the read DATA frame has not been reached, this transition shall occur and include a Retry argument after this state receives one of the following confirmations for a read DATA frame:

- a)   Transmission Status (NAK Received);
- b)   Transmission Status (ACK/NAK Timeout); or
- c)   Transmission Status (Connection Lost Without ACK/NAK).

9.2.6.3.3.3.3 Transition ST_TTS2:Target_Send_Frame to ST_TTS5:Receive_Data_Out

This transition shall occur after this state sends a Transmission Complete (Xfer_Rdy Delivered) message to the ST_TFR state machine.

9.2.6.3.3.4 ST_TTS3:Prepare_Data_In state

9.2.6.3.3.4.1 State description

This state retrieves the data from the Device Server Buffer and constructs a read DATA frame.

This state shall construct a read DATA frame using the Data-In arguments as follows:

- a)   FRAME TYPE field set to 01h (i.e., DATA frame);
- b)   HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Address Data-In argument;
- c)   HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
- d)   RETRY DATA FRAMES bit set to zero;
- e)   RETRANSMIT bit set to zero;
- f)   CHANGING DATA POINTER set as specified in this subclause;
- g)   NUMBER OF FILL BYTES field set to the number of fill bytes needed for the specified read data;
- h)   TAG field set to the Tag Data-In argument;
- i)   TARGET PORT TRANSFER TAG field set to zero;
- j)   DATA OFFSET field set as specified in this subclause;
- k)   in the information unit, DATA field set as specified in this subclause; and
- l)   fill bytes, if required.

If this state is entered without a Retry argument then this state shall:

- a)   set the CHANGING DATA POINTER bit set to zero;
- b)   set the DATA OFFSET field to the Read Data Offset state machine variable; and
- c)   in the information unit, DATA field set to the information that starts at the location in the specified device server buffer pointed to by the Read Data Offset state machine variable and shall contain the amount of data that is the lesser of:
  - A)   the Data-In Request Byte Count argument minus the Read Data Offset state machine variable; and
  - B)   the maximum size of the DATA information unit.

If this state is entered with a Retry argument then this state shall:

- a)   set the CHANGING DATA POINTER bit in the frame to one;
- b)   set the DATA OFFSET field to Balance Point Read Data Offset state machine variable; and

    c) in the information unit, DATA field set to the information that starts at the location in the specified device server buffer pointed to by the Balance Point Read Data Offset state machine variable and shall contain the amount of data that is the lesser of:
        A) the Data-In Request Byte Count argument minus the Balance Point Read Data Offset state machine variable; and
        B) the maximum size of the DATA information unit.

If a confirmation of Transmission Status (Frame Transmitted) is received, then this state shall set the Read Data Offset state machine variable to the current read data offset plus the number of read data bytes in the transmitted read DATA frame.

If a Transmission Status (ACK Received) confirmation is received, and the Transmit Frame request was for a read DATA frame, then this state shall set the Balance Point Read Data Offset state machine variable to the current balance point read data offset plus the number of read data bytes transmitted in the read DATA frame associated with Transmission Status (ACK Received) confirmation.

9.2.6.3.3.4.2 Transition ST_TTS3:Prepare_Data_In to ST_TTS2:Target_Send_Frame

This transition shall occur after this state:

    a) constructs a read DATA frame; or
    b) receives a Cancel message.

This transition shall include the received Transmission Status, if any, as an argument and the:

    a) read DATA frame as an argument; or
    b) if a Cancel message was received, then a Cancel argument.

9.2.6.3.3.5 ST_TTS4:Prepare_Xfer_Rdy state

9.2.6.3.3.5.1 State description

This state shall construct an XFER_RDY frame using the Data-Out state machine arguments:

    a) FRAME TYPE field set to 05h (i.e., XFER_RDY frame);
    b) HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Destination SAS Data-Out address argument;
    c) HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
    d) RETRY DATA FRAMES bit set to the value of the TRANSPORT LAYER RETRIES bit in the Protocol-Specific Logical Unit mode page (see 10.2.7.3);
    e) set the RETRANSMIT bit set to zero;
    f) CHANGING DATA POINTER bit set to zero;
    g) NUMBER OF FILL BYTES field set to zero;
    h) TAG field set to the Tag Data-Out argument;
    i) TARGET PORT TRANSFER TAG field set to a vendor-specific value, unless the TRANSPORT LAYER RETRIES bit is set to one in the Protocol-Specific Logical Unit mode page, in which case the TARGET PORT TRANSFER TAG field is set to a value that is different from:
        A) the target port transfer tag in the previous XFER_RDY frame associated with the Data-Out arguments; and
        B) any other target port transfer tag currently in use.

        If write data is received for a subsequent XFER_RDY frame for a command, then all target port transfer tags used for previous XFER_RDY frames for the command are no longer in use;

    j) DATA OFFSET field set to zero; and
    k) in the information unit, REQUESTED OFFSET field set to the Requested Write Data Offset state machine variable;
    l) in the information unit, WRITE DATA LENGTH field set as specified in this subclause; and
    m) no fill bytes.

If the SSP target port has the resources available to receive all of the write data as indicated by the Requested Write Data Length state machine variable, then this state shall set the WRITE DATA LENGTH field in the XFER_RDY information unit to the Requested Write Data Length state machine variable.

If the SSP target port does not have the resources available to receive all of the write data as indicated by the Requested Write Data Length state machine variable (e.g., the SSP target port has a vender specific limit as to

how much write data may be received during one operation), then this state shall set the WRITE DATA LENGTH field in the XFER_RDY information unit and the Requested Write Data Length state machine variable to a value representing the amount of write data for which the SSP target port has available resources to receive.

9.2.6.3.3.5.2 Transition ST_TTS4:Prepare_Xfer_Rdy to ST_TTS2:Target_Send_Frame

This transition shall occur after this state:

   a) constructs an XFER_RDY frame; or
   b) receives a Cancel message.

This transition shall include the:

   a) if a Cancel message was received, then a Cancel argument; or
   b) XFER_RDY frame as an argument.

9.2.6.3.3.6 ST_TTS5:Receive_Data_Out state

9.2.6.3.3.6.1 State description

On entry into this state the Write Data Received variable is set to the Requested Write Data Offset state machine variable.

If this state receives a Data-Out Arrived message, then this state shall verify the write DATA frame received with the Data-Out Arrived values as specified in table 141. If the verification test fails, then this state sends the message specified in table 141 to the ST_TFR state machine.

Table 141 — Reception Complete message for write DATA frame verification failures

Message sent to ST_TFR [a]	Condition
Reception Complete (Data Offset Error)	Transport layer retries are disabled, and the DATA OFFSET field is not equal to the Write Data Received variable.
	The DATA OFFSET field is: a) less than the Requested Write Data Offset state machine variable; or b) greater than or equal to the Requested Write Data Offset state machine variable plus the Requested Write Data Length state machine variable.
Reception Complete (Too Much Write Data)	The number of bytes in the DATA field plus the Write Data Received variable is greater than the Request Byte Count Data-Out state machine argument.
Reception Complete (Information Unit Too Short)	The number of bytes in the DATA field is zero.

[a] If more than one condition is true, then this state shall select which message to send to the ST_TFR state machine using the following order:
   1) Reception Complete (Data Offset Error);
   2) Reception Complete (Too Much Write Data); or
   3) Reception Complete (Information Unit Too Short).	

If:

   a) transport layer retries are enabled;
   b) the CHANGING DATA POINTER bit is set to zero; and
   c) the value in the DATA OFFSET field is not equal to the Write Data Received variable,

then this state should discard all Data-Out Arrived messages until the CHANGING DATA POINTER bit is set to one. This state shall resume processing additional Data-Out Arrived messages when it receives a Data-Out Arrived message with the CHANGING DATA POINTER bit set to one.

If the WRITE data frame verification is successful and the Data-Out Arrived message in not discarded, then this state shall:

   a) set the Write Data Received variable to the current Write Data Received variable plus the number of bytes received in the DATA field of the write information unit; and

b)   process the write data as indicated in the Data-Out state machine arguments using the Device Server Buffer (e.g., logical block address) to which the write data is to be transferred.

If the WRITE data frame verification is successful and the CHANGING DATA POINTER bit set to one, then this state shall:

a)   set the Write Data Received variable to the Requested Write Data Offset state machine variable; and
b)   process the write data as indicated in the Data-Out state machine arguments using the Device Server Buffer (e.g., logical block address) to which the write data is to be transferred.

If the Initiator Response Timeout timer is implemented, then this state shall initialize and start the Initiator Response Timeout timer:

a)   upon entry into this state; and
b)   when this state receives and verifies the write DATA frame received with the Data-Out Arrived values (i.e., Data-Out data was received and processed).

If the Initiator Response Timeout timer is running, then this state shall stop the timer before transitioning from this state.

If the Initiator Response Timeout timer expires, then this state shall send a Reception Complete (Initiator Response Timeout) message to the ST_TFR state machine.

If Write Data Received variable equals the Request Byte Count Data-Out state machine argument, then this state shall send a Reception Complete (Data-Out Received) message to the ST_TFR state machine after a Reception Complete (ACK Transmitted) confirmation is received for each write DATA frame previously received.

If this state receives a Cancel message, then this state shall send a Reception Complete (Data Transfer Terminated) message to the ST_TFR state machine.

The Reception Complete message, if any, shall include the tag as an argument.

9.2.6.3.3.6.2 Transition ST_TTS5:Receive_Data_Out to ST_TTS1:Target_Start

This transition shall occur after this state sends a Reception Complete message to the ST_TFR state machine.

9.2.6.3.3.6.3 Transition ST_TTS5:Receive_Data_Out to ST_TTS4:Prepare_Xfer_Rdy

This transition shall occur:

1)   if the Write Data Received variable is less than Request Byte Count Data-Out state machine argument and equal to the Requested Write Data Offset state machine variable plus the Requested Write Data Length state machine variable;
2)   a Reception Complete (ACK Transmitted) confirmation is received for each write DATA frame previously received;
3)   after setting the Requested Write Data Length state machine variable to the Request Byte Count Data-Out state machine argument minus the Requested Write Data Offset state machine variable; and
4)   after setting the Requested Write Data Offset state machine variable to the Write Data Received state machine variable.

9.2.6.3.3.7 ST_TTS6:Prepare_Response state

9.2.6.3.3.7.1 State description

This state shall construct a RESPONSE frame using the received Application Response arguments or the received Transport Response arguments as follows:

a)   FRAME TYPE field set to 07h (i.e., RESPONSE frame);
b)   HASHED DESTINATION SAS ADDRESS field set to the hashed value of the Application Response or Transport Response destination SAS address argument;
c)   HASHED SOURCE SAS ADDRESS field set to the hashed value of the SSP target port's SAS address;
d)   RETRY DATA FRAMES bit set to zero;
e)   RETRANSMIT bit set to zero;
f)   CHANGING DATA POINTER bit set to zero;
g)   TAG field set to the Tag Application Response argument or the Tag Transport Response argument;
h)   TARGET PORT TRANSFER TAG field set to zero;
i)   DATA OFFSET field set to zero;

    j) information unit set as specified in this subclause; and

    k) fill bytes, if needed as specified in this subclause.

If this state was entered with the Transport Response arguments, then this state shall set the fields as follows:

    a) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the response data, if any;

    b) in the information unit, set the DATAPRES field to RESPONSE DATA;

    c) in the information unit, set the STATUS field to zero;

    d) in the information unit, set the SENSE DATA LENGTH field to zero;

    e) in the information unit, set the RESPONSE DATA LENGTH field to 00000004h;

    f) in the information unit, set the RESPONSE DATA field as specified in table 142;

    g) in the information unit, do not include the SENSE DATA field; and

    h) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the response data, if needed.

Table 142 defines how the RESPONSE DATA field shall be set based on the arguments received with the Request (Send Transport Response) message.

Table 142 — Request argument to RESPONSE frame RESPONSE DATA field mapping

Request argument	RESPONSE frame RESPONSE DATA field
Invalid Frame	INVALID FRAME
Function Complete	TASK MANAGEMENT FUNCTION COMPLETE
Function Succeeded	TASK MANAGEMENT FUNCTION SUCCEEDED
Function Not Supported	TASK MANAGEMENT FUNCTION NOT SUPPORTED
Function Failed	TASK MANAGEMENT FUNCTION FAILED
Incorrect Logical Unit Number	INCORRECT LOGICAL UNIT NUMBER
Overlapped Tag Attempted	OVERLAPPED TAG ATTEMPTED

If this state was entered with the Application Response arguments, then this state shall set the fields as follows:

    a) in the information unit, set the DATAPRES field to SENSE_DATA if sense data is to be included in the information unit or NO_DATA if sense data is not to be included in the information unit;

    b) in the information unit, set the STATUS field to the status;

    c) in the information unit, set the SENSE DATA LENGTH field to the length of the sense data, if any;

    d) in the information unit, set the RESPONSE DATA LENGTH field to zero;

    e) in the information unit, do not include the RESPONSE DATA field;

    f) in the information unit, set the SENSE DATA field to the sense data, if any; and

    g) NUMBER OF FILL BYTES field set to the number of fill bytes, based on the length of the sense data, if needed.

9.2.6.3.3.7.2 Transition ST_TTS6:Prepare_Response to ST_TTS2:Target_Send_Frame

This transition shall occur after this state constructs a RESPONSE frame.

This transition shall include:

    a) the RESPONSE frame; or

    b) if a Cancel message was received, then a Cancel argument.

9.3 STP transport layer

9.3.1 Initial FIS

A SATA device phy transmits a Register - Device to Host FIS after completing the link reset sequence (see G.5 for exceptions to this). The expander device shall update a set of shadow registers with the contents of this FIS and shall not deliver it to any STP initiator port. SMP initiator ports may read the shadow register contents using the SMP REPORT PHY SATA function (see 10.4.3.7). The expander device generates a BROADCAST (CHANGE) after receiving the Register - Device to Host FIS (see 7.11).

After the Register - Device to Host FIS is accepted, if the SATA device sends a SATA_X_RDY before an affiliation is established, the expander device shall not send SATA_R_RDY.

9.3.2 BIST Activate FIS

STP initiator ports and STP target ports shall not generate BIST Activate FISes and shall process any BIST Activate FISes received as frames having invalid FIS types (i.e., have the link layer generate SATA_R_ERR in response).

9.3.3 TT (transport layer for STP ports) state machines

The STP transport layer uses the transport layer state machines defined in SATA, modified to communicate with the port layer rather than directly with the link layer. These modifications are not described in this standard.

9.4 SMP transport layer

9.4.1 SMP transport layer overview

Table 143 defines the SMP frame format.

Table 143 — SMP frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE							
1	Frame-type dependent bytes							
	Fill bytes, if needed							
n - 3	(MSB)							
n	CRC							(LSB)

Table 144 defines the SMP FRAME TYPE field, which defines the format of the frame-type dependent bytes.

Table 144 — SMP FRAME TYPE **field**

Code	Name	Frame type	Originator	Reference
40h	SMP_REQUEST	SMP function request	SMP initiator port	9.4.2
41h	SMP_RESPONSE	SMP function response	SMP target port	9.4.3
All others	Reserved.			

The SMP target port in an expander device shall support the SMP_REQUEST and SMP_RESPONSE frames. Other SMP target ports may support these frames.

Fill bytes shall be included after the frame-type dependent bytes so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field contains a CRC value (see 7.5) that is computed over the entire SMP frame prior to the CRC field, and shall begin on a four-byte boundary. The CRC field is checked by the SMP link layer (see 7.18).

9.4.2 SMP_REQUEST frame

The SMP_REQUEST frame is sent by an SMP initiator port to request an SMP function be performed by a management device server. Table 145 defines the SMP_REQUEST frame format.

Table 145 — SMP_REQUEST frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	REQUEST BYTES							
	Fill bytes, if needed							
n - 3	(MSB)							
n	CRC							(LSB)

The SMP FRAME TYPE field shall be set to 40h indicating this is an SMP_REQUEST frame.

The REQUEST BYTES field definition and length is based on the SMP function (see 10.4.3.1). The maximum size of the REQUEST BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL REQUEST BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is defined in 9.4.1.

9.4.3 SMP_RESPONSE frame

The SMP_RESPONSE frame is sent by an SMP target port in response to an SMP_REQUEST frame. Table 146 defines the SMP_RESPONSE frame format.

Table 146 — SMP_RESPONSE frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	RESPONSE BYTES							
	Fill bytes, if needed							
n - 3	(MSB)							
n	CRC							(LSB)

The SMP FRAME TYPE field shall be set to 41h indicating this is an SMP_RESPONSE frame.

The RESPONSE BYTES field definition and length is based on the SMP function (see 10.4.3.2). The maximum size of the RESPONSE BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL RESPONSE BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is defined in 9.4.1.

9.4.4 Sequence of SMP frames

Inside an SMP connection, the source phy transmits a single SMP_REQUEST frame and the destination phy replies with a single SMP_RESPONSE frame.

Figure 177 shows the sequence of SMP frames.
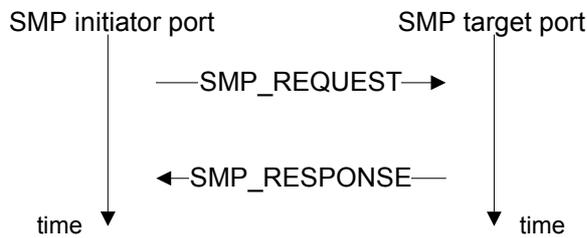
Figure 177 — Sequence of SMP frames

9.4.5 MT (transport layer for SMP ports) state machines

9.4.5.1 SMP transport layer state machines overview

The SMP transport layer contains state machines that process requests from the management application layer and returns confirmations to the management application layer. The SMP transport state machines are as follows:

   a) MT_IP (transport layer for SMP initiator ports) state machine (see 9.4.5.2); and
   b) MT_TP (transport layer for SMP target ports) state machine (see 9.4.5.3).

9.4.5.2 MT_IP (transport layer for SMP initiator ports) state machine

9.4.5.2.1 MT_IP state machine overview

The MT_IP state machine processes requests from the management application layer. These management requests are sent to the port layer and the resulting SMP frame or error condition is sent to the management application layer as a confirmation.

This state machine consists of the following states:

   a) MT_IP1:Idle (see 9.4.5.2.2)(initial state);
   b) MT_IP2:Send (see 9.4.5.2.3); and
   c) MT_IP3:Receive (see 9.4.5.2.4).

This state machine shall start in the MT_IP1:Idle state.

The MT_IP state machine shall maintain the timers listed in table 147.

Table 147 — MT_IP timers

Timer	Initial value
SMP Frame Receive Timeout timer	Vendor specific

Figure 178 describes the MT_IP state machine.
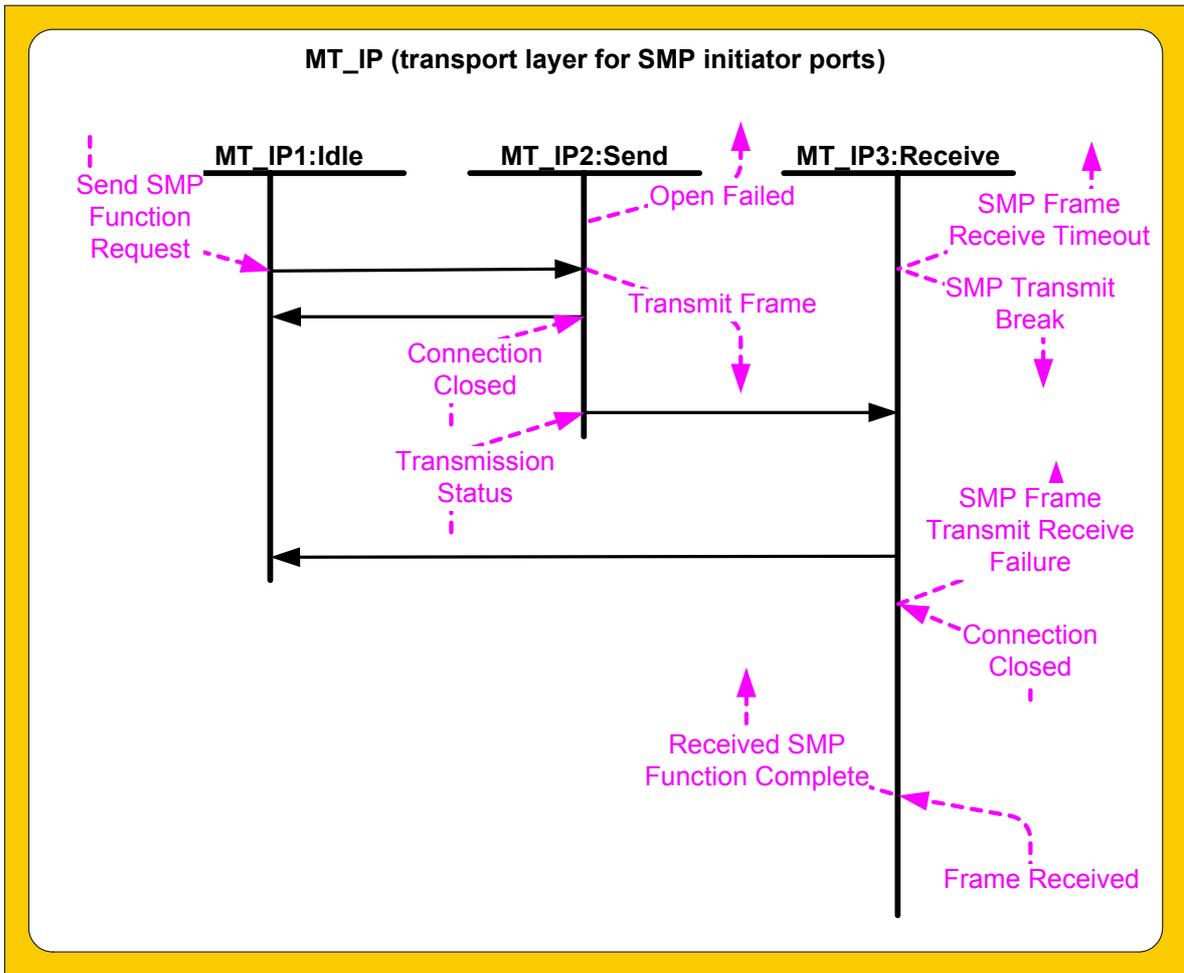
Figure 178 — MT_IP (transport layer for SMP initiator ports) state machine

9.4.5.2.2 MT_IP1:Idle state

9.4.5.2.2.1 State description

This state is the initial state of the MT_IP state machine.

This state waits for a Send SMP Function Request request, which includes the following arguments:

    a) connection rate;
    b) destination SAS address; and
    c) request bytes.

9.4.5.2.2.2 Transition MT_IP1:Idle to MT_IP2:Send

This transition shall occur after a Send SMP Function Request request is received. This transition shall include the following arguments:

    a) connection rate;
    b) destination SAS address; and
    c) request bytes.

9.4.5.2.3 MT_IP2:Send state

9.4.5.2.3.1 State description

This state constructs an SMP_REQUEST frame using the following arguments received in the transition into this state:

    a)   request bytes;

and sends a Transmit Frame request to the port layer with the following arguments:

    a)   initiator port bit set to one;
    b)   protocol set to SMP;
    c)   connection rate;
    d)   initiator connection tag set to FFFFh;
    e)   destination SAS address;
    f)   source SAS address set to the SAS address of the SMP initiator port; and
    g)   request bytes.

9.4.5.2.3.2 Transition MT_IP2:Send to MT_IP1:Idle

This transition shall occur after receiving either a Connection Closed confirmation or a Transmission Status confirmation other than a Transmission Status (Frame Transmitted) confirmation, and after sending an Open Failed confirmation to the management application layer.

9.4.5.2.3.3 Transition MT_IP2:Send to MT_IP3:Receive

This transition shall occur after receiving a Transmission Status (Frame Transmitted) confirmation.

9.4.5.2.4 MT_IP3:Receive state

9.4.5.2.4.1 State description

This state waits for a confirmation from the port layer that either an SMP frame has been received or a failure occurred.

Upon entry into this state, this state shall initialize and start the SMP Frame Receive Timeout timer.

If a Frame Received confirmation is received and the SMP frame type is equal to 41h, this state shall send a Received SMP Function Complete confirmation to the management application layer.

If a Frame Received confirmation is received and the SMP frame type is not equal to 41h, this state shall send a SMP Frame Transmit Receive Failure confirmation to the management application layer.

If a Connection Closed or Frame Received (SMP Failure) confirmation is received, this state shall send an SMP Frame Transmit Receive Failure confirmation to the management application layer.

If the SMP Frame Receive Timeout timer expires before a Received SMP Function Complete confirmation is received, this state shall send an SMP Frame Receive Timeout confirmation to the management application layer and send an SMP Transmit Break request to the port layer.

9.4.5.2.4.2 Transition MT_IP3:Receive to MT_IP1:Idle

This transition shall occur after one of the following:

    a)   sending a Received SMP Function Complete confirmation;
    b)   sending an SMP Frame Transmit Receive Failure confirmation; or
    c)   sending an SMP Transmit Break request.

9.4.5.3 MT_TP (transport layer for SMP target ports) state machine

9.4.5.3.1 MT_TP state machine overview

The MT_TP state machine informs the management application layer of the receipt of an SMP frame. Confirmation of the receipt of an SMP frame is sent to the management application layer. The management application layer creates the corresponding SMP_RESPONSE frame and this state sends it to the port layer.

This state machine consists of the following states:

    a)   MT_TP1:Idle (see 9.4.5.3.2)(initial state); and

    b)   MT_TP2:Respond (see 9.4.5.3.3).

This state machine shall start in the MT_TP1:Idle state.

Figure 179 describes the MT_TP state machine.
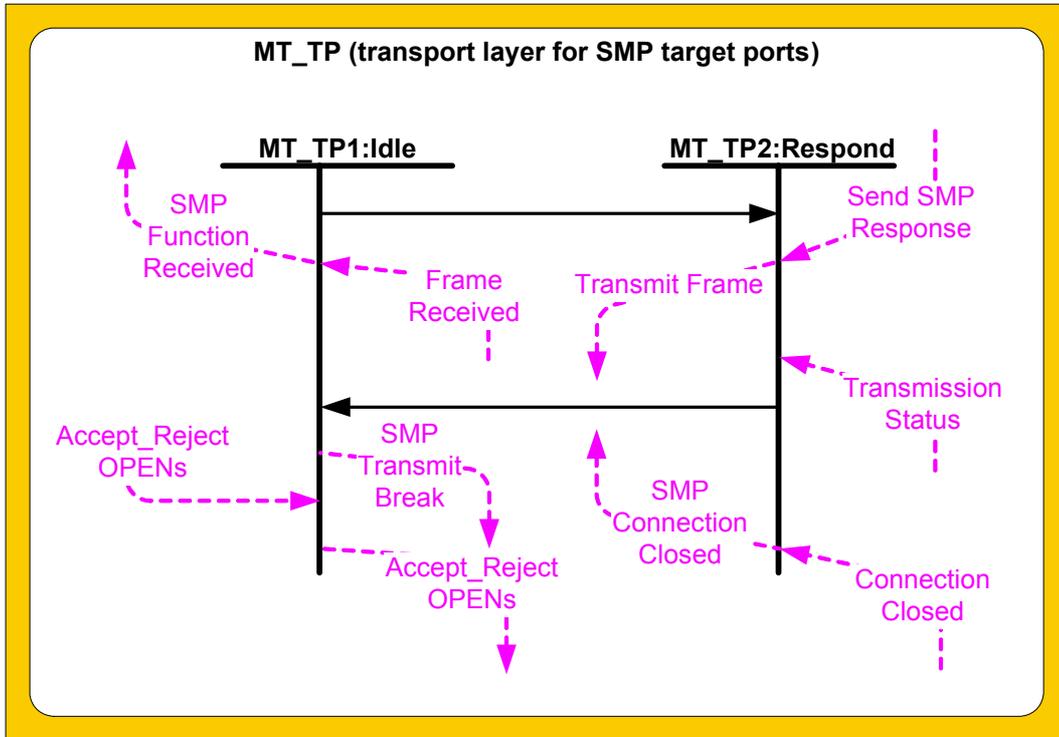
Figure 179 — MT_TP (transport layer for SMP target ports) state machine

9.4.5.3.2 MT_TP1:Idle state

9.4.5.3.2.1 State description

This state is the initial state of the MT_TP state machine.

This state waits for a Frame Received confirmation. If the SMP frame type is not equal to 40h, this state shall discard the frame and send a SMP Transmit Break request to the port layer. Otherwise, this state shall send an SMP Function Received confirmation to the management application layer.

If an Accept_Reject OPENs (Accept SMP) or Accept_Reject OPENs (Reject SMP) request is received, this state shall send an Accept_Reject OPENs request with the same arguments to the port layer.

9.4.5.3.2.2 Transition MT_TP1:Idle to MT_TP2:Respond

This transition shall occur after sending an SMP Function Received confirmation.

9.4.5.3.3 MT_TP2:Respond state

9.4.5.3.3.1 State description

This state waits for a Send SMP Response request, which includes the following arguments:

    a)   response bytes.

After receiving a Send SMP Response request, this state shall construct an SMP_RESPONSE frame using the arguments from the Send SMP Response request and send a Transmit Frame request to the port layer.

If this state receives a Connection Closed confirmation, this state shall send an SMP Connection Closed confirmation to the management application layer.

9.4.5.3.3.2 Transition MT_TP2:Respond to MT_TP1:Idle

This transition shall occur after one of the following:

    a)   receiving a Transmission Status (Frame Transmitted) confirmation; or

    b)   sending an SMP Connection Closed confirmation.

10 Application layer

10.1 Application layer overview

The application layer defines SCSI, ATA, and management specific features.

10.2 SCSI application layer

10.2.1 SCSI transport protocol services

10.2.1.1 SCSI transport protocol services overview

An application client requests the processing of a SCSI command by invoking SCSI transport protocol services, the collective operation of which is conceptually modeled in the following remote procedure call (see SAM-3):

> Service response = Execute Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Task Priority]), OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status))

An application client requests the processing of a SCSI task management function by invoking SCSI transport protocol services, the collective operation of which is conceptually modeled in the following remote procedure calls (see SAM-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 148 describes the mapping of the remote procedure calls to transport protocol services and the SSP implementation of each transport protocol service.

Table 148 — SCSI architecture mapping

Remote procedure call	Type of transport protocol service	Transport protocol service interaction	Transport protocol service	I/T [a]	SSP implementation
Execute Command	Request/ Confirmation	Request	Send SCSI Command	I	COMMAND frame
		Indication	SCSI Command Received	T	Receipt of the COMMAND frame
		Response	Send Command Complete	T	RESPONSE frame
		Confirmation	Command Complete Received	I	Receipt of the RESPONSE frame or problem transmitting the COMMAND frame
	Data-In Transfer [b]	Request	Send Data-In	T	Read DATA frames
		Confirmation	Data-In Delivered	T	Receipt of ACKs for the read DATA frames
	Data-Out Transfer [b]	Request	Receive Data-Out	T	XFER_RDY frame
		Confirmation	Data-Out Received	T	Receipt of write DATA frames
	Terminate Data Transfer [b]	Request	Terminate Data Transfer	T	
		Confirmation	Data Transfer Terminated	T	
ABORT TASK, ABORT TASK SET, CLEAR ACA, CLEAR TASK SET, 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 SSP initiator port (I) or the SSP target port (T) implements the transport protocol service.					
[b] Data transfer transport protocol services for SCSI initiator ports are not specified by SAM-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.

10.2.1.2 Send SCSI Command transport protocol service

An application client uses the Send SCSI Command transport protocol service request to request that an SSP initiator port transmit a COMMAND frame.

Send SCSI Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Task Priority], [Command Reference Number], [First Burst

Enabled]))

Table 149 shows how the arguments to the Send SCSI Command transport protocol service are used.

Table 149 — Send SCSI Command transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to send the COMMAND frame; b) T specifies the target port to which the COMMAND frame is to be sent; c) L specifies the LOGICAL UNIT NUMBER field in the COMMAND frame header; and d) Q specifies the TAG field in the COMMAND frame header.
CDB	Specifies the CDB field in the COMMAND frame.
Task Attribute	Specifies the TASK ATTRIBUTE field in the COMMAND frame.
[Data-In Buffer Size]	Maximum of 2^{32}
[Data-Out Buffer]	Internal to the SSP initiator port.
[Data-Out Buffer Size]	Maximum of 2^{32}
[Task Priority]	Specifies the TASK PRIORITY field in the COMMAND frame.
[First Burst Enabled]	Specifies the ENABLE FIRST BURST field in the COMMAND frame and to cause the SSP initiator port to transmit the number of bytes indicated by the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1.5) for the SCSI target port without waiting for an XFER_RDY frame.

10.2.1.3 SCSI Command Received transport protocol service

An SSP target port uses the SCSI Command Received transport protocol service indication to notify a device server that it has received a COMMAND frame.

SCSI Command Received (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Task Priority], [Command Reference Number], [First Burst Enabled]))

Table 150 shows how the arguments to the SCSI Command Received transport protocol service are determined.

Table 150 — SCSI Command Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that sent the COMMAND frame; b) T indicates the target port that received the COMMAND frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the COMMAND frame header; and d) Q indicates the value of the TAG field in the COMMAND frame header.
CDB	Indicates the value of the CDB field in the COMMAND frame.
Task Attribute	Indicates the value of the TASK ATTRIBUTE field in the COMMAND frame.
[Task Priority]	Indicates the value of the TASK PRIORITY field in the COMMAND frame.
[Command Reference Number]	Ignored
[First Burst Enabled]	Indicates that first burst data is being delivered based on the ENABLE FIRST BURST field in the COMMAND frame and the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see 10.2.7.1.5).

10.2.1.4 Send Command Complete transport protocol service

A device server uses the Send Command Complete transport protocol service response to request that an SSP target port transmit a RESPONSE frame.

Send Command Complete (IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, Service Response))

A device server shall only call Send Command Complete () after receiving SCSI Command Received ().

A device server shall not call Send Command Complete () for a given I_T_L_Q nexus until all its outstanding Receive Data-Out () calls have been responded to with Data-Out Received () and all its outstanding Send Data-In () calls have been responded to with Data-In Delivered ().

Table 151 shows how the arguments to the Send Command Complete transport protocol service are used.

Table 151 — Send Command Complete transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the RESPONSE frame is to be sent; b) T specifies the target port to send the RESPONSE frame; c) L specifies the LOGICAL UNIT NUMBER field in the RESPONSE frame header; and d) Q specifies the TAG field in the RESPONSE frame header.
[Sense Data]	Specifies the SENSE DATA field in the RESPONSE frame.
[Sense Data Length]	Specifies the SENSE DATA LENGTH field in the RESPONSE frame.
Status	Specifies the STATUS field in the RESPONSE frame.
Service Response	Specifies the DATAPRES field and STATUS field in the RESPONSE frame: a) TASK COMPLETE: The DATAPRES field is set to NO_DATA or SENSE_DATA and the STATUS field is set to a value other than INTERMEDIATE or INTERMEDIATE-CONDITION MET; b) LINKED COMMAND COMPLETE: The DATAPRES field is set to NO_DATA or SENSE_DATA and the STATUS field is set to INTERMEDIATE or INTERMEDIATE-CONDITION MET; or c) SERVICE DELIVERY OR TARGET FAILURE: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to INVALID FRAME or OVERLAPPED TAG ATTEMPTED.

10.2.1.5 Command Complete Received transport protocol service

An SSP initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received a response for its COMMAND frame (e.g., a RESPONSE frame or a NAK).

Command Complete Received (IN (I_T_L_Q Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Table 152 shows how the arguments to the Command Complete Received transport protocol service are determined.

Table 152 — Command Complete Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that received the RESPONSE frame; b) T indicates the target port that sent the RESPONSE frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the RESPONSE frame header or COMMAND frame header; and d) Q indicates the value of the TAG field in the RESPONSE frame header or COMMAND frame header.
[Data-In Buffer]	Internal to the SSP initiator port.
[Sense Data]	Indicates the value of the SENSE DATA field in the RESPONSE frame.
[Sense Data Length]	The smaller of the value of the SENSE DATA LENGTH field in the RESPONSE frame and the actual number of sense data bytes received by the SSP initiator port.
Status	Indicates the value of the STATUS field in the RESPONSE frame.
Service Response	From the DATAPRES field and STATUS field in the RESPONSE frame, or from a NAK on the COMMAND frame: a) TASK COMPLETE: The RESPONSE frame contains a DATAPRES field set to NO_DATA or SENSE_DATA and a STATUS field set to a value other than INTERMEDIATE or INTERMEDIATE-CONDITION MET; b) LINKED COMMAND COMPLETE: The RESPONSE frame contains a DATAPRES field set to NO_DATA or SENSE_DATA and a STATUS field set to INTERMEDIATE or INTERMEDIATE-CONDITION MET; or c) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to INVALID FRAME or OVERLAPPED TAG ATTEMPTED, or a NAK was received for the COMMAND frame, or the length of the RESPONSE frame is incorrect.

10.2.1.6 Send Data-In transport protocol service

A device server uses the Send Data-In transport protocol service request to request that an SSP target port transmit a read DATA frame.

> Send Data-In (IN (I_T_L_Q Nexus, Device Server Buffer, Application Client Buffer Offset, Request Byte Count))

A device server shall only call Send Data-In () during a read or bidirectional command.

A device server shall not call Send Data-In () for a given I_T_L_Q nexus after it has called Send Command Complete () for that I_T_L_Q nexus (e.g., a RESPONSE frame with for that I_T_L_Q nexus) or called Task Management Function Executed for a task management function that terminates that task (e.g., an ABORT TASK).

Table 153 shows how the arguments to the Send Data-In transport protocol service are used.

Table 153 — Send Data-In transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the read DATA frame is to be sent; b) T specifies the target port to send the read DATA frame; c) L specifies the LOGICAL UNIT NUMBER field in the read DATA frame header; and d) Q specifies the TAG field in the read DATA frame header.
Device Server Buffer	Internal to the device server.
Application Client Buffer Offset	Specifies the DATA OFFSET field in the read DATA frame.
Request Byte Count	Specifies the size of the read DATA frame.

10.2.1.7 Data-In Delivered transport protocol service

An SSP target port uses the Data-In Delivered transport protocol service indication to notify a device server of the results of transmitting a read DATA frame.

> Data-In Delivered (IN (I_T_L_Q Nexus, Delivery Result))

Table 154 shows how the arguments to the Data-In Delivered transport protocol service are determined.

Table 154 — Data-In Delivered transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port that sent the read DATA frame; b) T indicates the target port that received the read DATA frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the read DATA frame header; and d) Q indicates the value of the TAG field in the read DATA frame header.
Delivery Result	From the response to the outgoing read DATA frame: a) DELIVERY SUCCESSFUL: The read DATA frame received an ACK; or b) DELIVERY FAILURE: The read DATA frame received a NAK or no response.

10.2.1.8 Receive Data-Out transport protocol service

A device server uses the Receive Data-Out transport protocol service request to request that an SSP target port transmit an XFER_RDY frame.

> Receive Data-Out (IN (I_T_L_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device Server Buffer))

A device server shall only call Receive Data-Out () during a write or bidirectional command.

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus until Data-Out Received () has completed successfully for the previous Receive Data-Out () call (i.e., no XFER_RDY frame until all write DATA frames for the previous XFER_RDY frame, if any, and has provided link layer acknowledgement for all of the previous write DATA frames for that I_T_L_Q nexus).

A device server shall not call Receive Data-Out () for a given I_T_L_Q nexus after a Send Command Complete () has been called for that I_T_L_Q nexus or after a Task Management Function Executed () has been called for a task management function that terminates that task (e.g., an ABORT TASK).

Table 155 shows how the arguments to the Receive Data-Out transport protocol service are used.

Table 155 — Receive Data-Out transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I specifies the initiator port to which the XFER_RDY frame is to be sent; b) T specifies the target port to send the XFER_RDY frame; c) L specifies the LOGICAL UNIT NUMBER field in the XFER_RDY frame header; and d) Q specifies the TAG field in the XFER_RDY frame header.
Application Client Buffer Offset	Specifies the REQUESTED OFFSET field in the XFER_RDY frame.
Request Byte Count	Specifies WRITE DATA LENGTH field in the XFER_RDY frame.
Device Server Buffer	Internal to the device server.

10.2.1.9 Data-Out Received transport protocol service

An SSP target port uses the Data-Out Received transport protocol service indication to notify a device server of the result of transmitting an XFER_RDY frame (e.g., receiving write DATA frames in response).

Data-Out Received (IN (I_T_L_Q Nexus, Delivery Result))

Table 156 shows how the arguments to the Data-Out Received transport protocol service are determined.

Table 156 — Data-Out Received transport protocol service arguments

Argument	SAS SSP implementation
I_T_L_Q nexus	I_T_L_Q nexus, where: a) I indicates the initiator port to which the XFER_RDY frame was sent; b) T indicates the target port that sent the XFER_RDY frame; c) L indicates the value of the LOGICAL UNIT NUMBER field in the XFER_RDY frame header; and d) Q indicates the value of the TAG field in the XFER_RDY frame header.
Delivery Result	From the response to the XFER_RDY: a) DELIVERY SUCCESSFUL: The XFER_RDY frame was successfully transmitted and all the write DATA frames for the requested write data were received; or b) DELIVERY FAILURE: The XFER_RDY frame received a NAK or no response.

10.2.1.10 Terminate Data Transfer transport protocol service

A device server uses the Terminate Data Transfer transport protocol service request to request that an SSP target port terminate any Send Data-In () or Receive Data-Out () transport protocol services, if any, being processed using the specified nexus.

Terminate Data Transfer (IN (Nexus))

Table 157 shows how the arguments to the Terminate Data Transfer transport protocol service are used.

Table 157 — Terminate Data Transfer transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus, specifying the scope of the data transfer(s) to terminate.

10.2.1.11 Data Transfer Terminated transport protocol service

An SSP target port uses the Data Transfer Terminated transport protocol service indication to notify a device server that all data transfers for the indicated nexus have been terminated.

Data Transfer Terminated (IN (Nexus))

Table 158 shows how the arguments to the Data Transfer Terminated transport protocol service are determined.

Table 158 — Data Transfer Terminated transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T nexus, I_T_L nexus, or I_T_L_Q nexus indicated by the preceding Terminate Data Transfer () call.

10.2.1.12 Send Task Management Request transport protocol service

An application client uses the Send Task Management Request transport protocol service request to request that an SSP initiator port transmit a TASK frame.

Send Task Management Request (IN (Nexus, Function Identifier, [Association]))

Table 159 shows how the arguments to the Send Task Management Request transport protocol service are used.

Table 159 — Send Task Management Request transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the Function Identifier), where: a) I specifies the initiator port to send the TASK frame; b) T specifies the target port to which the TASK frame is sent; c) L specifies the LOGICAL UNIT NUMBER field in the TASK frame header; and d) Q (for an I_T_L_Q nexus) specifies the TAG OF TASK TO BE MANAGED field in the TASK frame header.
Function Identifier	Specifies the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK (Nexus argument specifies an I_T_L_Q Nexus); b) ABORT TASK SET (Nexus argument specifies an I_T_L Nexus); c) CLEAR ACA (Nexus argument specifies an I_T_L Nexus); d) CLEAR TASK SET (Nexus argument specifies an I_T_L Nexus); e) LOGICAL UNIT RESET (Nexus argument specifies an I_T_L Nexus); and f) QUERY TASK (Nexus argument specifies an I_T_L_Q Nexus).
[Association]	Specifies the TAG field in the TASK frame header.

10.2.1.13 Task Management Request Received transport protocol service

An SSP target port uses the Task Management Request Received transport protocol service indication to notify a task manager that it has received a TASK frame.

Task Management Request Received (IN (Nexus, Function Identifier, [Association]))

Table 160 shows how the arguments to the Task Management Request Received transport protocol service are determined.

Table 160 — Task Management Request Received transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the Function Identifier), where: a) I indicates the initiator port that sent the TASK frame; b) T indicates the target port that received the TASK frame; c) L indicated by the LOGICAL UNIT NUMBER field in the TASK frame header; and d) Q (for an I_T_L_Q nexus) indicated by the TAG OF TASK TO BE MANAGED field in the TASK frame header.
Function Identifier	Indicates the TASK MANAGEMENT FUNCTION field in the TASK frame. Only these task management functions are supported: a) ABORT TASK (Nexus argument specifies an I_T_L_Q Nexus); b) ABORT TASK SET (Nexus argument specifies an I_T_L Nexus); c) CLEAR ACA (Nexus argument specifies an I_T_L Nexus); d) CLEAR TASK SET (Nexus argument specifies an I_T_L Nexus); e) LOGICAL UNIT RESET (Nexus argument specifies an I_T_L Nexus); and f) QUERY TASK (Nexus argument specifies an I_T_L_Q Nexus).
[Association]	Indicates the TAG field in the TASK frame header.

10.2.1.14 Task Management Function Executed transport protocol service

A task manager uses the Task Management Function Executed transport protocol service response to request that an SSP target port transmit a RESPONSE frame.

Task Management Function Executed (IN (Nexus, Service Response, [Association]))

A task manager shall only call Task Management Function Executed () after receiving Task Management Request Received ().

Table 161 shows how the arguments to the Task Management Function Executed transport protocol service are used.

Table 161 — Task Management Function Executed transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the function), where: a) I specifies the initiator port to which the RESPONSE frame is sent; b) T specifies the target port to send the RESPONSE frame; c) L specifies the LOGICAL UNIT NUMBER field in the RESPONSE frame header; and d) Q (for an I_T_L_Q nexus) indirectly specifies the TAG field in the RESPONSE frame header.
Service Response	Specifies the DATAPRES field and RESPONSE CODE field in the RESPONSE frame: a) FUNCTION COMPLETE: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION COMPLETE; b) FUNCTION SUCCEEDED: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION SUCCEEDED; c) FUNCTION REJECTED: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to TASK MANAGEMENT FUNCTION NOT SUPPORTED; d) INCORRECT LOGICAL UNIT NUMBER: The DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to INCORRECT LOGICAL UNIT NUMBER; or e) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame DATAPRES field is set to RESPONSE_DATA and the RESPONSE CODE field is set to: A) INVALID FRAME; B) TASK MANAGEMENT FUNCTION FAILED; or C) OVERLAPPED TAG ATTEMPTED.
[Association]	Specifies the TAG field in the RESPONSE frame header.

10.2.1.15 Received Task Management Function Executed transport protocol service

An SSP initiator port uses the Received Task Management Function Executed transport protocol service confirmation to notify an application client that it has received a response to a TASK frame (e.g., received a RESPONSE frame or a NAK).

Received Task Management Function Executed (IN (Nexus, Service Response, [Association]))

Table 162 shows how the arguments to the Received Task Management Function Executed transport protocol service are determined.

Table 162 — Received Task Management Function Executed transport protocol service arguments

Argument	SAS SSP implementation
Nexus	I_T_L nexus or I_T_L_Q nexus (depending on the function), where: a) I indicates the initiator port that received the RESPONSE frame; b) T indicates the target port that sent the RESPONSE frame; c) L indicates the LOGICAL UNIT NUMBER field in the RESPONSE frame header or TASK frame header; and d) Q (for an I_T_L_Q nexus) indirectly indicates the TAG field in the RESPONSE frame header, or indicates the TAG OF TASK TO BE MANAGED field TASK frame header.

Service Response	Indicates the response to the TASK frame:
	a) FUNCTION COMPLETE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION COMPLETE;
	b) FUNCTION SUCCEEDED: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION SUCCEEDED;
	c) FUNCTION REJECTED: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to TASK MANAGEMENT FUNCTION NOT SUPPORTED;
	d) INCORRECT LOGICAL UNIT NUMBER: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to INCORRECT LOGICAL UNIT NUMBER; or
	e) SERVICE DELIVERY OR TARGET FAILURE: The RESPONSE frame contains a DATAPRES field set to RESPONSE_DATA and a RESPONSE CODE field set to:
	  A) INVALID FRAME;
	  B) TASK MANAGEMENT FUNCTION FAILED; or
	  C) OVERLAPPED TAG ATTEMPTED;
	
	or a NAK was received for the TASK frame, or the length of the RESPONSE frame is incorrect.

[Association]	Indicates the TAG field in the RESPONSE frame header or the TASK frame header.

10.2.2 Application client error handling

If an SSP initiator port calls Command Complete Received () with a Service Response of:

a) Service Delivery or Target Failure - XFER_RDY Information Unit Too Short;
b) Service Delivery or Target Failure - XFER_RDY Information Unit Too Long;
c) Service Delivery or Target Failure - XFER_RDY Incorrect Write Data Length;
d) Service Delivery or Target Failure - XFER_RDY Requested Offset Error;
e) Service Delivery or Target Failure - XFER_RDY Not Expected;
f) Service Delivery or Target Failure - DATA Information Unit Too Short;
g) Service Delivery or Target Failure - DATA Too Much Read Data;
h) Service Delivery or Target Failure - DATA Data Offset Error;
i) Service Delivery or Target Failure - DATA Not Expected;
j) Service Delivery or Target Failure - NAK Received,

then the application client shall abort the command (e.g., by sending an ABORT TASK task management function).

After an application client calls Send SCSI Command (), if Command Complete Received () returns a Service Response of Service Delivery or Target Failure - ACK/NAK Timeout, the application client shall send a QUERY TASK task management function with Send Task Management Request () to determine whether the command was received successfully. If Received Task Management Function Executed () returns a Service Response of FUNCTION SUCCEEDED, the application client shall assume the command was delivered successfully. If Received Task Management Function Executed () returns a Service Response of FUNCTION COMPLETE, and Command Complete Received () has not yet been invoked a second time for the command in question (e.g.,

indicating a RESPONSE frame arrived for the command before the QUERY TASK was processed), the application client shall assume the command was not delivered successfully and may reuse the tag. The application client should call Send SCSI Command () again with identical arguments.

After a Received Task Management Function Executed () call with a Service Response of Service Delivery or Target Failure - ACK/NAK Timeout, an application client should call Send Task Management Request () with identical arguments, including the same tag.

After a Command Complete Received () or Received Task Management Function Executed () call returns a Service Response other than Service Delivery or Target Failure - ACK/NAK Timeout, an application client shall not reuse the tag until it determines the tag is no longer in use by the logical unit (e.g., the ACK for the RESPONSE frame was seen by the SSP target port). Examples of ways the application client may determine that a tag may be used are:

   a) receiving another frame in the same connection;
   b) receiving a DONE (NORMAL) or DONE (CREDIT TIMEOUT) in the same connection; or
   c) receiving a DONE (ACK/NAK TIMEOUT) in the same connection, then running a QUERY TASK task management function to confirm that the tag is no longer active in the logical unit.

10.2.3 Device server error handling

If the SCSI target device performs tag checking and an SSP target port calls SCSI Command Received () with a tag already in use by another SCSI command (i.e., an overlapped command) in any logical unit, the task router and device server(s) shall abort all task management functions received on that I_T nexus and shall respond to the overlapped command as defined in SAM-3.

If an SSP target port calls Data-Out Received () with a Delivery Result set to a value in table 163, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set as indicated in table 163.

Table 163 — Delivery Result to additional sense code mapping

Delivery Result	Additional sense code
DELIVERY FAILURE - DATA OFFSET ERROR	DATA OFFSET ERROR
DELIVERY FAILURE - TOO MUCH WRITE DATA	TOO MUCH WRITE DATA
DELIVERY FAILURE - INFORMATION UNIT TOO SHORT	INFORMATION UNIT TOO SHORT
DELIVERY FAILURE - ACK/NAK TIMEOUT	ACK/NAK TIMEOUT
DELIVERY FAILURE - NAK RECEIVED	NAK RECEIVED
DELIVERY FAILURE - INITIATOR RESPONSE TIMEOUT	INITIATOR RESPONSE TIMEOUT

10.2.4 Task router and task manager error handling

If the SCSI target device performs tag checking and:

   a) an SSP target port calls SCSI Command Received () with a tag already in use by a SCSI task management function in any logical unit; or
   b) an SSP target port calls Task Management Request Received () with a tag already in use by a SCSI command or SCSI task management function in any logical unit,

then the task router and task manager(s) shall:

   a) abort all commands received on that I_T nexus;
   b) abort all task management functions received on that I_T nexus; and
   c) call Task Management Function Executed () with the Service Response set to FUNCTION REJECTED - Overlapped Tag Attempted (i.e., requesting that the target port set the DATAPRES field to RESPONSE_DATA and the RESPONSE CODE field set to OVERLAPPED TAG ATTEMPTED).

10.2.5 SCSI transport protocol event notifications

Table 164 lists the SCSI transport protocol event notifications supported by this standard.

Table 164 — SCSI transport protocol events

Event notification	SAS SSP implementation
Transport Reset	Receipt of a hard reset sequence (see 4.4.2)
Nexus Loss	Receipt of specific OPEN_REJECTs for a specific time period (see 4.5).

10.2.6 SCSI commands

10.2.6.1 INQUIRY command

The vital product data returned by the INQUIRY command (see SPC-3) that shall be returned for a SAS device is described in 10.2.11.

10.2.6.2 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 10.2.8.

10.2.6.3 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 10.2.7.

10.2.6.4 START STOP UNIT command

The power condition states controlled by the START STOP UNIT command (see SBC-2) for a SAS device are described in 10.2.10.

10.2.7 SCSI mode parameters

10.2.7.1 Disconnect-Reconnect mode page

10.2.7.1.1 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 165 defines the parameters which are applicable to SSP. If any field in the Disconnect-Reconnect mode page is not implemented, the value assumed for the functionality of the field shall be zero (i.e., as if the field in the mode page is implemented and the field is set to zero).

The application client sends the values in the fields to be used by the device server to control the SSP connections by means of a MODE SELECT command. The device server shall then communicate the field values to the SSP target port. The field values are communicated from the device server to the SSP target port in a vendor-specific manner.

SAS devices shall only use the parameter fields defined below in this subclause. If any other fields within the Disconnect-Reconnect mode page of the MODE SELECT command contain a non-zero value, the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 165 — Disconnect-Reconnect mode page for SSP

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (02h)					
1	PAGE LENGTH (0Eh)							
2	Reserved							
3	Reserved							
4	(MSB)	BUS INACTIVITY TIME LIMIT						
5								(LSB)
6		Reserved						
7								
8	(MSB)	MAXIMUM CONNECT TIME LIMIT						
9								(LSB)
10	(MSB)	MAXIMUM BURST SIZE						
11								(LSB)
12	Reserved							
13	Reserved							
14	(MSB)	FIRST BURST SIZE						
15								(LSB)

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-3.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to the short format mode page.

The PAGE CODE (PS) field shall be set to 02h.

The PAGE LENGTH field shall be set to 0Eh.

The BUS INACTIVITY TIME LIMIT field is defined in 10.2.7.1.2.

The MAXIMUM CONNECT TIME LIMIT field is defined in 10.2.7.1.3.

The MAXIMUM BURST SIZE field is defined in 10.2.7.1.4.

The FIRST BURST SIZE field is defined in 10.2.7.1.5.

10.2.7.1.2 BUS INACTIVITY TIME LIMIT **field**

The value in the BUS INACTIVITY TIME LIMIT field contains the maximum period that an SSP target port is permitted to maintain a connection (see 4.1.10) without transferring a frame to the SSP initiator port. This value shall be the number of 100 μs increments between frames that the SSP target port transmits during a connection. When this number is exceeded, the SSP target port shall prepare to close the connection (i.e., by requesting to have the link layer transmit DONE). This value may be rounded as defined in SPC-3. A value of zero in this field shall specify that there is no bus inactivity time limit. The bus inactivity time limit is enforced by the port layer (see 8.2.3).

10.2.7.1.3 MAXIMUM CONNECT TIME LIMIT **field**

The value in the MAXIMUM CONNECT TIME LIMIT field contains the maximum duration of a connection (see 4.1.10). This value shall be the number of 100 µs increments that an SSP target port transmits during a connection after which the SSP target port shall prepare to close the connection (e.g., a value of one in this field means that the time is less than or equal to 100 µs and a value of two in this field means that the time is less than or equal to 200 µs). If an SSP target port is transferring a frame when the maximum connection time limit is exceeded, the SSP target port shall complete transfer of the frame before preparing to close the connection. This value may be rounded as defined in SPC-3. A value of zero in this field shall specify that there is no maximum connection time limit. The maximum connection time limit is enforced by the port layer (see 8.2.3).

10.2.7.1.4 MAXIMUM BURST SIZE **field**

For read data, the value in the MAXIMUM BURST SIZE field contains the maximum amount of data that is transferred during a connection by an SSP target port per I_T_L_Q nexus without transferring at least one frame for a different I_T_L_Q nexus. If the SSP target port:

   a) has read data to transfer for only one I_T_L_Q nexus, and
   b) has no requests to transfer write data for any I_T_L_Q nexus;

then the SSP target port shall prepare to close the connection after the amount of data specified by the MAXIMUM BURST SIZE field is transferred to the SSP initiator port.

For write data, the value shall specify the maximum amount of data that an SSP target port requests via a single XFER_RDY frame (see 9.2.2.3).

This value shall be specified in 512-byte increments (e.g., a value of one in this field means that the number of bytes transferred to the SSP initiator port for the nexus is less than or equal to 512 and a value of two in this field means that the number of bytes transferred to the SSP initiator port for the nexus is less than or equal to 1 024). A value of zero in this field shall specify that there is no maximum burst size.

In terms of the SCSI transport protocol services (see 10.2.1), the device server shall limit the Request Byte Count argument to the Receive Data-Out () protocol service and the Send Data-In () protocol service to the amount specified in this field.

10.2.7.1.5 FIRST BURST SIZE **field**

If the ENABLE FIRST BURST field in the COMMAND frame is set to zero, the FIRST BURST SIZE field is ignored.

If the ENABLE FIRST BURST field in the COMMAND frame is set to one, the value in the FIRST BURST SIZE field contains the maximum amount of write data in 512-byte increments that may be sent by the SSP initiator port to the SSP target port without having to receive an XFER_RDY frame (see 9.2.2.3) from the SSP target port (e.g., a value of one in this field means that the number of bytes transferred by the SSP initiator port is less than or equal to 512 and a value of two in this field means that the number of bytes transferred by the SSP initiator port is less than or equal to 1 024).

Specifying a non-zero value in the FIRST BURST SIZE field is equivalent to an implicit XFER_RDY frame for each command requiring write data where the WRITE DATA LENGTH field of the XFER_RDY frame is set to 512 times the value of the FIRST BURST SIZE field.

The rules for data transferred using the value in the FIRST BURST SIZE field are the same as those used for data transferred for an XFER_RDY frame (i.e., the number of bytes transferred using the value in the FIRST BURST SIZE field is as if that number of bytes was requested by an XFER_RDY frame).

If the amount of data to be transferred for the command is less than the amount of data specified by the FIRST BURST SIZE field, the SSP target port shall not transmit an XFER_RDY frame for the command. If the amount of data to be transferred for the command is greater than the amount of data specified by the FIRST BURST SIZE field, the SSP target port shall transmit an XFER_RDY frame after it has received all of the data specified by the FIRST BURST SIZE field from the SSP initiator port. All data for the command is not required to be transferred during the same connection in which the command is transferred.

A value of zero in this field shall specify that there is no first burst size (i.e., an SSP initiator port shall transmit no write DATA frames to the SSP target port before receiving an XFER_RDY frame).

The first burst size is handled by the SCSI transport protocol services (see 10.2.1) and the SSP transport layer (see 9.2.6).

10.2.7.2 Protocol-Specific Port mode page

10.2.7.2.1 Protocol-Specific Port mode page overview

The Protocol-Specific Port mode page (see SPC-3) contains parameters that affect SSP target port operation. If the mode page is implemented, all logical units in SCSI target devices in SAS domains supporting the MODE SELECT or MODE SENSE commands shall implement the page.

If a SAS target device has multiple SSP target ports, changes in the short page parameters for one SSP target port should not affect other SSP target ports.

Table 166 defines the subpages of this mode page.

Table 166 — Protocol-Specific Port mode page subpages

Subpage	Description	Reference
Short page	Short format	10.2.7.2.2
Long page 00h	Not allowed	
Long page 01h	Phy Control And Discover subpage	10.2.7.2.3
Long page E0h - FEh	Vendor specific	
Long page FFh	Return all subpages for the Protocol-Specific Port mode page	SPC-3
All others	Reserved	

10.2.7.2.2 Protocol-Specific Port mode page - short format

The mode page policy (see SPC-3) for the Protocol-Specific mode page short format subpage shall be either shared or per target port. If a SAS target device has multiple SSP target ports, the mode page policy should be per target port.

Parameters in this page shall affect all phys in the SSP target port if the mode page policy is per target port, and shall affect all SSP target ports in the SAS target device if the mode page policy is shared.

Table 167 defines the format of the page for SAS SSP.

Table 167 — Protocol-Specific Port mode page for SAS SSP - short format

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (19h)					
1	PAGE LENGTH (06h)							
2	Reserved			READY LED MEANING	PROTOCOL IDENTIFIER (6h)			
3	Reserved							
4	(MSB)	I_T NEXUS LOSS TIME						
5								(LSB)
6	(MSB)	INITIATOR RESPONSE TIMEOUT						
7								(LSB)

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-3.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to the short format mode page.

The PAGE CODE field shall be set to 19h.

The PAGE LENGTH field shall be set to 06h.

The READY LED MEANING bit specifies the READY LED signal behavior (see 10.4.1). Regardless of the mode page policy (see SPC-3) for this mode page, the shared mode page policy shall be applied to the READY LED MEANING bit.

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The I_T NEXUS LOSS TIME field contains the time that the SSP target port shall retry connection requests to an SSP initiator port that are rejected with responses indicating the SSP initiator port may no longer be present (see 8.2.2) before recognizing an I_T nexus loss (see 4.5). Table 168 defines the values of the I_T NEXUS LOSS TIME field. If this mode page is not implemented, the I_T nexus loss time is vendor specific. This value is enforced by the port layer (see 8.2.2).

Table 168 — I_T NEXUS LOSS TIME **field**

Code	Description
0000h	Vendor-specific amount of time.
0001h to FFFEh	Time in milliseconds.
FFFFh	The SSP target port shall never recognize an I_T nexus loss (i.e., it shall retry the connection requests forever).

NOTE 60 - If this mode page is implemented, the default value of the I_T NEXUS LOSS TIME field should be non-zero. It is recommended that this value be 2 000 ms.

NOTE 61 - An SSP initiator port should retry connection requests for the time indicated by the I_T NEXUS LOSS field in the Protocol-Specific Port mode page for the SSP target port to which it is trying to establish a connection (see 4.5).

The INITIATOR RESPONSE TIMEOUT field contains the time in milliseconds that the SSP target port shall wait for the receipt of a frame (e.g., a write DATA frame) before aborting the command associated with that frame. An INITIATOR RESPONSE TIMEOUT field value of zero indicates that the SSP target port shall disable the initiator response timeout timer. If this mode page is not implemented, the logical unit shall not implement an initiator response timeout timer. This value is enforced by the transport layer (see 9.2.6.3).

10.2.7.2.3 Protocol-Specific Port mode page - Phy Control And Discover subpage

The Phy Control And Discover subpage contains phy-specific parameters. The mode page policy (see SPC-3) for this subpage shall be shared. Parameters in this subpage shall affect only the referenced phy.

Table 169 defines the format of the subpage for SAS SSP.

Table 169 — Protocol-Specific Port mode page for SAS SSP - Phy Control And Discover subpage

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (19h)					
1	SUBPAGE CODE (01h)							
2	(MSB)	PAGE LENGTH (n - 3)						
3								(LSB)
4	Reserved							
5	Reserved				PROTOCOL IDENTIFIER (6h)			
6	Reserved							
7	NUMBER OF PHYS							
SAS phy mode descriptor list								
8	SAS phy mode descriptor (first)(see table 170)							
...	...							
n	SAS phy mode descriptor (last)(see table 170)							

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-3.

The SUBPAGE FORMAT (SPF) bit shall be set to one to access the long format mode pages.

The PAGE CODE field shall be set to 19h.

The SUBPAGE CODE field shall be set to 01h.

The PAGE LENGTH field shall be set to (4 + (the value of the NUMBER OF PHYS field) × (the length in bytes of the SAS phy mode descriptor)).

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

The NUMBER OF PHYS field contains the number of phys in the SAS target device and indicates the number of SAS phy mode descriptors that follow. This field shall not be changeable with MODE SELECT.

A SAS phy mode descriptor shall be included for each phy in the SAS target device, not just the SAS target port, starting with the lowest numbered phy and ending with the highest numbered phy.

Table 170 defines the SAS phy mode descriptor.

Table 170 — SAS phy mode descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	PHY IDENTIFIER							
2	Reserved							
3								
4	Reserved	ATTACHED DEVICE TYPE			Reserved			
5	Reserved				NEGOTIATED PHYSICAL LINK RATE			
6	Reserved				ATTACHED SSP INITIATOR PORT	ATTACHED STP INITIATOR PORT	ATTACHED SMP INITIATOR PORT	Reserved
7	Reserved				ATTACHED SSP TARGET PORT	ATTACHED STP TARGET PORT	ATTACHED SMP TARGET PORT	Reserved
8	SAS ADDRESS							
15								
16	ATTACHED SAS ADDRESS							
23								
24	ATTACHED PHY IDENTIFIER							
25	Reserved							
31								
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
34	Reserved							
41								
42	Vendor specific							
43								
44	Reserved							
47								

The PHY IDENTIFIER field, ATTACHED DEVICE TYPE field, NEGOTIATED PHYSICAL LINK RATE field, ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, ATTACHED SMP TARGET PORT bit, SAS ADDRESS field, ATTACHED SAS ADDRESS field, ATTACHED PHY IDENTIFIER, HARDWARE MINIMUM PHYSICAL LINK RATE field, and HARDWARE MAXIMUM PHYSICAL LINK RATE field are defined in the SMP DISCOVER function (see 10.4.3.5). These fields shall not be changeable with MODE SELECT.

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field are defined in the SMP PHY CONTROL function (see 10.4.3.10).

10.2.7.3 Protocol-Specific Logical Unit mode page

10.2.7.3.1 Protocol-Specific Logical Unit mode page overview

The Protocol-Specific Logical Unit mode page (see SPC-3) contains parameters that affect SSP target port operation on behalf of the logical unit.

Table 171 defines the subpages of this mode page.

Table 171 — Protocol-Specific Logical Unit mode page subpages

Subpage	Description	Reference
Short page	Short format	10.2.7.3.2
Long page 00h	Not allowed	
Long page E0h - FEh	Vendor specific	
Long page FFh	Return all subpages for the Protocol-Specific Logical Unit mode page	SPC-3
All others	Reserved	

10.2.7.3.2 Protocol-Specific Logical Unit mode page - short format

The mode page policy (see SPC-3) for the Protocol-Specific Logical Unit mode page short format subpage shall be either shared or per target port. If a SAS target device has multiple SSP target ports, the mode page policy should be per target port. Parameters in this page shall affect all phys in the SSP target port if the mode page policy is per target port, and shall affect all SSP target ports in the SAS target device if the mode page policy is shared.

Table 172 defines the format of the page for SAS SSP.

Table 172 — Protocol-Specific Logical Unit mode page for SAS SSP - short format

Byte\Bit	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (18h)					
1	PAGE LENGTH (06h)							
2	Reserved			TRANSPORT LAYER RETRIES	PROTOCOL IDENTIFIER (6h)			
3	Reserved							
4	Reserved							
7								

The PARAMETERS SAVEABLE (PS) bit is defined in SPC-3.

The SUBPAGE FORMAT (SPF) bit shall be set to zero for access to the short format mode page.

The PAGE CODE field shall be set to 18h.

The PAGE LENGTH field shall be set to 06h.

The PROTOCOL IDENTIFIER field shall be set to 6h indicating this is a SAS SSP specific mode page.

A TRANSPORT LAYER RETRIES bit set to one specifies that the target port shall support transport layer retries for XFER_RDY and DATA frames for the logical unit as described in 9.2.4 (i.e., transport layer retries are enabled). A TRANSPORT LAYER RETRIES bit set to zero specifies that transport layer retries shall not be used (i.e., transport layer retries are disabled).

10.2.8 SCSI log parameters

10.2.8.1 Protocol-Specific Port log page

The Protocol-Specific Port log page for SAS defined in table 173 is used to report errors that have occurred on the SAS target device's phy(s).

Table 173 — Protocol-Specific Port log page for SAS

Byte\Bit	7	6	5	4	3	2	1	0
0	Reserved		PAGE CODE (18h)					
1	Reserved							
2	(MSB)		PAGE LENGTH (m - 3)					
3								(LSB)
Protocol-specific port log parameter list								
4	Protocol-specific port log parameter (first)(see table 174)							
...	...							
m	Protocol-specific port log parameter (last)(see table 174)							

The PAGE CODE field shall be set to 18h.

The PAGE LENGTH field shall be set to the total length in bytes of the log parameter list.

Table 174 defines the format for the Protocol-Specific Port log parameter for SAS.

Table 174 — Protocol-Specific Port log parameter for SAS

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)							
1			PARAMETER CODE (relative target port identifier)					(LSB)
2	Parameter control byte							
	DU	DS	TSD	ETC	TMC		LBIN	LP
3	PARAMETER LENGTH (y - 3)							
4	Reserved				PROTOCOL IDENTIFIER (6h)			
5	Reserved							
6								
7	NUMBER OF PHYS							
	SAS phy log descriptor list							
8	SAS phy log descriptor (first)(see table 176)							
	...							
y	SAS phy log descriptor (last)(see table 176)							

The PARAMETER CODE field contains the relative target port identifier (see SPC-3) of the SSP target port that the log parameter describes.

Table 175 defines the values of the fields in the parameter control byte for the log parameter.

Table 175 — Parameter control byte in the Protocol-Specific Port log parameter for SAS

Field	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 the ETC bit is set to 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 phys in the SAS target port (not in the entire SAS target device) and indicates the number of SAS phy log descriptors that follow.

Table 176 defines the SAS phy log descriptor. Each SAS phy log descriptor is the same length.

Table 176 — SAS phy log descriptor

Byte\Bit	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 PORT	ATTACHED STP INITIATOR PORT	ATTACHED SMP INITIATOR PORT	Reserved
7	Reserved				ATTACHED SSP TARGET PORT	ATTACHED STP TARGET PORT	ATTACHED SMP TARGET PORT	Reserved
8	SAS ADDRESS							
15								
16	ATTACHED SAS ADDRESS							
23								
24	ATTACHED PHY IDENTIFIER							
25	Reserved							
31								
32	(MSB)			INVALID DWORD COUNT				
35								(LSB)
36	(MSB)			RUNNING DISPARITY ERROR COUNT				
39								(LSB)
40	(MSB)			LOSS OF DWORD SYNCHRONIZATION				
43								(LSB)
44	(MSB)			PHY RESET PROBLEM				
47								(LSB)

The PHY IDENTIFIER field, ATTACHED DEVICE TYPE field, NEGOTIATED PHYSICAL LINK RATE field, ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, ATTACHED SMP TARGET PORT bit, SAS ADDRESS field, ATTACHED SAS ADDRESS field, and ATTACHED PHY IDENTIFIER field are defined in the SMP DISCOVER function (see 10.4.3.5).

The INVALID DWORD COUNT field, RUNNING DISPARITY ERROR COUNT field, LOSS OF DWORD SYNCHRONIZATION field, and PHY RESET PROBLEM COUNT field are each defined in the SMP REPORT PHY ERROR LOG response data (see 10.4.3.6).

10.2.9 SCSI diagnostic parameters

10.2.9.1 Protocol-Specific diagnostic page

The Protocol-Specific diagnostic page for SAS provides a method for an application client to enable and disable phy test functions (see 4.8) for selected phys. The diagnostic page format is specified in SPC-3.

The Protocol-Specific diagnostic page is transmitted using the SEND DIAGNOSTIC command. If the device server receives a RECEIVE DIAGNOSTIC RESULTS command with the PAGE CODE field set to 3Fh, it shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. Table 177 defines the Protocol-Specific diagnostic page for SAS.

Table 177 — Protocol-Specific diagnostic page for SAS

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (3Fh)							
1	Reserved				PROTOCOL IDENTIFIER (6h)			
2	(MSB)							
3	PAGE LENGTH (001Ch)							(LSB)
4	PHY IDENTIFIER							
5	PHY TEST FUNCTION							
6	PHY TEST PATTERN							
7	Reserved				PHY TEST PATTERN PHYSICAL LINK RATE			
8	Reserved							
31								

The PHY IDENTIFIER field specifies the phy identifier (see 4.2.7) of the phy that is to perform or to stop performing a phy test function (i.e., the selected phy). If the PHY IDENTIFIER field specifies a phy that does not exist, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The PHY TEST FUNCTION field specifies the phy test function to be performed and is defined in table 178. If the PHY TEST FUNCTION field specifies a phy test function that is not supported, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 178 — PHY TEST FUNCTION field

Code	Name	Description
00h	STOP	If the selected phy is performing a phy test function, then the selected phy shall stop performing the phy test function and originate a link reset sequence. If the selected phy is not performing a phy test function, then this function has no effect on the selected phy. [a]
01h	TRANSMIT_PATTERN	If the selected phy is not performing a phy test function, the selected phy shall be set to transmit the phy test pattern specified by the PHY TEST PATTERN field at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field and set to ignore its receiver. If the selected phy receives data while transmitting the pattern, then the selected phy shall ignore the received data. If the selected phy is performing a phy test function, the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to PHY TEST FUNCTION IN PROGRESS. [a]
02h - EFh	Reserved	
F0h - FFh	Vendor specific	
[a] If there is no SSP target port available to receive a SEND DIAGNOSTIC command to stop a phy from performing a phy test function, then a power on may be required to cause the phy to stop performing the function and originate a phy reset sequence.		

If the PHY TEST FUNCTION field is set to TRANSMIT_PATTERN (i.e., 01h), then the PHY TEST PATTERN field specifies the phy test pattern to be transmitted as defined by table 179. If the PHY TEST PATTERN field specifies a phy test pattern that is not supported by the specified SAS phy, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 179 — PHY TEST PATTERN field

Code	Name	Description
00h	Reserved	
01h	JTPAT	The selected phy shall continuously transmit the JTPAT for RD+ and RD- (see A.1).
02h	CJTPAT	The selected phy shall continuously transmit the CJTPAT (see A.2).
03h - FFh	Reserved	

The PHY TEST PATTERN PHYSICAL LINK RATE field specifies the physical link rate at which the phy test pattern shall be transmitted and is defined in table 180. If the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field is less than the hardware minimum physical link rate or greater than the hardware maximum physical link rate, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table 180 — PHY TEST PATTERN PHYSICAL LINK RATE **field**

Code	Description
0h - 7h	Reserved
8h	1,5 Gbit/s
9h	3,0 Gbit/s
Ah - Fh	Reserved

10.2.10 SCSI power conditions

10.2.10.1 SCSI power conditions overview

The logical unit power condition states from the Power Condition mode page (see SPC-3) and START STOP UNIT command (see SBC-2), if implemented, shall interact with the NOTIFY (ENABLE SPINUP) primitive (see 7.2.5.9) to control temporary consumption of additional power (e.g., spin-up of rotating media) as described in this subclause.

The logical unit uses NOTIFY (ENABLE SPINUP) to:

a) initiate spin-up after power on; and
b) delay spin-ups requested by START STOP UNIT commands.

10.2.10.2 SA_PC (SCSI application layer power condition) state machine

10.2.10.2.1 SA_PC state machine overview

The SA_PC (SCSI application layer power condition) state machine describes how the SAS target device processes logical unit power condition state change requests and NOTIFY (ENABLE SPINUP) if it is a SCSI target device.

> NOTE 62 - This state machine is an enhanced version of the logical unit power condition state machines described in SPC-3 and SBC-2.

This state machine consists of the following states:

a) SA_PC_0:Powered_On (see 10.2.10.2.2)(initial state);
b) SA_PC_1:Active (see 10.2.10.2.3);
c) SA_PC_2:Idle (see 10.2.10.2.4);
d) SA_PC_3:Standby (see 10.2.10.2.5);
e) SA_PC_4:Stopped (see 10.2.10.2.6)(specific to SBC-2 logical units);
f) SA_PC_5:Active_Wait (see 10.2.10.2.7)(specific to SAS devices); and
g) SA_PC_6:Idle_Wait (see 10.2.10.2.8)(specific to SAS devices).

This state machine shall start in the SA_PC_0:Powered_On state after power on.

If the device server processes a START STOP UNIT command (see SBC-2) with the IMMED bit set to one, it may complete the command before completing the transition, if any, specified by the POWER CONDITION field and the START bit.

Figure 180 describes the SA_PC state machine.
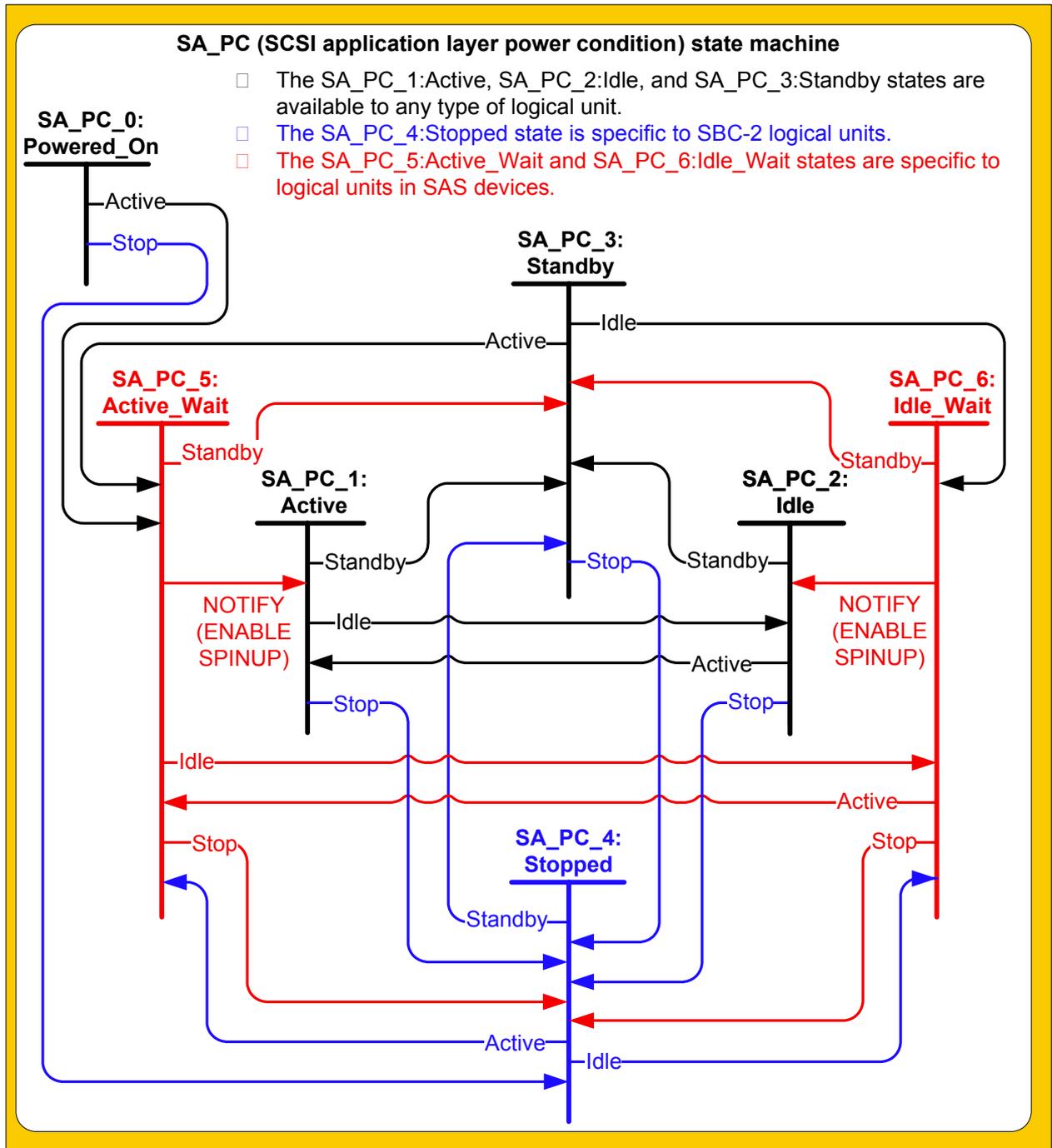
Figure 180 — SA_PC (SCSI application layer power condition) state machine for SAS

10.2.10.2.2 SA_PC_0:Powered_On state

10.2.10.2.2.1 State description

This state shall be entered upon power on. This state consumes zero time.

10.2.10.2.2.2 Transition SA_PC_0:Powered_On to SA_PC_4:Stopped

This transition shall occur if the SAS device has been configured to start in the SA_PC_4:Stopped state.

10.2.10.2.2.3 Transition SA_PC_0:Powered_On to SA_PC_5:Active_Wait

This transition shall occur if the SAS device has been configured to start in the SA_PC_5:Active_Wait state.

10.2.10.2.3 SA_PC_1:Active state

10.2.10.2.3.1 State description

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-3 for more details about this state.

10.2.10.2.3.2 Transition SA_PC_1:Active to SA_PC_2:Idle

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is received; or
c) the Power Condition mode page idle condition timer expires.

10.2.10.2.3.3 Transition SA_PC_1:Active to SA_PC_3:Standby

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is received; or
c) the Power Condition mode page standby condition timer expires.

10.2.10.2.3.4 Transition SA_PC_1:Active to SA_PC_4:Stopped

This transition shall occur if:

a) a START STOP UNIT command with the START bit set to zero is received.

10.2.10.2.4 SA_PC_2:Idle state

10.2.10.2.4.1 State description

While in this state, rotating media in block devices shall be active (i.e., rotating or spinning).

See SPC-3 for more details about this state.

10.2.10.2.4.2 Transition SA_PC_2:Idle to SA_PC_1:Active

This transition shall occur if:

a) a START STOP UNIT command with the START bit set to one is 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.

10.2.10.2.4.3 Transition SA_PC_2:Idle to SA_PC_3:Standby

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is received; or
c) the Power Condition mode page standby condition timer expires.

10.2.10.2.4.4 Transition SA_PC_2:Idle to SA_PC_4:Stopped

This transition shall occur if:

a) a START STOP UNIT command with the START bit set to zero is received.

10.2.10.2.5 SA_PC_3:Standby state

10.2.10.2.5.1 State description

While in this state, rotating media in block devices shall be stopped.

See SPC-3 for more details about this state.

10.2.10.2.5.2 Transition SA_PC_3:Standby to SA_PC_4:Stopped

This transition shall occur if:

    a)   a START STOP UNIT command with the START bit set to zero is received.

10.2.10.2.5.3 Transition SA_PC_3:Standby to SA_PC_5:Active_Wait

This transition shall occur if:

    a)   a START STOP UNIT command with the START bit set to one is 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.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_1:Active state.

10.2.10.2.5.4 Transition SA_PC_3:Standby to SA_PC_6:Idle_Wait

This transition shall occur if:

    a)   a START STOP UNIT command with the POWER CONDITION field set to IDLE is received;
    b)   a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is received; or
    c)   a command is received which requires the idle power condition.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.6 SA_PC_4:Stopped state

10.2.10.2.6.1 State description

This state is only implemented in block devices.

While in this state, rotating media shall be stopped.

See SBC-2 for more details about this state.

10.2.10.2.6.2 Transition SA_PC_4:Stopped to SA_PC_3:Standby

This transition shall occur if:

    a)   a START STOP UNIT command with the POWER CONDITION field set to STANDBY is received; or
    b)   a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is received.

10.2.10.2.6.3 Transition SA_PC_4:Stopped to SA_PC_5:Active_Wait

This transition shall occur if:

    a)   a START STOP UNIT command with the START bit set to one is received; or
    b)   a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is received.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_1:Active state.

10.2.10.2.6.4 Transition SA_PC_4:Stopped to SA_PC_6:Idle_Wait

This transition shall occur if:

    a)   a START STOP UNIT command with the POWER CONDITION field set to IDLE is received; or
    b)   a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is received.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.7 SA_PC_5:Active_Wait state

10.2.10.2.7.1 State description

This state shall only be implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped and the device server is not capable of processing media access commands. Any media access commands received while in this state shall cause the device server to terminate the command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED.

10.2.10.2.7.2 Transition SA_PC_5:Active_Wait to SA_PC_1:Active

This transition shall occur if:

a) a NOTIFY (ENABLE SPINUP) is detected; or
b) the SAS device does not consume additional power as a result of the transition to SA_PC_1:Active.

10.2.10.2.7.3 Transition SA_PC_5:Active_Wait to SA_PC_3:Standby

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is received; or
c) the Power Condition mode page standby condition timer expires.

10.2.10.2.7.4 Transition SA_PC_5:Active_Wait to SA_PC_4:Stopped

This transition shall occur if a START STOP UNIT command with the START bit set to zero is received.

10.2.10.2.7.5 Transition SA_PC_5:Active_Wait to SA_PC_6:Idle_Wait

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to IDLE is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0 is received; or
c) the Power Condition mode page idle condition timer expires.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_2:Idle state.

10.2.10.2.8 SA_PC_6:Idle_Wait state

10.2.10.2.8.1 State description

This state shall only be implemented in SAS devices.

While in this state, rotating media in block devices shall be stopped and the device server is not capable of processing media access commands. Any media access commands received while in this state shall cause the device server to terminate the command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, NOTIFY (ENABLE SPINUP) REQUIRED.

10.2.10.2.8.2 Transition SA_PC_6:Idle_Wait to SA_PC_2:Idle

This transition shall occur if:

a) a NOTIFY (ENABLE SPINUP) is detected; or
b) the SAS device does not consume additional power as a result of the transition to SA_PC_2:Idle.

10.2.10.2.8.3 Transition SA_PC_6:Idle_Wait to SA_PC_3:Standby

This transition shall occur if:

a) a START STOP UNIT command with the POWER CONDITION field set to STANDBY is received;
b) a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0 is received; or
c) the Power Condition mode page standby condition timer expires.

10.2.10.2.8.4 Transition SA_PC_6:Idle_Wait to SA_PC_4:Stopped

This transition shall occur if a START STOP UNIT command with the START bit set to zero is received.

10.2.10.2.8.5 Transition SA_PC_6:Idle_Wait to SA_PC_5:Active_Wait

This transition shall occur if:

    a) a START STOP UNIT command with the POWER CONDITION field set to ACTIVE is received; or

    b) a command is received which requires the active power condition.

If the transition is based on a START STOP UNIT command with the IMMED bit set to zero, the device server shall not complete the command with GOOD status until this state machine reaches the SA_PC_1:Active state.

10.2.11 SCSI vital product data (VPD)

In the Device Identification VPD page (83h) returned by the INQUIRY command (see SPC-3), each logical unit in a SAS target device shall include the identification descriptors for the target port identifier (see 4.2.6) and the relative target port identifier (see SAM-3 and SPC-3) listed in table 181.

Table 181 — Device Identification VPD page identification descriptors for the SAS target port

Field in identification descriptor	Identification descriptor	
	Target port identifier	**Relative target port identifier**
IDENTIFIER TYPE	3h (i.e., NAA)	4h (i.e., relative target port identifier)
ASSOCIATION	01b (i.e., SCSI target port)	01b (i.e., SCSI target port)
CODE SET	1h (i.e., binary)	1h (i.e., binary)
IDENTIFIER LENGTH	8	4
PIV (protocol identifier valid)	1	1
PROTOCOL IDENTIFIER	6h (i.e., SAS)	6h (i.e., SAS)
IDENTIFIER	SAS address [a] in NAA IEEE Registered format	Relative port identifier [b] as described in SAM-3 and SPC-3
[a] The IDENTIFIER field contains the SAS address of the SSP target port through which the INQUIRY command was received. [b] The IDENTIFIER field contains the relative port identifier of the SSP target port through which the INQUIRY command was received.		

In the Device Identification VPD page (83h) returned by the INQUIRY command (see SPC-3), each logical unit in a SAS target device shall include an identification descriptor for the SAS target device name (see 4.2.4) using NAA

format and may include an identification descriptor for the SAS target device name using the SCSI name string format as listed in table 182.

Table 182 — Device Identification VPD page identification descriptors for the SAS target device

Field in identification descriptor	Identification descriptor for SAS target device	
	NAA format (required)	**SCSI name string format (optional)**
IDENTIFIER TYPE	3h (i.e., NAA)	8h (i.e., SCSI name string)
ASSOCIATION	10b (i.e., SCSI target device)	10b (i.e., SCSI target device)
CODE SET	1h (i.e., binary)	3h (i.e., UTF-8)
IDENTIFIER LENGTH	8	24
PIV (protocol identifier valid)	1	1
PROTOCOL IDENTIFIER	6h (i.e., SAS)	6h (i.e., SAS)
IDENTIFIER	SAS address of the SAS target device in NAA IEEE Registered format	SAS address of the SAS target device in SCSI name string format (e.g., "naa." followed by 16 hexadecimal digits followed by 4 ASCII null characters)

Logical units may include identification descriptors in addition to those required by this standard (e.g., SCSI target devices with SCSI target ports using other SCSI transport protocols may return additional target device names for those other SCSI transport protocols).

10.3 ATA application layer

No SAS-specific ATA features are defined.

10.4 Management application layer

10.4.1 READY LED signal behavior

A SAS target device uses the READY LED signal to activate an externally visible LED that indicates the state of readiness and activity of the SAS target device. The READY LED signal electrical characteristics are described in 5.4. All SAS target devices using the SAS Drive plug connector (see 5.2.3.2.1.1) shall support the READY LED signal.

The system is not required to generate any visual output when the READY LED signal is asserted. Additional vendor-specific flashing patterns may be used to signal vendor-specific conditions.

SAS target devices without SSP target ports may transmit the READY LED signal using vendor-specific patterns.

SAS target devices with SSP target ports shall follow the READY LED MEANING bit in the Protocol-Specific Port mode page (see 10.2.7.2) as described in table 183.

Table 183 — READY LED signal behavior

Power condition [a] (see 10.2.10) or activity	READY LED MEANING **bit set to zero** [b]	READY LED MEANING **bit set to one**
Active or Idle power condition	The SAS target device shall: a) when not processing a command, assert the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually on, but flashes on and off when commands are processed.)	The SAS target device shall: a) when not processing a command, negate the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually off, but flashes on and off when commands are processed.)
Standby or Stopped power condition	The SAS target device shall: a) when not processing a command, negate the READY LED signal continuously; and b) when processing a command, toggle the READY LED signal between the asserted and negated states in a vendor-specified manner. (i.e., The LED is usually off, but flashes on and off when commands are processed.) After a vendor-specific amount of time in this state, SAS target devices with rotating media may be removed with minimum risk of mechanical or electrical damage.	
Spinup/ spindown	If the SAS target device has rotating media and is in the process of performing a spin-up or spin-down, then the SAS target device shall toggle the READY LED signal between the asserted and negated states with a 1 s ± 0,1 s cycle using a 50 % ± 10 % duty cycle (e.g., the LED is on for 0,5 s and off for 0,5 s).	
Formatting media	If the SAS target device is in the process of formatting media, then the SAS target device shall toggle the READY LED signal between the asserted and negated states in a vendor-specified manner (e.g., with each cylinder change on a disk drive).	

[a] If the SAS target device has more than one logical unit and any logical unit is active or idle, its power condition should be used to control the READY LED signal.
[b] If the target device has rotating media, a READY LED MEANING bit set to zero results in a READY LED signal behavior that provides an indication of the target device's readiness for removal. A target device with rotating media that is not in a state for safe removal shall either toggle the READY LED signal at a significant rate during spin-up, during spin-down, and while formatting media, or assert the READY LED signal continuously. When removal is safe from a mechanical standpoint, the READY LED signal shall be deasserted.

10.4.2 Management protocol services

The management application client and management device server use a four-step process to perform management functions:

1) The management application client invokes Send SMP Function;
2) The SMP target port invokes SMP Function Received;

3) The management device server invokes Send SMP Function Response; and
4) The SMP initiator port invokes Received SMP Function Complete.

10.4.3 SMP functions

10.4.3.1 SMP function request frame format

An SMP request frame is sent by an SMP initiator port to request an SMP function be performed by a management device server. Table 184 defines the SMP request frame format.

Table 184 — SMP request frame format

Byte\Bit	7	6	5	4	3	2	1	0
0				SMP FRAME TYPE (40h)				
1				FUNCTION				
2				Reserved				
3								
4				ADDITIONAL REQUEST BYTES				
m								
				Fill bytes, if needed				
n - 3	(MSB)			CRC				
n								(LSB)

The SMP FRAME TYPE field is included in each frame format defined in this clause, although that field is parsed by the SMP transport layer (see 9.4). The SMP FRAME TYPE field is set to 40h.

The FUNCTION field specifies which SMP function is being requested and is defined in table 185. If the value in the FUNCTION field is not supported by the SMP target port, it shall return a function result of UNKNOWN SMP FUNCTION as described in table 187.

Table 185 — SMP functions (FUNCTION field)

Code	SMP function	Description	Request frame size (in bytes)	Response frame size (in bytes)	Reference
00h	REPORT GENERAL	Return general information about the device	8	32	10.4.3.3
01h	REPORT MANUFACTURER INFORMATION	Return vendor and product identification	8	64	10.4.3.4
02h	READ GPIO REGISTER	See SFF-8485			
03h - 0Fh	Reserved for general SMP input functions				
10h	DISCOVER	Return information about the specified phy	16	56	10.4.3.5
11h	REPORT PHY ERROR LOG	Return error logging information about the specified phy	16	32	10.4.3.6
12h	REPORT PHY SATA	Return information about a phy currently attached to a SATA phy	16	60	10.4.3.7
13h	REPORT ROUTE INFORMATION	Return route table information	16	44	10.4.3.8
14h - 1Fh	Reserved for phy-based SMP input functions				
20h - 3Fh	Reserved for SMP input functions				
40h - 7Fh	Vendor specific				
80h - 81h	Reserved for general SMP output functions				
82h	WRITE GPIO REGISTER	See SFF-8485			
83h - 8Fh	Reserved for general SMP output functions				
90h	CONFIGURE ROUTE INFORMATION	Change route table information	44	8	10.4.3.9
91h	PHY CONTROL	Request actions by the specified phy	44	8	10.4.3.10
92h	PHY TEST FUNCTION	Request a test function by the specified phy	44	8	10.4.3.11
92h - 9Fh	Reserved for phy-based SMP output functions				
A0h - BFh	Reserved for SMP output functions				
C0h - FFh	Vendor specific				

The ADDITIONAL REQUEST BYTES field definition and length are based on the SMP function. The maximum size of the ADDITIONAL REQUEST BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL REQUEST BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is included in each request frame format defined in this clause, although that field is defined by the SMP transport layer (see 9.4.1) and parsed by the SMP link layer (see 7.18).

10.4.3.2 SMP function response frame format

An SMP response frame is sent by an SMP target port in response to an SMP request frame. Table 186 defines the SMP response frame format.

Table 186 — SMP response frame format

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION							
2	FUNCTION RESULT							
3	Reserved							
4	ADDITIONAL RESPONSE BYTES							
m								
	Fill bytes, if needed							
n - 3	(MSB)							
n	CRC							(LSB)

The SMP FRAME TYPE field is included in each frame format defined in this clause, although that field is parsed by the SMP transport layer (see 9.4). The SMP FRAME TYPE field is set to 41h.

The FUNCTION field indicates the SMP function to which this frame is a response, and is defined in table 185 in 10.4.3.1.

The FUNCTION RESULT field is defined in table 187.

Table 187 — FUNCTION RESULT **field (Sheet 1 of 2)**

Code	Name	SMP function(s)	Description
00h	SMP FUNCTION ACCEPTED	All	The SMP target port supports the SMP function. The ADDITIONAL RESPONSE BYTES field contains the requested information.
01h	UNKNOWN SMP FUNCTION	Unknown	The SMP target port does not support the requested SMP function. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
02h	SMP FUNCTION FAILED	All	The SMP target port supports the SMP function, but the requested SMP function failed. The ADDITIONAL RESPONSE BYTES may be present but shall be ignored.
03h	INVALID REQUEST FRAME LENGTH	All	The SMP target port supports the SMP function, but the SMP request frame length was invalid (i.e., did not match the frame size defined for the function). The ADDITIONAL RESPONSE BYTES may be present but shall be ignored.
10h	PHY DOES NOT EXIST	DISCOVER, REPORT PHY ERROR LOG, REPORT PHY SATA, REPORT ROUTE INFORMATION, CONFIGURE ROUTE INFORMATION, PHY CONTROL, PHY TEST FUNCTION	The phy specified by the PHY IDENTIFIER field in the SMP request frame does not exist (e.g., the value is not within the range of zero to the value of the NUMBER OF PHYS field reported in the REPORT GENERAL function). The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
11h	INDEX DOES NOT EXIST	REPORT ROUTE INFORMATION, CONFIGURE ROUTE INFORMATION	The phy specified by the PHY IDENTIFIER field in the SMP request frame does not have the table routing attribute (see 4.6.7.1), or the expander route index specified by the EXPANDER ROUTE INDEX field does not exist (i.e., the value is not in the range of 0000h to the value of the EXPANDER ROUTE INDEXES field in the REPORT GENERAL function). The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.

Table 187 — FUNCTION RESULT **field (Sheet 2 of 2)**

Code	Name	SMP function(s)	Description
12h	PHY DOES NOT SUPPORT SATA	REPORT PHY SATA and PHY CONTROL (TRANSMIT SATA PORT SELECTION SIGNAL)	The phy specified by the PHY IDENTIFIER field in the SMP request frame is not part of an STP target port. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
13h	UNKNOWN PHY OPERATION	PHY CONTROL	The operation specified by the PHY OPERATION field in the SMP request frame is unknown. The SMP function had no affect. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
14h	UNKNOWN PHY TEST FUNCTION	PHY TEST FUNCTION	The operation specified by the PHY TEST FUNCTION field in the SMP request frame is unknown. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
15h	PHY TEST FUNCTION IN PROGRESS	PHY TEST FUNCTION	The specified phy is already performing a phy test function. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
16h	PHY VACANT	DISCOVER, REPORT PHY ERROR LOG, REPORT PHY SATA, REPORT ROUTE INFORMATION, CONFIGURE ROUTE INFORMATION, PHY CONTROL	The SMP target port processing the SMP request frame does not have access to the phy, although the value is within the range of zero to the value of the NUMBER OF PHYS field reported in the REPORT GENERAL function. The ADDITIONAL RESPONSE BYTES field may be present but shall be ignored.
All others	Reserved		

The ADDITIONAL RESPONSE BYTES field definition depends on the SMP function requested. The maximum size of the ADDITIONAL RESPONSE BYTES field is 1 024 bytes, making the maximum size of the frame 1 032 bytes (i.e., 1 024 bytes of data + 4 bytes of header + 4 bytes of CRC).

Fill bytes shall be included after the ADDITIONAL RESPONSE BYTES field so the CRC field is aligned on a four byte boundary. The contents of the fill bytes are vendor specific.

The CRC field is included in each response frame format defined in this clause, although that field is defined by the SMP transport layer (see 9.4.1) and parsed by the SMP link layer (see 7.18).

10.4.3.3 REPORT GENERAL function

The REPORT GENERAL function returns general information about the SAS device (e.g., a SAS device contained in an expander device). This SMP function shall be implemented by all SMP target ports.

Table 188 defines the request format.

Table 188 — REPORT GENERAL request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (00h)							
2	Reserved							
3								
4	(MSB)							
				CRC				
7								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 00h.

The CRC field is defined in 10.4.3.1.

Table 189 defines the response format.

Table 189 — REPORT GENERAL response

Byte\Bit	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						CONFIGURING	CONFIGURABLE ROUTE TABLE
11	Reserved							
12	ENCLOSURE LOGICAL IDENTIFIER							
19								
20	Reserved							
27								
28	(MSB)			CRC				
31								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 00h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The EXPANDER CHANGE COUNT field counts the number of BROADCAST (CHANGE)s originated by an expander device (see 7.11). SMP target ports in expander devices shall support this field. SMP target ports in other device types (e.g., end devices) shall set the EXPANDER CHANGE COUNT field to 0000h. This field shall be set to 0000h at power on. If the SMP target port has transmitted BROADCAST (CHANGE) for any reason described in 7.11 other than forwarding a BROADCAST (CHANGE) since transmitting a REPORT GENERAL response, it shall increment this field at least once from the value in the previous REPORT GENERAL response. This field shall not be incremented when forwarding a BROADCAST (CHANGE) from another expander device. This field shall wrap to zero after the maximum value (i.e., FFFFh) has been reached.

> NOTE 63 - 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 per phy for the expander device (see 4.6.7.3). SMP target ports in expander devices shall support this field. SMP target ports in other device types (e.g., end devices) shall set the EXPANDER ROUTE INDEXES field to zero. Not all phys in an edge expander device are required to support the maximum number indicated by this field.

The NUMBER OF PHYS field contains the number of phys in the device, including any virtual phys and any vacant phys.

A CONFIGURING bit set to one indicates that a self-configuring expander device has not completed configuring its expander route table. A CONFIGURING bit set to zero indicates that configuration is complete and the expander device is ready for connection requests. Changes in this bit from one to zero result in a BROADCAST (CHANGE) being originated. SMP target ports in self-configuring expander devices shall support this bit. SMP target ports in configurable expander devices and in other device types shall set the CONFIGURING bit to zero.

The CONFIGURABLE ROUTE TABLE bit indicates whether the expander device has an expander route table that is required to be configured with the SMP CONFIGURE ROUTE INFORMATION function (see 4.6.7.3). An expander device with a configurable route table shall set the CONFIGURABLE ROUTE TABLE bit to one. An expander device without a configurable route table or a device with any other device type shall set the CONFIGURABLE ROUTE TABLE bit to zero.

The ENCLOSURE LOGICAL IDENTIFIER field identifies the enclosure, if any, in which the device is located, and is defined in SES-2. The ENCLOSURE LOGICAL IDENTIFIER field shall be set to the same value reported by the enclosure services process, if any, for the enclosure. An ENCLOSURE LOGICAL IDENTIFIER field set to zero indicates no enclosure information is available.

The CRC field is defined in 10.4.3.2.

10.4.3.4 REPORT MANUFACTURER INFORMATION function

The REPORT MANUFACTURER INFORMATION function returns vendor and product identification. This SMP function may be implemented by any SMP target port.

Table 190 defines the request format.

Table 190 — REPORT MANUFACTURER INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (01h)							
2	Reserved							
3								
4	(MSB)							
...	CRC							
7								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 01h.

The CRC field is defined in 10.4.3.1.

Table 191 defines the response format.

Table 191 — REPORT MANUFACTURER INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)								
1	FUNCTION (01h)								
2	FUNCTION RESULT								
3								
7	Reserved								
8	Reserved								SAS-1.1 FORMAT
9								
11	Reserved								
12	(MSB)							
19	VENDOR IDENTIFICATION								(LSB)
20	(MSB)							
35	PRODUCT IDENTIFICATION								(LSB)
36	(MSB)							
39	PRODUCT REVISION LEVEL								(LSB)
40	(MSB)							
47	COMPONENT VENDOR IDENTIFICATION								(LSB)
48	(MSB)							
49	COMPONENT ID								(LSB)
50	COMPONENT REVISION ID								
51	Reserved								
52								
59	Vendor specific								
60	(MSB)							
63	CRC								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 01h.

The FUNCTION RESULT field is defined in 10.4.3.2.

A SAS-1.1 FORMAT bit set to one indicates that bytes 40 through 59 are as defined in this standard. A SAS-1.1 FORMAT bit set to zero indicates that bytes 40 through 59 are vendor-specific as defined in the original version of this standard.

ASCII data fields (e.g., the VENDOR IDENTIFICATION field, the PRODUCT IDENTIFICATION field, and PRODUCT REVISION LEVEL field, and the COMPONENT VENDOR IDENTIFICATION field) shall contain only graphic codes (i.e., code values 20h through 7Eh). Left-aligned fields shall place any unused bytes at the end of the field (i.e., at the highest offset) and the unused bytes shall be filled with space characters (i.e., 20h).

The VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the subsystem (e.g., the board or enclosure) containing the component. The data shall be left-aligned within the field. The vendor identification string shall be one assigned by INCITS for use in the standard INQUIRY data VENDOR IDENTIFICATION field. A list of assigned vendor identification strings is in SPC-3 and on the T10 web site (http://www.t10.org).

The PRODUCT IDENTIFICATION field contains sixteen bytes of ASCII data identifying the type of the subsystem (e.g., the board or enclosure model number) containing the component, as defined by the vendor of the subsystem. The data shall be left-aligned within the field. The PRODUCT IDENTIFICATION field should be changed whenever the subsystem design changes in a way noticeable to a user (e.g., a different stock-keeping unit (SKU)).

The PRODUCT REVISION LEVEL field contains four bytes of ASCII data identifying the revision level of the subsystem (e.g., the board or enclosure) containing the component, as defined by the vendor of the subsystem. The data shall be left-aligned within the field. The PRODUCT REVISION LEVEL field should be changed whenever the subsystem design changes (e.g., any component change, even including resistor values).

All components on a subsystem should have the same values for their VENDOR IDENTIFICATION fields, PRODUCT IDENTIFICATION fields, and PRODUCT REVISION LEVEL fields.

> NOTE 64 - Application clients may use the VENDOR IDENTIFICATION field and PRODUCT IDENTIFICATION field to identify the subsystem (e.g., for a user interface). Application clients may use the VENDOR IDENTIFICATION field, PRODUCT IDENTIFICATION field, PRODUCT REVISION LEVEL field to perform workarounds for problems in a specific revision of a subsystem.

The COMPONENT VENDOR IDENTIFICATION field contains eight bytes of ASCII data identifying the vendor of the component (e.g., the expander device) containing the SMP target port. The data shall be left-aligned within the field. The component vendor identification string shall be one assigned by INCITS for use in the standard INQUIRY data VENDOR IDENTIFICATION field. A list of assigned vendor identification strings is in SPC-3 and on the T10 web site (http://www.t10.org).

The COMPONENT ID field contains a 16-bit identifier identifying the type of the component (e.g., the expander device model number) containing the SMP target port, as defined by the vendor of the component. The COMPONENT ID field should be changed whenever the component's programming interface (e.g., the SMP target port definition) changes.

The COMPONENT REVISION LEVEL field contains an 8-bit identifier identifying the revision level of the component (e.g., the expander device) containing the SMP target port, as defined by the vendor of the component. The COMPONENT REVISION LEVEL field should be changed whenever the component changes but its programming interface does not change.

> NOTE 65 - Application clients may use the COMPONENT VENDOR IDENTIFICATION field and the COMPONENT ID field to interpret vendor-specific information (e.g., vendor-specific SMP functions) correctly for that component. Application clients may use the COMPONENT VENDOR IDENTIFICATION field, the COMPONENT ID field, and the COMPONENT REVISION LEVEL field to perform workarounds for problems in a specific revision of a component.

The vendor-specific bytes are defined by the vendor of the subsystem (e.g., the board or enclosure) containing the component.

The CRC field is defined in 10.4.3.2.

10.4.3.5 DISCOVER function

The DISCOVER function returns the physical link configuration information for the specified phy. This SMP function provides information from the IDENTIFY address frame received by the phy and additional phy-specific information. This SMP function shall be implemented by all SMP target ports.

Table 192 defines the request format.

Table 192 — DISCOVER request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (10h)							
2	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)							
	CRC							
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 10h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) for the link configuration information being requested.

The CRC field is defined in 10.4.3.1.

Table 193 defines the response format.

Table 193 — DISCOVER response (Sheet 1 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (10h)							
2	FUNCTION RESULT							
3	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	Reserved	ATTACHED DEVICE TYPE		Reserved				
13	Reserved				NEGOTIATED PHYSICAL LINK RATE			
14	Reserved				ATTACHED SSP INITIATOR	ATTACHED STP INITIATOR	ATTACHED SMP INITIATOR	ATTACHED SATA HOST
15	ATTACHED SATA PORT SELECTOR	Reserved			ATTACHED SSP TARGET	ATTACHED STP TARGET	ATTACHED SMP TARGET	ATTACHED SATA DEVICE
16	SAS ADDRESS							
23								

Table 193 — DISCOVER response (Sheet 2 of 2)

Byte\Bit	7	6	5	4	3	2	1	0
24								
31	ATTACHED SAS ADDRESS							
32	ATTACHED PHY IDENTIFIER							
33								
39	Reserved							
40	PROGRAMMED MINIMUM PHYSICAL LINK RATE				HARDWARE MINIMUM PHYSICAL LINK RATE			
41	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				HARDWARE MAXIMUM PHYSICAL LINK RATE			
42	PHY CHANGE COUNT							
43	VIRTUAL PHY	Reserved			PARTIAL PATHWAY TIMEOUT VALUE			
44	Reserved				ROUTING ATTRIBUTE			
45	Reserved		CONNECTOR TYPE					
46	CONNECTOR ELEMENT INDEX							
47	CONNECTOR PHYSICAL LINK							
48								
49	Reserved							
50								
51	Vendor specific							
52	(MSB)							
55	CRC							(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 10h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The PHY IDENTIFIER field indicates the phy for which physical configuration link information is being returned.

The ATTACHED DEVICE TYPE field indicates the DEVICE TYPE value received during the link reset sequence and is defined in table 194.

Table 194 — ATTACHED DEVICE TYPE **field**

Code	Description
000b	No device attached
001b	End device
010b	Edge expander device
011b	Fanout expander device
All others	Reserved

The ATTACHED DEVICE TYPE field shall only be set to a value other than 000b after:

a) the identification sequence is complete if a SAS device or expander device is attached; or
b) the initial Register - Device to Host FIS has been received if a SATA phy is attached.

The NEGOTIATED PHYSICAL LINK RATE field is defined in table 195 and indicates the physical link rate negotiated during the link reset sequence. The negotiated physical link rate may be less than the programmed minimum physical link rate or greater than the programmed maximum physical link rate if the programmed physical link rates have been changed since the last link reset sequence.

Table 195 — NEGOTIATED PHYSICAL LINK RATE **field**

Code	Name	Description
0h	UNKNOWN	Phy is enabled; unknown physical link rate. [a]
1h	DISABLED	Phy is disabled.
2h	PHY_RESET_PROBLEM	Phy is enabled; the phy obtained dword synchronization for at least one physical link rate during the SAS speed negotiation sequence (see 6.7.4.2), but the SAS speed negotiation sequence failed (i.e., the last speed negotiation window, using a physical link rate expected to succeed, failed). These failures may be logged in the SMP REPORT PHY ERROR LOG function (see 10.4.3.6) and/or the Protocol-Specific Port log page (see 10.2.8.1).
3h	SPINUP_HOLD	Phy is enabled; detected a SATA device and entered the SATA spinup hold state. The LINK RESET and HARD RESET operations in the SMP PHY CONTROL function (see 10.4.3.10) may be used to release the phy. This field shall be updated to this value at SATA spinup hold time (see 6.8.7 and 6.10)(i.e., after the COMSAS Detect Timeout timer expires during the SATA OOB sequence) if SATA spinup hold is supported.
4h	PORT_SELECTOR	Phy is enabled; detected a SATA port selector. The physical link rate has not been negotiated since the last time the phy's SP state machine entered the SP0:OOB_COMINIT state. The SATA spinup hold state has not been entered since the last time the phy's SP state machine entered the SP0:OOB_COMINIT state. The value in this field may change to 3h, 8h, or 9h if attached to the active phy of the SATA port selector. Presence of a SATA port selector is indicated by the ATTACHED SATA PORT SELECTOR bit.
8h	G1	Phy is enabled; 1,5 Gbit/s physical link rate. This field shall be updated to this value after the speed negotiation sequence completes.
9h	G2	Phy is enabled; 3,0 Gbit/s physical link rate. This field shall be updated to this value after the speed negotiation sequence completes.
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 discover process has queried the phy).		

Table 196 describes the ATTACHED SATA PORT SELECTOR bit and the ATTACHED SATA DEVICE bit.

Table 196 — ATTACHED SATA PORT SELECTOR **and** ATTACHED SATA DEVICE **bits**

ATTACHED SATA PORT SELECTOR bit value [a] [b]	ATTACHED SATA DEVICE bit value [c] [d]	Description
0	0	Neither a SATA port selector nor a SATA device is attached and ready on the selected phy.
0	1	The attached phy is a SATA device phy. No SATA port selector is present (i.e., the SP state machine did not detect COMWAKE in response to the initial COMINIT, but sequenced through the normal (non-SATA port selector) SATA device OOB sequence).
1	0	The attached phy is a SATA port selector host phy, and either: a) the attached phy is the inactive host phy, or b) the attached phy is the active host phy and a SATA device is either not present or not ready behind the SATA port selector (i.e., the SP state machine detected COMWAKE while waiting for COMINIT).
1	1	The attached phy is a SATA port selector's active host phy and a SATA device is present behind the SATA port selector (i.e., the SP state machine detected COMWAKE while waiting for COMINIT, timed out waiting for COMSAS, and exchanged COMWAKE with an attached SATA device).

[a] The ATTACHED SATA PORT SELECTOR bit is invalid if the NEGOTIATED PHYSICAL LINK RATE field is set to UNKNOWN (i.e., 0h) or DISABLED (i.e., 1h).
[b] Whenever the ATTACHED SATA PORT SELECTOR bit changes, the phy shall generate a BROADCAST(CHANGE) notification.
[c] For the purposes of the ATTACHED SATA DEVICE bit, the SATA port selector is not considered a SATA device.
[d] The ATTACHED SATA DEVICE bit shall be updated at SATA spin-up hold time (see 6.8.7 and 6.10).

An ATTACHED SATA HOST bit set to one indicates a SATA host port is attached. An ATTACHED SATA HOST bit set to zero indicates a SATA host port is not attached.

NOTE 66 - Support for SATA hosts is outside the scope of this standard.

If a SAS phy reset sequence occurs (see 6.7.4)(i.e., one or more of the ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, the ATTACHED SMP INITIATOR PORT bit, the ATTACHED SSP TARGET PORT bit, the ATTACHED STP TARGET PORT bit, and/or the ATTACHED SMP TARGET PORT bit is set to one), then the ATTACHED SATA PORT SELECTOR bit, the ATTACHED SATA DEVICE bit, and the ATTACHED SATA HOST bit shall each be set to zero.

The ATTACHED SSP INITIATOR PORT bit indicates the value of the SSP INITIATOR PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED STP INITIATOR PORT bit indicates the value of the STP INITIATOR PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED SMP INITIATOR PORT bit indicates the value of the SMP INITIATOR PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED SSP TARGET PORT bit indicates the value of the SSP TARGET PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED STP TARGET PORT bit indicates the value of the STP TARGET PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED SMP TARGET PORT bit indicates the value of the SMP TARGET PORT field received in the IDENTIFY address frame (see 7.8.2) during the identification sequence.

The ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, and ATTACHED SMP TARGET PORT bit shall be updated at the end of the identification sequence.

If a SATA phy reset sequence occurs (see 6.7.3)(i.e., the ATTACHED SATA PORT SELECTOR bit is set to one, the ATTACHED SATA DEVICE bit is set to one, or the ATTACHED SATA HOST bit is set to one), then the ATTACHED SSP INITIATOR PORT bit, ATTACHED STP INITIATOR PORT bit, ATTACHED SMP INITIATOR PORT bit, ATTACHED SSP TARGET PORT bit, ATTACHED STP TARGET PORT bit, and ATTACHED SMP TARGET PORT bit shall each be set to zero.

The SAS ADDRESS field contains the value of the SAS ADDRESS field transmitted in the IDENTIFY address frame during the identification sequence. If the phy is an expander phy, the SAS ADDRESS field contains the SAS address of the expander device (see 4.2.4). If the phy is a SAS phy, the SAS ADDRESS field contains the SAS address of the SAS port (see 4.2.6).

The ATTACHED SAS ADDRESS field contains the value of the SAS ADDRESS field received in the IDENTIFY address frame during the identification sequence. If the attached port is an expander port, the ATTACHED SAS ADDRESS field contains the SAS address of the attached expander device (see 4.2.4). If the attached port is a SAS port, the ATTACHED SAS ADDRESS field contains SAS address of the attached SAS port (see 4.2.6). If the attached port is a SATA device port, the ATTACHED SAS ADDRESS field contains the SAS address of the STP/SATA bridge (see 4.6.2).

The ATTACHED SAS ADDRESS field shall be updated:

  a) after the identification sequence completes, if a SAS phy or expander phy is attached; or
  b) after the COMSAS Detect Timeout timer expires (see 6.8.3.9), if a SATA phy is attached.

An STP initiator port should not make a connection request to the attached SAS address until the ATTACHED DEVICE TYPE field is set to a value other than 000b.

The ATTACHED PHY IDENTIFIER field contains a phy identifier for the attached phy:

  a) If the attached phy is a SAS phy or an expander phy, the ATTACHED PHY IDENTIFIER field contains the value of the PHY IDENTIFIER field received in the IDENTIFY address frame during the identification sequence:
       A) If the attached phy is a SAS phy, the ATTACHED PHY IDENTIFIER field contains the phy identifier of the attached SAS phy in the attached SAS device;
       B) If the attached phy is an expander phy, the ATTACHED PHY IDENTIFIER field contains the phy identifier (see 4.2.7) of the attached expander phy in the attached expander device; and
  b) If the attached phy is a SATA device phy, the ATTACHED PHY IDENTIFIER field contains 00h;
  c) If the attached phy is a SATA port selector phy and the expander device is able to determine the port of the SATA port selector to which it is attached, the ATTACHED PHY IDENTIFIER field contains 00h or 01h; and
  d) If the attached phy is a SATA port selector phy and the expander device is not able to determine the port of the SATA port selector to which it is attached, the ATTACHED PHY IDENTIFIER field contains 00h.

The ATTACHED PHY IDENTIFIER field shall be updated:

  a) after the identification sequence completes, if a SAS phy or expander phy is attached; or
  b) after the COMSAS Detect Timeout timer expires (see 6.8.3.9), if a SATA phy is attached.

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate set by the PHY CONTROL function (see 10.4.3.10). The values are defined in table 197. The default value shall be the value of the HARDWARE MINIMUM PHYSICAL LINK RATE field.

The HARDWARE MINIMUM PHYSICAL LINK RATE field indicates the minimum physical link rate supported by the phy. The values are defined in table 198.

The PROGRAMMED MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate set by the PHY CONTROL function (see 10.4.3.10). The values are defined in table 197. The default value shall be the value of the HARDWARE MAXIMUM PHYSICAL LINK RATE field.

Table 197 — PROGRAMMED MINIMUM PHYSICAL LINK RATE **and** PROGRAMMED MAXIMUM PHYSICAL LINK **rate fields**

Code	Description
0h	Not programmable
8h	1,5 Gbit/s
9h	3,0 Gbit/s
All others	Reserved

The HARDWARE MAXIMUM PHYSICAL LINK RATE field indicates the maximum physical link rate supported by the phy. The values are defined in table 198.

Table 198 — HARDWARE MINIMUM PHYSICAL LINK RATE **and** HARDWARE MAXIMUM PHYSICAL LINK RATE **fields**

Code	Description
8h	1,5 Gbit/s
9h	3,0 Gbit/s
All others	Reserved

The PHY CHANGE COUNT field counts the number of BROADCAST (CHANGE)s originated by an expander phy. Expander devices shall support this field. Other device types shall not support this field. This field shall be set to zero at power on. The expander device shall increment this field at least once when it transmits a BROADCAST (CHANGE) for any reason described in 7.11 originating from the expander phy other than forwarding a BROADCAST (CHANGE).

The expander device is not required to increment the PHY CHANGE COUNT field again unless a DISCOVER response is transmitted. This field shall not be incremented when forwarding a BROADCAST (CHANGE) from another expander device.The PHY CHANGE COUNT field shall wrap to zero after the maximum value (i.e., FFh) has been reached.

> NOTE 67 - Application clients that use the PHY CHANGE COUNT field should read it often enough to ensure that it does not increment a multiple of 256 times between reading the field.

A VIRTUAL PHY bit set to one indicates the phy is part of an internal port and the attached device is contained within the expander device. A VIRTUAL PHY bit set to zero indicates the phy is a physical phy and the attached device is not contained within the expander device.

The PARTIAL PATHWAY TIMEOUT VALUE field indicates the partial pathway timeout value in microseconds (see 7.12.4.5).

> NOTE 68 - The recommended default value for PARTIAL PATHWAY TIMEOUT VALUE is 7 μs. The partial pathway timeout value may be set by the PHY CONTROL function (see 10.4.3.10).

The ROUTING ATTRIBUTE field indicates the routing attribute supported by the phy (see 4.6.7.1) and is defined in table 199.

Table 199 — ROUTING ATTRIBUTE **field**

Code	Name	Description
0h	Direct routing attribute	Direct routing method for attached end devices. Attached expander devices are not supported on this phy.
1h	Subtractive routing attribute	Either: a) subtractive routing method for attached expander devices; or b) direct routing method for attached end devices.
2h	Table routing attribute	Either: a) table routing method for attached expander devices; or b) direct routing method for attached end devices.
All others	Reserved	

The ROUTING ATTRIBUTE field shall not change based on the attached device type.

The CONNECTOR TYPE field indicates the type of connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2). A CONNECTOR TYPE field set to 00h indicates no connector information is available and that the CONNECTOR ELEMENT INDEX field and the CONNECTOR PHYSICAL LINK fields are invalid and shall be ignored.

The CONNECTOR ELEMENT INDEX indicates the element index of the SAS Connector element representing the connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2).

The CONNECTOR PHYSICAL LINK field indicates the physical link in the connector used to access the phy, as reported by the enclosure services process for the enclosure (see the SAS Connector element in SES-2).

The CRC field is defined in 10.4.3.2.

10.4.3.6 REPORT PHY ERROR LOG function

The REPORT PHY ERROR LOG function returns error logging information about the specified phy. This SMP function may be implemented by any SMP target port.

Table 200 defines the request format.

Table 200 — REPORT PHY ERROR LOG request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (11h)							
2	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)				CRC			
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 11h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) for which information shall be reported.

The CRC field is defined in 10.4.3.1.

Table 201 defines the response format.

Table 201 — REPORT PHY ERROR LOG response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (11h)							
2	FUNCTION RESULT							
3	Reserved							
8								
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)							
			INVALID DWORD COUNT					
15								(LSB)
16	(MSB)							
			RUNNING DISPARITY ERROR COUNT					
19								(LSB)
20	(MSB)							
			LOSS OF DWORD SYNCHRONIZATION COUNT					
23								(LSB)
24	(MSB)							
			PHY RESET PROBLEM COUNT					
27								(LSB)
28	(MSB)							
			CRC					
31								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 11h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The PHY IDENTIFIER field indicates the phy (see 4.2.7) for which information is being reported.

The INVALID DWORD COUNT field indicates the number of invalid dwords (see 3.1.100) that have been received outside of phy reset sequences (i.e., between when the SP state machine (see 6.8) sends a Phy Layer Ready (SAS) confirmation or Phy Layer Ready (SATA) confirmation and when it sends a Phy Layer Not Ready confirmation to the link layer). The count shall stop at the maximum value. The INVALID DWORD COUNT field is set to a vendor-specific value after power on.

The RUNNING DISPARITY ERROR COUNT field indicates the number of dwords containing running disparity errors (see 6.2) that have been received outside of phy reset sequences. The count shall stop at the maximum value. The RUNNING DISPARITY ERROR COUNT field is set to a vendor-specific value after power on.

The LOSS OF DWORD SYNCHRONIZATION COUNT field indicates the number of times the phy has restarted the link reset sequence because it lost dword synchronization (see 6.9) (i.e., the SP state machine transitioned from SP15:SAS_PHY_Ready or SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)). The count shall stop at the maximum value. The LOSS OF DWORD SYNCHRONIZATION COUNT field is set to a vendor-specific value after power on.

The PHY RESET PROBLEM COUNT field indicates the number of times the phy did not obtain dword synchronization during the final SAS speed negotiation window (see 6.7.4.2). The count shall stop at the maximum value. The PHY RESET PROBLEM COUNT field is set to a vendor-specific value after power on.

The CRC field is defined in 10.4.3.2.

10.4.3.7 REPORT PHY SATA function

The REPORT PHY SATA function returns information about the SATA state for a specified phy. This SMP function shall be implemented by SMP target ports that share SAS addresses with STP target ports and by SMP target ports in expander devices with STP/SATA bridges. This SMP function shall not be implemented by any other type of SMP target port.

Table 202 defines the request format.

Table 202 — REPORT PHY SATA request

Byte\Bit	7	6	5	4	3	2	1	0
0				SMP FRAME TYPE (40h)				
1				FUNCTION (12h)				
2								
8				Reserved				
9				PHY IDENTIFIER				
10								
11				Reserved				
12	(MSB)							
15				CRC				(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 12h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) for which information shall be reported.

The CRC field is defined in 10.4.3.1.

Table 203 defines the response format.

Table 203 — REPORT PHY SATA response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (12h)							
2	FUNCTION RESULT							
3 8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11	Reserved						AFFILIATIONS SUPPORTED	AFFILIATION VALID
12 15	Reserved							
16 23	STP SAS ADDRESS							
24 43	REGISTER DEVICE TO HOST FIS							
44 47	Reserved							
48 55	AFFILIATED STP INITIATOR SAS ADDRESS							
56	(MSB)							
59	CRC							(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 12h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The PHY IDENTIFIER field indicates the phy (see 4.2.7) for which information is being reported.

An AFFILIATIONS SUPPORTED bit set to one indicates that affiliations are supported by the STP target port containing the specified phy. An AFFILIATIONS SUPPORTED bit set to zero indicates that affiliations are not supported by the STP target port containing the specified phy.

The AFFILIATION VALID bit shall be set to one when the AFFILIATED STP INITIATOR SAS ADDRESS field is valid and the STP target port containing the specified phy is maintaining an affiliation (see 7.17.5). The AFFILIATION VALID bit shall be set to zero when no affiliation is being maintained.

The STP SAS ADDRESS field contains the SAS address (see 4.2.2) of the STP target port that contains the specified phy.

The REGISTER DEVICE TO HOST FIS field contains the contents of the initial Register - Device to Host FIS. For an STP/SATA bridge, this is delivered by the attached SATA device after a link reset sequence (see ATA/ATAPI-7 V3 and SATAII-EXT). For a native STP target port in an end device, this is directly provided.

The FIS contents shall be stored with little-endian byte ordering (i.e., the first byte, byte 24, contains the FIS Type).

For an STP/SATA bridge, the first byte of the field (i.e., the FIS Type) shall be initialized to zero on power on and whenever the phy has restarted the link reset sequence after losing dword synchronization (see 6.9)(i.e., the SP state machine transitioned from SP22:SATA_PHY_Ready to SP0:OOB_COMINIT (see 6.8)) to indicate the field is invalid and the attached SATA device has not delivered a Register – Device to Host FIS. The first byte of the field shall be set to 34h when the attached SATA device has delivered the initial Register – Device to Host FIS. The remaining contents of the REGISTER DEVICE TO HOST FIS field shall remain constant until a link reset sequence causes the attached SATA device to deliver another initial Register – Device to Host FIS.

An STP/SATA bridge that receives a connection request for a SATA device that has not successfully delivered the initial Register – Device to Host FIS shall return an OPEN_REJECT (NO DESTINATION).

> NOTE 69 - If there is a problem receiving the expected initial Register - Device to Host FIS, the STP/SATA bridge should use SATA_R_ERR to retry until it succeeds. In the DISCOVER response, the ATTACHED SATA DEVICE bit is set to one and the ATTACHED SAS ADDRESS field is valid, but the ATTACHED DEVICE TYPE field is set to 000b (i.e., no device attached) during this time.

The AFFILIATED STP INITIATOR SAS ADDRESS field contains the SAS address (see 4.2.2) of the STP initiator port that currently has an affiliation with the STP target port that contains the specified phy.

The CRC field is defined in 10.4.3.2.

10.4.3.8 REPORT ROUTE INFORMATION function

The REPORT ROUTE INFORMATION function returns an expander route entry from the expander route table within an expander device. This SMP function shall be supported by SMP target ports in expander devices if the EXPANDER ROUTE INDEXES field is non-zero in the REPORT GENERAL function. This SMP function may be used as a diagnostic tool to resolve topology issues.

Table 204 defines the request format.

Table 204 — REPORT ROUTE INFORMATION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (13h)							
2	Reserved							
5								
6	(MSB)	EXPANDER ROUTE INDEX						
7								(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11								
12	(MSB)	CRC						
15								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 13h.

The EXPANDER ROUTE INDEX field specifies the expander route index for the expander route entry being requested (see 4.6.7.3).

The PHY IDENTIFIER field specifies the phy for which the expander route entry is being requested.

The CRC field is defined in 10.4.3.1.

Table 205 defines the response format.

Table 205 — REPORT ROUTE INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (13h)							
2	FUNCTION RESULT							
3	Reserved							
5								
6	(MSB)			EXPANDER ROUTE INDEX				
7								(LSB)
8	Reserved							
9	PHY IDENTIFIER							
10	Reserved							
11								
12	EXPANDER ROUTE ENTRY DISABLED	Reserved						
13	Reserved							
15								
16	ROUTED SAS ADDRESS							
23								
24	Reserved							
39								
40	(MSB)			CRC				
43								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 13h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The EXPANDER ROUTE INDEX field contains the expander route index for the expander route entry being returned (see 4.6.7.3).

The PHY IDENTIFIER field contains the phy identifier for the expander route entry being returned.

The EXPANDER ROUTE ENTRY DISABLED bit indicates whether the ECM shall use the expander route entry to route connection requests (see 4.6.7.3). If the EXPANDER ROUTE ENTRY DISABLED bit is set to zero, then the ECM shall use the expander route entry to route connection requests. If the EXPANDER ROUTE ENTRY DISABLED bit is set to one, the ECM shall not use the expander route entry to route connection requests.

The ROUTED SAS ADDRESS field contains the SAS address in the expander route entry (see 4.6.7.3).

The CRC field is defined in 10.4.3.2.

10.4.3.9 CONFIGURE ROUTE INFORMATION function

The CONFIGURE ROUTE INFORMATION function sets an expander route entry within the expander route table of a configurable expander device. This SMP function shall be supported by SMP target ports in expander devices if the CONFIGURABLE ROUTE TABLE field is set to one in the REPORT GENERAL response data. Other SMP target ports shall not support this SMP function.

Table 206 defines the request format.

Table 206 — CONFIGURE ROUTE INFORMATION request

Byte\Bit	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 EXPANDER ROUTE ENTRY	Reserved						
13	Reserved							
15								
16	ROUTED SAS ADDRESS							
23								
24	Reserved							
39								
40	(MSB)							
			CRC					
43								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 90h.

The EXPANDER ROUTE INDEX field specifies the expander route index for the expander route entry being configured (see 4.6.7.3).

The PHY IDENTIFIER field specifies the phy for which the expander route entry is being configured (see 4.6.7.3).

The DISABLE EXPANDER ROUTE ENTRY bit specifies whether the ECM shall use the expander route entry to route connection requests (see 4.6.7.3). If the DISABLE EXPANDER ROUTE ENTRY bit is set to zero, then the ECM shall use the expander route entry to route connection requests. If the DISABLE EXPANDER ROUTE ENTRY bit is set to one, the ECM shall not use the expander route entry to route connection requests.

The ROUTED SAS ADDRESS field contains the SAS address for the expander route entry being configured (see 4.6.7.3).

The CRC field is defined in 10.4.3.1.

Table 207 defines the response format.

Table 207 — CONFIGURE ROUTE INFORMATION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (90h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)				CRC			
7								(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 90h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The CRC field is defined in 10.4.3.2.

10.4.3.10 PHY CONTROL function

The PHY CONTROL function requests actions by the specified phy. This SMP function may be implemented by any SMP target port.

Table 208 defines the request format.

Table 208 — PHY CONTROL request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (91h)							
2	Reserved							
8								
9	PHY IDENTIFIER							
10	PHY OPERATION							
11	Reserved							UPDATE PARTIAL PATHWAY TIMEOUT VALUE
12	Reserved							
31								
32	PROGRAMMED MINIMUM PHYSICAL LINK RATE				Reserved			
33	PROGRAMMED MAXIMUM PHYSICAL LINK RATE				Reserved			
34	Reserved							
35								
36	Reserved				PARTIAL PATHWAY TIMEOUT VALUE			
37	Reserved							
39								
40	(MSB)							
				CRC				
43								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 91h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) to which the PHY CONTROL request applies.

Table 209 defines the PHY OPERATION field.

Table 209 — PHY OPERATION field (Sheet 1 of 2)

Code	Operation	Description
00h	NOP	No operation.
01h	LINK RESET	If the specified phy is not a virtual phy, perform a link reset sequence (see 4.4) on the specified phy and enable the specified phy. If the specified phy is a virtual phy, perform an internal reset and enable the specified phy. See 7.11 for BROADCAST (CHANGE) requirements related to this phy operation in an expander device. Any affiliation (see 7.17.5) shall continue to be present. The phy shall bypass the SATA spinup hold state, if implemented (see 6.8.3.9). The SMP response shall be returned without waiting for the link reset to complete.
02h	HARD RESET	If the specified phy is not a virtual phy, perform a link reset sequence (see 4.4) on the specified phy and enable the specified phy. If the attached phy is a SAS phy or an expander phy, the link reset sequence shall include a hard reset sequence (see 4.4.2). If the attached phy is a SATA phy, the phy shall bypass the SATA spinup hold state. See 7.11 for BROADCAST (CHANGE) requirements related to this phy operation in an expander device. If the specified phy is a virtual phy, perform an internal reset and enable the specified phy. Any affiliation (see 7.17.5) shall be cleared. The SMP response shall be returned without waiting for the hard reset to complete.
03h	DISABLE	Disable the specified phy (i.e., stop transmitting valid dwords and receiving dwords on the specified phy). The LINK RESET and HARD RESET operations may be used to enable the phy. See 7.11 for BROADCAST (CHANGE) requirements related to this phy operation in an expander device.
04h	Reserved	
05h	CLEAR ERROR LOG	Clear the error log counters (see 10.4.3.6) for the specified phy.

Table 209 — PHY OPERATION **field (Sheet 2 of 2)**

Code	Operation	Description
06h	CLEAR AFFILIATION	Clear an affiliation (see 7.17.5) 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 SMP target port shall return a function result of SMP FUNCTION FAILED in the response frame.
07h	TRANSMIT SATA PORT SELECTION SIGNAL	This function shall only be supported by phys in an expander device. If the expander phy incorporates an STP/SATA bridge and supports SATA port selectors, the phy shall transmit the SATA port selection signal (see 6.6) which causes the SATA port selector to select the attached phy as the active host phy and make its other host phy inactive. See 7.11 for BROADCAST (CHANGE) requirements related to this phy operation in an expander device. Any affiliation (see 7.17.5) shall be cleared. If the expander phy does not support SATA port selectors, then the SMP target port shall return a function result of PHY DOES NOT SUPPORT SATA. If the expander phy supports SATA port selectors but is attached to a SAS phy or an expander phy, the SMP target port shall return a function result of SMP FUNCTION FAILED.
All others	Reserved	

If the PHY IDENTIFIER field specifies the phy which is being used for the SMP connection and a phy operation of LINK RESET, HARD RESET, or DISABLE is requested, the SMP target port shall not perform the requested operation and shall return a function result of SMP FUNCTION FAILED in the response frame.

An UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit set to one specifies that the PARTIAL PATHWAY TIMEOUT VALUE field shall be honored. An UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit set to zero specifies that the PARTIAL PATHWAY TIMEOUT VALUE field shall be ignored.

The PROGRAMMED MINIMUM PHYSICAL LINK RATE field specifies the minimum physical link rate the phy shall support during a link reset sequence (see 4.4.1). Table 210 defines the values for this field. If this field is changed along with a phy operation of LINK RESET or HARD RESET, that phy operation shall utilize the new value for this field.

The PROGRAMMED MAXIMUM PHYSICAL LINK RATE field specifies the maximum physical link rates the phy shall support during a link reset sequence (see 4.4.1).Table 210 defines the values for this field. If this field is changed along with a phy operation of LINK RESET or HARD RESET, that phy operation shall utilize the new value for this field.

Table 210 — PROGRAMMED MINIMUM PHYSICAL LINK RATE **and** PROGRAMMED MAXIMUM PHYSICAL LINK RATE **fields**

Code	Description
0h	Do not change current value
8h	1,5 Gbit/s
9h	3,0 Gbit/s
All others	Reserved

If the PROGRAMMED MINIMUM PHYSICAL LINK RATE field or the PROGRAMMED MAXIMUM PHYSICAL LINK RATE field is set to an unsupported or reserved value, or the PROGRAMMED MINIMUM PHYSICAL LINK RATE field and PROGRAMMED MAXIMUM PHYSICAL LINK RATE field are set to an invalid combination of values (e.g., the minimum is greater than the maximum), the SMP target port shall not change either of their values and may return a function result of SMP

FUNCTION FAILED in the response frame. If it returns a function result of SMP FUNCTION FAILED, it shall not perform the requested phy operation.

The PARTIAL PATHWAY TIMEOUT VALUE field specifies the amount of time in microseconds the expander phy shall wait after receiving an Arbitrating (Blocked On Partial) confirmation from the ECM before requesting that the ECM resolve pathway blockage (see 7.12.4.6). A PARTIAL PATHWAY TIMEOUT VALUE field value of zero (i.e., 0 µs) specifies that partial pathway resolution shall be requested by the expander phy immediately upon reception of an Arbitrating (Blocked On Partial) confirmation from the ECM. The PARTIAL PATHWAY TIMEOUT VALUE field is only honored when the UPDATE PARTIAL PATHWAY TIMEOUT VALUE bit is set to one.

The CRC field is defined in 10.4.3.1.

Table 211 defines the response format.

Table 211 — PHY CONTROL response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (91h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)							
7				CRC				(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 91h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The CRC field is defined in 10.4.3.2.

10.4.3.11 PHY TEST FUNCTION function

The PHY TEST FUNCTION function requests actions by the specified phy. This SMP function may be implemented by any SMP target port.

Table 212 defines the request format.

Table 212 — PHY TEST FUNCTION request

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (40h)							
1	FUNCTION (92h)							
2	Reserved							
8								
9	PHY IDENTIFIER							
10	PHY TEST FUNCTION							
11	PHY TEST PATTERN							
12	Reserved							
14								
15	Reserved				PHY TEST PATTERN PHYSICAL LINK RATE			
16	Reserved							
39								
40	(MSB)							
				CRC				
43								(LSB)

The SMP FRAME TYPE field shall be set to 40h.

The FUNCTION field shall be set to 92h.

The PHY IDENTIFIER field specifies the phy (see 4.2.7) to which the PHY TEST PATTERN request applies.

If the PHY IDENTIFIER field specifies the phy which is being used for the SMP connection, the SMP target port shall not perform the requested operation and shall return a function result of SMP FUNCTION FAILED in the response frame.

The PHY TEST FUNCTION field specifies the phy test function to be performed, and is defined in table 213. If the PHY TEST FUNCTION field specifies a phy test function that is not supported by the phy, the SMP target port shall return a function result of UNKNOWN PHY TEST FUNCTION in the response frame.

Table 213 — PHY TEST FUNCTION **field**

Code	Name	Description
00h	STOP	If the selected phy is performing a phy test function, then the selected phy shall stop performing the phy test function and originate a link reset sequence. If the selected phy is not performing a phy test function, then this function has no effect on the selected phy.
01h	TRANSMIT_ PATTERN	If the selected phy is not performing a phy test function, the selected phy shall be set to transmit the phy test pattern specified by the PHY TEST PATTERN field at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field and set to ignore its receiver. If the selected phy receives data while transmitting the pattern, then the selected phy shall ignore the received data. If the selected phy is performing a phy test function, the SMP target port shall return a function result of PHY TEST FUNCTION IN PROGRESS in the response frame.
02h - EFh	Reserved	
F0h - FFh	Vendor specific	

If the PHY TEST FUNCTION field is set to 01h, the PHY TEST PATTERN field specifies the phy test pattern to be performed, and the same as that defined in table 179 for the Protocol-Specific diagnostic page (see 10.2.9.1). The phy test pattern shall be sent at the physical link rate specified by the PHY TEST PATTERN PHYSICAL LINK RATE field.

The PHY TEST PATTERN PHYSICAL LINK RATE field specifies the physical link rate at which the phy test function, if any, shall be performed. Table 214 defines the values for this field.

Table 214 — PHY TEST PATTERN PHYSICAL LINK RATE **field**

Code	Description
8h	1,5 Gbit/s
9h	3,0 Gbit/s
All others	Reserved

The CRC field is defined in 10.4.3.1.

Table 215 defines the response format.

Table 215 — PHY TEST FUNCTION response

Byte\Bit	7	6	5	4	3	2	1	0
0	SMP FRAME TYPE (41h)							
1	FUNCTION (92h)							
2	FUNCTION RESULT							
3	Reserved							
4	(MSB)							
7	CRC							(LSB)

The SMP FRAME TYPE field shall be set to 41h.

The FUNCTION field shall be set to 92h.

The FUNCTION RESULT field is defined in 10.4.3.2.

The CRC field is defined in 10.4.3.2.

Annex A
(normative)

Jitter tolerance patterns

A.1 Jitter tolerance pattern (JTPAT)

The jitter tolerance pattern (JTPAT) consists of:

1) a long run of low transition density pattern;
2) a long run of high transition density pattern; and
3) another short run of low transition density pattern.

The transitions between the pattern segments stress the receiver. The JTPAT is designed to contain the phase shift in both polarities, from 0 to 1 and from 1 to 0. The critical pattern sections with the phase shifts are underlined in table A.1 and table A.2.

Table A.1 shows the JTPAT when there is positive running disparity (RD+) (see 6.2) at the beginning of the pattern. The 8b and 10b values of each character are shown.

Table A.1 — JTPAT for RD+

Beginning RD	First character	Second character	Third character	Fourth character	Ending RD
RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD+
	1000011100b	0111100011b	1000011100b	0111100011b	
The above dword of low transition density pattern is sent a total of 41 times					
RD+	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D20.3 (74h)	RD-
	1000011100b	0111100011b	1000011100b	0010111100b	
The above dword containing phase shift 11100001011b is sent 1 time					
RD-	D30.3 (7Eh)	D11.5 (ABh)	D21.5 (B5h)	D21.5 (B5h)	RD+
	0111100011b	1101001010b	1010101010b	1010101010b	
The above dword containing phase shift 00011110100b is sent 1 time					
RD+	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	D21.5 (B5h)	RD+
	1010101010b	1010101010b	1010101010b	1010101010b	
The above dword of high transition density pattern is sent a total of 12 times					
RD+	D21.5 (B5h)	D30.2 (5Eh)	D10.2 (4Ah)	D30.3 (7Eh)	RD+
	1010101010b	1000010101b	0101010101b	0111100011b	
The above dword containing phase shift 01010000b and 10101111b is sent 1 time					

If the same 8b characters specified in table A.1 are used when there is negative running disparity (RD-) at the beginning of the pattern, the resulting 10b pattern is different and does not provide the critical phase shifts. To achieve the same phase shift effects with RD-, a different 8b pattern is required. Table A.2 shows the JTPAT when there is negative running disparity (RD-) at the beginning of the pattern. The 8b and 10b values of each character are shown.

Table A.2 — JTPAT for RD-

Beginning RD	First character	Second character	Third character	Fourth character	Ending RD
RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	RD-
	0111100011b	1000011100b	0111100011b	1000011100b	
The above dword of low transition density pattern is sent a total of 41 times					
RD-	D30.3 (7Eh)	D30.3 (7Eh)	D30.3 (7Eh)	D11.3 (6Bh)	RD+
	0111100011b	1000011100b	0111<u>00011</u>b	<u>101000</u>011b	
The above dword containing phase shift <u>00011110100</u>b is sent 1 time					
RD+	D30.3 (7Eh)	D20.2 (54h)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
	10000<u>11100</u>b	<u>001011</u>0101b	0101010101b	0101010101b	
The above dword containing phase shift <u>11100001011</u>b is sent 1 time					
RD-	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	D10.2 (4Ah)	RD-
	0101010101b	0101010101b	0101010101b	0101010101b	
The above dword of high transition density pattern is sent a total of 12 times					
RD-	D10.2 (4Ah)	D30.5 (BEh)	D21.5 (B5h)	D30.3 (7Eh)	RD-
	0101010<u>101</u>b	<u>0111</u>101010b	1010101<u>010</u>b	<u>10000</u>11100b	
The above dword containing phase shift <u>10101111</u>b and <u>01010000</u>b is sent 1 time					

Table A.3 shows a pattern containing both JTPAT for RD+ and JTPAT for RD-. The 10b pattern resulting from encoding the 8b pattern contains the desired bit sequences for the phase shifts with both starting running disparities.

Table A.3 — JTPAT for RD+ and RD-

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.

A.2 Compliant jitter tolerance pattern (CJTPAT)

The compliant jitter tolerance pattern (CJTPAT) is the JTPAT for RD+ and RD- (see table A.3 in A.1) included as the payload in an SSP DATA frame or an SSP DATA frame modified to reduce the percentage of the transfer that is not the JTPAT.The CJTPAT shall include at least:

1) SOF;
2) JTPAT for RD+ and RD-;
3) CRC; and
4) EOF.

Other SSP frame header information (see 9.2.1) may be included in the CJTPAT. ALIGN and/or NOTIFY primitives may be included in the transmission of the CJTPAT, but the number of ALIGN and/or NOTIFY primitives transmitted should be as small as possible so that the percentage of the transfer that is the JTPAT is as high as possible.

When the JTPAT is encapsulated in an SSP frame, the 8b data is scrambled by the transmitter scrambler before transmission. 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 an example of a CJTPAT where the JTPAT for RD+ and RD- is embedded in an SSP DATA frame. In this example, a 24-byte header is included following the SOF before the JTPAT for RD+ and RD-.

The second column (8b data dword) in table A.4 lists the desired 8b pattern data that is to be 8b10b encoded.

The third column (Scrambler output dword) in table A.4 lists the output, in dword format, of the transmitter scrambler.

The fourth column (Scrambled 8b data dword) in table A.4 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 and/or NOTIFYs 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 (Sheet 1 of 4)

Frame contents	8b data dword	Scrambler output dword	Scrambled 8b data dword = (8b data dword) XOR (scrambler output dword)
SOF	<primitive>	<primitive>	<primitive>
SSP DATA frame header (optional)	unknown	C2D2768Dh	unknown
	unknown	1F26B368h	unknown
	unknown	A508436Ch	unknown
	unknown	3452D354h	unknown
	unknown	8A559502h	unknown
	unknown	BB1ABE1Bh	unknown
SSP frame data (part 1) (dwords 0 - 7)	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 (part 2) (dwords 8-15)	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 (part 3) (dwords 16 - 23)	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 (Sheet 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 (part 4) (dwords 24 - 31)	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 (part 5) (dwords 32 - 39)	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 (part 6) (dwords 40 - 47)	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 (part 7) (dwords 48 -55)	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 (Sheet 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 (part 8) (dwords 56 - 63)	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 (part 9) (dwords 64 - 71)	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 (part 10) (dwords 72 - 79)	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 (part 11) (dwords 80 - 87)	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 (Sheet 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 (part 12) (dwords 88 - 95)	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 (part 13) (dwords 96 - 103)	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 (part 14) (dwords 104 - 111)	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	9AFCB3DDh	unknown
EOF	<primitive>	<primitive>	<primitive>

Annex B
(informative)

Signal performance measurements

B.1 Signal performance measurements overview

This annex specifies the configuration requirements for making electrical performance measurements. These measurements consist of signal output, signal tolerance, and return loss. Standard loads are used in all cases so that independent specification of connection components and transportability of the measurement results are possible.

> NOTE 70 - The methodology for making return loss measurements are specified in this annex although this standard does not define any return loss values. Statements in this annex that imply return loss values are specified should be ignored.

B.2 Simple physical link

B.2.1 Simple physical link overview

The physical link is considered to consist of the following component parts:

 a) the transmitter device;
 b) the interconnect; and
 c) the receiver device.

Each of those components is connected by a separable connector. On a TxRx connection, signals travel in opposite directions down the same nominal path.

Figure B.1 shows a physical link and the location of the connectors.
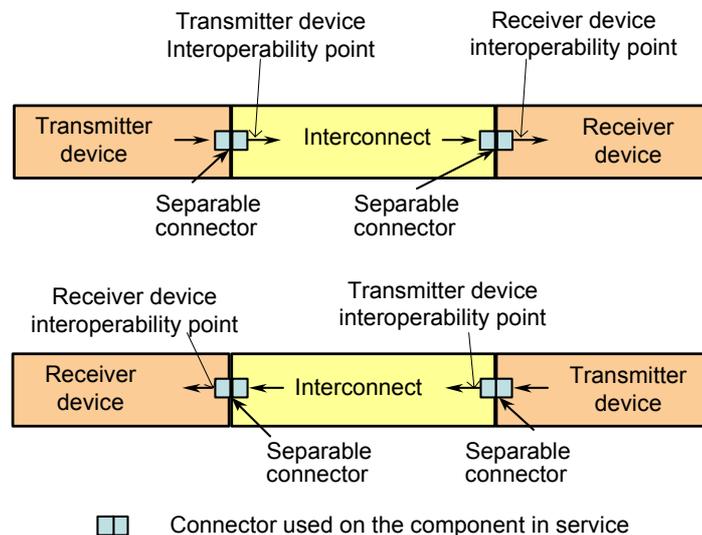
Figure B.1 — A simple physical link

Since connectors are always used in the mated condition, the only access to the signals is before the signal enters the mated connector (i.e., upstream) or after the signal exits the connector (i.e., downstream). Even if signals were able to be practically accessed at the point of mating within the connector, such access would disturb the connector to the point that the measurement of the signal would be compromised (e.g., attempting to access the unmated connector with probes does not provide valid results because the connector is not in the same condition when unmated as when mated and the probe contact points are not at the same location as the connector contact points).

In this annex, signal outputs are always measured downstream of the mated connector (see figure B.1) so that the contribution of the mated connector to the signal properties is included in the measurement. This approach assigns a portion of the connector losses to the upstream component, but it also makes the signal measurement

conservative. If the connectors on both ends of the interconnect are the same, the additional loss at the downstream connector is offset by the reduced loss at the upstream connector. For transmitter devices, a slightly stronger transmitter is required to pass the signal through the downstream half of the connector that does not belong to the transmitter device. The signal coming into receiver devices is specified after the signal has passed through the connector.

Examination of the details of the measurement methods described in this annex shows that the mated connector issue may not be as severe as it appears.

B.2.2 Assumptions for the structure of the transmitter device and the receiver device

Figure B.2 shows the details of a transmitter device.

All signals measured with full activity present
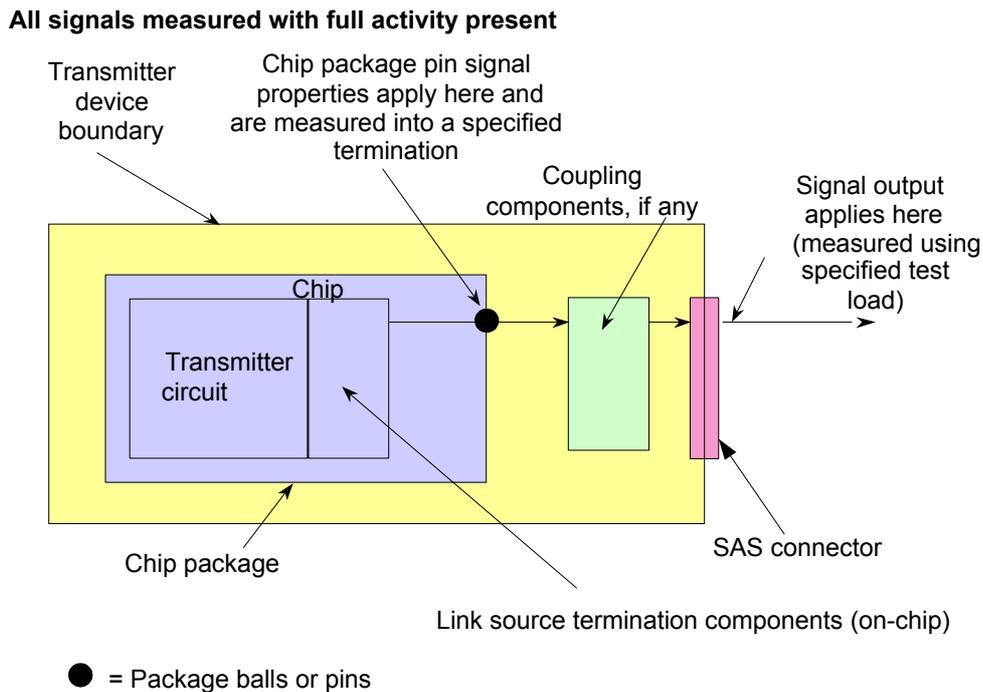
Figure B.2 — Transmitter device details

As figure B.2 shows, any of the following internal parts of this transmitter device may be labeled as the transmitter:

    a)   the transmitter circuit in the chip;
    b)   the chip itself; and
    c)   the chip and its associated chip package.

The term transmitter is therefore not well defined and is not used in the terminology without a modifier.

The transmitter device contains:

    a)   a connector (i.e., half a mated pair);
    b)   coupling components. if any;
    c)   PCB traces and vias;
    d)   the chip package;
    e)   ESD protection devices, if any;
    f)   the source termination; and
    g)   the transmitter circuit.

It is assumed that the source termination is contained within the chip package.

Interoperability points may be defined at the chip package pins in some standards (e.g., Ethernet XAUI). This standard does not define requirements at chip package pins.

Figure B.3 shows the details of a receiver device. It is similar to the transmitter device.

Figure B.3 — Receiver device details

As figure B.3 shows, any of the following internal parts of this receiver device may be labeled as the receiver:

    a)   the receiver circuit in the chip;
    b)   the chip itself; and
    c)   the chip and its associated chip package.

The term receiver is therefore not well defined and is not used in the terminology without a modifier.

The receiver device contains:

    a)   a connector (i.e., half a mated pair);
    b)   coupling components, if any;
    c)   PCB traces and vias;
    d)   the chip package;
    e)   ESD protection devices, if any;
    f)   the physical link termination; and
    g)   the receiver circuit.

It is assumed that the physical link termination is contained within the chip package.

B.2.3 Definition of receiver sensitivity and receiver device sensitivity

The term receiver sensitivity is problematic in common usage. This term is not used for interoperability in standards. A related term applicable to the receiver device input signal is receiver device sensitivity. While these two terms are related, they are significantly different because of the noise environment assumed. The description in this subclause is used to uniquely define these terms with the understanding that this standard discourages usage of either term.

Receiver sensitivity is defined as the minimum vertical inner eye opening at which the receiver chip delivers the required BER (see 5.3.3) with the horizontal eye opening at its minimum (i.e., maximum jitter is present) and all activity quiesced except for the receiver itself. The receiver in the receiver sensitivity refers to signal properties at the chip package pin for the chip package that contains the receiver circuit. However, receiver sensitivity is not defined in this standard because there are no chip package pin specifications.

Receiver device sensitivity is defined as the minimum vertical inner eye opening measured at the signal output point for the input to the receiver device at which the receiver chip (i.e., the receiver circuit in the chip package on

the board containing the receiver device interoperability point as shown in figure B.3) delivers the required BER (see 5.3.3) with:

    h)   the minimum horizontal eye opening;

    a)   all activity expected in the application for the receiver circuit present (i.e., not quiesced as for the receiver sensitivity definition); and

    b)   the CJTPAT pattern being received (see Annex A).

Special test conditions are required to measure these sensitivities (see B.8). The terminology used in this standard is signal tolerance instead of receiver device sensitivity.

B.3 Measurement architecture requirements

B.3.1 General

Signal specifications are only meaningful if the signals are able to be measured with practical instrumentation. Another requirement is that different laboratories making measurements on the same signal get the same results within acceptable measurement error (i.e., the measurements need to be accessible, verifiable, and transportable). As of the publishing of this standard, there are no accepted standard for creating signals with traceable properties and with all the properties required for an effective signal specification architecture for high speed serial applications.

Some of the elements required for practical, verifiable, and transportable signal measurements are included in this standard.

Having signal specifications at interoperability points that do not depend on the actual properties of the other physical link components not under test requires that specified known test loads be used for the signal measurements. In service, the load presented to the interoperability point is that of the actual component and environment.

Interfacing with practical instruments also requires that specified impedance environments be used. This forces a signal measurement architecture where the impedance environment is 50 Ω single-ended (i.e., 100 Ω differential). It also forces the requirement for instrumentation-quality loads of the correct value.

Instrumentation-quality loads are readily available for simple transmission line termination. However, none are available for more complex loads that include specified propagation time, insertion loss properties, crosstalk properties, and jitter creation properties.

For signal tolerance measurements, the signal is calibrated before applying it to the interoperability point under test. This signal calibration is done by adjusting the properties of the specified signal source system as measured into a laboratory-quality test load until the desired signal tolerance specifications are met. The signal source system is then disconnected from the laboratory-quality test load and connected to the interoperability point under test. It is assumed that any changes to the signal from the calibration state to the measurement state are caused by the interoperability point under test and are therefore part of the performance sought by the measurement.

B.3.2 Relationship between signal compliance measurements at interoperability points and operation in systems

The signal measurements in this standard apply under specified test conditions that simulate some parts of the conditions existing in service (e.g., this simulation includes full-duplex traffic on all phys and under all applicable environmental conditions). Other features existing in service (e.g., non-ideal return loss in parts of the physical link that are not present when measuring signals in the specified test conditions) may be included in the specifications themselves. This methodology is required to specify signal performance requirements for each side of the interoperability point that do not depend on knowing the properties of the other side.

Measuring signals in an actual functioning system at an interoperability point does not verify compliance for the components on either side of the interoperability point, although it does verify that the specific combination of components in the system at the time of the measurement produces compliant signals. Interaction between components on either side of the interoperability point may allow the signal measured to be compliant, but this compliance may have resulted because one component does not meet the specification while the other exceeds the specification.

It is recommended that additional margin be allowed when performing signal compliance measurements to account for conditions existing in service that may not have been accounted for in the specified measurements and specifications.

B.4 De-embedding connectors in test fixtures

Connectors are part of the test fixtures (e.g. test loads) required for obtaining access to the interoperability points. This is intrinsic for most practical measurements because the connectors used on the service components are different from those used on the instrumentation.

The effects of the portions of the connector that is used on the test fixture need to be accounted for in order to not penalize the interoperability point under test by the performance of the test fixture connector. This accounting process is termed de-embedding.

Figure B.4 shows two cases that apply.
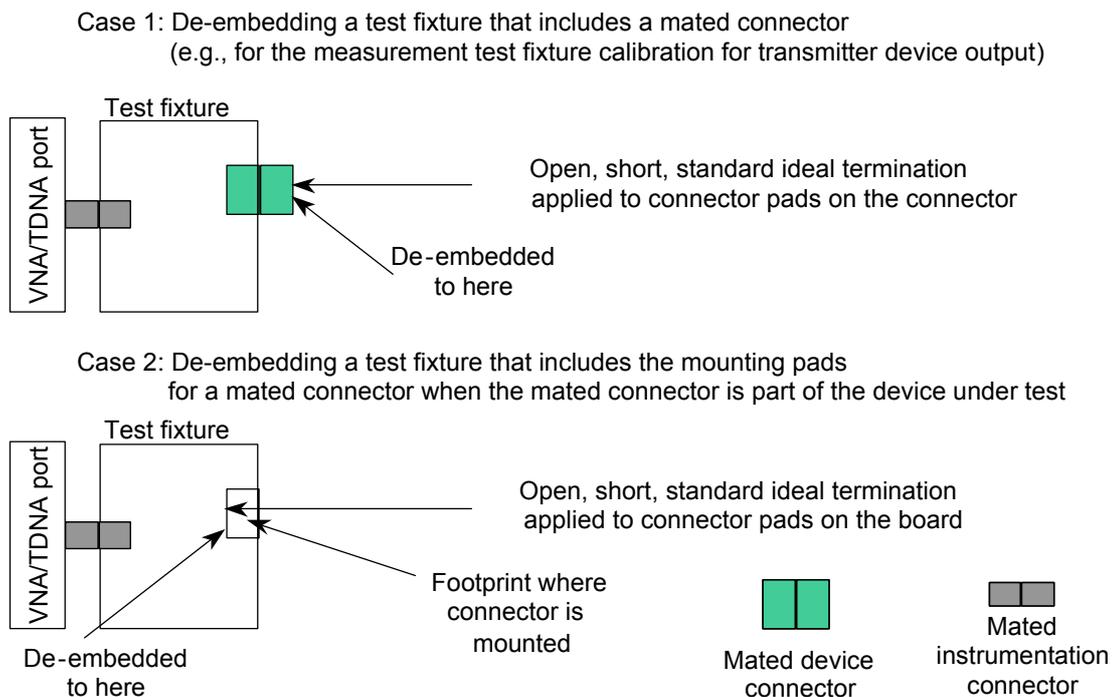
Figure B.4 — De-embedding of connectors in test fixtures

The de-embedding process assumes that the test fixture is linear and that S-parameter methodologies (see B.9) are used. Fundamentally, an S-parameter model for the test fixture with or without the connector in place is the result. Knowing this model for the test fixture, with or without the connector in place, allows simulation of the impact of the test fixture on the signal measurement.

B.5 Measurement conditions for signal output at the transmitter device

The measurement conditions required for a differential transmitter device signal output are shown in figure B.5. Figure B.5 applies to the following cases:

   a)   the transmitter device is directly attached to the zero length test load (see 5.3.6.2); and
   b)   the transmitter device is attached to the TCTF test load (see 5.3.6.3).

To simulate the properties of the interconnect assembly, instrumentation-quality test loads as shown in figure B.6 are used.

Measurement of signal output at the transmitter device：
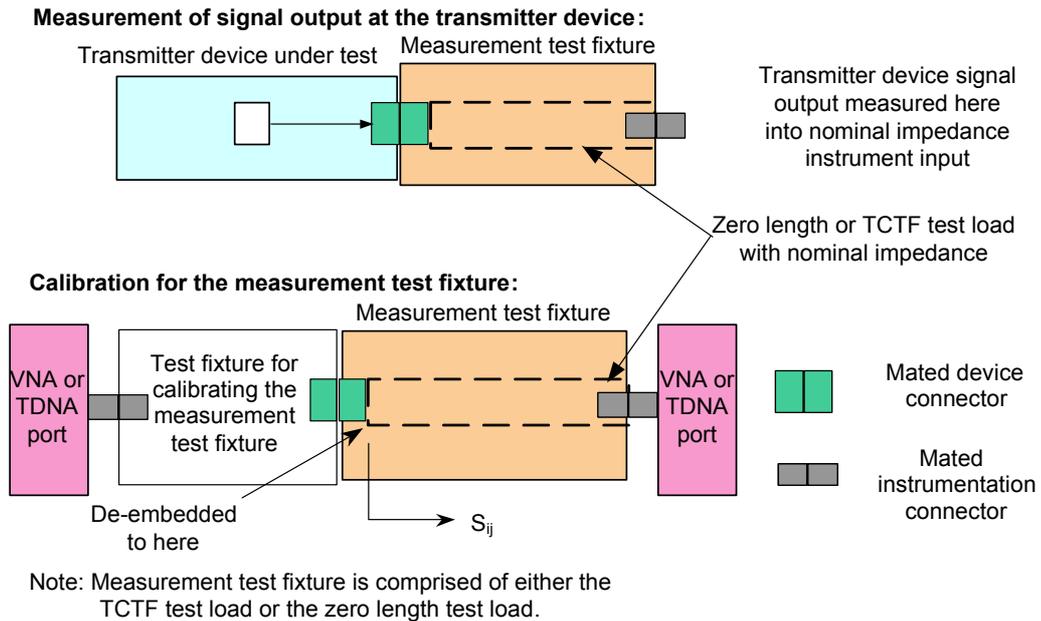
Figure B.5 — Measurement conditions for signal output at the transmitter device

An instrumentation-quality cable assembly connecting the measurement test fixture to the instrumentation port is usually required. This cable assembly is considered part of the instrumentation and is not specifically shown in figure B.5, figure B.6, figure B.7, figure B.8, figure B.9, figure B.10, and figure B.11.

A measurement test fixture may be constructed from an instrumentation-quality TCTF test load with instrumentation-quality connectors and a connector adapter as shown in figure B.6. This scheme may be useful when using multiple device connector types but adds extra components that may increase loss and delay. For best accuracy, this scheme is not recommended. Extra components make it more difficult for the transmitter device to meet the required output specifications.
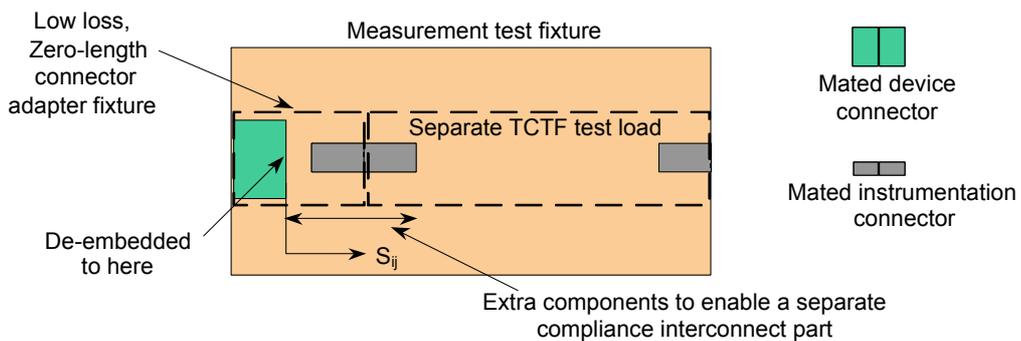
Figure B.6 — Transmitter device signal output measurement test fixture details

B.6 Measurement conditions for signal tolerance at the transmitter device

The measurement conditions required for the signal tolerance at the differential transmitter device interoperability point are shown in figure B.7. Figure B.7 shows the test signal is launched into the interconnect assembly (e.g., cable assembly or PCB) that is attached to the receiver device.

This standard does not include this performance requirement but is included here for completeness.

Measurement of signal tolerance at transmitter device (e.g., IT and CT):
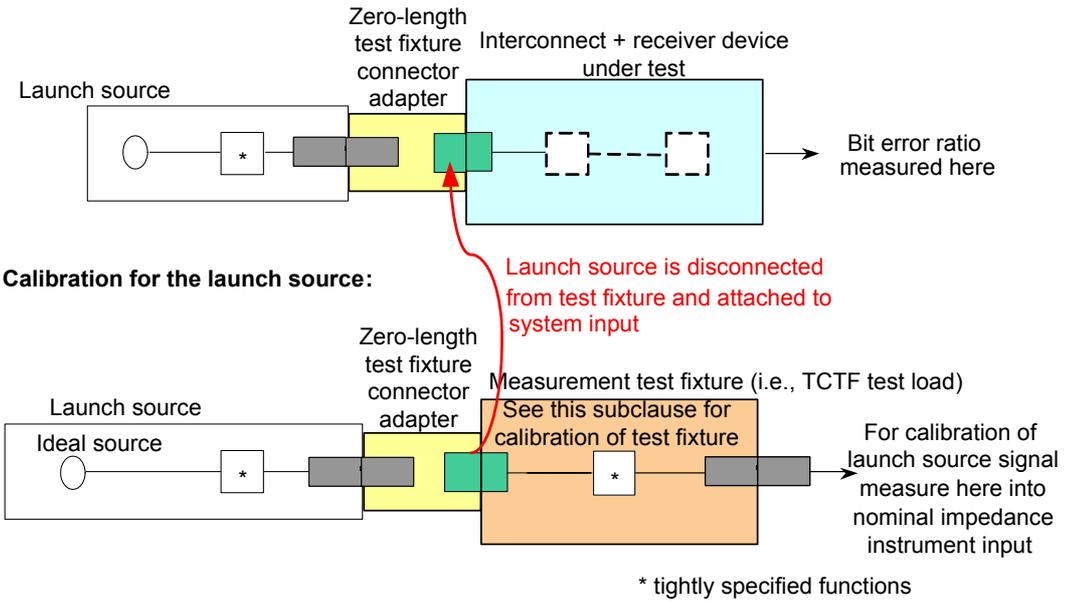
Figure B.7 — Measurement conditions for signal tolerance at the transmitter device

Figure B.8 shows calibration of the measurement test fixture.
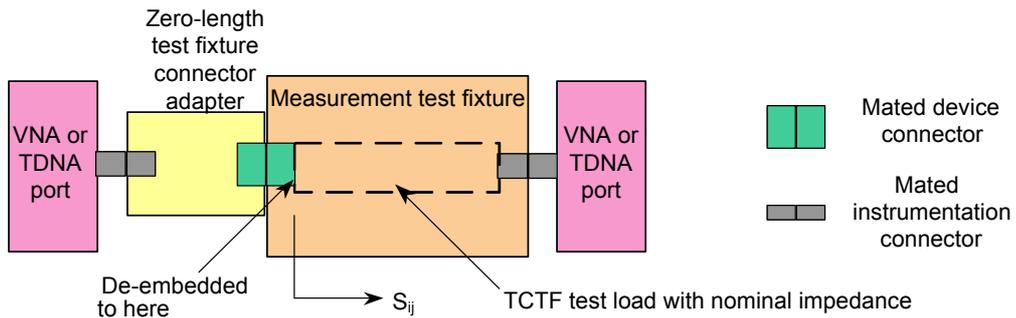
Figure B.8 — Calibration of test fixture for signal tolerance at the transmitter device

B.7 Measurement conditions for signal output at the receiver device

Figure B.9 shows the measurement conditions for the signal output at the receiver device.
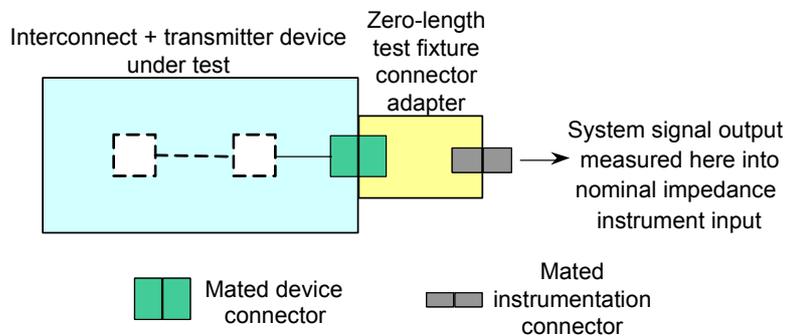
Figure B.9 — Measurement conditions for signal output at the receiver device

The interconnect may be the zero-length connector adaptor where the transmitter device is connected directly to the receiver device.

B.8 Measurement conditions for signal tolerance at the receiver device

Figure B.10 shows the measurement conditions required for the signal tolerance at the differential receiver device interoperability point (see 5.3.7.2).
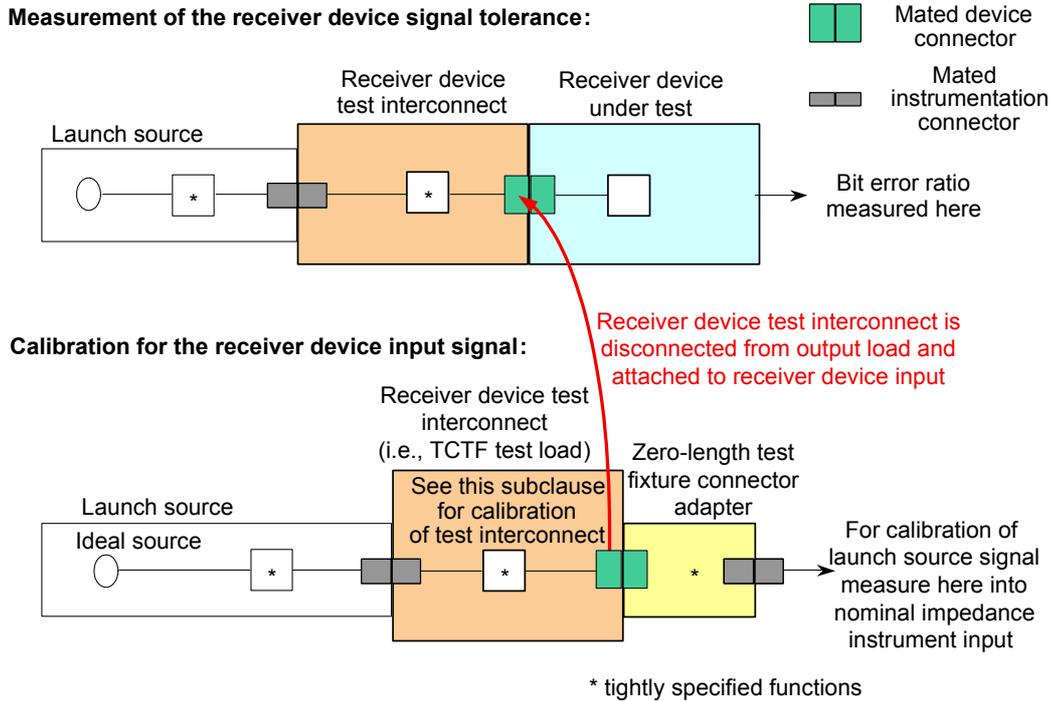
Figure B.10 — Measurement conditions for signal tolerance at the receiver device

Figure B.11 shows calibration of the measurement test fixture.
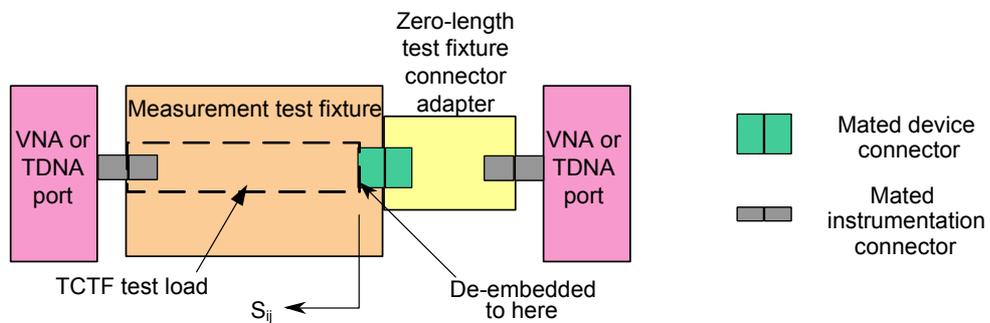
Notes:

1 This is not identical to the measurement test fixture used for the transmitter output signal even though the connector genders are the same. The pins used in the SAS connector are for the Rx (i.e., not the Tx) signals and the signals flow the other way. The S_{22} measurement here is the same as the S_{11} measurement for the transmitter output signal but on different pins.

2 The S_{21} and S_{12} are used to create the desired jitter in this application and are not as critical.

Figure B.11 — Calibration of test fixture for signal tolerance at the receiver device

B.9 S-parameter measurements

B.9.1 S-parameter overview

Properties of physical link elements that are linear may be represented by S-parameter (i.e., scattering parameter) spectra. There are two problematic areas when applying S-parameters to differential electrical physical links:

   a) naming conventions; and
   b) use of single-ended vector network methods on differential and common-mode systems.

This subclause explores both of these areas.

Measurement architecture for the most common conditions are described in some detail.

B.9.2 S-parameter naming conventions

There are two types of measurements performed with S-parameters:

    a) return loss from the same port of the element; and
    b) transfer function or insertion loss across the element.

Each s-parameter is a function of frequency returning complex numbers and is expressed with:

    a) a magnitude component, usually expressed in dB; and
    b) a phase component.

For a two-port linear element having ports i and j with the signals being either differential or common-mode, S_{ij} is the ratio of the signal coming out of the ith port (i.e., the response) to the signal coming into the jth port (i.e., the stimulus).

A port number convention is used where the downstream port is always port 2 and the upstream port is always port 1. The stream direction is determined by the direction of the primary signal launched from the transmitter device to the receiver device (e.g., in this standard, since each differential pair carries a signal in only one direction, the port nearest the transmitter device is port 1 and the port nearest the receiver device is port 2).

There are four combinations of ports for a two-ported system yielding the following S-parameters:

    a) S_{11}: return loss measured at port 1;
    b) S_{21}: transfer function or insertion loss measured at port 1;
    c) S_{22}: upstream return loss. Return loss measure at port 2 of the element. The measurement is the same kind of measurement that is done at port 1 to measure S_{11}; and
    d) S_{12}: upstream insertion loss. Insertion loss measured at port 2 of the element. The measurement is the same kind of measurement that is done at port 1 to measure S_{21}.

Figure B.12 shows the port naming conventions for physical link elements, loads, and where those elements exit.

Port definitions for loads used for signal output testing and
S-parameter measurements in multiline configurations
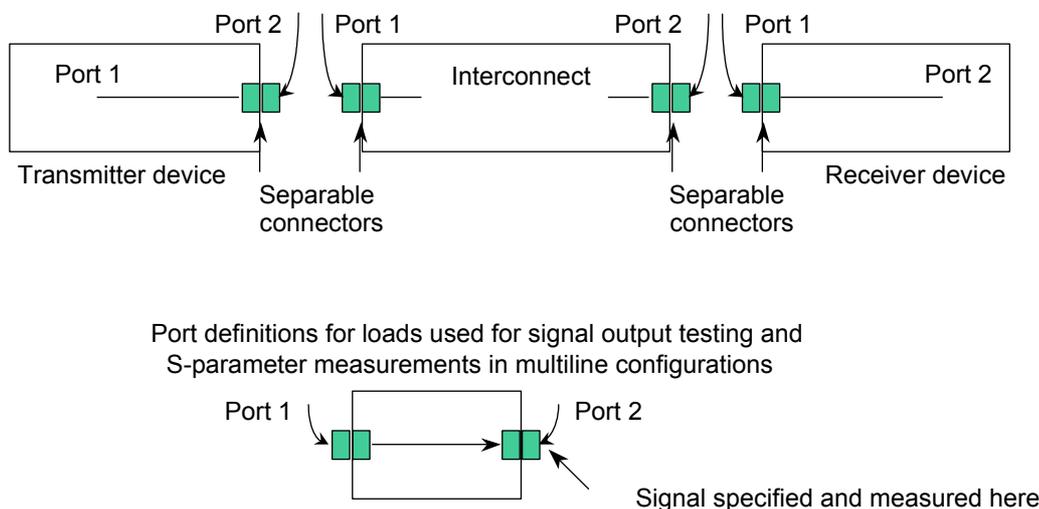
This load has ideal differential and common mode properties

Note: The transmitter device port 1 and receiver device port 2 are internal and are not defined. They are required to be an ideal source and an ideal sink, respectively.

Figure B.12 — S-parameter port naming conventions

B.9.3 Use of single-ended instrumentation in differential applications

There are four categories of S-parameters for a differential system:

    a) S_{DDij}: differential stimulus, differential response;
    b) S_{CDij}: differential stimulus, common-mode response (i.e., mode conversion causing emissions);

c) S_{DCij}: common-mode stimulus, differential response (i.e., mode conversion causing susceptibility); and

d) S_{CCij}: common-mode stimulus, common-mode response.

All the measurements specified in this standard relate to differential signal pairs, and all specified S-parameters are of the S_{DDij} form.

Figure B.13 shows the connections that are made to a four port VNA or TDNA for measuring S-parameters on a four single-ended port black box device.

VNA ports are all single-ended; the differential and common-mode properties for differential ports are calculated internal to the VNA or may mathematically derived. If using a TDNA, consult the details for the specific instrument. Four analyzer ports are required to measure the properties of two differential ports.

$$V_1 = V_{A1} + V_{B1}$$
$$V_2 = V_{A2} + V_{B2}$$
$$V_3 = V_{A3} + V_{B3}$$
$$V_4 = V_{A4} + V_{B4}$$

V_{A1}, V_{A2}, V_{A3}, and V_{A4} are incident signals
V_{B1}, V_{B2}, V_{B3}, and V_{B4} are reflected signals
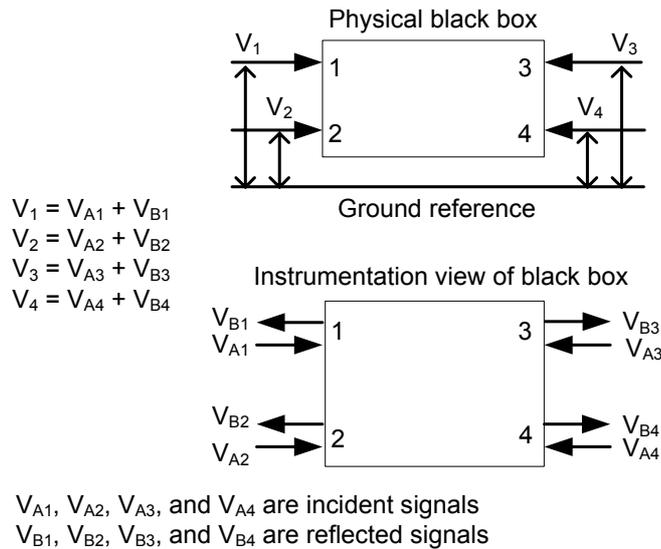
Figure B.13 — Four single-ended port or two differential port element

Figure B.14 shows the set of S-parameters for a single-ended system and for a differential system.
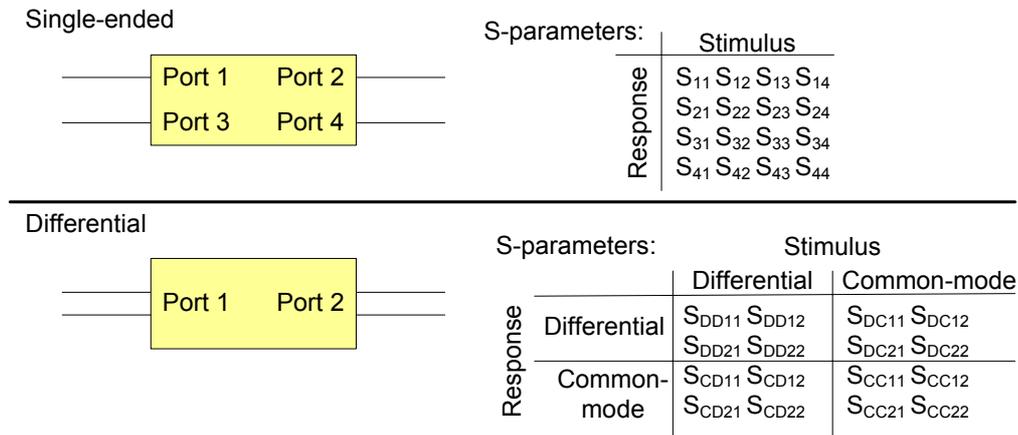
Figure B.14 — S-parameters for single-ended and differential systems

See SFF-8416 for details on connections required for differential S-parameter measurements.

B.9.4 Measurement configurations for physical link elements

B.9.4.1 Measurement configuration overview

Special test fixtures are required to make S-parameter measurements partly because the connectors used on real physical link elements are different from those used on instrumentation. The goal is for these test fixtures to be as invisible as possible.

See B.9.4 for the description of the measurement configurations. All of these measurements are return loss in this annex. A more complete set of S-parameters is used as part of the calibration process for test fixtures.

B.9.4.2 Transmitter device return loss

Figure B.15 shows the configuration to be used for the transmitter device return loss measurement.
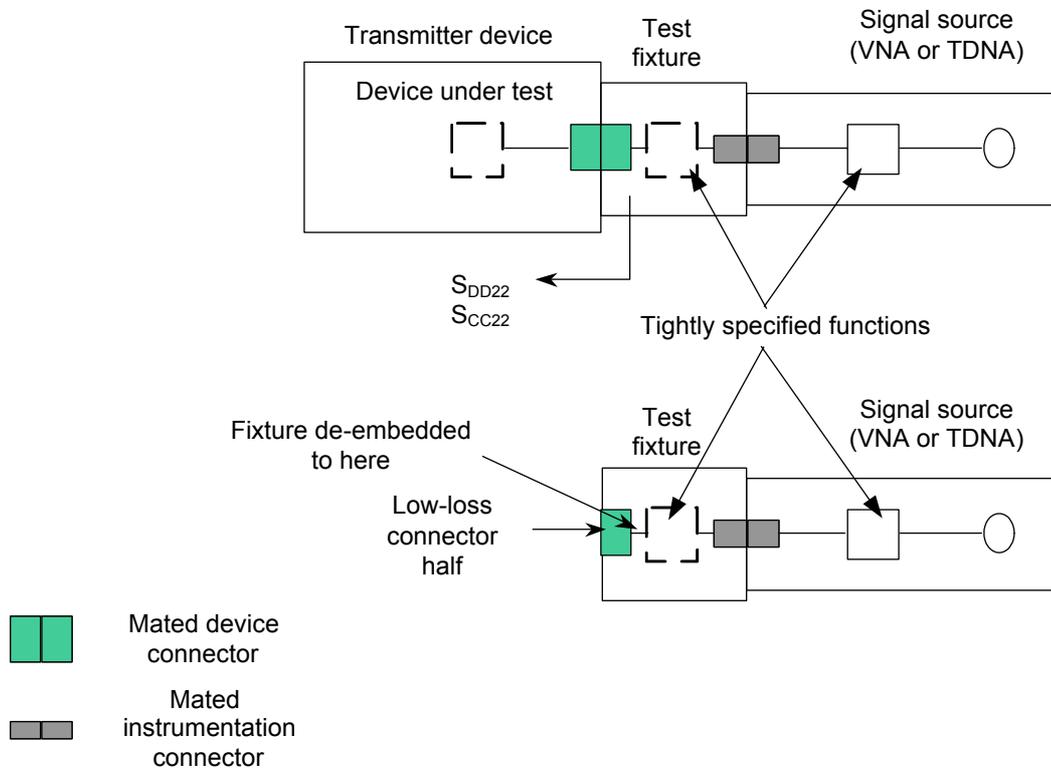
Figure B.15 — Measurement conditions for upstream return loss at the transmitter device connector

The test fixture in figure B.15 uses low loss connectors to avoid penalizing the transmitter device under test for the test fixture half of the connector. If the test fixture half of the device connector is poor then the transmitter device has to compensate for those losses.

The test fixture losses up to the mounting points for the device connector are de-embedded using the methods described in figure B.4.

B.9.4.3 Receiver device return loss

Figure B.16 shows the configuration to be used for the receiver device return loss measurement.
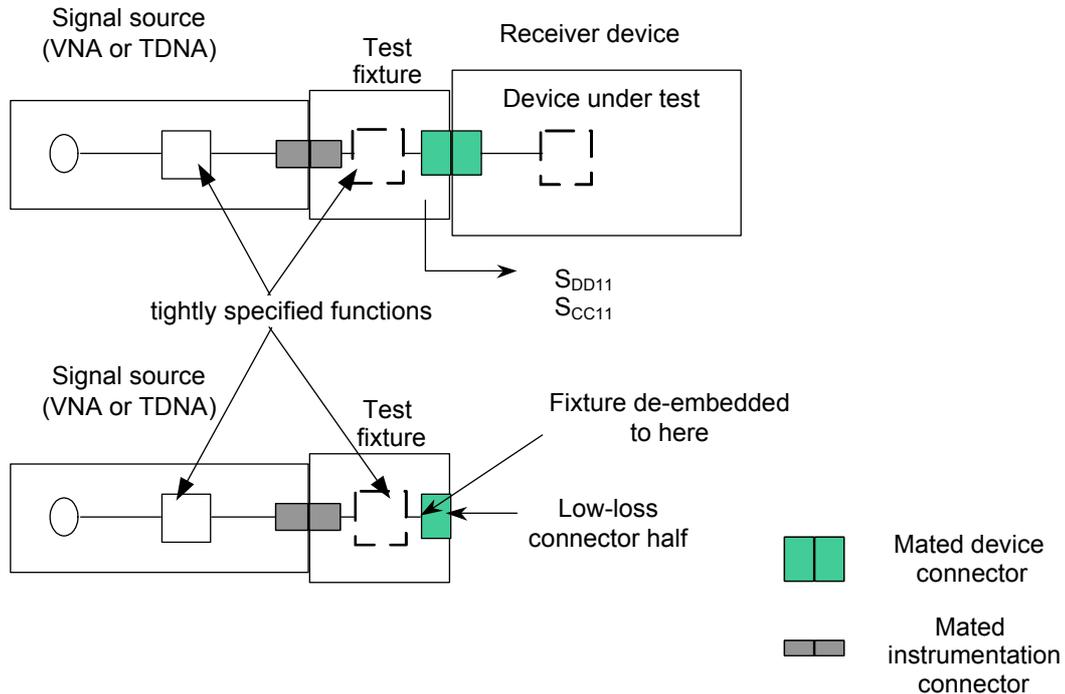
Figure B.16 — Measurement conditions for downstream return loss at the receiver device connector

The test fixture in figure B.16 uses low loss connectors to avoid penalizing the receiver device under test for the test fixture half of the connector. If the test fixture half of the device connector is poor then the receiver device has to compensate for those losses.

The test fixture losses up to the mounting points for the device connector are de-embedded using the methods described in figure B.4.

B.9.4.4 Return loss (S_{11}) at IT or CT

Figure B.17 shows the conditions for making the return loss measurement into the interconnect attached to the transmitter device.

This measurement, like the signal tolerance measurement at the transmitter device connector, requires both the interconnect and the receiver device to be in place and the combination is measured. If the receiver device is replaced by an ideal load then the return loss does not represent in service conditions. If the interconnect is very lossy then the effects of the load on the far end (i.e., where the receiver device is located) are not significant and an ideal load may be used. However, if the interconnect is not very lossy as in the zero length case, then the measured return loss may be dominated by the properties of the receiver device and not the properties of the interconnect.

For short physical links, this return loss performance may be the limiting factor for the entire physical link due to severe unattenuated reflections that create large DJ.
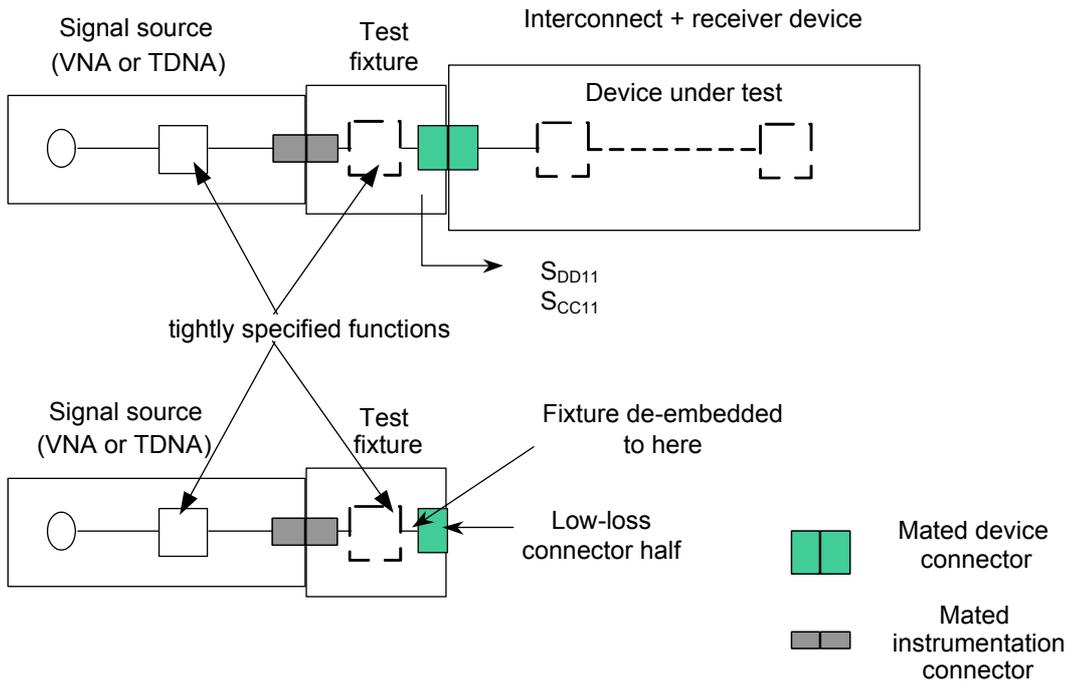
Figure B.17 — Measurement conditions for downstream return loss at IT or CT

B.9.4.5 Upstream return loss (S_{22}) at IR or CR

Figure B.18 shows the conditions for making the return loss measurement out of the interconnect attached to the receiver device.

This measurement is unique in that it requires both the interconnect and the transmitter device to be in place and the combination is measured. This may be considered a reverse direction signal tolerance measurement. If the transmitter device is replaced by an ideal load then the return loss does not represent in service conditions. If the interconnect is very lossy then the effects of the load on the far end (i.e., where the transmitter device is located) are not significant and an ideal load may be used. However, if the interconnect is not very lossy as in the zero length case, then the measured return loss may be dominated by the properties of the transmitter device and not the properties of the interconnect.

For short physical links, this return loss performance may be the limiting factor for the entire physical link due to severe unattenuated reflections that create large DJ.
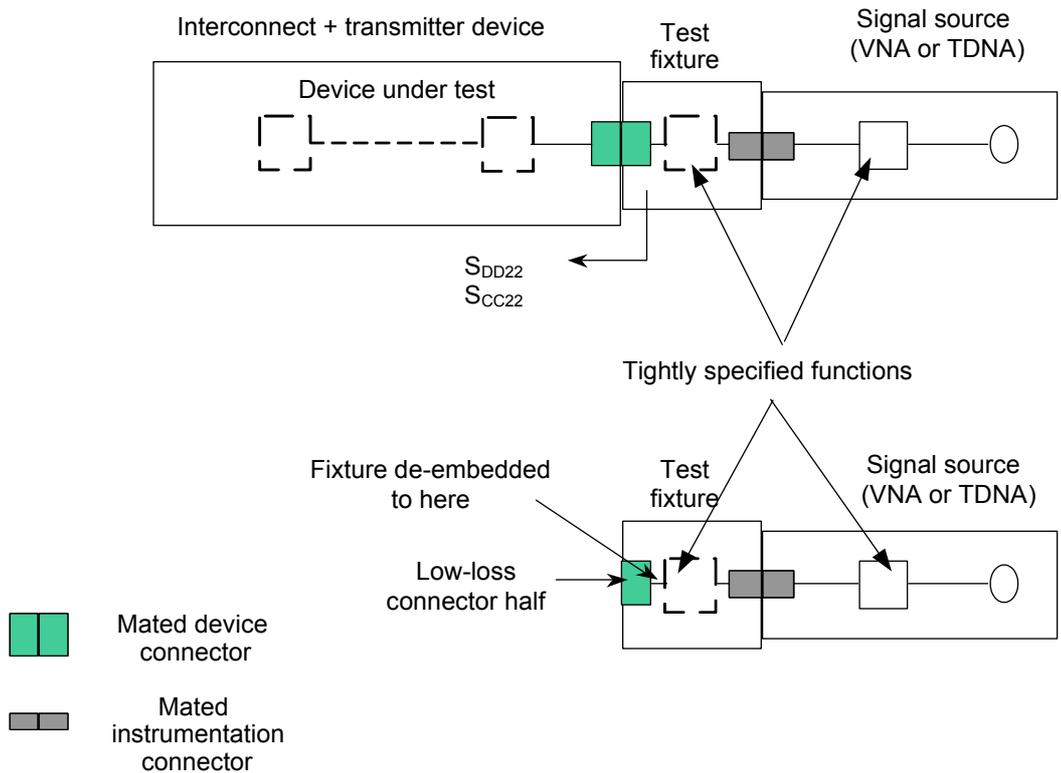
Figure B.18 — Measurement conditions for upstream return loss at IR or CR

B.9.5 Summary for S-parameter measurements

S-parameters are the preferred method of capturing the linear properties of physical link elements. Complex, but tractable, methods are required to use single-ended instruments for differential and common-mode applications. Careful attention to test configuration details is required.

A frequency domain spectrum output is required for all S-parameters and specifying pass/fail limits to such a spectrum may overconstrain the system because some peaks and properties are benign to the application.

Annex C
(informative)

SAS to SAS phy reset sequence examples

Figure C.1 shows a speed negotiation between a phy A that supports only G1 attached to a phy B that only supports G1. Both phys run:

    1)   the G1 speed negotiation window, supported by both phys; and
    2)   the G2 speed negotiation window, supported by neither phy.

Both phys then select G1 for the final speed negotiation window used to establish the negotiated physical link rate.
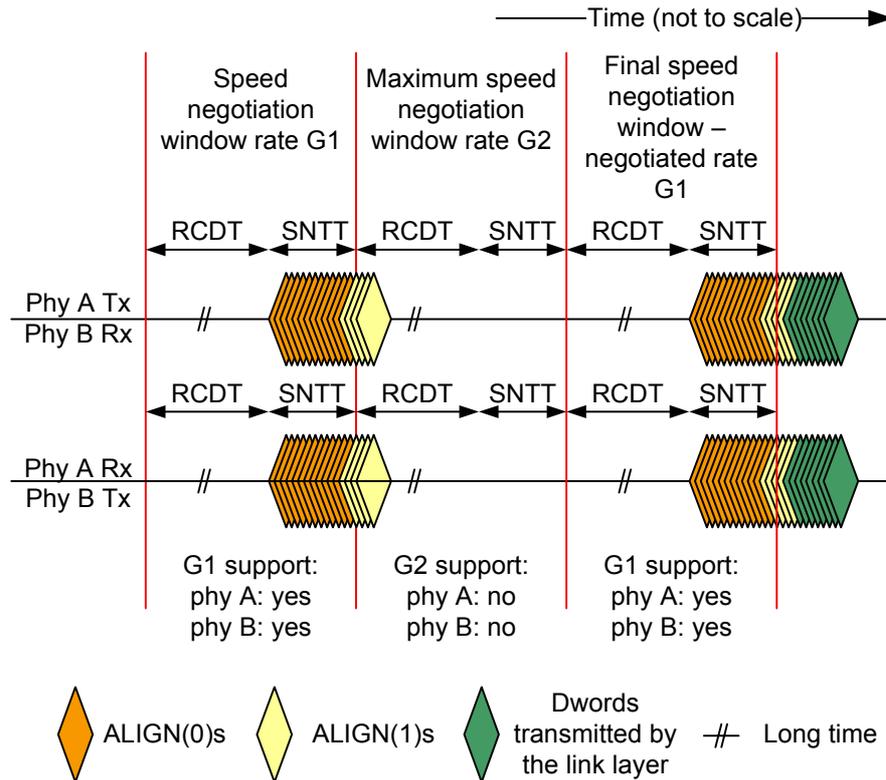
Figure C.1 — SAS speed negotiation sequence (phy A: G1 only, phy B: G1 only)

Figure C.2 shows a speed negotiation between a phy A that supports G1 and G2 attached to a phy B that supports G1 and G2. Both phys run:

    1)   the G1 speed negotiation window, supported by both phys;
    2)   the G2 speed negotiation window, supported by both phys; and
    3)   the G3 speed negotiation window, supported by neither phy.

Both phys then select G2 for the final speed negotiation window used to establish the negotiated physical link rate.
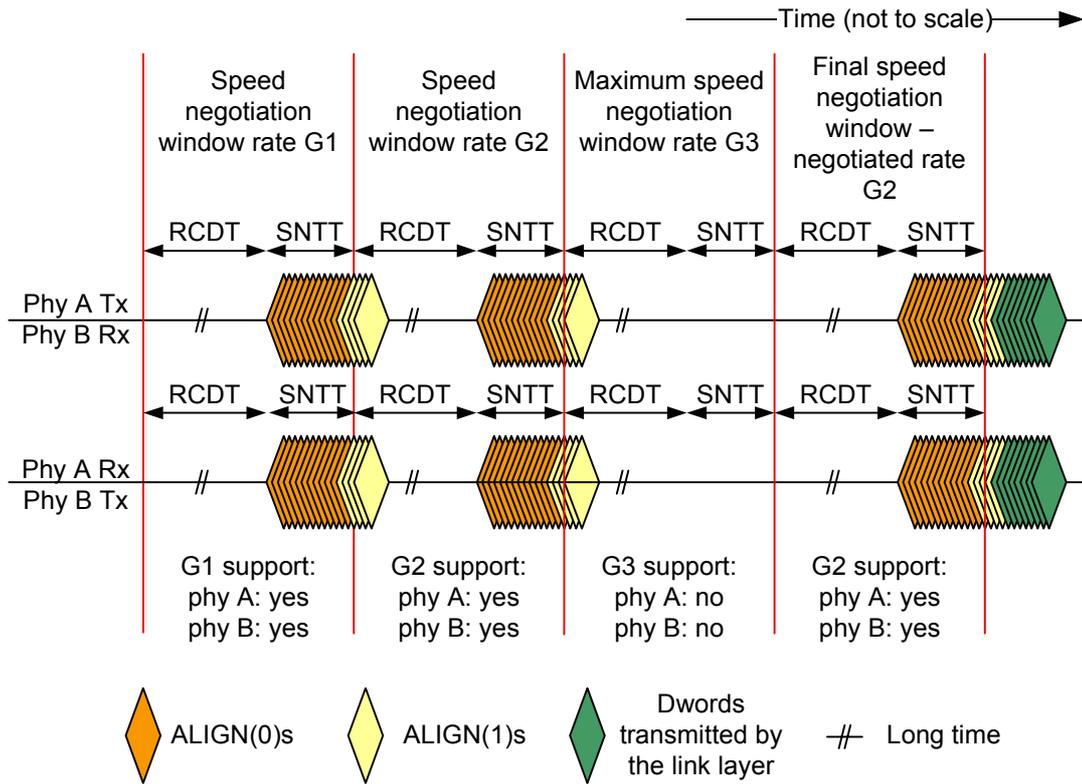
Figure C.2 — SAS speed negotiation sequence (phy A: G1, G2, phy B: G1, G2)

Figure C.3 shows a speed negotiation between a phy A that supports G1 through G3 attached to a phy B that only supports G1 and G2. Both phys run:

1) the G1 speed negotiation window, supported by both phys;
2) the G2 speed negotiation window, supported by both phys; and
3) the G3 speed negotiation window, supported by phy A but not by phy B.

Both phys then select G2 for the final speed negotiation window used to establish the negotiated physical link rate.
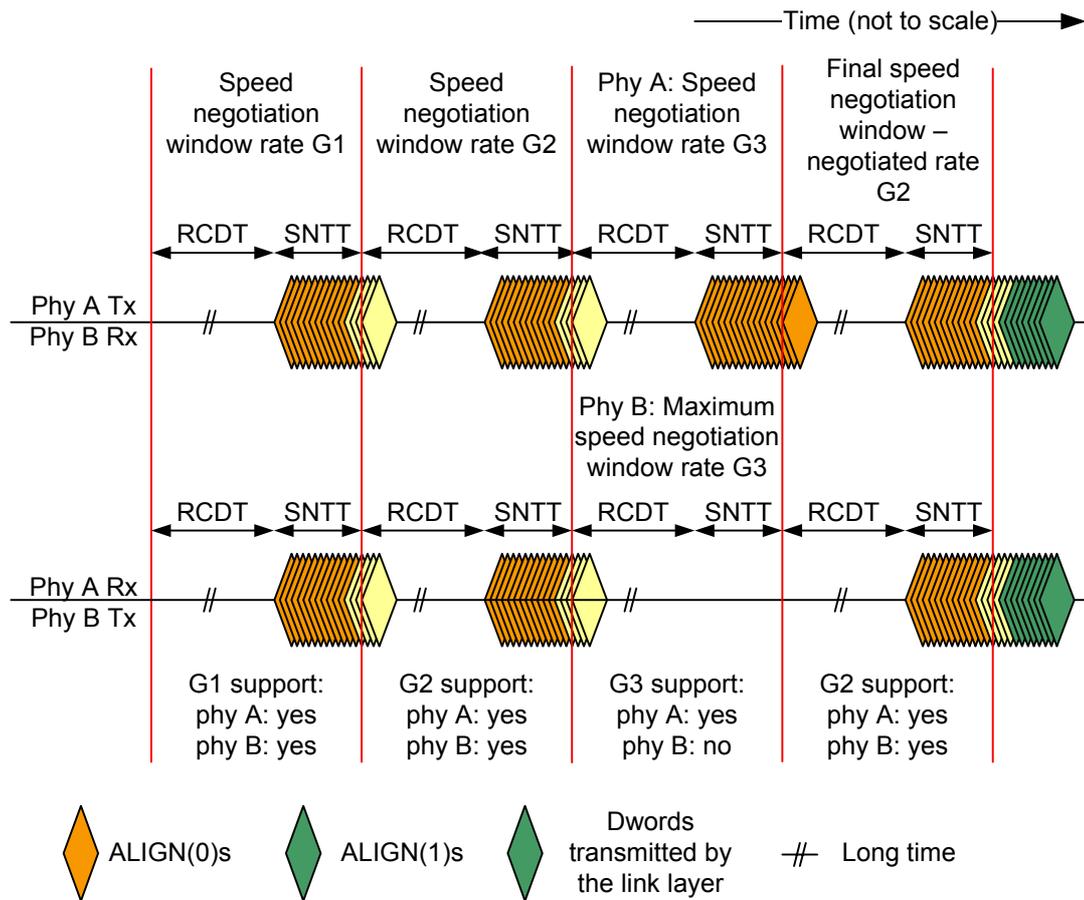
Figure C.3 — SAS speed negotiation sequence (phy A: G1, G2, G3, phy B: G1, G2)

Figure C.4 shows a speed negotiation between a phy A that supports G2 through G3 attached to a phy B that only supports G1 and G2. Both phys run:

1) the G1 speed negotiation window, supported by phy B but not by phy A;
2) the G2 speed negotiation window, supported by both phys; and
3) the G3 speed negotiation window, supported by phy A but not by phy B.

Both phys then select G2 for the final speed negotiation window used to establish the negotiated physical link rate.
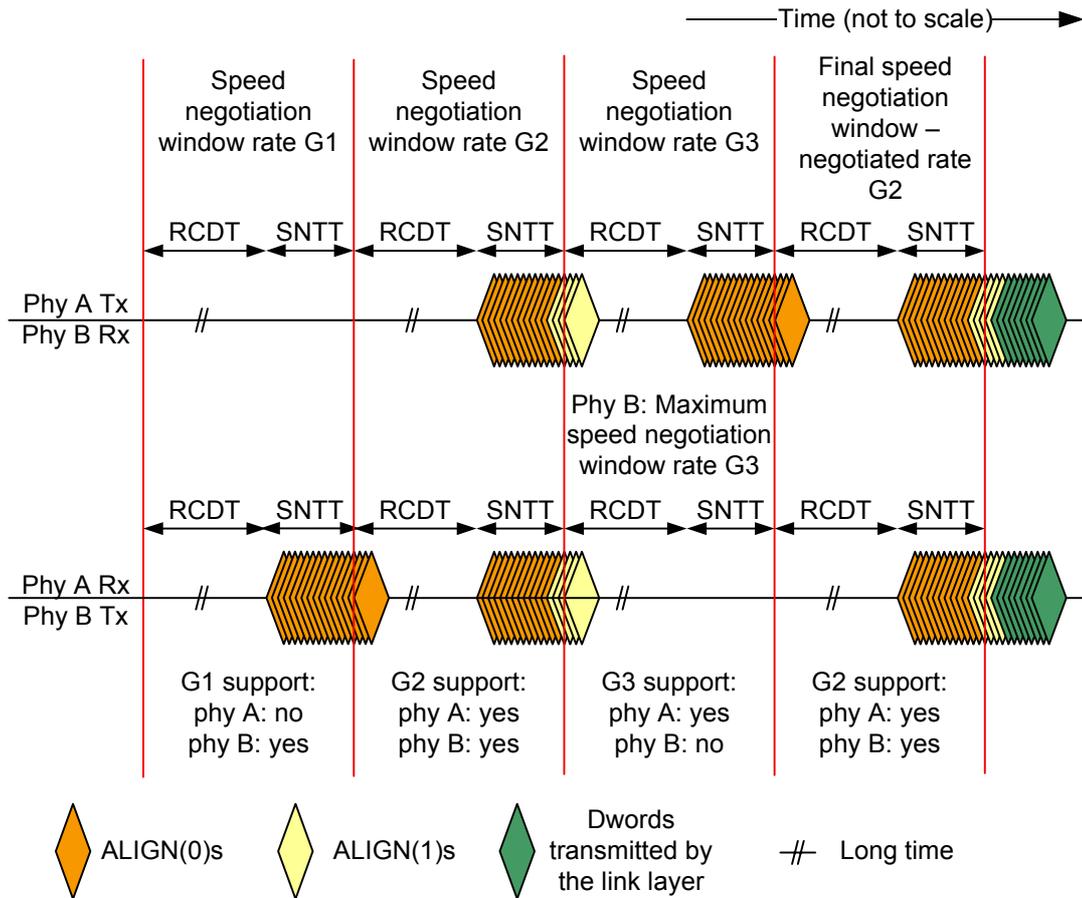
Figure C.4 — SAS speed negotiation sequence (phy A: G2, G3, phy B: G1, G2)

Figure C.5 shows a speed negotiation between a phy A that only supports G1 attached to a phy B that only supports G2. Both phys run:

1) the G1 speed negotiation window, supported by phy A but not by phy B; and
2) the G2 speed negotiation window, supported by phy B but not by phy A.

Phy B continues to run the G3 speed negotiation window, but phy A determines speed negotiation is unsuccessful and may attempt another phy reset sequence after a hot-plug timeout.

Phy B determines speed negotiation is unnecessarily after the G3 window and may retry the phy reset sequence after a hot-plug timeout.

Time (not to scale)

| Speed negotiation window rate G1 | Maximum speed negotiation window rate G2 | Phy A does not participate in the third speed negotiation window. It retries the phy reset sequence. |

RCDT   SNTT   RCDT   SNTT   Hot-plug timeout   COMINIT

Phy A Tx
Phy B Rx

Maximum speed negotiation window rate G3

RCDT   SNTT   RCDT   SNTT   RCDT   SNTT   Hot-plug timeout   COMINIT

Phy A Rx
Phy B Tx

G1 support:
phy A: yes
phy B: no

G2 support:
phy A: no
phy B: yes

G3 support:
phy A: N/A
phy B: no

ALIGN(0)s   ALIGN(1)s   Long time

Figure C.5 — SAS speed negotiation sequence (phy A: G1 only, phy B: G2 only)

Annex D
(informative)

CRC

D.1 CRC generator and checker implementation examples

Figure D.1 shows an example of a CRC generator.
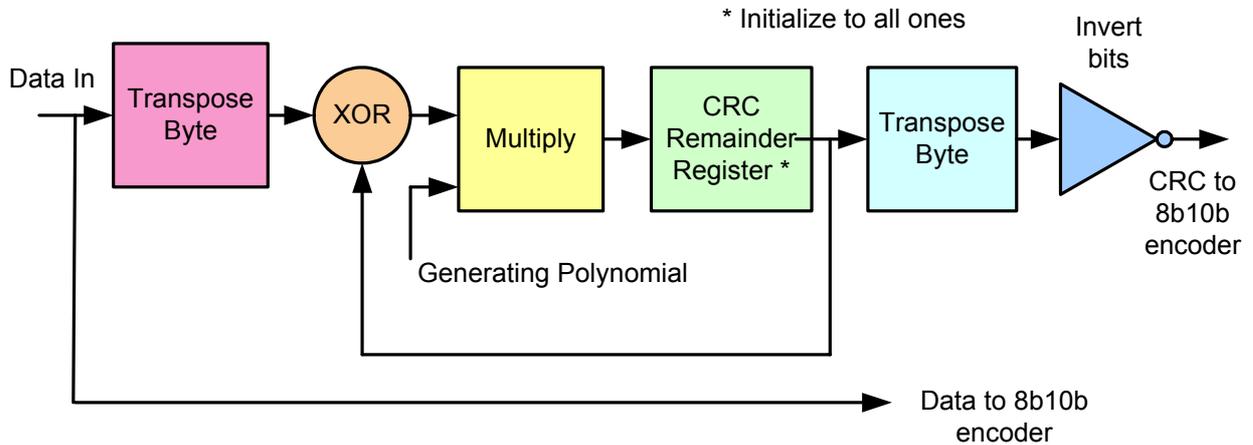
Figure D.1 — CRC generator example

Figure D.2 shows an example of a CRC checker.
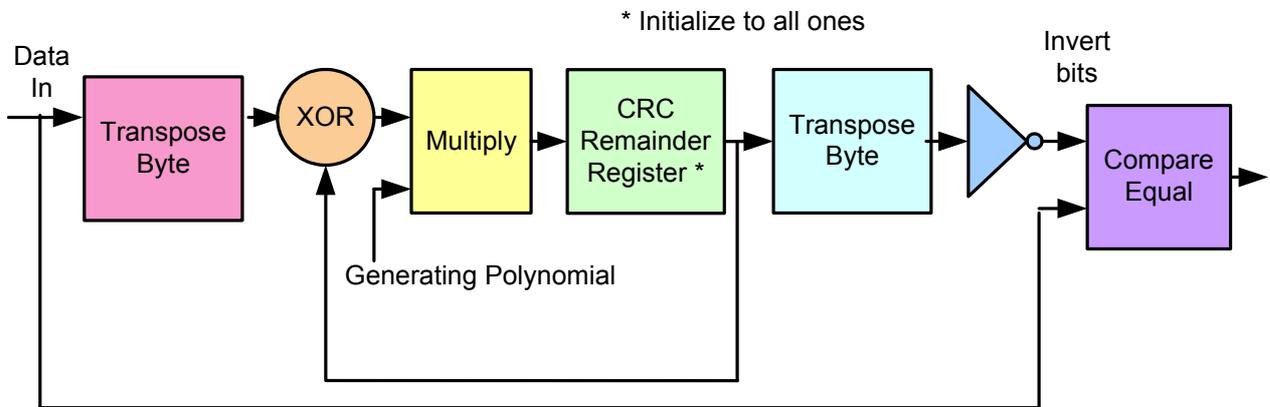
Figure D.2 — CRC checker example

D.2 CRC implementation in C

The following is an example C program that generates the value for the CRC field in frames. The inputs are the data dwords for the frame and the number of data dwords.

```
#include <stdio.h>

void main (void) {

        static unsigned long data_dwords[] = {
                0x06D0B992L, 0x00B5DF59L, 0x00000000L,
                0x00000000L, 0x1234FFFFL, 0x00000000L,
                0x00000000L, 0x00000000L, 0x00000000L,
                0x08000012L, 0x01000000L, 0x00000000L,
                0x00000000L};  /* example data dwords */

        unsigned long calculate_crc(unsigned long *, unsigned long);
```

```
        unsigned long crc;

        crc = calculate_crc(data_dwords, 13);
        printf ("Example CRC is %x\n", crc);
}


/* returns crc value */
unsigned long calculate_crc(unsigned long *frame, unsigned long length) {
long poly = 0x04C11DB7L;
unsigned long crc_gen, x;
union {
        unsigned long lword;
        unsigned char byte[4];
} b_access;
static unsigned char xpose[] = {
                0x0, 0x8, 0x4, 0xC, 0x2, 0xA, 0x6, 0xE,
                0x1, 0x9, 0x5, 0xD, 0x3, 0xB, 0x7, 0xF};
unsigned int i, j, fb;

crc_gen = ~0;   /* seed generator with all ones */
for (i = 0; i < length; i++) {
        x = *frame++; /* get word */
          b_access.lword = x; /* transpose bits in byte */
          for (j = 0; j < 4; j++) {
              b_access.byte[j] = xpose[b_access.byte[j] >> 4] |
                        xpose[b_access.byte[j] & 0xF] << 4;
          }
          x = b_access.lword;

          for (j = 0; j < 32; j++) { /* serial shift register implementation */
              fb = ((x & 0x80000000L) > 0) ^ ((crc_gen & 0x80000000L) > 0);
              x <<= 1;
              crc_gen <<= 1;
              if (fb)
                   crc_gen ^= poly;
      }
}

b_access.lword = crc_gen; /* transpose bits in CRC */
for (j = 0; j < 4; j++) {
        b_access.byte[j] = xpose[b_access.byte[j] >> 4] |
                    xpose[b_access.byte[j] & 0xF] << 4;
}
crc_gen = b_access.lword;

return ~crc_gen; /* invert output */
}
```

D.3 CRC implementation with XORs

These equations implement the multiply function shown in figure D.1 and figure D.2. The ^ symbol represents an XOR operation.

```
crc00 = d00^d06^d09^d10^d12^d16^d24^d25^d26^d28^d29^d30^d31;
crc01 = d00^d01^d06^d07^d09^d11^d12^d13^d16^d17^d24^d27^d28;
crc02 = d00^d01^d02^d06^d07^d08^d09^d13^d14^d16^d17^d18^d24^d26^d30^d31;
crc03 = d01^d02^d03^d07^d08^d09^d10^d14^d15^d17^d18^d19^d25^d27^d31;
crc04 = d00^d02^d03^d04^d06^d08^d11^d12^d15^d18^d19^d20^d24^d25^d29^d30^d31;
crc05 = d00^d01^d03^d04^d05^d06^d07^d10^d13^d19^d20^d21^d24^d28^d29;
crc06 = d01^d02^d04^d05^d06^d07^d08^d11^d14^d20^d21^d22^d25^d29^d30;
crc07 = d00^d02^d03^d05^d07^d08^d10^d15^d16^d21^d22^d23^d24^d25^d28^d29;
```

```
crc08 = d00^d01^d03^d04^d08^d10^d11^d12^d17^d22^d23^d28^d31;
crc09 = d01^d02^d04^d05^d09^d11^d12^d13^d18^d23^d24^d29;
crc10 = d00^d02^d03^d05^d09^d13^d14^d16^d19^d26^d28^d29^d31;
crc11 = d00^d01^d03^d04^d09^d12^d14^d15^d16^d17^d20^d24^d25^d26^d27^d28^d31;
crc12 = d00^d01^d02^d04^d05^d06^d09^d12^d13^d15^d17^d18^d21^d24^d27^d30^d31;
crc13 = d01^d02^d03^d05^d06^d07^d10^d13^d14^d16^d18^d19^d22^d25^d28^d31;
crc14 = d02^d03^d04^d06^d07^d08^d11^d14^d15^d17^d19^d20^d23^d26^d29;
crc15 = d03^d04^d05^d07^d08^d09^d12^d15^d16^d18^d20^d21^d24^d27^d30;
crc16 = d00^d04^d05^d08^d12^d13^d17^d19^d21^d22^d24^d26^d29^d30;
crc17 = d01^d05^d06^d09^d13^d14^d18^d20^d22^d23^d25^d27^d30^d31;
crc18 = d02^d06^d07^d10^d14^d15^d19^d21^d23^d24^d26^d28^d31;
crc19 = d03^d07^d08^d11^d15^d16^d20^d22^d24^d25^d27^d29;
crc20 = d04^d08^d09^d12^d16^d17^d21^d23^d25^d26^d28^d30;
crc21 = d05^d09^d10^d13^d17^d18^d22^d24^d26^d27^d29^d31;
crc22 = d00^d09^d11^d12^d14^d16^d18^d19^d23^d24^d26^d27^d29^d31;
crc23 = d00^d01^d06^d09^d13^d15^d16^d17^d19^d20^d26^d27^d29^d31;
crc24 = d01^d02^d07^d10^d14^d16^d17^d18^d20^d21^d27^d28^d30;
crc25 = d02^d03^d08^d11^d15^d17^d18^d19^d21^d22^d28^d29^d31;
crc26 = d00^d03^d04^d06^d10^d18^d19^d20^d22^d23^d24^d25^d26^d28^d31;
crc27 = d01^d04^d05^d07^d11^d19^d20^d21^d23^d24^d25^d26^d27^d29;
crc28 = d02^d05^d06^d08^d12^d20^d21^d22^d24^d25^d26^d27^d28^d30;
crc29 = d03^d06^d07^d09^d13^d21^d22^d23^d25^d26^d27^d28^d29^d31;
crc30 = d04^d07^d08^d10^d14^d22^d23^d24^d26^d27^d28^d29^d30;
crc31 = d05^d08^d09^d11^d15^d23^d24^d25^d27^d28^d29^d30^d31;
```

D.4 CRC examples

Table D.1 shows several CRC examples. Data is shown in dwords, from first to last.

Table D.1 — CRC examples

Frame contents	CRC	Frame contents	CRC
<SOF> 00010203h 04050607h 08090A0Bh 0C0D0E0Fh 10111213h 14151617h 18191A1Bh 1C1D1E1Fh <CRC> <EOF>	8A7E2691h	<SOF> 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000001h <CRC> <EOF>	3B650D6Eh
<SOF> 00000001h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h 00000000h <CRC> <EOF>	898C0D7Ah	<SOF> 06D0B992h 00B5DF59h 00000000h 00000000h 1234FFFFh 00000000h 00000000h 00000000h 08000012h 01000000h 00000000h 00000000h <CRC> <EOF>	3F4F1C26h

Annex E
(informative)

SAS address hashing

E.1 SAS address hashing overview

See 4.2.2 for a description of hashed SAS addresses and the algorithm used to create them.

E.2 Hash collision probability

The following are Monte-Carlo simulations evaluating the probability of collision in a system containing 128 addressable SAS ports. Four models were used for the models for the simulations:

   a)   random model;
   b)   sequential mode;
   c)   lots model; and
   d)   three lots model.

The random model uses a system with 128 randomly chosen 64-bit integers as SAS addresses.

The sequential model uses a system with 128 sequentially-assigned SAS addresses starting from a random 64-bit base.

The lots model uses:

   a)   Two sequentially assigned SAS addresses with unique company IDs and random vendor-specific identifiers;
   b)   125 randomly drawn SAS addresses from a 10 000-unit production lot. The vendor-specific identifiers within the lot were assigned by 10 SAS address-writers, randomly drawn from a pool of 4 096 possible SAS address-writers. Each SAS address-writer assigns vendor-specific identifiers sequentially within its own subset of the vendor-specific identifiers, starting from a randomly chosen base at the beginning of the production run; and
   c)   One randomly chosen SAS address with another unique company ID, representing a replacement unit.

The three lots model uses:

   a)   Two sequentially assigned SAS addresses with unique company IDs and random vendor-specific identifiers;
   b)   125 randomly drawn SAS addresses from three 10 000-unit lots. The vendor-specific identifiers within each lot were assigned by 10 SAS address-writers, randomly drawn from a pool of 4 096 possible SAS address-writers for that vendor. Each SAS address-writer assigns vendor-specific identifiers sequentially within its own subset of the vendor-specific identifiers, starting from a randomly chosen base at the beginning of the production run. Each of the three lots has a different company ID; and
   c)   One randomly chosen SAS address with another unique company ID, representing a replacement unit.

Table E.1 lists the results of Monte-Carlo simulation.

Table E.1 — Monte-Carlo simulation results

SAS address model	Trials	Collisions	Average collisions per system
lots	2 000 000 000	45 063	0,000 022 531 5
three lots	2 000 000 000	662 503	0,000 331 251 5
random	10 000 000	4 882	0,000 488 2
sequential	10 000 000	0	0

E.3 Hash generation

One way to implement the hashing encoder in hardware is to use serial shift registers as shown in figure E.1. For error correction purposes, the number of data bits is limited to 39. For hashing purposes, the circuit shown serves as a divider. Because the period of this generator polynomial is 63, any binary sequence of length exceeding 63 is

treated as a 63-bit sequence with (bit 63) $\times$ 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 is treated as an equivalent binary sequence of 63 bits, which, in turn, is treated as a degree-62 polynomial. After feeding this equivalent degree-62 polynomial into the circuit shown, the shift register contains the remainder from dividing the degree-62 input polynomial by the generator polynomial. This remainder is the hashed result.
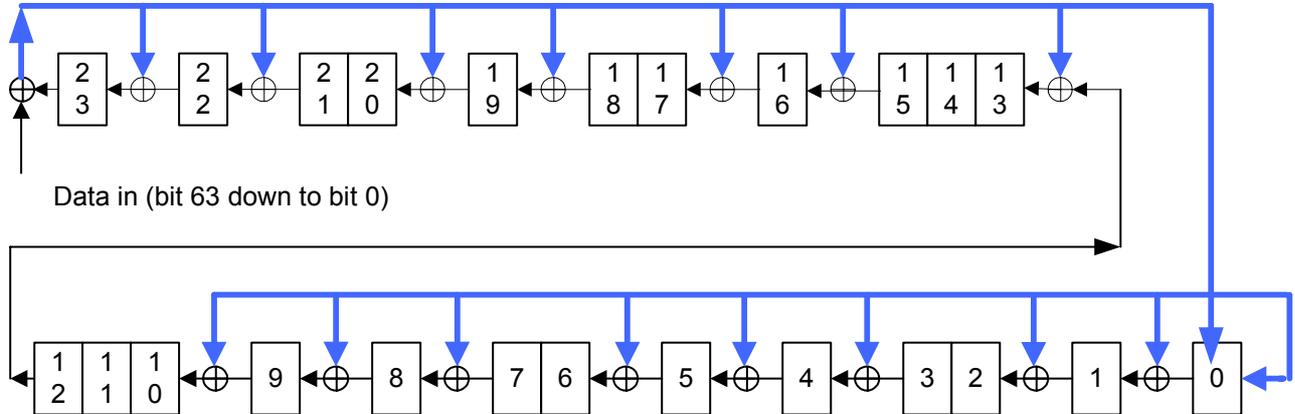
Figure E.1 — BCH(69, 39, 9) code generator

E.4 Hash implementation in C

The following is an example C program that generates a 24-bit hashed value from a 64-bit value.

```c
typedef unsigned int uint32_t;
uint32_t hash(uint32_t upperbits, uint32_t lowerbits)
{
      const unsigned distance_9_poly = 0x01DB2777;
      uint32_t msb = 0x01000000;
      uint32_t moving_one, leading_bit;
      int i;
      unsigned regg;
      regg = 0;
      moving_one = 0x80000000;
      for (i = 31; i >= 0; i--) {
            leading_bit = 0;
            if (moving_one & upperbits) leading_bit = msb;
            regg <<= 1;
            regg ^= leading_bit;
            if (regg & msb) regg ^= distance_9_poly;
            moving_one >>= 1;
      }
      moving_one = 0x80000000;
      for (i = 31; i >= 0; i--) { // note lower limit of i = 0;
            leading_bit = 0;
            if (moving_one & lowerbits) leading_bit = msb;
            regg <<= 1;
            regg ^= leading_bit;
            if (regg & msb) regg ^= distance_9_poly;
            moving_one >>= 1;
      }
      return regg & 0x00FFFFFF;
}
```

E.5 Hash implementation with XORs

These equations generate the 24-bit hashed SAS address for the SSP frame header from a 64-bit SAS address. The ^ symbol represents an XOR.

```
hash00=d00^d01^d03^d05^d07^d09^d10^d11^d12^d15^d16^d17^d18^d19^d20^d21^d22^d23
      ^d24^d25^d28^d30^d31^d33^d34^d36^d38^d39^d63;
hash01=d00^d02^d03^d04^d05^d06^d07^d08^d09^d13^d15^d26^d28^d29^d30^d32^d33^d35
      ^d36^d37^d38^d40^d63;
hash02=d00^d04^d06^d08^d11^d12^d14^d15^d17^d18^d19^d20^d21^d22^d23^d24^d25^d27
      ^d28^d29^d37^d41^d63;
hash03=d01^d05^d07^d09^d12^d13^d15^d16^d18^d19^d20^d21^d22^d23^d24^d25^d26^d28
      ^d29^d30^d38^d42;
hash04=d00^d01^d02^d03^d05^d06^d07^d08^d09^d11^d12^d13^d14^d15^d18^d26^d27^d28
      ^d29^d33^d34^d36^d38^d43^d63;
hash05=d00^d02^d04^d05^d06^d08^d11^d13^d14^d17^d18^d20^d21^d22^d23^d24^d25^d27
      ^d29^d31^d33^d35^d36^d37^d38^d44^d63;
hash06=d00^d06^d10^d11^d14^d16^d17^d20^d26^d31^d32^d33^d37^d45^d63;
hash07=d01^d07^d11^d12^d15^d17^d18^d21^d27^d32^d33^d34^d38^d46;
hash08=d00^d01^d02^d03^d05^d07^d08^d09^d10^d11^d13^d15^d17^d20^d21^d23^d24^d25
      ^d30^d31^d35^d36^d38^d47^d63;
hash09=d00^d02^d04^d05^d06^d07^d08^d14^d15^d17^d19^d20^d23^d26^d28^d30^d32^d33
      ^d34^d37^d38^d48^d63;
hash10=d00^d06^d08^d10^d11^d12^d17^d19^d22^d23^d25^d27^d28^d29^d30^d35^d36^d49
      ^d63;
hash11=d01^d07^d09^d11^d12^d13^d18^d20^d23^d24^d26^d28^d29^d30^d31^d36^d37^d50
      ;
hash12=d02^d08^d10^d12^d13^d14^d19^d21^d24^d25^d27^d29^d30^d31^d32^d37^d38^d51
      ;
hash13=d00^d01^d05^d07^d10^d12^d13^d14^d16^d17^d18^d19^d21^d23^d24^d26^d32^d34
      ^d36^d52^d63;
hash14=d01^d02^d06^d08^d11^d13^d14^d15^d17^d18^d19^d20^d22^d24^d25^d27^d33^d35
      ^d37^d53;
hash15=d02^d03^d07^d09^d12^d14^d15^d16^d18^d19^d20^d21^d23^d25^d26^d28^d34^d36
      ^d38^d54;
hash16=d00^d01^d04^d05^d07^d08^d09^d11^d12^d13^d18^d23^d25^d26^d27^d28^d29^d30
      ^d31^d33^d34^d35^d36^d37^d38^d55^d63;
hash17=d00^d02^d03^d06^d07^d08^d11^d13^d14^d15^d16^d17^d18^d20^d21^d22^d23^d25
      ^d26^d27^d29^d32^d33^d35^d37^d56^d63;
hash18=d01^d03^d04^d07^d08^d09^d12^d14^d15^d16^d17^d18^d19^d21^d22^d23^d24^d26
      ^d27^d28^d30^d33^d34^d36^d38^d57;
hash19=d00^d01^d02^d03^d04^d07^d08^d11^d12^d13^d21^d27^d29^d30^d33^d35^d36^d37
      ^d38^d58^d63;
hash20=d00^d02^d04^d07^d08^d10^d11^d13^d14^d15^d16^d17^d18^d19^d20^d21^d23^d24
      ^d25^d33^d37^d59^d63;
hash21=d01^d03^d05^d08^d09^d11^d12^d14^d15^d16^d17^d18^d19^d20^d21^d22^d24^d25
      ^d26^d34^d38^d60;
hash22=d00^d01^d02^d03^d04^d05^d06^d07^d11^d13^d24^d26^d27^d28^d30^d31^d33^d34
      ^d35^d36^d38^d61^d63;
hash23=d00^d02^d04^d06^d08^d09^d10^d11^d14^d15^d16^d17^d18^d19^d20^d21^d22^d23
      ^d24^d27^d29^d30^d32^d33^d35^d37^d38^d62^d63;
```

E.6 Hash examples

Table E.2 shows examples using simple SAS addresses as input values. Two of the input values hash to the same value.

Table E.2 — Hash results for simple SAS addresses

64-bit input value	24-bit hashed value
00000000_00000000h	000000h
00000000_00000001h	DB2777h
FFFFFFFF_FFFFFFFFh	DB2777h

Table E.3 shows examples using realistic SAS addresses as input values.

Table E.3 — Hash results for realistic SAS addresses

64-bit input value	24-bit hashed value
50010753_4F0CFC88h	D0B992h
50010B92_B3CBF639h	B5DF59h
5002037E_157FEC63h	B064F7h
50004CF6_FBCE3889h	88FF12h
50020374_C4657EC7h	F36570h
50010D92_A016E450h	9F9571h
50002A58_850ACC66h	64B6B9h
50008C7B_EE7910DEh	8D6135h
500508BD_C22CAC94h	86ECF1h
500805F3_334B0AD3h	752AB2h
500A0B8A_FAA6A820h	5543A7h
500805E6_BCC55C68h	463DEDh

Table E.4 shows examples using a walking ones pattern to generate the input values.

Table E.4 — Hash results for a walking ones pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
0000000000000001h	DB2777h	0000000100000000h	8232C2h
0000000000000002h	6D6999h	0000000200000000h	DF42F3h
0000000000000004h	DAD332h	0000000400000000h	65A291h
0000000000000008h	6E8113h	0000000800000000h	CB4522h
0000000000000010h	DD0226h	0000001000000000h	4DAD33h
0000000000000020h	61233Bh	0000002000000000h	9B5A66h
0000000000000040h	C24676h	0000004000000000h	ED93BBh
0000000000000080h	5FAB9Bh	0000008000000000h	000001h
0000000000000100h	BF5736h	0000010000000000h	000002h
0000000000000200h	A5891Bh	0000020000000000h	000004h
0000000000000400h	903541h	0000040000000000h	000008h
0000000000000800h	FB4DF5h	0000080000000000h	000010h
0000000000001000h	2DBC9Dh	0000100000000000h	000020h
0000000000002000h	5B793Ah	0000200000000000h	000040h
0000000000004000h	B6F274h	0000400000000000h	000080h
0000000000008000h	B6C39Fh	0000800000000000h	000100h
0000000000010000h	B6A049h	0001000000000000h	000200h
0000000000020000h	B667E5h	0002000000000000h	000400h
0000000000040000h	B7E8BDh	0004000000000000h	000800h
0000000000080000h	B4F60Dh	0008000000000000h	001000h
0000000000100000h	B2CB6Dh	0010000000000000h	002000h
0000000000200000h	BEB1ADh	0020000000000000h	004000h
0000000000400000h	A6442Dh	0040000000000000h	008000h
0000000000800000h	97AF2Dh	0080000000000000h	010000h
0000000001000000h	F4792Dh	0100000000000000h	020000h
0000000002000000h	33D52Dh	0200000000000000h	040000h
0000000004000000h	67AA5Ah	0400000000000000h	080000h
0000000008000000h	CF54B4h	0800000000000000h	100000h
0000000010000000h	458E1Fh	1000000000000000h	200000h
0000000020000000h	8B1C3Eh	2000000000000000h	400000h
0000000040000000h	CD1F0Bh	4000000000000000h	800000h
0000000080000000h	411961h	8000000000000000h	DB2777h

Table E.5 shows examples using a walking zeros pattern to generate the input values.

Table E.5 — Hash results for a walking zeros pattern

64-bit input value	24-bit hashed value	64-bit input value	24-bit hashed value
FFFFFFFFFFFFFFFEh	000000h	FFFFFFFEFFFFFFFFh	5915B5h
FFFFFFFFFFFFFFFDh	B64EEEh	FFFFFFFDFFFFFFFFh	046584h
FFFFFFFFFFFFFFBh	01F445h	FFFFFFFBFFFFFFFFh	BE85E6h
FFFFFFFFFFFFFFF7h	B5A664h	FFFFFFF7FFFFFFFFh	106255h
FFFFFFFFFFFFFFEFh	062551h	FFFFFFEFFFFFFFFFh	968A44h
FFFFFFFFFFFFFFDFh	BA044Ch	FFFFFFDFFFFFFFFFh	407D11h
FFFFFFFFFFFFFFBFh	196101h	FFFFFFBFFFFFFFFFh	36B4CCh
FFFFFFFFFFFFFF7Fh	848CECh	FFFFFF7FFFFFFFFFh	DB2776h
FFFFFFFFFFFFFEFFh	647041h	FFFFFEFFFFFFFFFFh	DB2775h
FFFFFFFFFFFFFDFFh	7EAE6Ch	FFFFFDFFFFFFFFFFh	DB2773h
FFFFFFFFFFFFFBFFh	4B1236h	FFFFFBFFFFFFFFFFh	DB277Fh
FFFFFFFFFFFFF7FFh	206A82h	FFFFF7FFFFFFFFFFh	DB2767h
FFFFFFFFFFFFEFFFh	F69BEAh	FFFFEFFFFFFFFFFFh	DB2757h
FFFFFFFFFFFFDFFFh	805E4Dh	FFFFDFFFFFFFFFFFh	DB2737h
FFFFFFFFFFFFBFFFh	6DD503h	FFFFBFFFFFFFFFFFh	DB27F7h
FFFFFFFFFFFF7FFFh	6DE4E8h	FFFF7FFFFFFFFFFFh	DB2677h
FFFFFFFFFFFEFFFFh	6D873Eh	FFFEFFFFFFFFFFFFh	DB2577h
FFFFFFFFFFFDFFFFh	6D4092h	FFFDFFFFFFFFFFFFh	DB2377h
FFFFFFFFFFFBFFFFh	6CCFCAh	FFFBFFFFFFFFFFFFh	DB2F77h
FFFFFFFFFFF7FFFFh	6FD17Ah	FFF7FFFFFFFFFFFFh	DB3777h
FFFFFFFFFFEFFFFFh	69EC1Ah	FFEFFFFFFFFFFFFFh	DB0777h
FFFFFFFFFFDFFFFFh	6596DAh	FFDFFFFFFFFFFFFFh	DB6777h
FFFFFFFFFFBFFFFFh	7D635Ah	FFBFFFFFFFFFFFFFh	DBA777h
FFFFFFFFFF7FFFFFh	4C885Ah	FF7FFFFFFFFFFFFFh	DA2777h
FFFFFFFFFEFFFFFFh	2F5E5Ah	FEFFFFFFFFFFFFFFh	D92777h
FFFFFFFFFDFFFFFFh	E8F25Ah	FDFFFFFFFFFFFFFFh	DF2777h
FFFFFFFFFBFFFFFFh	BC8D2Dh	FBFFFFFFFFFFFFFFh	D32777h
FFFFFFFFF7FFFFFFh	1473C3h	F7FFFFFFFFFFFFFFh	CB2777h
FFFFFFFFEFFFFFFFh	9EA968h	EFFFFFFFFFFFFFFFh	FB2777h
FFFFFFFFDFFFFFFFh	503B49h	DFFFFFFFFFFFFFFFh	9B2777h
FFFFFFFFBFFFFFFFh	16387Ch	BFFFFFFFFFFFFFFFh	5B2777h
FFFFFFFF7FFFFFFFh	9A3E16h	7FFFFFFFFFFFFFFFh	000000h

Annex F
(informative)

Scrambling

F.1 Scrambler implementation example

Figure F.1 shows an example of a data scrambler. This example generates the value to XOR with the dword input with two 16 bit parallel multipliers. 16 bits wide is the maximum width for the multiplier as the generating polynomial is 16 bits.
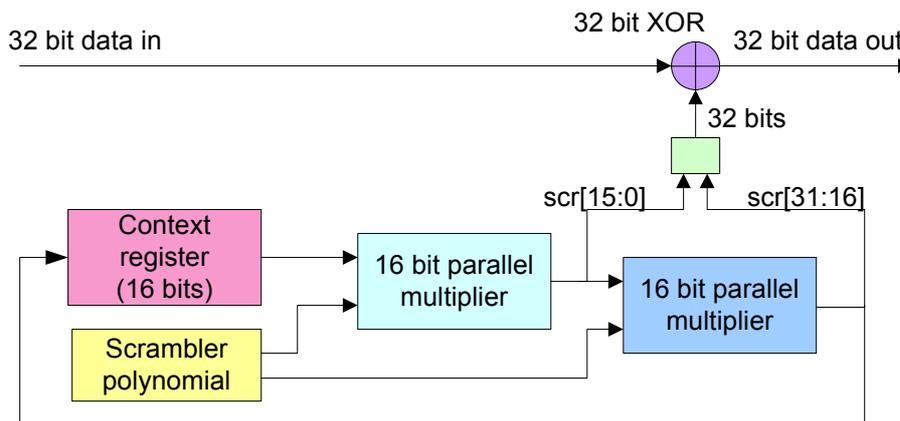
Figure F.1 — Scrambler

The generator polynomial is:

$$G(x) = x^{16} + x^{15} + x^{13} + x^4 + 1$$

For all implementations, the context register is initialized to produce a first dword output of C2D2768Dh for a dword input of all zeros.

F.2 Scrambler implementation in C

The following is an example C program that generates the scrambled data dwords for transmission. The inputs are the data dword to scramble and control indication to reinitialize the residual value (e.g., following an SOF).

```
#include <stdio.h>

unsigned long scramble(int reset, unsigned long dword);
void main(void)
{
int i;

for (i  = 0; i < 12; i++)
     printf(" %08X \n",scramble(i==0, 0));/* scramble all 0s */
}

#define poly 0xA011
unsigned long scramble(int reset, unsigned long dword) {
static unsigned short scramble;
int i;

  if (reset)
    scramble = 0xFFFF;
  for (i = 0; i < 32; i++)  /* serial shift register implementation */
  {
      dword ^= (scramble & 0x8000)? (1 << i):0;
      scramble = (scramble << 1) ^ ((scramble & 0x8000)? poly:0);
```

```
      }
      return dword;
   }
```

F.3 Scrambler implementation with XORs

These equations generate the scrambled dwords to XOR with dwords before transmission and dword reception to recover the original data. The ^ symbol represents an XOR operation. The initialized value for d[15:0] is F0F6h (i.e., 0xF0F6) in this example.

```
   scr0=d15^d13^d4^d0;
   scr1=d15^d14^d13^d5^d4^d1^d0;
   scr2=d14^d13^d6^d5^d4^d2^d1^d0;
   scr3=d15^d14^d7^d6^d5^d3^d2^d1;
   scr4=d13^d8^d7^d6^d3^d2^d0;
   scr5=d14^d9^d8^d7^d4^d3^d1;
   scr6=d15^d10^d9^d8^d5^d4^d2;
   scr7=d15^d13^d11^d10^d9^d6^d5^d4^d3^d0;
   scr8=d15^d14^d13^d12^d11^d10^d7^d6^d5^d1^d0;
   scr9=d14^d12^d11^d8^d7^d6^d4^d2^d1^d0;
   scr10=d15^d13^d12^d9^d8^d7^d5^d3^d2^d1;
   scr11=d15^d14^d10^d9^d8^d6^d3^d2^d0;
   scr12=d13^d11^d10^d9^d7^d3^d1^d0;
   scr13=d14^d12^d11^d10^d8^d4^d2^d1;
   scr14=d15^d13^d12^d11^d9^d5^d3^d2;
   scr15=d15^d14^d12^d10^d6^d3^d0;
   scr16=d11^d7^d1^d0;
   scr17=d12^d8^d2^d1;
   scr18=d13^d9^d3^d2;
   scr19=d14^d10^d4^d3;
   scr20=d15^d11^d5^d4;
   scr21=d15^d13^d12^d6^d5^d4^d0;
   scr22=d15^d14^d7^d6^d5^d4^d1^d0;
   scr23=d13^d8^d7^d6^d5^d4^d2^d1^d0;
   scr24=d14^d9^d8^d7^d6^d5^d3^d2^d1;
   scr25=d15^d10^d9^d8^d7^d6^d4^d3^d2;
   scr26=d15^d13^d11^d10^d9^d8^d7^d5^d3^d0;
   scr27=d15^d14^d13^d12^d11^d10^d9^d8^d6^d1^d0;
   scr28=d14^d12^d11^d10^d9^d7^d4^d2^d1^d0;
   scr29=d15^d13^d12^d11^d10^d8^d5^d3^d2^d1;
   scr30=d15^d14^d12^d11^d9^d6^d3^d2^d0;
   scr31=d12^d10^d7^d3^d1^d0;
```

F.4 Scrambler examples

Table F.1 shows several scrambler examples. Data is shown in dwords, from first to last.

Table F.1 — Scrambler examples

Frame contents	Scrambled output
<SOF>	<SOF>
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 [a]	CF79E22Ah [a]
<EOF>	<EOF>
<SOF>	<SOF>
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]
<EOF>	<EOF>
[a] The last dword represents a CRC dword.	

Annex G
(informative)

ATA architectural notes

G.1 STP differences from Serial ATA (SATA)

Some of the differences of STP compared with SATA are:

a) STP adds addressing of multiple SATA devices. Each SATA device is assigned a SAS address by its attached expander device with STP/SATA bridge functionality. The STP initiator port understands addressing more than one STP target port;
b) STP allows multiple STP initiator ports to share access to a SATA device using affiliations (see 7.17.5);
c) interface power management is not supported;
d) far-end analog loopback testing is not supported;
e) far-end retimed loopback testing is not supported;
f) near-end analog loopback testing is not supported;
g) use of SATA_CONT is required; and
h) BIST Activate frames are not supported.

G.2 STP differences from Serial ATA II

The following features of Serial ATA II are specifically excluded from SAS STP or handled differently in a SAS domain:

a) extended differential voltages;
b) enclosure services;
c) staggered spin-up (see 6.10);
d) device activity indication;
e) presence detect; and
f) power management improvements.

G.3 Affiliation policies

G.3.1 Affiliation policies overview

SATA is based on a model that assumes a SATA device is controlled by a single SATA host, and does not address the notion of multiple SATA hosts having the ability to access any given SATA device.

With STP/SATA bridges, SATA devices are cast into an environment where multiple STP initiator ports, by sharing the SATA host port of the STP/SATA bridge, have access to the same SATA device. The SATA protocol used inside STP connections does not account for the possibility that more than one STP initiator port might be vying for access to the SATA device. Affiliations provide a way to ensure contention for a SATA device does not result in incoherent access to the SATA device when commands from different STP initiator ports collide at the SATA device.

To prevent a SATA device from confusing commands from one STP initiator port with commands from another, an STP initiator port needs a means to maintain exclusive access to a SATA or STP device for the duration of the processing of a command.

For example, consider the case where an STP initiator port establishes a connection to send a command (e.g., a read), and then closes the connection while the SATA device (e.g., a disk drive) retrieves the data (e.g., performs a seek operation to the track containing the data). If, after the connection is closed, another STP initiator port is allowed to establish a connection and send another command, the SATA device would no longer have a means to determine which STP initiator port should receive the data when the device requests the connection to send the data for the first command. This is because, unlike SCSI target devices, SATA devices have no notion of multiple SATA hosts.

The consequences are worse for write commands since the result could be wrong data written to media, with the original data being overwritten and permanently lost.

Affiliation provides a means for an STP initiator port to establish atomic access to a SATA device across the processing of a command or series of commands to the SATA device, without requiring the STP initiator port to maintain a connection open to the STP target port for the duration of command processing.

G.3.2 Affiliation policy for static STP initiator port to STP target port mapping

Affiliations should not be used to enforce policies establishing fixed associations between STP initiator ports and STP target ports.

G.3.3 Affiliation policy with SATA queued commands and multiple STP initiator ports

STP initiator ports using queued commands when other STP initiator ports may be accessing the same STP target port should, at vendor-specific intervals, allow commands to complete and release the affiliation to allow other STP initiator ports access to the STP target port.

G.3.4 Applicability of affiliation for STP target ports

Affiliation may or may not be necessary for STP target ports depending on whether the STP target port tracks the STP initiator port's SAS address on each command received. If the STP target port has the means to manage and track commands from each STP initiator port independently, then affiliations are not necessary because the STP target port is capable of associating each information transfer with the appropriate STP initiator port, and is capable of establishing a connection to the appropriate STP initiator port when sending information back for a command.

An STP target port that behaves the same as a SATA device, in that it maintains only a single ATA task file register image to be shared among all STP initiator ports, supports affiliations in order to provide a way for STP initiator ports to maintain exclusive access to the STP target port while commands remain outstanding. In this model, an STP target port is capable of establishing connections to an STP initiator port, but is only capable of remembering the SAS address of the last STP initiator port to establish a connection, and therefore is only capable of requesting a connection back to that same STP initiator port.

See 10.4.3.7 for an explanation of how an STP target port reports support for affiliations.

G.4 SATA port selector considerations

Not all the protocol elements for STP initiator ports to manage a SATA port selector (see SATAII-PS) in a SAS domain are defined in this standard. Additional coordination between STP initiator ports may be needed to avoid conflicting usage of the SATA port selector between STP initiator ports (e.g., between two SAS domains). Such additional coordination is outside the scope of this standard.

G.5 SATA device not transmitting initial Register Device-to-Host FIS

Some SATA devices do not return the initial Register Device-to-Host FIS after a link reset sequence if they did not detect the COMINIT during the link reset sequence (e.g., if the SATA device originated the link reset sequence). While waiting for the initial Register Device-to-Host FIS, an STP/SATA bridge returns:

   a) In the SMP DISCOVER response (see 10.4.3.5):
      A) the ATTACHED DEVICE TYPE field is set to 000b;
      B) the NEGOTIATED PHYSICAL LINK RATE field is set to G1 (i.e., 8h) or G2 (i.e., 9h);
      C) the ATTACHED SATA DEVICE bit is set to one; and
      D) the ATTACHED SAS ADDRESS field is set to the SAS address of the STP target port of the STP/SATA bridge;

      and

   b) OPEN_REJECT (NO DESTINATION) for connection requests to the SAS address of the STP target port.

If an STP initiator port detects this situation for a vendor-specific amount of time, an SMP application client should send an SMP PHY CONTROL function requesting a phy operation of LINK RESET or HARD RESET to originate a new link reset sequence. The SATA device is expected to detect the COMINIT during this link reset sequence and provide the initial Register Device-to-Host FIS.

Annex H
(informative)

ALIGN and/or NOTIFY insertion rate summary

Table H.1 shows all the possible combinations of ALIGN and/or NOTIFY insertion rates for clock skew management (see 7.3), rate matching (see 7.13), and STP initiator phy throttling (see 7.17.2).

Table H.1 — ALIGN and/or NOTIFY insertion rate examples

Physical link rate	Connection rate	Type of dword stream	ALIGN and/or NOTIFY insertion rate (per specified number of dwords)
3,0 Gbit/s	3,0 Gbit/s	all but to STP target	2 per 4 096 (clock skew management)
		to STP target	2 per 4 096 (clock skew management) + 2 per 256 (STP initiator phy throttling)
	1,5 Gbit/s	all but to STP target	2 per 4 096 (clock skew management) + 1 per 2 (rate matching)
		to STP target	2 per 4 096 (clock skew management) + 1 per 2 (rate matching) + 2 per 256 (STP initiator phy throttling)
1,5 Gbit/s	1,5 Gbit/s	all but to STP target	1 per 2 048 (clock skew management)
		to STP target	1 per 2 048 (clock skew management) + 2 per 256 (STP initiator phy throttling)

Annex I
(informative)

Expander device handling of connections

I.1 Expander device handling of connections overview

This annex provides examples of how expander devices process connection requests.

Figure I.1 shows the topology used by examples in this annex.
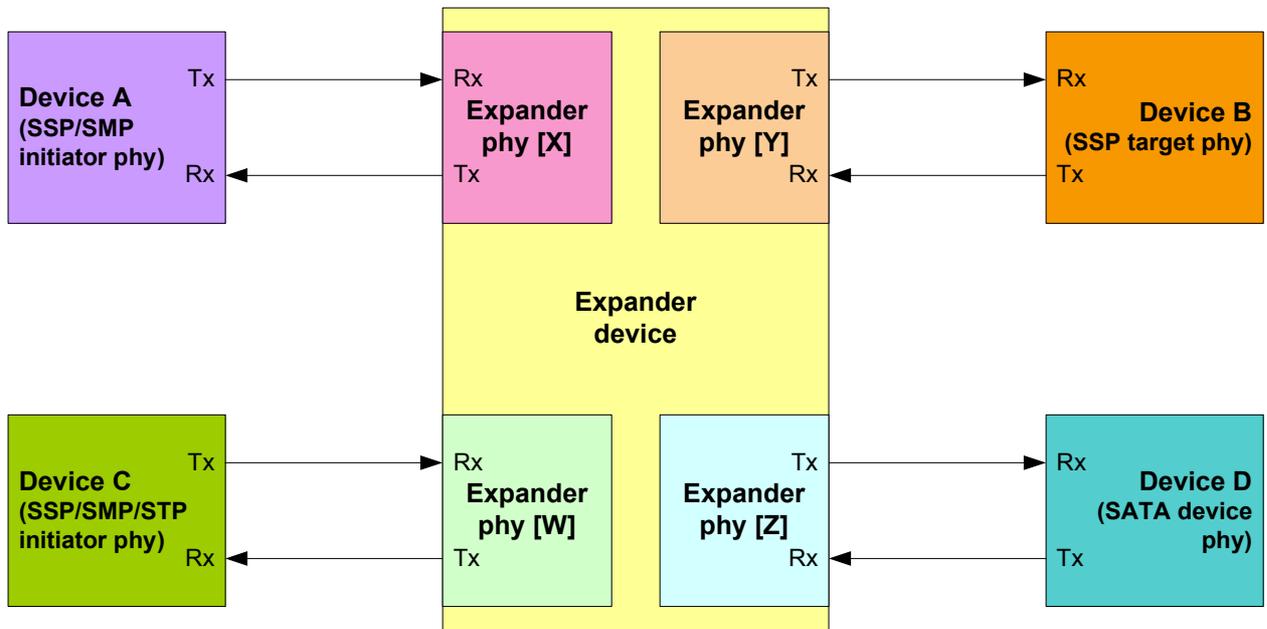
Figure I.1 — Example topology

Table I.1 defines the column headers used within the figures contained within this annex.

Table I.1 — Column descriptions for connection examples

Column header	Description
Phy [W] Rx	Expander phy [W] Receive from device C
Phy [W] Tx	Expander phy [W] Transmit to device C
Phy [W] XL state	Expander phy [W] XL state machine state (see 7.15)
Phy [W] XL req/rsp	Expander phy [W] XL requests and responses (see 4.6.6)
Phy [W] XL cnf/ind	Expander phy [W] XL confirmations and indications (see 4.6.6)
Phy [X] Rx	Expander phy [X] Receive from device A
Phy [X] Tx	Expander phy [X] Transmit to device A
Phy [X] XL state	Expander phy [X] XL state machine state (see 7.15)
Phy [X] XL req/rsp	Expander phy [X] XL requests and responses (see 4.6.6)
Phy [X] XL cnf/ind	Expander phy [X] XL confirmations and indications (see 4.6.6)
Phy [Y] XL cnf/ind	Expander phy [Y] XL confirmations and indications (see 4.6.6)
Phy [Y] XL req/rsp	Expander phy [Y] XL requests and responses (see 4.6.6)
Phy [Y] XL state	Expander phy [Y] XL state machine state (see 7.15)
Phy [Y] Tx	Expander phy [Y] Transmit to device B
Phy [Y] Rx	Expander phy [Y] Receive from device B
Phy [Z] XL cnf/ind	Expander phy [Y] XL confirmations and indications (see 4.6.6)
Phy [Z] XL req/rsp	Expander phy [Y] XL requests and responses (see 4.6.6)
Phy [Z] XL state	Expander phy [Y] XL state machine state (see 7.15)
Phy [Z] Tx	Expander phy [Y] Transmit to device D
Phy [Z] Rx	Expander phy [Y] Receive from device D

I.2 Connection request - OPEN_ACCEPT

Figure I.2 shows the establishment of a successful connection between two end devices.

Figure I.2 — Connection request - OPEN_ACCEPT

I.3 Connection request - OPEN_REJECT by end device

Figure I.3 shows failure to establish a connection due to rejection of the connection request by an end device.

Figure I.3 — Connection request - OPEN_REJECT by end device

I.4 Connection request - OPEN_REJECT by expander device

Figure I.4 shows failure to establish a connection due to rejection of the connection request by an expander device.

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN (A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path						
				Arbitrating (Normal)					
				Arb Reject					
	OPEN_REJECT	XL4:Open_Reject							
	idle dwords	XL0:Idle							

Figure I.4 — Connection request - OPEN_REJECT by expander device

I.5 Connection request - arbitration lost

Figure I.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 I.5 — Connection request - arbitration lost

I.6 Connection request - backoff and retry

Figure I.6 shows a higher priority OPEN address frame (A to B) received by a phy which has previously forwarded an OPEN address frame (B to C) whose source (A) differs from the winning destination (C). In this case expander phy [X] is required to back off and retry path arbitration (see 7.15.9).

Figure I.6 — Connection request - backoff and retry

I.7 Connection request - backoff and reverse path

Figure I.7 shows a higher priority OPEN address frame (B to A) received by a phy which has previously forwarded an OPEN address frame (A to B) whose source (A) matches the winning destination (A). In this case expander phy [Y] forwards the higher priority OPEN to expander phy [X] (see 7.15.9).

Figure I.7 — Connection request - backoff and reverse path

I.8 Connection close - single step

Figure I.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 I.8 — Connection close - single step

I.9 Connection close - simultaneous

Figure I.9 shows two end devices simultaneously transmitting CLOSE to each other.

	Expander phy [X]						Expander phy [Y]						
Rx	**Tx**	**XL state**	**XL req/rsp**	**XL cnf/ind**	**XL cnf/ind**	**XL req/rsp**	**XL state**	**Tx**	**Rx**
connection dwords	connection dwords	XL7:Connected	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	Forward Dword (connection dwords)	XL7:Connected	connection dwords	connection dwords
CLOSE	CLOSE	XL8:Close_Wait	Forward Close	Forward Close	Forward Close	Forward Close	XL8:Close_Wait	CLOSE	CLOSE
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords

Figure I.9 — Connection close - simultaneous

I.10 BREAK handling during path arbitration

Figure I.10 shows an expander device responding to the reception of a BREAK during path arbitration.

Expander phy [X]					Expander phy [Y]				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle					XL0:Idle	idle dwords	idle dwords
SOAF									
OPEN(A to B)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
BREAK	BREAK	XL9:Break							
idle dwords	idle dwords	XL0:Idle							

Figure I.10 — BREAK handling during path arbitration

I.11 BREAK handling during connection

Figure I.11 shows an expander device responding to the reception of a BREAK during a connection.

Figure I.11 — BREAK handling during a connection

I.12 STP connection - originated by STP initiator port

Figure I.12 shows an STP initiator port originating a connection to an STP target port in an STP/SATA bridge.

Expander phy [W] - STP target port in an STP/SATA bridge					Expander phy [Z] - SATA host port in an STP/SATA bridge				
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
idle dwords	idle dwords	XL0:Idle						SYNC/CONT	SATA device dwords
SOAF									
OPEN (C to D)									
EOAF									
idle dwords	AIP (NORMAL) and/or idle dwords	XL1: Request_Path	Request Path	Arbitrating (Normal)					
		XL2: Request_Open	Forward Open	Arb Won	Forward Open				
		XL3: Open_Cnf_Wait	Forward Dword (idle dwords)		Forward Dword (idle dwords)				
	AIP (WAITING ON DEVICE)			Arb Status (Waiting On Device)	Arb Status (Waiting On Device)				
	idle dwords			Open Accept	Open Accept				
	OPEN_ACCEPT			Forward Dword (SATA device dwords [1])	Forward Dword (SATA device dwords [1])				
	SATA device dwords [1]	XL7:Connected	Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords)	Forward Dword (SATA device dwords)	Forward Dword (STP connection dwords)			
STP connection dwords	SATA device dwords							STP connection dwords	

[1] STP/SATA bridge duplicates the dword stream which is being received from the SATA device before forwarding dwords - this ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

Figure I.12 — STP connection - originated by STP initiator port

I.13 STP connection - originated by STP target port in an STP/SATA bridge

Figure I.13 shows an STP target port in an STP/SATA bridge originating a connection on behalf of a SATA device which is requesting to transmit a frame.

Figure I.13 — STP connection - originated by STP target port in an STP/SATA bridge

[a] STP/SATA bridge duplicates the dword stream which is being received from the SATA device before forwarding dwords. This ensures that a continued SATA primitive is correctly forwarded to the STP initiator port.

I.14 STP connection close - originated by STP initiator port

Figure I.14 shows an STP initiator port closing a connection to an STP target port in an STP/SATA bridge.

| Expander phy [W] - STP target port in an STP/SATA bridge | | | | | Expander phy [Z] - SATA host port in an STP/SATA bridge | | | | |
Rx	Tx	XL state	XL req/rsp	XL cnf/ind	XL cnf/ind	XL req/rsp	XL state	Tx	Rx
STP connection dwords	SATA device dwords	XL7:Connected	Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords)	Forward Dword (STP connection dwords)	Forward Dword (SATA device dwords)		STP connection dwords	SATA device dwords
SYNC/CONT	SYNC/CONT					Forward Dword (SYNC/CONT)			SYNC/CONT
			Forward Dword (SYNC/CONT)	Forward Close					
CLOSE	CLOSE	XL8:Close_Wait	Forward Close		Forward Close	Forward Close		SYNC/CONT	
idle dwords	idle dwords	XL0:Idle							

Figure I.14 — STP connection close - originated by STP initiator port

I.15 STP connection close - originated by STP target port in an STP/SATA bridge

Figure I.15 shows an STP target port in an STP/SATA bridge closing an STP connection.

Figure I.15 — STP connection close - originated by STP target port in an STP/SATA bridge

I.16 Connection request - XL1:Request_Path to XL5:Forward_Open transition

Figure I.16 shows the establishment of a connection following a XL1:Request_Path to XL5:Forward_Open transition by expander phy [Y].

Figure I.16 — XL1:Request_Path to XL5:Forward_Open transition

I.17 Pathway blocked and pathway recovery example

Figure I.17 shows a topology used to illustrate pathway recovery. Exp[1] and Exp[2] are expander devices. A, B, and C are end devices. A attempts to open a connection to B while B attempts to open a connection to A.
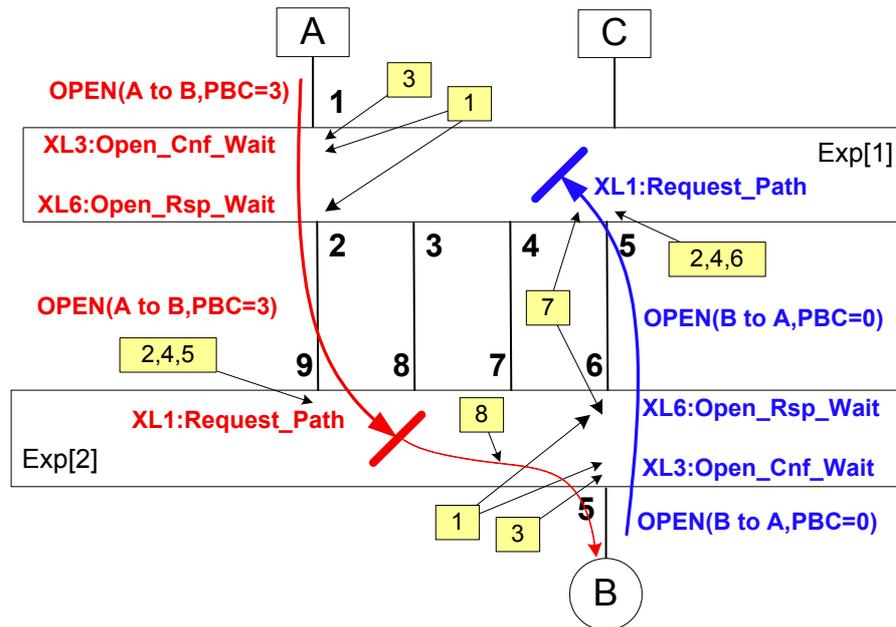
Figure I.17 — Partial pathway recovery

The sequence of events used to identify pathway blockage and to perform pathway recovery are as follows:

1) Exp[1].Phy[1,2] and Exp[2].Phy[5,6] each send Phy Status (Partial Pathway) responses to the ECM to indicate that they contain partial pathways;
2) Exp[1].Phy[5] and Exp[2].Phy[9] each receive Arbitrating (Waiting On Partial) confirmations from the ECM which cause them to transmit AIP (WAITING ON PARTIAL);
3) AIP (WAITING ON PARTIAL) is received by Exp[1].Phy[2] and Exp[2].Phy[6] then forwarded to Exp[1].Phy[1] and Exp[2].Phy[5] as an Arb Status (Waiting On Partial) confirmation. Exp[1].Phy[1] and Exp[2].Phy[5] each send Phy Status (Blocked On Partial) responses to ECM as confirmation that they are blocked waiting on a partial pathway in another expander device;
4) Exp[1].Phy[5] and Exp[2].Phy[9] each receive Arbitrating (Blocked On Partial) confirmations from the ECM while all destination phys send Phy Status (Blocked On Partial) responses which cause them to run their Partial Pathway Timeout timers;
5) The Partial Pathway Timeout timer expires in Exp[2].Phy[9]. This causes a request to the ECM to resolve pathway blockage. The pathway recovery priority for this phy is not lower than all phys within the destination port which are also blocked (i.e., the Request Path from Exp[2].Phy[9] has higher priority than the Request Path from Exp[2].Phy[5], which is receiving Arbitrating (Blocked On Partial)). The ECM does not provide an Arb Reject (Pathway Blocked) confirmation to Exp[2].Phy[9], so this expander phy waits for pathway resolution to occur elsewhere in the topology;
6) the Partial Pathway Timeout timer expires in Exp[1].Phy[5]. This causes a request to the ECM to resolve pathway blockage. The pathway recovery priority for this expander phy is lower than all expander phys within the destination port which are also blocked (i.e., the Request Path from Exp[1].Phy[5] is lower priority than the Request Path from Exp[1].Phy[5], which is receiving Arbitrating (Blocked On Partial)). The ECM provides an Arb Reject (Pathway Blocked) confirmation to Exp[1].Phy[5] which instructs this expander phy to reject the connection request using OPEN_REJECT (PATHWAY BLOCKED);

   NOTE 71 - The Partial Pathway Timeout timer in Exp[1].Phy[5] might expire before, after, or at the same time the Partial Pathway Timeout timer expires in Exp[2].Phy[9].

7) OPEN_REJECT (PATHWAY BLOCKED) tears down partial pathway all the way to the originating end device (Device B);
8) Exp[2].Phy[9] receives Arb Won and the partial pathway is extended through Exp[2].Phy[5]; and
9) OPEN (A to B) is delivered to device B.

Annex J
(informative)

Primitive encoding

This annex describes a set of the K28.5-based primitive encodings whose 40-bit values (after 8b10b encoding with either starting running disparity) have a Hamming distance (i.e., the number of bits different in two patterns) of at least 8. All the primitive encodings in 7.2 were selected from this list. Unassigned encodings may be used by future versions of this standard.

Table J.1 — Primitives with Hamming distance of 8 (Sheet 1 of 3)

1st	2nd	3rd	4th	Assignment
K28.5	D01.3	D01.3	D01.3	ALIGN (2)
K28.5	D01.4	D01.4	D01.4	ACK
K28.5	D01.4	D02.0	D31.4	RRDY (RESERVED 0)
K28.5	D01.4	D04.7	D24.0	NAK (RESERVED 1)
K28.5	D01.4	D07.3	D30.0	CREDIT_BLOCKED
K28.5	D01.4	D16.7	D07.3	NAK (RESERVED 2)
K28.5	D01.4	D24.0	D16.7	RRDY (NORMAL)
K28.5	D01.4	D27.4	D04.7	NAK (CRC ERROR)
K28.5	D01.4	D30.0	D02.0	RRDY (RESERVED 1)
K28.5	D01.4	D31.4	D29.7	NAK (RESERVED 0)
K28.5	D02.0	D01.4	D29.7	ERROR
K28.5	D02.0	D02.0	D02.0	HARD_RESET
K28.5	D02.0	D04.7	D01.4	CLOSE (RESERVED 1)
K28.5	D02.0	D07.3	D04.7	CLOSE (CLEAR AFFILIATION)
K28.5	D02.0	D16.7	D31.4	
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	BROADCAST (RESERVED 2)
K28.5	D04.7	D07.3	D29.7	BROADCAST (SES)
K28.5	D04.7	D16.7	D02.0	BROADCAST (RESERVED 3)
K28.5	D04.7	D24.0	D31.4	BROADCAST (RESERVED CHANGE 0)
K28.5	D04.7	D27.4	D07.3	BROADCAST (RESERVED CHANGE 1)
K28.5	D04.7	D29.7	D30.0	BROADCAST (RESERVED 4)
K28.5	D04.7	D31.4	D27.4	
K28.5	D07.0	D07.0	D07.0	ALIGN (1)
K28.5	D07.3	D01.4	D31.4	
K28.5	D07.3	D02.0	D04.7	
K28.5	D07.3	D04.7	D30.0	

Table J.1 — Primitives with Hamming distance of 8 (Sheet 2 of 3)

1st	2nd	3rd	4th	Assignment
K28.5	D07.3	D07.3	D07.3	
K28.5	D07.3	D24.0	D29.7	
K28.5	D07.3	D27.4	D16.7	
K28.5	D07.3	D29.7	D27.4	
K28.5	D07.3	D30.0	D24.0	
K28.5	D07.3	D31.4	D02.0	
K28.5	D10.2	D10.2	D27.3	ALIGN (0)
K28.5	D16.7	D01.4	D02.0	
K28.5	D16.7	D02.0	D07.3	
K28.5	D16.7	D04.7	D31.4	
K28.5	D16.7	D16.7	D16.7	OPEN_ACCEPT
K28.5	D16.7	D24.0	D27.4	
K28.5	D16.7	D27.4	D30.0	
K28.5	D16.7	D29.7	D24.0	
K28.5	D16.7	D30.0	D04.7	
K28.5	D16.7	D31.4	D01.4	
K28.5	D24.0	D01.4	D16.7	
K28.5	D24.0	D02.0	D29.7	
K28.5	D24.0	D04.7	D07.3	SOF
K28.5	D24.0	D07.3	D31.4	EOAF
K28.5	D24.0	D16.7	D27.4	EOF
K28.5	D24.0	D24.0	D24.0	
K28.5	D24.0	D27.4	D02.0	
K28.5	D24.0	D29.7	D04.7	
K28.5	D24.0	D30.0	D01.4	SOAF
K28.5	D27.3	D27.3	D27.3	ALIGN (3)
K28.5	D27.4	D01.4	D07.3	AIP (RESERVED WAITING ON PARTIAL)
K28.5	D27.4	D04.7	D02.0	
K28.5	D27.4	D07.3	D24.0	AIP (WAITING ON CONNECTION)
K28.5	D27.4	D16.7	D30.0	AIP (RESERVED 1)
K28.5	D27.4	D24.0	D04.7	AIP (WAITING ON PARTIAL)
K28.5	D27.4	D27.4	D27.4	AIP (NORMAL)
K28.5	D27.4	D29.7	D01.4	AIP (RESERVED 2)
K28.5	D27.4	D30.0	D29.7	AIP (WAITING ON DEVICE)
K28.5	D27.4	D31.4	D16.7	AIP (RESERVED 0)
K28.5	D29.7	D02.0	D30.0	OPEN_REJECT (RESERVED CONTINUE 0)
K28.5	D29.7	D04.7	D27.4	OPEN_REJECT (RESERVED STOP 1)
K28.5	D29.7	D07.3	D16.7	OPEN_REJECT (RESERVED INITIALIZE 1)
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)

Table J.1 — Primitives with Hamming distance of 8 (Sheet 3 of 3)

1st	2nd	3rd	4th	Assignment
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	D10.2	D10.2	NOTIFY (RESERVED 2)
K28.5	D31.3	D31.3	D31.3	NOTIFY (ENABLE SPINUP)
K28.5	D31.4	D01.4	D30.0	OPEN_REJECT (RESERVED ABANDON 3)
K28.5	D31.4	D02.0	D27.4	OPEN_REJECT (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 K
(informative)

Messages between state machines

K.1 Messages between phy layer and other layers

Table K.1 lists the requests from the management application layer and the link layer to the phy layer.

Table K.1 — Requests from management application layer or link layer to phy layer

Phy layer	← Request —	Application layer or link layer
SP	Enter Partial	MA
	Enter Slumber	
	Exit Partial	
	Exit Slumber	
	Power on, hard reset, Management Reset, or Disable Phy	
SP_DWS	Power on, hard reset, Management Reset, or Disable Phy	MA
SP	Stop SNTT	SL_IR

Table K.2 lists the confirmations from the phy layer to the link layer.

Table K.2 — Confirmations from phy layer to link layer

Phy layer	— Confirmation →	Link layer
SP	Phy Layer Not Ready	XL (in expander phy)
	SATA Spinup Hold	
SP	Start SL_IR Receiver	SL_IR
	Phy Layer Not Ready	
	Phy Layer Ready (SAS)	
	Phy Layer Ready (SATA)	
SP_DWS receiver	Dword Received	SL_IR, SL, SSP, SMP, and XL receivers

K.2 Messages between link layer, port layer, and management application layer for all protocols

Table K.3 lists the requests between the link layer and the port layer for all protocols.

Table K.3 — Requests between link layer and port layer

Link layer	← Request —	Port layer
SL	Open Connection	PL
	Stop Arb	

Table K.4 lists the confirmations between the link layer and the port layer for all protocols.

Table K.4 — Confirmations between link layer and port layer

Link layer	— Confirmation →	Port layer
SL	Open Failed (Retry)	PL
	Open Failed (No Destination)	
	Open Failed (Bad Destination)	
	Open Failed (Wrong Destination)	
	Open Failed (Connection Rate Not Supported)	
	Open Failed (Protocol Not Supported)	
	Open Failed (Pathway Blocked)	
	Open Failed (Port Layer Request)	
	Open Failed (Break Received)	
	Open Failed (STP Resources Busy)	
	Inbound Connection Rejected	
	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 (Break Timeout)	

Table K.5 lists the requests from the management application layer to the link layer for all protocols.

Table K.5 — Requests from management application layer to link layer

Link layer	← Request —	Application layer
SL_IR	Tx HARD_RESET	MA
	Tx IDENTIFY Address Frame	
SL	Transmit Broadcast (type)	MA

Table K.6 lists the confirmations between the link layer and the expander function or management application layer for all protocols.

Table K.6 — Confirmations between link layer and port layer, link layer, or application layer

Link layer	— Confirmation →	Port layer, link layer, or application layer
SL	Change Received	MA
SL_IR	Phy Enabled	MA
	Phy Disabled	
	Address Frame Failed	
	HARD_RESET Transmitted	
	Identify Timeout	
	Identification Sequence Complete	
SL_IR	HARD_RESET Received	PL
	Phy Enabled	
	Phy Disabled	

K.3 Messages between link layer, port layer, and transport layer for SSP

Table K.7 lists the requests between the link layer, port layer, and SSP transport layer.

Table K.7 — Requests between link layer, port layer, and transport layer for SSP

Link layer	← Request —	Port layer	← Request —	Transport layer
SL		PL	Cancel	ST
	Accept_Reject Opens (Accept SSP)		Accept_Reject Opens (Accept SSP)	
	Accept_Reject Opens (Reject SSP)		Accept_Reject Opens (Reject SSP)	
SSP		PL	Transmit Frame (Interlocked)	ST
			Transmit Frame (Non-Interlocked)	
	Tx Frame (Balance Required)			
	Tx Frame (Balance Not Required)			
	Close Connection			

Table K.8 lists the confirmations from the port layer to the SSP transport layer.

Table K.8 — Confirmations from port layer to transport layer for SSP

Port layer	— Confirmation →	Transport layer
PL	HARD_RESET Received	ST

Table K.9 lists the confirmations between the SL link layer, port layer, and SSP transport layer.

Table K.9 — Confirmations between SL link layer, port layer, and SSP transport layer

Link layer	— Confirmation →	Port layer	— Confirmation →	Transport layer
SL	Open Failed (Port Layer Request)	PL	Transmission Status (Cancel Acknowledge)	ST
	Open Failed (Wrong Destination)		Transmission Status (Wrong Destination)	
	Open Failed (Connection Rate Not Supported)		Transmission Status (Connection Rate Not Supported)	
	Open Failed (Protocol Not Supported)		Transmission Status (Protocol Not Supported)	
	Open Failed (No Destination)		Transmission Status (No Destination) or Transmission Status (I_T Nexus Loss)	
	Open Failed (Bad Destination)		Transmission Status (Bad Destination)	
	Open Failed (STP Resources Busy)		Transmission Status (STP Resources Busy)	
	Open Failed (Open Timeout Occurred)		Transmission Status (Open Timeout Occurred) or Transmission Status (I_T Nexus Loss)	
	Open Failed (Break Received)		Transmission Status (Break Received)	
	Open Failed (Pathway Blocked)			
	Connection Closed (any reason)		no confirmation or Transmission Status (Connection Lost Without ACK/NAK)	

Table K.10 lists the confirmations between the SSP link layer, port layer, and SSP transport layer.

Table K.10 — 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		Transmission Status (ACK/NAK Timeout)	
	Credit Timeout			
	DONE Transmitted			
	ACK Received		Transmission Status (ACK Received)	
	ACK Transmitted		ACK Transmitted	
	NAK Received		Transmission Status (NAK Received)	
	DONE Received			
	Frame Received (ACK/NAK Balanced)		Frame Received (ACK/NAK Balanced)	
	Frame Received (ACK/NAK Not Balanced)		Frame Received (ACK/NAK Not Balanced)	
			Transmission Status (No Phys In Port)	

K.4 Messages between link layer, port layer, and transport layer for SMP

Table K.11 lists the requests between the link layer, port layer, and SMP transport layer.

Table K.11 — Requests between SL/SMP link layer, port layer, and SMP transport layer

Link layer	← Request —	Port layer	← Request —	Transport layer
SL	Accept_Reject Opens (Accept SMP)	PL	Accept_Reject Opens (Accept SMP)	MT
	Accept_Reject Opens (Reject SMP)		Accept_Reject Opens (Reject SMP)	
SMP	Tx Frame	PL	Transmit Frame	MT
	SMP Transmit Break		SMP Transmit Break	

Table K.12 lists the confirmations between the link layer, port layer, and SMP transport layer.

Table K.12 — 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		Frame Received	
	Frame Received (SMP Failure)		Frame Received (SMP Failure)	
SL	Open Failed (Wrong Destination)	PL	Transmission Status (Wrong Destination)	MT
	Open Failed (Connection Rate Not Supported)		Transmission Status (Connection Rate Not Supported)	
	Open Failed (Protocol Not Supported)		Transmission Status (Protocol Not Supported)	
	Open Failed (No Destination)		Transmission Status (No Destination)	
	Open Failed (Bad Destination)		Transmission Status (Bad Destination)	
	Open Failed (STP Resources Busy)		Transmission Status (STP Resources Busy)	
	Open Failed (Open Timeout Occurred)		Transmission Status (Open Timeout Occurred)	
	Open Failed (Break Received)		Transmission Status (Break Received)	
	Open Failed (Pathway Blocked)			
	Connection Closed (any reason)		Connection Closed	

K.5 Messages from transport layer to application layer for SSP

Table K.13 lists the requests and responses from the SCSI application layer to the SSP transport layer.

Table K.13 — Requests and responses from SCSI application layer to SSP transport layer

Transport layer	← Request or response —	Application layer
ST_I (SSP initiator port)	Send SCSI Command	SA (SCSI initiator device)
	Send Task Management Request	
	Accept_Reject OPENs (Accept SSP)	
	Accept_Reject OPENs (Reject SSP)	
ST_T (SSP target port)	Send Command Complete	SA (SCSI target device)
	Task Management Function Executed	
	Send Data-In	
	Receive Data-Out	
	Accept_Reject OPENs (Accept SSP)	
	Accept_Reject OPENs (Reject SSP)	

Table K.14 lists the confirmations and indications from the SSP transport layer to the SCSI application layer.

Table K.14 — Confirmations and indications from SSP transport layer to SCSI application layer

Transport layer	— Confirmation or indication →	Application layer
ST_I (SSP initiator port)	Command Complete Received	SA (SCSI initiator device)
	Received Task Management Function Executed	
	Nexus Loss	
	Transport Reset	
ST_T (SSP target port)	SCSI Command Received	SA (SCSI target device)
	Task Management Request Received	
	Data-In Delivered	
	Data-Out Received	
	Nexus Loss	
	Transport Reset	

K.6 Messages from transport layer to application layer for SMP

Table K.15 lists the requests from the management application layer to the SMP transport layer.

Table K.15 — Requests from management application layer to SMP transport layer

Transport layer	← Request —	Application layer
MT_IP (SMP initiator port)	Send SMP Function Request	MA (SAS initiator device)
MT_TP (SMP target port)	Send SMP Response	MA (SAS target device)
	Accept_Reject OPENs (Accept SMP)	
	Accept_Reject OPENs (Reject SMP)	

Table K.16 lists the confirmations from the SMP transport layer to the management application layer.

Table K.16 — Confirmations from SMP transport layer to management application layer

Transport layer	— Confirmation →	Application layer
MT_IP (SMP initiator port)	Open Failed	MA (SAS initiator device)
	SMP Frame Receive Timeout	
	SMP Frame Transmit Receive Failure	
	Received SMP Function Complete	
MT_TP (SMP target port)	SMP Function Received	MA (SAS target device)
	SMP Connection Closed	

Annex L
(informative)

Discover process example implementation

L.1 Discover process example implementation overview

This annex includes a C program implementing the discover process. Table L.1 describes the source files.

Table L.1 — C program files

Filename	Description
SASDiscoverSimulation.h	header file
SASDiscoverSimulation.cpp	C source file

L.2 Header file

The following is the C header file for the discover process.

```
// SASDiscoverSimulation.h
// updated 2005/05/29

// assume the maximum number of phys in an expander device is 128
#define MAXIMUM_EXPANDER_PHYS           128

// assume the maximum number of indexes per phy is 128
#define MAXIMUM_EXPANDER_INDEXES        128

// limit to 8 initiators for this example
#define MAXIMUM_INITIATORS              8

// defines for address frame types
#define ADDRESS_IDENTIFY_FRAME          0x00
#define ADDRESS_OPEN_FRAME              0x01

// defines for SMP frame types
#define SMP_REQUEST_FRAME               0x40
#define SMP_RESPONSE_FRAME              0x41

// defines for SMP request functions
#define REPORT_GENERAL                  0x00
#define REPORT_MANUFACTURER_INFORMATION 0x01
#define DISCOVER                        0x10
#define REPORT_PHY_ERROR_LOG            0x11
#define REPORT_PHY_SATA                 0x12
#define REPORT_ROUTE_INFORMATION        0x13
#define CONFIGURE_ROUTE_INFORMATION     0x90
#define PHY_CONTROL                     0x91
#define PHY_TEST                        0x92

// defines for the protocol bits
#define SATA                            0x01
#define SMP                             0x02
#define STP                             0x04
#define SSP                             0x08

// defines for open responses, arbitrary values, not defined in the spec
#define OPEN_ACCEPT                         0
#define OPEN_REJECT_BAD_DESTINATION         1
```

```
    #define OPEN_REJECT_RATE_NOT_SUPPORTED      2
    #define OPEN_REJECT_NO_DESTINATION          3
    #define OPEN_REJECT_PATHWAY_BLOCKED         4
    #define OPEN_REJECT_PROTOCOL_NOT_SUPPORTED  5
    #define OPEN_REJECT_RESERVE_ABANDON         6
    #define OPEN_REJECT_RESERVE_CONTINUE        7
    #define OPEN_REJECT_RESERVE_INITIALIZE      8
    #define OPEN_REJECT_RESERVE_STOP            9
    #define OPEN_REJECT_RETRY                   10
    #define OPEN_REJECT_STP_RESOURCES_BUSY      11
    #define OPEN_REJECT_WRONG_DESTINATION       12
    #define OPEN_REJECT_WAITING_ON_BREAK        13

    // definitions for discovery algorithm use
    enum
    {
        SAS_SIMPLE_LEVEL_DESCENT = 0,
        SAS_UNIQUE_LEVEL_DESCENT
    };

    enum
    {
        SAS_10_COMPATIBLE = 0,
        SAS_11_COMPATIBLE
    };

    // definitions for SMP function results
    enum SMPFunctionResult
    {
        SMP_REQUEST_ACCEPTED = 0,           // from original example

        SMP_FUNCTION_ACCEPTED = 0,
        SMP_UNKNOWN_FUNCTION,
        SMP_FUNCTION_FAILED,
        SMP_INVALID_REQUEST_FRAME_LENGTH,
        SMP_PHY_DOES_NOT_EXIST = 0x10,
        SMP_INDEX_DOES_NOT_EXIST,
        SMP_PHY_DOES_NOT_SUPPORT_SATA,
        SMP_UNKNOWN_PHY_OPERATION,
        SMP_UNKNOWN_PHY_TEST_FUNCTION,
        SMP_UNKNOWN_PHY_TEST_FUNCTION_IN_PROGRESS,
        SMP_PHY_VACANT
    };

    // DeviceTypes
    enum DeviceTypes
    {
        NO_DEVICE = 0,
        END_DEVICE,
        EDGE_EXPANDER_DEVICE,
        FANOUT_EXPANDER_DEVICE,

        END = END_DEVICE,                   // from original example
        EDGE = EDGE_EXPANDER_DEVICE,        // from original example
        FANOUT = FANOUT_EXPANDER_DEVICE     // from original example
    };

    // RoutingAttribute
    enum RoutingAttribute
    {
```

```
   DIRECT = 0,
   SUBTRACTIVE,
   TABLE,

   // this attribute is a psuedo attribute, used to reflect the function
   // result of SMP_PHY_VACANT in a fabricated discover response
   PHY_NOT_USED = 15
};

// ConnectorType
enum ConnectorType
{
   UNKNOWN_CONNECTOR = 0,
   SFF_8470_EXTERNAL_WIDE,
   SFF_8484_INTERNAL_WIDE = 16,
   SFF_8482_BACKPLANE = 32,
   SATA_HOST_PLUG,
   SAS_DEVICE_PLUG,
   SATA_DEVICE_PLUG
};

// RouteFlag
enum DisableRouteEntry
{
   ENABLED = 0,
   DISABLED
};

// PhyLinkRate(s)
enum PhysicalLinkRate
{
   RATE_UNKNOWN = 0,
   PHY_DISABLED,
   PHY_FAILED,
   SPINUP_HOLD_OOB,
   PORT_SELECTOR_DETECTED,

   // this is a psuedo link rate, used to reflect the function
   // result of SMP_PHY_VACANT in a fabricated discover response
   PHY_DOES_NOT_EXIST,

   GBPS_1_5 = 8,
   GBPS_3_0
};

// PhyOperation
enum PhyOperation
{
   NOP = 0,
   LINK_RESET,
   HARD_RESET,
   DISABLE,
   CLEAR_ERROR_LOG = 5,
   CLEAR_AFFILIATION,
   TRANSMIT_SATA_PORT_SELECTION_SIGNAL
};

// provide the simple type definitions
typedef unsigned char byte;
typedef unsigned short word;
```

```c
        typedef unsigned long dword;
        typedef unsigned _int64 quadword;

        // the structures assume a char bitfield is valid, this is compiler
        // dependent defines would be more portable, but less descriptive

        // the Identify frame is exchanged following OOB, for this
        // code it contains the identity information for the attached device
        // and the initiator application client
        struct Identify
        {
           // byte 0
           byte AddressFrameType:4;              // ADDRESS_IDENTIFY_FRAME
           byte DeviceType:3;                    // END_DEVICE
                                                 //
           byte RestrictedByte0Bit7:1;

           // byte 1
           byte RestrictedByte1;

           // byte 2
           union
           {
              struct
              {
                 byte RestrictedByte2Bit0:1;
                 byte SMPInitiatorPort:1;
                 byte STPInitiatorPort:1;
                 byte SSPInitiatorPort:1;
                 byte ReservedByte2Bit4_7:4;
              };
              byte InitiatorBits;
           };

           // byte 3
           union
           {
              struct
              {
                 byte RestrictedByte3Bit0:1;
                 byte SMPTargetPort:1;
                 byte STPTargetPort:1;
                 byte SSPTargetPort:1;
                 byte ReservedByte3Bit4_7:4;
              };
              byte TargetBits;
           };

           // byte 4-11
           byte RestrictedByte4_11[8];

           // byte 12-19
           quadword SASAddress;

           // byte 20
           byte PhyIdentifier;

           // byte 21-27
           byte ReservedByte21_27[4];
```

```
   // byte 28-31
   dword CRC;
}; // struct Identify

// the Open address frame is used to send open requests
struct OpenAddress
{
   // byte 0
   byte AddressFrameType:4;          // ADDRESS_OPEN_FRAME
   byte Protocol:3;                  // SMP
                                     // STP
                                     // SSP
   byte InitiatorPort:1;

   // byte 1
   byte ConnectionRate:4;            // GBPS_1_5
                                     // GBPS_3_0
   byte Features:4;

   // byte 2-3
   word InitiatorConnectionTag;

   // byte 4-11
   quadword DestinationSASAddress;

   // byte 12-19
   quadword SourceSASAddress;

   // byte 20
   byte CompatibleFeatures;

   // byte 21
   byte PathwayBlockedCount;

   // byte 22-23
   word ArbitrationWaitTime;

   // byte 24-27
   byte MoreCompatibleFeatures[4];

   // byte 28-31
   dword CRC[4];
}; // struct OpenAddress

// request specific bytes for a general input function
struct SMPRequestGeneralInput
{
   // byte 4-7
   dword CRC;
};

// request specific bytes for a phy input function
struct SMPRequestPhyInput
{
   // byte 4-7
   byte IgnoredByte4_7[4];

   // byte 8
   byte ReservedByte8;
```

```
    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte ReservedByte11;

    // byte 12-15
    dword CRC;
}; // struct SMPRequestPhyInput

// the ConfigureRouteInformation structure is used to provide the
// expander route entry for the expander route table, it is intended
// to be referenced by the SMPRequestConfigureRouteInformation struct
struct ConfigureRouteInformation
{
    // byte 12
    byte IgnoredByte12Bit0_6:7;
    byte DisableRouteEntry:1;      // if a routing error is detected
                                   // then the route is disabled by
                                   // setting this bit

    // byte 13-15
    byte IgnoredByte13_15[3];

    // byte 16-23
    quadword RoutedSASAddress;     // identical to the AttachedSASAddress
                                   // found through discovery

    // byte 24-35
    byte IgnoredByte24_35[12];

    // byte 36-39
    byte ReservedByte36_39[4];
}; // struct ConfigureRouteInformation

// request specific bytes for SMP ConfigureRouteInformation function
struct SMPRequestConfigureRouteInformation
{
    // byte 4-5
    byte ReservedByte4_5[2];

    // byte 6-7
    word ExpanderRouteIndex;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10-11
    byte ReservedByte10_11[2];

    // byte 12-39
    struct ConfigureRouteInformation Configure;

    // byte 40-43
```

```
   dword CRC;
}; // struct SMPRequestConfigureRouteInformation

// the PhyControlInformation structure is used to provide the
// expander phy control values, it is intended
// to be referenced by the SMPRequestPhyControl struct
struct PhyControlInformation
{
   // byte 12-31
   byte IgnoredByte12_31[20];

   // byte 32
   byte IgnoredByte32Bit0_3:4;
   byte ProgrammedMinimumPhysicalLinkRate:4;

   // byte 33
   byte IgnoredByte33Bit0_3:4;
   byte ProgrammedMaximumPhysicalLinkRate:4;

   // byte 34-35
   byte IgnoredByte34_35[2];

   // byte 36
   byte PartialPathwayTimeoutValue:4;
   byte ReservedByte36Bit4_7:4;

   // byte 37-39
   byte ReservedByte37_39[3];
}; // struct PhyControlInformation

// request specific bytes for SMP Phy Control function
struct SMPRequestPhyControl
{
   // byte 4-7
   byte IgnoredByte4_7[4];

   // byte 8
   byte ReservedByte8;

   // byte 9
   byte PhyIdentifier;

   // byte 10
   byte PhyOperation;

   // byte 11
   byte UpdatePartialPathwayTimeoutValue:1;
   byte ReservedByte11Bit1_7:7;

   // byte 12-39
   struct PhyControlInformation Control;

   // byte 40-43
   dword CRC;
}; // struct SMPRequestPhyControl

// request specific bytes for SMP Phy Test function
struct SMPRequestPhyTest
{
   // byte 4-7
```

```
    byte IgnoredByte4_7[4];

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte PhyTestFunction;

    // byte 11
    byte PhyTestPattern;

    // byte 12-14
    byte ReservedByte12_14[3];

    // byte 15
    byte PhyTestPatternPhysicalLinkRate:4;
    byte ReservedByte15Bit4_7:4;

    // byte 16-39
    byte ReservedByte16_39[24];

    // byte 40-43
    dword CRC;
}; // struct SMPRequestPhyTest

// generic structure referencing an SMP Request, must be initialized
// before being used
struct SMPRequest
{
    // byte 0
    byte SMPFrameType;                    // always SMP_REQUEST_FRAME

    // byte 1
    byte Function;                        // REPORT_GENERAL
                                          // REPORT_MANUFACTURER_INFORMATION
                                          // DISCOVER
                                          // REPORT_PHY_ERROR_LOG
                                          // REPORT_PHY_SATA
                                          // REPORT_ROUTE_INFORMATION
                                          // CONFIGURE_ROUTE_INFORMATION
                                          // PHY_CONTROL
                                          // PHY_TEST

    // byte 2-3
    byte ReservedByte2_3[2];

    // bytes 4-n
    union
    {
        struct SMPRequestGeneralInput ReportGeneral;
        struct SMPRequestGeneralInput ReportManufacturerInformation;
        struct SMPRequestPhyInput Discover;
        struct SMPRequestPhyInput ReportPhyErrorLog;
        struct SMPRequestPhyInput ReportPhySATA;
        struct SMPRequestPhyInput ReportRouteInformation;
        struct SMPRequestConfigureRouteInformation ConfigureRouteInformation;
        struct SMPRequestPhyControl PhyControl;
```

```
      struct SMPRequestPhyTest PhyTest;
   } Request;
}; // struct SMPRequest

// request specific bytes for SMP Report General response, intended to be
// referenced by SMPResponse
struct SMPResponseReportGeneral
{
   // byte 4-5
   word ExpanderChangeCount;

   // byte 6-7
   word ExpanderRouteIndexes;

   // byte 8
   byte ReservedByte8;

   // byte 9
   byte NumberOfPhys;

   // byte 10
   byte ConfigurableRouteTable:1;
   byte Configuring:1;
   byte ReservedByte10Bit2_7:6;

   // byte 11
   byte ReservedByte11;

   // byte 12-19
   byte EnclosureLogicalIdentifier[8];

   // byte 20-27
   byte ReservedByte20_27[8];

   // byte 28-31
   dword CRC;
}; // struct SMPResponseReportGeneral

struct SAS11FormatReportManufacturerInformation
{
   // byte 40-47
   byte ComponentVendorIdentification[8];

   // byte 48-49
   byte ComponentID[2];

   // byte 50
   byte ComponentRevisionID;

   // byte 51
   byte Reserved;

   // byte 52-59
   byte VendorSpecific[8];
}; // struct SAS11FormatReportManufacturerInformation

// request specific bytes for SMP Report Manufacturer Information response,
// intended to be referenced by SMPResponse
struct SMPResponseReportManufacturerInformation
{
```

```
    // byte 4-7
    byte IgnoredByte4_7[4];

    // byte 8
    byte SAS11Format:1;
    byte ReservedByte8_Bit1_7:7;

    // byte 9-10
    byte IgnoredByte9_10[2];

    // byte 11
    byte ReservedByte11;

    // byte 12-19
    byte VendorIdentification[8];

    // byte 20-35
    byte ProductIdentification[16];

    // byte 36-39
    byte ProductRevisionLevel[4];

    union
    {
        struct SAS11FormatReportManufacturerInformation SAS11;

        // byte 40-59
        byte VendorSpecific[20];
    };

    // byte 60-63
    dword CRC;
}; // struct SMPResponseReportManufacturerInformation

// the Discover structure is used to retrieve expander port information
// it is intended to be referenced by the SMPResponseDiscover structure
struct Discover
{
    // byte 12
    byte ReservedByte12Bit0_3:4;
    byte AttachedDeviceType:3;
    byte IgnoredByte12Bit7:1;

    // byte 13
    byte NegotiatedPhysicalLinkRate:4;
    byte ReservedByte13Bit4_7:4;

    // byte 14
    union
    {
        struct
        {
            byte AttachedSATAHost:1;
            byte AttachedSMPInitiator:1;
            byte AttachedSTPInitiator:1;
            byte AttachedSSPInitiator:1;
            byte ReservedByte14Bit4_7:4;
        };
        byte InitiatorBits;
    };
```

```
   // byte 15
   union
   {
      struct
      {
         byte AttachedSATADevice:1;
         byte AttachedSMPTarget:1;
         byte AttachedSTPTarget:1;
         byte AttachedSSPTarget:1;
         byte ReservedByte15Bit4_6:3;
         byte AttachedSATAPortSelector:1;
      };
      byte TargetBits;
   };

   // byte 16-23
   quadword SASAddress;

   // byte 24-31
   quadword AttachedSASAddress;

   // byte 32
   byte AttachedPhyIdentifier;

   // byte 33-39
   byte ReservedByte33_39[7];

   // byte 40
   byte HardwareMinimumPhysicalLinkRate:4;
   byte ProgrammedMinimumPhysicalLinkRate:4;

   // byte 41
   byte HardwareMaximumPhysicalLinkRate:4;
   byte ProgrammedMaximumPhysicalLinkRate:4;

   // byte 42
   byte PhyChangeCount;

   // byte 43
   byte PartialPathwayTimeoutValue:4;
   byte IgnoredByte36Bit4_6:3;
   byte VirtualPhy:1;

   // byte 44
   byte RoutingAttribute:4;
   byte ReservedByte44Bit4_7:4;

   // byte 45
   byte ConnectorType:7;
   byte ReservedByte45Bit7:1;

   // byte 46
   byte ConnectorElementIndex;

   // byte 47
   byte ConnectorPhysicalLink;

   // byte 48-49
   byte ReservedByte48_49[2];
```

```
      // byte 50-51
      byte VendorSpecific[2];

      // byte 52-55
      dword CRC;
   }; // struct Discover

   // response specific bytes for SMP Discover, intended to be referenced by
   // SMPResponse
   struct SMPResponseDiscover
   {
      // byte 4-7
      byte IgnoredByte4_7;

      // byte 8
      byte ReservedByte8;

      // byte 9
      byte PhyIdentifier;

      // byte 10
      byte IgnoredByte10;

      // byte 11
      byte ReservedByte11;

      union                               // original example used Results instead
                                          // of Result, this allows both
      {
         // byte 12-55
         struct Discover Results;
         struct Discover Result;
      };
   }; // struct SMPResponseDiscover

   // response specific bytes for SMP Report Phy Error Log, intended to be
   // referenced by SMPResponse
   struct SMPResponseReportPhyErrorLog
   {
      // byte 4-7
      byte IgnoredByte4_7;

      // byte 8
      byte ReservedByte8;

      // byte 9
      byte PhyIdentifier;

      // byte 10
      byte IgnoredByte10;

      // byte 11
      byte ReservedByte11;

      // byte 12-15
      dword InvalidDwordCount;

      // byte 16-19
      dword DisparityErrorCount;
```

```
    // byte 20-23
    dword LossOfDwordSynchronizationCount;

    // byte 24-27
    dword PhyResetProblemCount;

    // byte 28-31
    dword CRC;
}; // struct SMPResponseReportPhyErrorLog

// this structure describes the Register Device to Host FIS defined in the
// SATA specification
struct RegisterDeviceToHostFIS
{
    // byte 24
    byte FISType;

    // byte 25
    byte ReservedByte25Bit0_5:6;
    byte Interrupt:1;
    byte ReservedByte25Bit7:1;

    // byte 26
    byte Status;

    // byte 27
    byte Error;

    // byte 28
    byte SectorNumber;

    // byte 29
    byte CylLow;

    // byte 30
    byte CylHigh;

    // byte 31
    byte  DevHead;

    // byte 32
    byte SectorNumberExp;

    // byte 33
    byte CylLowExp;

    // byte 34
    byte CylHighExp;

    // byte 35
    byte ReservedByte35;

    // byte 36
    byte SectorCount;

    // byte 37
    byte SectorCountExp;

    // byte 38-43
```

```
      byte ReservedByte38_43[6];
   }; // struct RegisterDeviceToHostFIS

   // response specific bytes for SMP Report Phy SATA, intended to be
   // referenced by SMPResponse
   struct SMPResponseReportPhySATA
   {
      // byte 4-7
      byte IgnoredByte4_7;

      // byte 8
      byte ReservedByte8;

      // byte 9
      byte PhyIdentifier;

      // byte 10
      byte IgnoredByte10;

      // byte 11
      byte AffilationValid:1;
      byte AffilationsSupported:1;
      byte ReservedByte11Bit2_7:6;

      // byte 12-15
      byte ReservedByte12_15[4];

      // byte 16-32
      quadword STPSASAddress;

      // byte 24-43
      struct RegisterDeviceToHostFIS FIS;

      // byte 44-47
      byte ReservedByte44_47[4];

      // byte 48-55
      quadword AffiliatedSTPInitiatorSASAddress;

      // byte 56-59
      dword CRC;
   }; // struct SMPResponseReportPhySATA

   struct ReportRouteInformation
   {
      // byte 12
      byte IgnoredByte12Bit0_6:7;
      byte ExpanderRouteEntryDisabled:1;

      // byte 13-15
      byte IgnoredByte13_15[3];

      // byte 16-23
      quadword RoutedSASAddress;

      // byte 24-35
      byte IgnoredByte24_35[12];

      // byte 36-39
      byte ReservedByte36_39[4];
```

```
}; // struct ReportRouteInformation

// response specific bytes for SMP Report Route Information, intended to be
// referenced by SMPResponse
struct SMPResponseReportRouteInformation
{
    // byte 4-5
    byte IgnoredByte4_5;

    // byte 6-7
    word ExpanderRouteIndex;

    // byte 8
    byte ReservedByte8;

    // byte 9
    byte PhyIdentifier;

    // byte 10
    byte IgnoredByte10;

    // byte 11
    byte ReservedByte11;

    // byte 12-39
    struct ReportRouteInformation Result;

    // byte 40-43
    dword CRC;
}; // struct SMPResponseReportRouteInformation

// response specific bytes for SMP Configure Route Information,
// intended to be referenced by SMPResponse
struct SMPResponseConfigureRouteInformation
{
    // byte 4-7
    dword CRC;
};

// response specific bytes for SMP Phy Control,
// intended to be referenced by SMPResponse
struct SMPResponsePhyControl
{
    // byte 4-7
    dword CRC;
};

// response specific bytes for SMP Phy Test,
// intended to be referenced by SMPResponse
struct SMPResponsePhyTest
{
    // byte 4-7
    dword CRC;
};

// generic structure referencing an SMP Response, must be initialized
// before being used
struct SMPResponse
{
    // byte 0
```

```
    byte SMPFrameType;                    // always 41h for SMP responses

    // byte 1
    byte Function;

    // byte 2
    byte FunctionResult;

    // byte 3
    byte ReservedByte3;

    // bytes 4-n
    union
    {
        struct SMPResponseReportGeneral ReportGeneral;
        struct SMPResponseReportManufacturerInformation
               ReportManufacturerInformation;
        struct SMPResponseDiscover Discover;
        struct SMPResponseReportPhyErrorLog ReportPhyErrorLog;
        struct SMPResponseReportPhySATA ReportPhySATA;
        struct SMPResponseReportRouteInformation ReportRouteInformation;
        struct SMPResponseConfigureRouteInformation ConfigureRouteInformation;
        struct SMPResponsePhyControl PhyControl;
        struct SMPResponsePhyTest PhyTest;
    } Response;
}; // struct SMPResponse

// this structure is how this simulation obtains its knowledge about the
// initiator port that is doing the discover, it is not defined as part of
// the standard...
struct ApplicationClientKnowledge
{
    quadword SASAddress;
    byte NumberOfPhys;
    byte InitiatorBits;
    byte TargetBits;
};

// the RouteTableEntry structure is used to contain the internal copy of
// the expander route table
struct RouteTableEntry
{
    byte ExpanderRouteEntryDisabled;
    quadword RoutedSASAddress;
};

// the TopologyTable structure is the summary of the information gathered
// during the discover process, the table presented here is not concerned
// about memory resources consumed, production code would be more concerned
// about specifying necessary elements explictly
struct TopologyTable
{
    // pointer to a simple list of expanders in topology
    // a walk thru this link will encounter all expanders in
    // discover order
    struct TopologyTable *Next;

    // simple reference to this device, primarily to keep identification of
    // this structure simple, otherwise, the only place the address is
    // located is within the Phy element
```

```
    quadword SASAddress;

    // information from REPORT_GENERAL
    struct SMPResponseReportGeneral Device;

    // information from DISCOVER
    struct SMPResponseDiscover Phy[MAXIMUM_EXPANDER_PHYS];

    // list of route indexes for each phy
    word RouteIndex[MAXIMUM_EXPANDER_PHYS];

    // internal copy of the route table for the expander
    struct RouteTableEntry
            RouteTable[MAXIMUM_EXPANDER_PHYS][MAXIMUM_EXPANDER_INDEXES];

    //
    // in production code there would also be links to the necessary device
    // information like end device; vendor, model, serial number, etc.
    // the gathering of that type of information is not done here...
    //
}; // struct TopologyTable
```

L.3 Source file

The following is the C source file for the discover process.

```
// SASDiscoverSimulation.cpp
// updated 2005/05/29
//
// This is a simple simulation and code implementation of the initiator
// based expander discovery and configuration.

// There is no attempt to handle phy errors, arbitration issues, etc.
// Production level implementation need to handle errors appropriately.

// Structure names used are equivalent to those referenced in the standard.

// Basic assumptions:
// 1. BROADCAST (CHANGE) primitives initiate rediscovery/reconfiguration
// 2. Table locations for SASAddresses are deterministic for a specific
//    topology only.  When the topology changes, the location of a SASAddress
//    in an ASIC table cannot be assumed.
// 3. A complete discovery level occurs before the configuration of the
//    level begins.  Multiple passes are required as the levels of expanders
//    encountered between the initiator and the end devices increases.
// 4. Configuration of a single expander occurs before proceeding to
//    subsequent attached expanders.
// 5. The Attached structure is filled in following OOB and is available
//    from the initialization routines.
// 6. The Iam structure is provide by the application client.

#include <malloc.h>
#include <memory.h>
#include <stdlib.h>

// include the SAS structures
#include „SASDiscoverSimulation.h“

// this defines the type of algorithm used for discover
int DiscoverAlgorithm = SAS_SIMPLE_LEVEL_DESCENT;
```

```
    int SASCompatibility = SAS_11_COMPATIBLE;


    // loaded by the application client, in this simulation it is provided
    // in a text file, SASDeviceSetExample.ini
    extern struct ApplicationClientKnowledge Iam[MAXIMUM_INITIATORS];

    // obtained following OOB from the attached phy, in this simulation
    // it is provided in a text file, SASDeviceSetExample.ini
    extern struct Identify Attached[MAXIMUM_INITIATORS];

    // buffers used to request and return SMP data
    extern struct SMPRequest SMPRequestFrame;
    extern struct SMPResponse SMPResponseFrame;

    // resulting discover information ends up in this table
    extern struct TopologyTable *SASDomain[MAXIMUM_INITIATORS];

    // this is the function used to send an SMPRequest and get a response back
    extern byte SMPRequest(byte PhyIdentifier,
                           quadword Source,
                           quadword Destination,
                           struct SMPRequest *SMPRequestFrame,
                           struct SMPResponse *SMPResponseFrame,
                           byte *OpenStatus,
                           byte Function,
                           ...);

    // this function is used to output error information, it mimics fprintf
    // functionality to an open trace file
    extern int TracePrint(char *String, ...);

    // this function gets the report general and discover information for
    // a specific expander, the discover process begins at the subtractive
    // boundary and progress downstream
    static
    struct TopologyTable *DiscoverExpander(byte PhyIdentifier,
                                           quadword SourceSASAddress,
                                           quadword DestinationSASAddress)
    {
       struct TopologyTable *expander = 0;
       byte phyCount = 0;
       int error = 1;

       byte openStatus = OPEN_ACCEPT;

       // get the report general information for the expander
       SMPRequest(PhyIdentifier,
                  SourceSASAddress,
                  DestinationSASAddress,
                  &SMPRequestFrame,
                  &SMPResponseFrame,
                  &openStatus,
                  REPORT_GENERAL);

       // don't worry about too much in the ,else' case for this example,
       // production code needs to handle
       if((openStatus == OPEN_ACCEPT) &&
          (SMPResponseFrame.FunctionResult == SMP_FUNCTION_ACCEPTED))
       {
```

```c
if(SMPResponseFrame.Response.ReportGeneral.NumberOfPhys <=
   MAXIMUM_EXPANDER_PHYS)
{
   // allocate space to retrieve the expander information
   expander = (struct TopologyTable *)
             calloc(1,
                    sizeof(struct TopologyTable));

   // make sure we only do this if the allocation is successful
   if(expander)
   {
      // save the address of this expander
      expander->SASAddress = DestinationSASAddress;

      // copy the result into the topology table
      memcpy((void *)&(expander->Device),
             (void *)&SMPResponseFrame.Response.ReportGeneral,
              sizeof(struct SMPResponseReportGeneral));

      // now walk through all the phys of the expander
      for(phyCount = 0;
          (phyCount < expander->Device.NumberOfPhys);
           phyCount++)
      {
         // get the discover information for each phy
         SMPRequest(PhyIdentifier,
                    SourceSASAddress,
                    DestinationSASAddress,
                    &SMPRequestFrame,
                    &SMPResponseFrame,
                    &openStatus,
                    DISCOVER,
                    phyCount);

         // don't worry about the ,else' case for this example,
         // production code needs to handle
         if((openStatus == OPEN_ACCEPT) &&
            (SMPResponseFrame.FunctionResult == SMP_FUNCTION_ACCEPTED))
         {
            // clear the error flag
            error = 0;

            // copy the result into the topology table
            memcpy((void *)&(expander->Phy[phyCount]),
                   (void *)&SMPResponseFrame.Response.Discover,
                    sizeof(struct SMPResponseDiscover));
         }
         else if((openStatus == OPEN_ACCEPT) &&
                 (SMPResponseFrame.FunctionResult == SMP_PHY_VACANT))
         {
            struct Discover *discover;

            discover = &SMPResponseFrame.Response.Discover.Result;

            // clear the error flag
            error = 0;

            // set the routing attribute and link rate to indicate that
            // the phy is not being used, this keeps it from being
            // included in the routing table information, these values
```

```
                        // are not defined in the spec at this time, but are listed
                        // as reserved values
                        discover->NegotiatedPhysicalLinkRate = PHY_DOES_NOT_EXIST;
                        discover->RoutingAttribute = PHY_NOT_USED;

                        // copy the result into the topology table
                        memcpy((void *)&(expander->Phy[phyCount]),
                                (void *)&SMPResponseFrame.Response.Discover,
                                sizeof(struct SMPResponseDiscover));
                    }
                    else
                    {
                        // if we had a problem on this link, then don't bother
                        // to do anything else, production code needs to be more
                        // robust...
                        // for this simulation example, the addresses are
                        // described as strings, so we can print them out...
                        // not true for production code...
                        TracePrint(„\n“
                                    „discover error, %02Xh at %s\n“,
                                    SMPResponseFrame.FunctionResult,
                                    (char *)&DestinationSASAddress);

                        // something happened so just bailout on this expander
                        error = 1;

                        // release the memory we allocated for this...
                        free(expander);
                        expander = 0;
                        break;
                    }
                }
            }
        }
        // the assumptions we made were exceeded, need to bump simulation
        // limits...
        else
        {
            TracePrint(„\n“
                        „report general error“
                        „, NumberOfPhys %d exceeded limit %d on %s\n“,
                        expander->Device.NumberOfPhys,
                        MAXIMUM_EXPANDER_PHYS,
                        (char *)&DestinationSASAddress);
        }
    }
    else
    {
        // if we had a problem getting report general for this expander,
        // something is wrong, can't go any further  down this path...
        // production code needs to be more robust...
        // for this simulation example, the addresses are
        // described as strings, so we can print them out...
        // not true for production code...
        TracePrint(„\n“
                    „report general error, open %02Xh result %02Xh at %s\n“,
                    openStatus,
                    SMPResponseFrame.FunctionResult,
                    (char *)&DestinationSASAddress);
    }
```

```c
    // the expander pointer is the error return, a null indicates something
    // bad happened...
    return(expander);
} // DiscoverExpander


// this routine searches upstream for the subtractive boundary that defines
// the edge expander device set
static
struct TopologyTable *FindBoundary(byte PhyIdentifier,
                                   quadword SourceSASAddress,
                                   struct TopologyTable *Expander,
                                   struct TopologyTable **DeviceSet)
{
    struct TopologyTable *expander = Expander;
    struct TopologyTable *nextExpander;

    struct Discover *discover;

    byte phyCount;

    int error = 0;
    int foundSubtractivePort = 0;

    quadword subtractiveSASAddress;
    byte attachedPhyIdentifier;

    // make sure the device set link is initialized
    *DeviceSet = 0;

    // the outer loop searches for subtractive phys and finds the SAS addresses
    // connected to them,  it validates that the subtractive phys all resolve
    // to the same expander address, then moves upstream searching for the
    // edge expander device set boundary
    do
    {
        // initialize the subtractive address, a zero value is not valid
        subtractiveSASAddress = 0;
        attachedPhyIdentifier = 0;

        // walk through all the phys of this expander
        for(phyCount = 0;
            (phyCount < expander->Device.NumberOfPhys);
            phyCount++)
        {
            // this is just a pointer helper
            discover = &(expander->Phy[phyCount].Result);

            // look for phys with edge or fanout devices attached...
            if((discover->RoutingAttribute == SUBTRACTIVE) &&
               ((discover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
                (discover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
            {
                // make sure all the subtractive phys point to the same address
                // when we are connected to an expander device
                if(!subtractiveSASAddress)
                {
                    subtractiveSASAddress = discover->AttachedSASAddress;
                    attachedPhyIdentifier = discover->AttachedPhyIdentifier;
                    foundSubtractivePort = 1;
```

```
            }
            // the addresses don't match... problem...
            else if(subtractiveSASAddress !=
                    discover->AttachedSASAddress)
            {
                // production code needs to deal with this better, for this
                // example, the SASAddresses are assumed to strings
                // so just print out the error information
                TracePrint(„\n"
                            „topology error, diverging subtractive phys"
                            „, ,%s' != ,%s' \n",
                            (char *)&subtractiveSASAddress,
                            (char *)&discover->AttachedSASAddress);
                error = 1;
                break;
            }
        }
    }

    // if no error, then decide if we need to go upstream or stop
    if(!error)
    {
        // if we have a subtractive address then go upstream to see
        // if it is part of the edge expander device set
        if(subtractiveSASAddress)
        {
            // get the discover information
            nextExpander = DiscoverExpander(PhyIdentifier,
                                            SourceSASAddress,
                                            subtractiveSASAddress);

            // if we successfully got the information from the next
            // expander then proceed upstream...
            if(nextExpander)
            {
                struct Discover *discover;

                // this is just a pointer helper
                discover = &(nextExpander->Phy[attachedPhyIdentifier].Result);

                // check to see if we are connected to the subtractive
                // port of the next expander, if we are then we have two
                // expander device sets connected together, stop here
                // and save the address of next expander in device set,
                // the return is expander
                if(discover->RoutingAttribute == SUBTRACTIVE)
                {
                    *DeviceSet = nextExpander;
                    break;
                }
                // go ahead and continue upstream looking for the boundary
                else
                {

                    // release the memory we allocated for this
                    free(expander);

                    // move upstream to the next expander
                    expander = nextExpander;
                }
```

```
            }
            // if there are no more upstream expanders stop here...
            else
            {
               break;
            }
         }
         // if we did not get a subtractive address this time around then stop
         else
         {
            // if we did find a subtractive port on a previous pass,
            // then return with expander pointing to the last device
            // with the subtractive port
            if(foundSubtractivePort)
            {
               break;
            }
            // if we never found a subtractive port, then return with a
            // null indicating there are no subtractive phys, don't free
            // the memory, because it is still in use by the calling routine
            else
            {
               expander = 0;
            }
         }
      }
      // if there was an error make sure we return a null expander pointer
      else
      {
         // to get here, we had to see more than one subtractive phy that
         // connect to different SAS addresses, this is a topology error
         // do cleanup on any memory allocated if necessary
         if((expander != Expander) &&
            (expander != *DeviceSet))
         {
            // release the memory we allocated for this and make sure
            // we return a null
            free(expander);
            expander = 0;
         }
      }
   }
   while(!error &&
         expander &&
         subtractiveSASAddress);

   // on return expander contains the subtractive boundary expander
   // or a null indicating there were no subtractive phys,
   // or a null indicating there was an error
   return(expander);
} // FindBoundary

// find the table structure associated with a specific SAS address
static
struct TopologyTable *FindExpander(struct TopologyTable *Expander,
                                   quadword SASAddress)
{
   // walk the list of expanders, when we find the one that matches, stop
   while(Expander)
   {
```

```
        // do the SASAdresses match
        if(SASAddress == Expander->SASAddress)
        {
            break;
        }

        Expander = Expander->Next;
    }

    return(Expander);
} // FindExpander

// this routine searches the subtractive phys for the upstream expander address
static
int UpstreamExpander(struct TopologyTable *Expander,
                     quadword *SASAddress,
                     byte *PhyIdentifier)
{
    struct Discover *discover;

    byte phyCount;

    int found = 0;

    // walk through all the phys of this expander, searching for subtractive
    // phys return the SASAddress and PhyIdentifier for the first subtractive
    // phy encountered, they are all be the same if they have anything
    // attached
    for(phyCount = 0;
         (phyCount < Expander->Device.NumberOfPhys);
         phyCount++)
    {
        // this is just a pointer helper
        discover = &(Expander->Phy[phyCount].Result);

        // look for phys with edge or fanout devices attached...
        if((discover->RoutingAttribute == SUBTRACTIVE) &&
           ((discover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
            (discover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
        {
            *SASAddress = discover->AttachedSASAddress;
            *PhyIdentifier = discover->AttachedPhyIdentifier;
            found = 1;
            break;
        }
    }

    return(found);
} // UpstreamExpander

// this routine determines whether a SAS address is directly attached to
// an expander
static
int DirectAttached(struct TopologyTable *Expander,
                   quadword SASAddress)
{
    int direct = 0;
    byte phyCount;

    for(phyCount = 0;
```

```
             phyCount < Expander->Device.NumberOfPhys;
             phyCount++)
    {
       // did we find the address attached locally
       if(SASAddress ==
          Expander->Phy[phyCount].Result.AttachedSASAddress)
       {
          direct = 1;
          break;
       }
    }

    return(direct);
} // DirectAttached

// this route determines whether a SAS address is already in the route table
static
int AlreadyInTable(struct TopologyTable *Expander,
                   quadword *SASAddress,
                   byte PhyIdentifier)
{
    int inTable = 0;
    int routeIndex;

    for(routeIndex = 0;
        routeIndex < Expander->Device.ExpanderRouteIndexes;
        routeIndex++)
    {
       struct RouteTableEntry *entry =
              &Expander->RouteTable[PhyIdentifier][routeIndex];

       if(entry->RoutedSASAddress ==
          SASAddress)
       {
          inTable = 1;
          break;
       }
    }

    return(inTable);
} // AlreadyInTable

// this routine determines whether the SAS address, can be optimized out
// of the route table
static
int QualifiedAddress(struct TopologyTable *Expander,
                     byte PhyIdentifier,
                     quadword SASAddress,
                     byte RoutingAttribute,
                     byte *DisableRouteEntry)
{
    int qualified = 1;
    word routeIndex;

    if(DiscoverAlgorithm == SAS_UNIQUE_LEVEL_DESCENT)
    {
       if(((RoutingAttribute == SUBTRACTIVE) ||
           (RoutingAttribute == TABLE)) &&
          ((SASAddress == 0) ||
         (SASAddress == Expander->SASAddress) ||
```

```
            !memcmp(SASAddress, Expander->SASAddress, 8) ||
            DirectAttached(Expander, SASAddress) ||
            AlreadyInTable(Expander, SASAddress, PhyIdentifier)))
      {
         // if we made it here then we are disqualifying the address,
         // rules 2, 3, 4, and 5
         qualified = 0;
      }

      // if qualified, but the address is zero, then make sure it is
      // disabled
      if(qualified &&
         (SASAddress == 0))
      {
         *DisableRouteEntry = DISABLE;
      }
   }

   return(qualified);
} // QualifiedAddress


// this function is the configuration cycle from the current expander to
// the hub expander
static
int ConfigureExpander(byte PhyIdentifier,
                      quadword SourceSASAddress,
                      struct TopologyTable *HubExpander,
                      struct TopologyTable *ThisExpander)
{
   struct TopologyTable *thisExpander = ThisExpander;
   struct TopologyTable *expander = ThisExpander;
   struct TopologyTable *configureExpander;

   struct Discover *discover;

   quadword upstreamSASAddress = 0;
   byte upstreamPhyIdentifier = 0;

   byte phyIndex;
   word routeIndex;
   byte openStatus = OPEN_ACCEPT;

   int error = 0;

   do
   {
      // move upstream from here to find the expander table to configure with
      // information from „thisExpander"
      if(!UpstreamExpander(thisExpander,
                           &upstreamSASAddress,
                           &upstreamPhyIdentifier))
      {
         break;
      }

      if(upstreamSASAddress)
      {
         // get the expander associated with the upstream address
         configureExpander = FindExpander(HubExpander,
```

```
                                    upstreamSASAddress);

// if we found an upstream expander, then program it's route
// table
if(configureExpander)
{
   byte disableRouteEntry = ENABLED;

   for(phyIndex = 0;
       phyIndex < configureExpander->Device.NumberOfPhys;
       phyIndex++)
   {
     if(configureExpander->Phy[phyIndex].Result.AttachedSASAddress
    == thisExpander->SASAddress)
     {
        // loop through all the phys of the attached expander
        for(routeIndex = 0;
            ((routeIndex <
             expander->Device.NumberOfPhys) &&
            (configureExpander->RouteIndex[phyIndex] <
             configureExpander->Device.ExpanderRouteIndexes));
            routeIndex++)
        {
           discover = &(expander->Phy[routeIndex].Result);

           // assume the route entry is enabled
           disableRouteEntry = ENABLED;

           // make sure if the attached device type is 0, that the
           // sas address is 0, to simplify the qualified address
           // check
           if(discover->AttachedDeviceType == 0)
           {
               discover->AttachedSASAddress = 0;
           }

           // check to see if the address needs to be configured
           // in the route table, this decision is based on the
           // optimization flag
           if(QualifiedAddress(configureExpander,
                          phyIndex,
                          discover->AttachedSASAddress,
                          discover->RoutingAttribute,
                          &disableRouteEntry))
           {

              word index = configureExpander->RouteIndex[phyIndex];

              struct RouteTableEntry *entry =
                     &configureExpander->RouteTable[phyIndex][index];

              // configure the route indexes for the expander
              // with the attached address information
              SMPRequest(PhyIdentifier,
                       SourceSASAddress,
                       configureExpander->SASAddress,
                       &SMPRequestFrame,
                       &SMPResponseFrame,
                       &openStatus,
                       CONFIGURE_ROUTE_INFORMATION,
```

```
                                        index,
                                        phyIndex,
                                        disableRouteEntry,
                                        discover->AttachedSASAddress);

                        if((openStatus != OPEN_ACCEPT) ||
                           (SMPResponseFrame.FunctionResult !=
                            SMP_FUNCTION_ACCEPTED))
                        {
                           error = 1;
                           break;
                        }


                        // add the address to the internal copy of the
                        // route table, if successfully configured
                        entry->RoutedSASAddress =
                           discover->AttachedSASAddress;

                        // increment the route index for this phy
                        configureExpander->RouteIndex[phyIndex]++;
                     }
                  }
               }
            }
         }

         // move upstream
         thisExpander = configureExpander;

      } while(!error &&
              thisExpander &&
              upstreamSASAddress);

      return(error);
   } // ConfigureExpander

   // this determines if the expander has a table routed phy attached to the
   // sas address and phyIndex provided
   static
   int CheckForTableAttribute(struct TopologyTable *Expander,
                              byte PhyIndex,
                              quadword SASAddress)
   {
      int table = 0;
      byte phyCount;

      for(phyCount = 0;
          phyCount < Expander->Device.NumberOfPhys;
          phyCount++)
      {
         // did we find the address attached locally
         if((SASAddress ==
             Expander->Phy[phyCount].Result.AttachedSASAddress) &&
            (phyCount == PhyIndex) &&
            (Expander->Phy[phyCount].Result.RoutingAttribute == TABLE))
         {
            table = 1;
            break;
```

```
        }
    }

    return(table);
} // CheckForTableAttribute

// this discovers then configures as necessary the expanders it finds
// within the SAS domain that are „downstream"
static
struct TopologyTable *DiscoverAndConfigure(byte PhyIdentifier,
                                           quadword SourceSASAddress,
                                           struct TopologyTable *HubExpander,
                                           struct TopologyTable **DeviceSet)
{
    struct TopologyTable *currentExpander = HubExpander;
    struct TopologyTable *nextExpander;

    struct Discover *currentDiscover;

    quadword sasAddress;
    byte phyIndex;

    int error = 0;

    // this is a level descent traversal with a configuration stage
    // at each transition to a new level, if a configuration is required
    // by the expander

    // the discover process moves forward through the topology, but the
    // configuration process stays anchored at the hub of the
    // topology, meaning the fanout expander, or the top most subtractive
    // edge expander device
    // this ensures that as each new expander is added to the
    // topology table list, it is in the configuration chain

    while(!error &&
          currentExpander)
    {
        // start at phy 0
        phyIndex = 0;

        // walk through all the phys of the current expander looking for
        // new expanders to add to the topology table
        do
        {
            // this is just a pointer helper
            currentDiscover = &(currentExpander->Phy[phyIndex].Result);

            // look for phys with edge or fanout devices attached...
            if((currentDiscover->RoutingAttribute == TABLE) &&
               ((currentDiscover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
                (currentDiscover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)
               ))
            {
                struct TopologyTable *thisExpander = currentExpander;
                struct TopologyTable *previousExpander = currentExpander;

                // check to see if we already have the address information
                // in our expander list
                while(thisExpander)
```

```
        {
            // if we do, then stop here
            if(currentDiscover->AttachedSASAddress ==
               thisExpander->SASAddress)
            {
               break;
            }

            // setup the pointer references
            previousExpander = thisExpander;
            thisExpander = thisExpander->Next;
        }

        // if we did not have the expander in our list, then get
        // the information
        if(!thisExpander)
        {
            // discover all the details about the attached expander
            // and insert into the master list
            thisExpander =
               DiscoverExpander(PhyIdentifier,
                                SourceSASAddress,
                                currentDiscover->AttachedSASAddress);

            // if we got the discover information, then add it to the
            // list
            if(thisExpander)
            {
            if(CheckForTableAttribute(thisExpander,
                                      phyIndex,
                                      currentDiscover->SASAddress))
               {
                  // output an error message
                  TracePrint(„\n"
                             „error table phys connected together\n");
               }
               else
               {

            previousExpander->Next = thisExpander;

            // go through the configure cycle progressively ascending
            // to each expander starting at „thisExpander"
            ConfigureExpander(PhyIdentifier,
                              SourceSASAddress,
                              HubExpander,
                              thisExpander);
               }
            }
            }
        }

        // look for subtractive phys with edge or fanout devices attached...
        else
        if(DeviceSet &&
           (currentDiscover->RoutingAttribute == SUBTRACTIVE) &&
           ((currentDiscover->AttachedDeviceType == EDGE_EXPANDER_DEVICE) ||
           (currentDiscover->AttachedDeviceType == FANOUT_EXPANDER_DEVICE)))
        {
           if(*DeviceSet == 0)
```

```
            {
                struct TopologyTable *thisExpander = currentExpander;
                struct TopologyTable *previousExpander = currentExpander;

                // check to see if we already have the address information
                // in our expander list
                while(thisExpander)
                {
                    // if we do, then stop here
                    if(!memcmp(&currentDiscover->AttachedSASAddress,
                                &thisExpander->SASAddress,
                                8))
                    {
                        break;
                    }

                    // setup the pointer references
                    previousExpander = thisExpander;
                    thisExpander = thisExpander->Next;
                }

                // if we did not have the expander in our list, then get
                // the information
                if(!thisExpander)
                {
                    // discover all the details about the attached expander
                    // and insert into the master list
                    thisExpander =
                        DiscoverExpander(PhyIdentifier,
                                        SourceSASAddress,
                                        currentDiscover->AttachedSASAddress);

                    // if we got the discover information, then set it as the
                    // other device set
                    if(thisExpander)
                    {
                        *DeviceSet = thisExpander;
                    }
                }
            }
        }

        // move to the next phy on this expander
        phyIndex++;

    } while(phyIndex <
            currentExpander->Device.NumberOfPhys);

    // cycle to the next expander to discover
    currentExpander = currentExpander->Next;
    }

    // return the top of expander list
    return(HubExpander);
} // DiscoverAndConfigure

// this routine appends the leaf to the tree domain
static
void ConcatenateSASDomains(struct TopologyTable *Tree,
                        struct TopologyTable *Leaf)
```

```
        {
           while(Tree)
           {
              if(Tree->Next == 0)
              {
                 Tree->Next = Leaf;
                 break;
              }

              Tree = Tree->Next;
           }
        } // ConcatenateSASDomains

        // validate the route table entries for all expanders
        static
        int ValidateRouteTables(byte PhyIdentifier,
                                quadword SourceSASAddress,
                                struct TopologyTable *Expander,
                                int SASCompatibility)
        {
           struct ReportRouteInformation *route;

           // buffers used to request and return SMP data
           struct SMPRequest request = { 0 };
           struct SMPResponse response = { 0 };

           byte phyIndex;
           word routeIndex;

           byte openStatus = OPEN_ACCEPT;

           int valid = 1;

           if(SASCompatibility == SAS_10_COMPATIBLE)
           {
              // this is just a pointer helper
              route = &(response.Response.ReportRouteInformation.Result);

              // walk the list of expanders
              while(valid &&
                    Expander)
              {
                 if(Expander->Device.ConfigurableRouteTable)
                 {
                    struct RouteTableEntry *entry;

                    word expanderRouteIndexes;

                    _swab( (char *)&Expander->Device.ExpanderRouteIndexes,
                           (char *)&expanderRouteIndexes,
                           sizeof(word) );

                    for(phyIndex = 0;
                        (valid &&
                         (phyIndex < Expander->Device.NumberOfPhys));
                        phyIndex++)
                    {
                       // loop through all the phys of the expander
                       for(routeIndex = 0;
                           (valid &&
```

```
                              ((routeIndex <
                                Expander->RouteIndex[phyIndex]) &&
                                (routeIndex <
                                expanderRouteIndexes)));
                             routeIndex++)
                    {
                        openStatus = OPEN_ACCEPT;

                        // report the route indexes for the expander
                        SMPRequest(PhyIdentifier,
                                   SourceSASAddress,
                                   Expander->SASAddress,
                                   &request,
                                   &response,
                                   &openStatus,
                                   REPORT_ROUTE_INFORMATION,
                                   routeIndex,
                                   phyIndex);

                        if((openStatus != OPEN_ACCEPT) ||
                           (response.FunctionResult !=
                            SMP_FUNCTION_ACCEPTED))
                        {
                            break;
                        }

                        entry = &(Expander->RouteTable[phyIndex][routeIndex]);

                        if((memcmp(&entry->RoutedSASAddress,
                                   &route->RoutedSASAddress,
                                   8)) ||
                           (entry->ExpanderRouteEntryDisabled !=
                            route->ExpanderRouteEntryDisabled))
                        {
                            valid = 0;
                        }
                    }
                }
            }

            Expander = Expander->Next;
        }
    }

    return(valid);
} // ValidateRouteTables

// validate that the change count for the hub expander is still the same
// as when we started
static
int ChangeCount(byte PhyIdentifier,
                quadword SourceSASAddress,
                struct TopologyTable *Expander)
{
    // buffers used to request and return SMP data
    struct SMPRequest request = { 0 };
    struct SMPResponse response = { 0 };

    int change = 0;
```

```
        byte openStatus = OPEN_ACCEPT;

        // get the report general information for the expander
        SMPRequest(PhyIdentifier,
                    SourceSASAddress,
                    Expander->SASAddress,
                    &request,
                    &response,
                    &openStatus,
                    REPORT_GENERAL);

        // don't worry about too much in the ,else' case for this example,
        // production code needs to handle
        if((openStatus == OPEN_ACCEPT) &&
            (response.FunctionResult == SMP_FUNCTION_ACCEPTED))
        {
            if(response.Response.ReportGeneral.ExpanderChangeCount !=
               Expander->Device.ExpanderChangeCount)
            {
                change = 1;
            }
        }

        return(change);
    } // ChangeCount

    // delete all the topology information and start over
    void DeleteSASDomain(struct TopologyTable **Expander)
    {
        struct TopologyTable *expander = *Expander;
        struct TopologyTable *nextExpander = 0;

        // walk the list of expanders
        while(expander)
        {
            nextExpander = expander->Next;

            free(expander);

            expander = nextExpander;
        }

        *Expander = 0;
    } // DeleteSASDomain

    // the application client for the initiator device makes a call to
    // this function to begin the discover process...
    // to simplify the setup for the simulation, the DiscoverProcess gets
    // the Initiator number to allow multiple initiators...
    void DiscoverProcess(byte Initiator,
                          byte PhyIdentifier)
    {
        // check to see if an expander is attached
        if((Attached[Initiator].DeviceType == EDGE_EXPANDER_DEVICE) ||
            (Attached[Initiator].DeviceType == FANOUT_EXPANDER_DEVICE))
        {
            struct TopologyTable *connectedExpander;

            // get some local variables to keep things simple
            quadword sourceSASAddress = Iam[Initiator].SASAddress;
```

```
quadword destinationSASAddress = Attached[Initiator].SASAddress;

// expander is attached, so begin by getting the information about
// the connected expander
connectedExpander = DiscoverExpander(PhyIdentifier,
                                     sourceSASAddress,
                                     destinationSASAddress);

// make sure we get the information from the expander
if(connectedExpander)
{
    struct TopologyTable *thisDeviceSet;
    struct TopologyTable *attachedDeviceSet = 0;

    int redoDiscover = 0;
    int changed = 0;

    do
    {
        // go upstream on the subtractive phys until we discover that we
        // are attached to another subtractive phy or a fanout expander
        // then begin the discover process from that point, this works
        // because any new address that we find naturally moves
        // upstream due to the subtractive addressing method
        // if during the discover cycle, it is determined that there
        // are two device sets connected, then a second discover
        // and configuration cycle is required for the other device set
        thisDeviceSet = FindBoundary(PhyIdentifier,
                                     sourceSASAddress,
                                     connectedExpander,
                                     &attachedDeviceSet);

        // set the root for the domain as the subtractive boundary
        if(thisDeviceSet)
        {
            // output a little information about the subtractive boundary
            TracePrint(„subtractive boundary at %s\n",
                       (char *)&thisDeviceSet->SASAddress);

            SASDomain[Initiator] = thisDeviceSet;
        }
        // if there was no subtractive boundary, then the root is the
        // expander connected to the initiator
        else
        {
            // output a little information about the subtractive boundary
            TracePrint(„connected expander at %s\n",
                       (char *)&connectedExpander->SASAddress);

            SASDomain[Initiator] = connectedExpander;
        }

        // begin the discover and configuration cycle
        DiscoverAndConfigure(PhyIdentifier,
                             sourceSASAddress,
                             SASDomain[Initiator],
                             &attachedDeviceSet);

        // if two device sets are connected, then the attached device set
        // has to be discovered and configured
```

```
            if(attachedDeviceSet)
            {
                // output a little information about the attached device set
                TracePrint("attached device set at %s\n",
                            (char *)&attachedDeviceSet->SASAddress);

                // discover and configure the attached device set
                DiscoverAndConfigure(PhyIdentifier,
                                    sourceSASAddress,
                                    attachedDeviceSet,
                                    0);

                // put the domains together
                ConcatenateSASDomains(SASDomain[Initiator],
                                    attachedDeviceSet);
            }

            // if the change count of the top most expander is different
            // from when we started, then the topology was not stable
            // so do the discover again
            changed = ChangeCount(PhyIdentifier,
                            sourceSASAddress,
                            SASDomain[Initiator]);

            // if we used the route table optimization,
            // check the route tables for each expander phy, if they are
            // incorrect then change back to the original discover algorithm
            // and redo discover, continue to use the original algorithm
            // for any new discover
            if(!changed &&
                (DiscoverAlgorithm == SAS_UNIQUE_LEVEL_DESCENT) &&
                !ValidateRouteTables(PhyIdentifier,
                                sourceSASAddress,
                                SASDomain[Initiator],
                                SASCompatibility))
            {
                redoDiscover = 1;

                // if the change count of the top most expander is the same
                // as when we started the validation of the route tables
                // then the topology was stable, so change the algorithm
                // before the rediscover
                if(!ChangeCount(PhyIdentifier,
                            sourceSASAddress,
                            SASDomain[Initiator]))
                {
                    DiscoverAlgorithm = SAS_SIMPLE_LEVEL_DESCENT;
                }

                // delete everything allocated and start over
                DeleteSASDomain(&SASDomain[Initiator]);
            }

        } while(redoDiscover ||
                changed);
    }
  }
} // DiscoverProcess
```
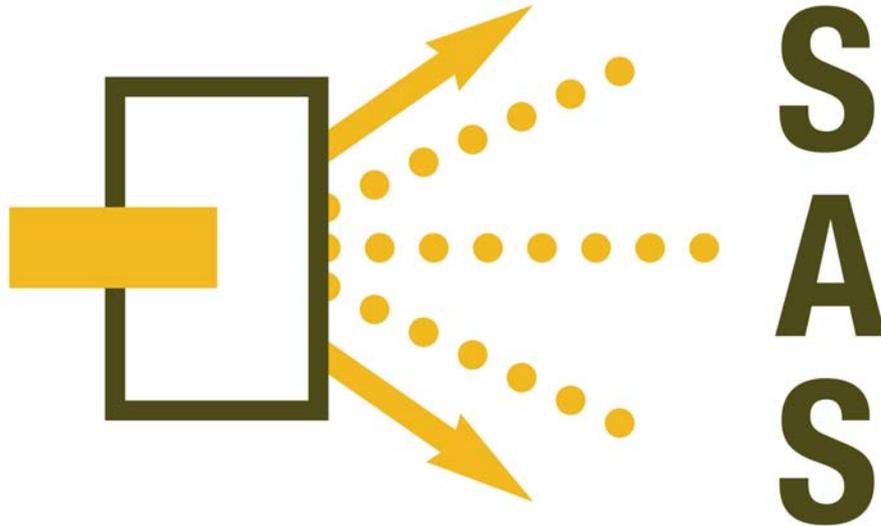
Annex M
(informative)

SAS icon

Figure M.1 shows the general SAS icon. This icon should be included on or near all connectors used by devices compliant with this standard.

Figure M.1 — SAS icon

NOTE 72 - Contact the SCSI Trade Association at http://www.scsita.org for versions of the SAS icons in various graphics formats.

Bibliography

Serial ATA II: Electrical Specification (SATAII-PHY). Revision 1.0. 26 May 2004

Serial ATA II: Extensions to Serial ATA 1.0a (SATAII-EXT). Revision 1.2. 27 August 2004

Serial ATA II: Port Selector (SATAII-PS). Revision 1.0. 28 July 2003

NOTE 73 - For information on the current status of Serial ATA documents, see the Serial ATA International Organization at http://www.sata-io.org.